

バージョン タグ名 HTML5 BODY要素	内容
形式サンプル	
5.5.4.2 og:ajaxTreeParam Tag	<p>【TAG】子データを取得するためのJSPに渡す引数のカラム名をセットします(初期値:``)。 子データを取得するためのJSPに渡す引数のカラム名をセットします。 親データから子データを展開する際に、ここで設定されたキーの値をDBTableModelから 取得し、子データ検索用のJSPに渡します。 子データ検索用のJSPでは通常のリクエスト変数(「@カラム名」)として値を取得することが可能です。</p> <p>keys="YYY, XXX, ZZZ" childJsp="getChildTag. jsp" /></p> <p>●形式: <og:ajaxTreeParam ●body: なし</p> <p>●Tag定義:</p> <pre><og:ajaxTreeParam childSearchKeys 【TAG】子データを取得するためのJSPに渡す引数のカラム名をセットします(初期値:``) childSearchJsp 【TAG】子データを取得するためのJSPをセットします(初期値:"getChildTag. jsp") levelClim 【TAG】階層レベルとして処理(展開を制御するための画像イメージを表示)するカラム名をセットします(初期値:"LVL") imgCollapsed 【TAG】折りたたまれた状態を表現するための画像ファイル名をセットします(初期値:"/jsp/image/collapsed.gif") imgExpanded 【TAG】展開された状態を表現するための画像ファイル名をセットします(初期値:"/jsp/image/expanded.gif") imgNoSub 【TAG】展開後子データが存在しない状態を表現するための画像ファイル名をセットします(初期値:"/jsp/image/nosub.gif") expandAll 【TAG】初期状態で全展開を行うかを指定します(初期値:false=全展開しない) childViewStartNo 【TAG】子データを表示するためのStartNoを指定します(初期値:-1) expCtrlClim 【TAG】初期全展開した際に最下位の要素の状態(展開済 or 下位あり)を制御するためのカラムを指定します(初期値:EXPAND_CONTROL) debug 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false) /></pre> <p>●使用例</p> <p>ViewFormTag の viewFormType が、HTMLAJaxTreeTable の場合に使用します。 useParam 属性を設定しておかないと、使用されません。</p> <pre><og:view viewFormType = "HTMLAJaxTreeTable" command = "{@command}" startNo = "0" pageSize = "20" useParam = "true" > <og:ajaxTreeParam childSearchKeys = "YYY, XXX, ZZZ" childSearchJsp = "getChildTag. jsp" levelClim = "LVL" imgCollapsed = "collapsed.gif" imgExpanded = "expanded.gif" imgNoSub = "nosub.gif" /> </og:view ></pre>
1.childSearchKeys	<p>【TAG】子データを取得するためのJSPに渡す引数のカラム名をセットします(初期値:``)。 子データを取得するためのJSPに渡す引数のカラム名をセットします。 親データから子データを展開する際に、ここで設定されたキーの値をDBTableModelから 取得し、子データ検索用のJSPに渡します。 子データ検索用のJSPでは通常のリクエスト変数(「@カラム名」)として値を取得することが可能です。</p>
2.childSearchJsp	<p>【TAG】子データを取得するためのJSPをセットします(初期値:"getChildTag. jsp")。 子データを取得するためのJSPをセットします。 このJSPで子データを取得し、viewタグでHTML出力した結果をパースし、子データとして 親データの直下に要素を挿入します。 子データ検索用のJSPでは通常のリクエスト変数(「@カラム名」)として値を取得することが可能です。</p>
3.levelClim	<p>【TAG】階層レベルとして処理(展開を制御するための画像イメージを表示)するカラム名をセットします(初期値:"LVL")。 階層レベルとして処理(展開を制御するための画像イメージを表示)するカラム名をセットします。 子データを取得した際に、ここで指定されたカラムは、親レベルに対して自動的に+1されます。</p>
4.imgCollapsed	<p>【TAG】折りたたまれた状態を表現するための画像ファイル名をセットします(初期値:"/jsp/image/collapsed.gif")。 折りたたまれた状態を表現するための画像ファイル名をセットします。 「折りたたまれた状態」とは、未展開の状態、及び展開後に折りたたんだ状態の両方を指します。</p>
5.imgExpanded	<p>【TAG】展開された状態を表現するための画像ファイル名をセットします(初期値:"/jsp/image/expanded.gif")。 展開された状態を表現するための画像ファイル名をセットします。</p>
6.imgNoSub	<p>【TAG】展開後子データが存在しない状態を表現するための画像ファイル名をセットします(初期値:"/jsp/image/nosub.gif")。 展開後子データが存在しない状態を表現するための画像ファイル名をセットします。</p>
7.expandAll	<p>【TAG】初期状態で全展開を行うかを指定します(初期値:false=全展開しない)。 初期状態で全展開を行うかを指定します。DBTableModel上のデータが展開済みのデータ であると仮定し、全展開を行った状態で表示します。 ※ この属性をtrueに指定しても子データ取得用のJSPが再帰的に呼び出される訳では ありません。子データの取得は、予めqueryで行っておく必要があります。</p>
8.childViewStartNo	<p>【TAG】子データを表示するためのStartNoを指定します(初期値:-1)。 子データを取得するためのJSP内で使用します。 子データを取得し、元のDBTableModelにマージした後、子データのみを表示するため、 ここでstartNoを指定して差分データを表示します。</p>
9.expCtrlClim	<p>【TAG】初期全展開した際に最下位の要素の状態(展開済 or 下位あり)を制御するためのカラムを指定します(初期値:EXPAND_CONTROL)。 初期全展開した際に最下位の要素の状態(展開済 or 下位あり)を制御するためのカラムを指定します。 ここで指定されたカラムの値により、全展開された結果の最下位の要素の状態指定することができます。 この機能を利用することで、第3レベルまで存在するデータの内の、第2レベルまでを展開しておく、 ということを実現することができます。 ここで指定したカラムが持つ値の意味は以下の通りです。 '0' or 'false' 展開済 (初期値) '1' or 'true' 未展開 ※展開ボタンを押すことで更に下位展開が可能</p> <p>なお、この機能は、全展開時(expandAll="true")のみ有効であり、通常の第1レベル展開の場合は、 このカラムは無視されます。</p>
10.debug	<p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない] を指定します。 出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:and JSP	【TAG】SQL条件句の最初の演算子を指定します(初期値:and)。 value を連結する場合の頭に置かれる文字列で、where句の最初には表示されず、それ以降について、表示されます。 (つまり、where VALUE1 and VALUE2 and VALUE3 … です。) startKey の初期値は、“and” です。
●形式: <og:and startKey=“[and or …]” value=“…” multi=“[false true]” /> ●body: あり	
●Tag定義: <og:and startKey value multi separator quotCheck instrVals instrType xssCheck caseKey caseVal debug > ... Body ... </og:and>	【TAG】SQL条件句の最初の演算子を指定します(初期値:and) 【TAG】条件の値を セットします 【TAG】複数の引数に対して処理するかどうか[true/false]を設定します(初期値:false) 【TAG】multi アクション時の文字列を分割する項目区切り文字をセットします 【TAG】リクエスト情報の クォーティション(') 存在チェックを実施するかどうか[true/false]を設定します (初期値:USE_SQL_INJECTION_CHECK[=true]) 【TAG】スペースで区切られた複数の値すべてを含む条件を作成します 【TAG】instrValsで複数の値を条件にする際の方法を指定します(初期値:and) 【TAG】リクエスト情報の HTMLTag開始/終了文字(><) 存在チェックを実施するかどうか[true/false]を設定します (初期値:USE_XSS_CHECK[=true]) 【TAG】このタグ自体を利用するかどうかの条件キーを指定します(初期値:null) 【TAG】このタグ自体を利用するかどうかの条件値を指定します(初期値:null) 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)
●使用例 <og:query command=“NEW”> select PN,YOBI,NMEN,HINM from XX01 <og:where> <og:and value=“PN = '[@PN]’” /> <og:and value=“YOBI like '[@YOBI]’” /> </og:where> order by PN </og:query> ・検索条件が入力された時(PN=AAA , YOBI=BBB) 作成されるSQL文⇒select PN,YOBI,NMEN,HINM from XX01 where PN = 'AAA' and YOBI like 'BBB%' order by PN ・検索条件が片方入力されなかった時(PNがNULLのとき, YOBI=BBB) 作成されるSQL文⇒select PN,YOBI,NMEN,HINM from XX01 where YOBI like 'BBB%' order by PN ・検索条件が入力されなかった時(PNがNULL, YOBIがNULL) WHERE句がなくなる。 作成されるSQL文⇒select PN,YOBI,NMEN,HINM from XX01 order by PN 注意:WhereTagを使わない場合に、検索条件が入力されなかった場合は、下記ようになります。 select PN,YOBI,NMEN,HINM from XX01 where PN = '' and YOBI like '%' order by PN	
1.startKey	【TAG】SQL条件句の最初の演算子を指定します(初期値:and)。 value を連結する場合の頭に置かれる文字列で、where句の最初には表示されず、それ以降について、表示されます。 (つまり、where VALUE1 and VALUE2 and VALUE3 … です。) startKey の初期値は、“and” です。
2.value	【TAG】条件の値を セットします。 条件値に、{@XXXX} 変数が含まれている場合、そのリクエスト値がない場合は、このタグそのものがなにも出力しません。(つまり条件から消えます。) BODY 部に記述することが可能です。その場合は、条件属性になにも設定できません。
3.multi	【TAG】複数の引数に対して処理するかどうか[true/false]を設定します(初期値:false)。 {@XXXX} 変数に、値が複数含まれている場合の処理を規定します。 multi=“true” に設定すると、複数の引数は、'xx1','xx2','xx3'、… という形式に変換します。 where 条件で言うと、“where PN in ([@PN])”という文字列に対して、“where PN in ('xx1','xx2','xx3')”を作成することになります。 初期値は、 false (マルチ変換しない) です。
4.separator	【TAG】multi アクション時の文字列を分割する項目区切り文字をセットします。 multi=“true” の場合、複数のリクエストを連結して、 in 句で問合せを行う文字列を作成しますが、separator を指定すると、さらに、separator で文字列を分割して、 in 句の引数を構築します。 具体的には、分割後の文字列が、複数の個々のリクエスト変数と同じ形式に加工されます。 この機能は、multi=“true” を指定した場合のみ有効になります。 初期値は、 null です。つまり、分割処理は行いません。
5.quotCheck	【TAG】リクエスト情報の クォーティション(') 存在チェックを実施するかどうか[true/false]を設定します (初期値:USE_SQL_INJECTION_CHECK[=true])。 SQLインジェクション対策の一つとして、暫定的ではありますが、SQLのパラメータに渡す文字列にクォーティション(') を許さない設定にすれば、ある程度は防止できます。 数字タイプの引数には、 or 5=5 などのクォーティションを使用しないコードを埋めても、数字チェックで検出可能です。文字タイプの場合は、必ず (') をはずして、 ' or 'A' like 'A' のような形式になる為、 (')チェックだけでも有効です。 (') が含まれていたエラーにする(true)／かノーチェックか(false)を指定します。 (初期値:システム定数のUSE_SQL_INJECTION_CHECK[=@og.value org.opengion.hayabusa.common.SystemData#USE_SQL_INJECTION_CHECK])。
6.instrVals	【TAG】スペースで区切られた複数の値すべてを含む条件を作成します。 通常、value=“CLM LIKE 'ABC%'”という文字列を指定しますが、value=“CLM” instrVals=“ABC DEF GHI”と指定すると、value=“CLM LIKE '%ABC%' AND CLM LIKE '%DEF%' AND CLM LIKE '%GHI%'”という文字列を作成します。 これは、instrVals に指定した引数に対して、スペース区切りで分割し、前方の value に複数のAND検索を同時に実現できるように指定します 個別にLIKE検索項目を AND 連結する為、現れる場所に依存しません。 逆に、現れる順序を指定する場合は、ABC%DEF の様に指定可能です。

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
7.instrType	<p>ただし、columnMarker の instrVals で、複数文字のマーカーを行う場合、ABC%DEF という文字列は、オリジナルでないので、マークアップされません。 ※instrType属性の指定により条件の生成方法を変更することができます。 詳細については、instrType属性のドキュメントを参照下さい。</p> <p>【TAG】instrValsで複数の値を条件にする際の方法を指定します(初期値:and)。 通常、instrValsに指定された値は、スペース区切りで分割した各値をLIKE条件としてand結合します。 しかし、instrType属性を変更することで、この条件式の生成方法を変更することができます。 具体的には、以下の通りです。</p> <p>①instrTypeに“and”が指定されている場合(初期値) タグの記述 : value="CLM" instrVals="ABC DEF GHI" 生成文字列 : “(CLM LIKE '%ABC%' AND CLM LIKE '%DEF%' AND CLM LIKE '%GHI%')”</p> <p>②instrTypeに“or”が指定されている場合 タグの記述 : value="CLM" instrVals="ABC DEF GHI" 生成文字列 : “(CLM LIKE '%ABC%' OR CLM LIKE '%DEF%' OR CLM LIKE '%GHI%')”</p> <p>③instrTypeに“in”が指定されている場合 タグの記述 : value="CLM" instrVals="ABC DEF GHI" 生成文字列 : “(CLM LIKE 'ABC' OR CLM LIKE 'DEF5' OR CLM LIKE 'GHI')”</p> <p>④instrTypeに“notin”が指定されている場合 タグの記述 : value="CLM" instrVals="ABC DEF GHI" 生成文字列 : “(CLM NOT LIKE 'ABC' AND CLM NOT LIKE 'DEF5' AND CLM NOT LIKE 'GHI')”</p> <p>※この属性を指定しない場合は、①のLIKE条件でのand結合となります。 ※③④について、LIKE条件で%を自動付加しないことにより、画面からの入力値に応じて、前方一致、後方一致、前後方一致の制御を行うことができます。</p> <p>【TAG】リクエスト情報の HTMLTag開始/終了文字(><) 存在チェックを実施するかどうか[true/false]を設定します (初期値:USE_XSS_CHECK[=true])。</p> <p>クロスサイトスクリプティング(XSS)対策の一環としてless/greater than signについてのチェックを行います。 >< が含まれていたエラーにする(true)／かノーチェックか(false)を指定します。 (初期値:システム定数のUSE_XSS_CHECK[=@og.value.org.opengion.hayabusa.common.SystemData#USE_XSS_CHECK])。</p>
8.xssCheck	<p>【TAG】このタグ自体を利用するかどうかの条件キーを指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。</p>
9.caseKey	<p>【TAG】このタグ自体を利用するかどうかの条件値を指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。</p>
10.caseVal	<p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。</p>
11.debug	<p>デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
<p>5.5.4.2 og:appear</p> <p>Tag</p> <p>●形式 : <og:appear startKey="order by …" value="…" defaultVal="…" /></p> <p>●body : なし</p> <p>●Tag定義</p> <pre> <og:appear startKey value defaultVal debug /> </pre> <p>●使用例</p> <pre> <!-- DB検索 SQL文記述 debug="true" でSQL文を確認できます。--> <og:query command="@command" debug="@debug" maxRowCount="@maxRowCount"> select CLM, NAME_JA, LABEL_NAME, URL, KBSAKU, SYSTEM_ID, LANG, FGJ, (CASE WHEN URL IS NULL THEN 0 ELSE 1 END) AS ONMARK from GF41 <!-- 検索条件でWhereTagを使用すれば{@xxxx}がNULLの場合、その条件は無視されます。 --> <og:where> <og:and value = "FGJ" in ('0','1') /> <og:and value = "SYSTEM_ID" = '@SYSTEM_ID' /> <og:and value = "LANG" = '@LANG' /> <og:and value = "CLM" like '@CLM' %' /> <og:and value = "NAME_JA" like '@NAME_JA' %' /> <og:and value = "LABEL_NAME" like '@LABEL_NAME' %' /> <og:and value = "KBSAKU" = '@KBSAKU' /> </og:where> <!-- ORDER BY句でAppearTagを使用すれば{@ORDER_BY}がNULLの場合、ORDER BY句は無視されます。 --> <!-- また、{@ORDER_BY}がNULLの場合に、defaultVal属性を指定すれば、その値でORDER BY表示されます。 --> <og:appear startKey = "order by" value = "@ORDER_BY" defaultVal = "SYSTEM_ID, CLM, LANG" /> </og:query> </pre>	<p>【TAG】 開始文字列を設定します(初期値:"")。</p> <p>このキーは、バリューと接続される場合に空白文字を一つ挿入します。</p> <p>この値は、固定値のみ設定可能です。{@XXXX}文字は使用できません。</p> <p>○【TAG】 値をセットします(指定された値が 設定されている場合のみ使用されます) (必須)</p> <p>【TAG】 初期値を設定します(value値がNULLの場合に、この初期値が表示)</p> <p>【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)</p>
1.startKey	<p>【TAG】 開始文字列を設定します(初期値:"")。</p> <p>このキーは、バリューと接続される場合に空白文字を一つ挿入します。</p> <p>この値は、固定値のみ設定可能です。{@XXXX}文字は使用できません。</p>
2.value	<p>【TAG】 値をセットします(指定された値が 設定されている場合のみ使用されます)。</p> <p>指定された値が 設定されている場合のみ、開始文字列(startKey)と組み合わせれて、使用されます。</p> <p>これは、一般にvalue値が変動する場合に、defaultVal 等に重複する値を設定したくない場合に使用します。{@XXXX}文字が使用できます。</p>
3.defaultVal	<p>【TAG】 初期値を設定します(value値がNULLの場合に、この初期値が表示)。</p> <p>value値がNULL(指定されない)の場合に、この初期値が値として使用されます。</p>
4.debug	<p>【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。</p> <p>デバッグ情報を [true:出力する/false:しない]を指定します。</p> <p>出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTML5 BODY要素	内容												
形式サンプル													
5.5.4.2 og:backGamen 拡張 JSP	<p>【TAG】 ボタンのタイプ[link/button/reLink/reButton/historyBack]を指定します(初期値:link)。button と設定すると、戻るボタンに、link と設定すると、戻るリンクになります。historyBack は、IE等の戻る操作と同じで、JavaScriptのヒストリーバックを行います。また、reButton、reLinkとすると、それぞれ相対パス(画面IDから飛び先のアドレスを元に生成)で戻り先のアドレスが生成されます。初期値は、戻るリンク(link)です。</p> <p>></p> <table><tr><th>タイプ</th><th>説明</th></tr><tr><td>link</td><td>戻るリンク</td></tr><tr><td>button</td><td>戻るボタン</td></tr><tr><td>reLink</td><td>戻るリンク(相対パス)</td></tr><tr><td>reButton</td><td>戻るボタン(相対パス)</td></tr><tr><td>historyBack</td><td>通常のヒストリバックボタン</td></tr></table>	タイプ	説明	link	戻るリンク	button	戻るボタン	reLink	戻るリンク(相対パス)	reButton	戻るボタン(相対パス)	historyBack	通常のヒストリバックボタン
タイプ	説明												
link	戻るリンク												
button	戻るボタン												
reLink	戻るリンク(相対パス)												
reButton	戻るボタン(相対パス)												
historyBack	通常のヒストリバックボタン												
<p>●形式: <og:backGamen keys="..." vals="..." > ..Body.. </og:backGamen></p> <p>●body: あり</p> <p>●Tag定義:</p> <pre><og:backGamen type command gamenId keys vals lbl href target id lang dir title style tabIndex accesskey clazz language onClick onBlur onFocus ondblClick onMouseDown onMouseUp onMouseMove onMouseOut onMouseOver debug > ... Body ... </og:backGamen></pre> <p>【TAG】 ボタンのタイプ[link/button/reLink/reButton/historyBack]を指定します(初期値:link)</p> <p>【TAG】 (通常使いません) 戻る時に指定する command を設定できます(初期値:RENEW)</p> <p>【TAG】 (通常使いません) 戻り先の画面をセットします(初期値:BACK_GAMENID)</p> <p>【TAG】 リンク先に渡すキーをCSV形式で複数指定します</p> <p>【TAG】 keys属性に対応する値をCSV形式で複数指定します</p> <p>【TAG】 ラベルリソースのラベルIDを指定します</p> <p>【HTML】 リンク先のURLを指定します</p> <p>【HTML】 リンク先の表示ターゲットを指定します(初期値:CONTENTS)</p> <p>【HTML】 要素に対して固有の名前(id)をつける場合に設定します</p> <p>【HTML】 要素の内容と他の属性値の言語(lang.xml:lang)を指定します</p> <p>【HTML】 文字表記の方向(dir)を指定します</p> <p>【HTML】 要素に対する補足的情報(title)を設定します</p> <p>【HTML】 この要素に対して適用させるスタイルシート(style)を設定します</p> <p>【HTML】 タブの移動順(tabindex)を指定します(0 ~ 32767)</p> <p>【HTML】 アクセスキー(alt+キーで直接指定)を割り当てます</p> <p>【HTML】 要素に対して class 属性を設定します</p> <p>【TAG】 タグ内部で使用する言語コード[ja/en/zh/...]を指定します</p> <p>【HTML】 JavaScriptのイベント onClick を設定します(例:onClick="renew(' query, jsp', ' QUERY');")</p> <p>【HTML】 JavaScriptのイベント onBlur を設定します(例:onBlur="this.value=value.toUpperCase();")</p> <p>【HTML】 JavaScriptのイベント onFocus を設定します</p> <p>【HTML】 JavaScriptのイベント ondblClick を設定します</p> <p>【HTML】 JavaScriptのイベント onMouseDown を設定します</p> <p>【HTML】 JavaScriptのイベント onMouseUp を設定します</p> <p>【HTML】 JavaScriptのイベント onMouseMove を設定します</p> <p>【HTML】 JavaScriptのイベント onMouseOut を設定します</p> <p>【HTML】 JavaScriptのイベント onMouseOver を設定します</p> <p>【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)</p> <p>●使用例</p> <pre><og:backGamen keys="displayMsg,clear" vals="MSG0065,true" > <og:message lbl="MSG0049" /> </og:backGamen></pre>													
1.type	<p>【TAG】 ボタンのタイプ[link/button/reLink/reButton/historyBack]を指定します(初期値:link)。button と設定すると、戻るボタンに、link と設定すると、戻るリンクになります。historyBack は、IE等の戻る操作と同じで、JavaScriptのヒストリーバックを行います。また、reButton、reLinkとすると、それぞれ相対パス(画面IDから飛び先のアドレスを元に生成)で戻り先のアドレスが生成されます。初期値は、戻るリンク(link)です。</p> <p>></p> <table><tr><th>タイプ</th><th>説明</th></tr><tr><td>link</td><td>戻るリンク</td></tr><tr><td>button</td><td>戻るボタン</td></tr><tr><td>reLink</td><td>戻るリンク(相対パス)</td></tr><tr><td>reButton</td><td>戻るボタン(相対パス)</td></tr><tr><td>historyBack</td><td>通常のヒストリバックボタン</td></tr></table>	タイプ	説明	link	戻るリンク	button	戻るボタン	reLink	戻るリンク(相対パス)	reButton	戻るボタン(相対パス)	historyBack	通常のヒストリバックボタン
タイプ	説明												
link	戻るリンク												
button	戻るボタン												
reLink	戻るリンク(相対パス)												
reButton	戻るボタン(相対パス)												
historyBack	通常のヒストリバックボタン												
2.command	<p>【TAG】 (通常使いません) 戻る時に指定する command を設定できます(初期値:RENEW)。通常は、RENEW で戻ります。(初期値は、RENEW なので設定不要です。)</p>												
3.gamenId	<p>【TAG】 (通常使いません) 戻り先の画面をセットします(初期値:BACK_GAMENID)。通常は、自動的に、BACK_GAMENID の値がセットされますが、先祖に戻る(画面A⇒画面B⇒画面C のときに、画面Aに戻る)場合や、別の画面に、進む場合に、直接指定します。(初期値は、来た画面:BACK_GAMENID です)ので、設定不要です。)値は、キャッシュではなく、session より HybsSystem.BACK_GAMENID_KEY をキーに取り出します。これは、command="NEW" で、BACK_GAMENID リクエストが存在し、BACK_GAMENID と自分自身の画面IDが異なる場合のみ、RequestCacheTag#backGamenIdSet メソッドでsession に登録されます。</p>												
4.keys	<p>【TAG】 リンク先に渡すキーをCSV形式で複数指定します。戻る時に、検索時のキャッシュに指定した引数以外に指定したり、別の値に置き換えたりする場合のキーを設定できます。カンマ区切りで複数指定できます。vals 属性には、キーに対応する値を、設定してください。例:keys="displayMsg,clear" vals="MSG0065,true" 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。こうしないとデータ自身にカンマを持っている場合に分解をミスの為です。</p>												
5.vals	<p>【TAG】 keys属性に対応する値をCSV形式で複数指定します。キーに設定した値を、カンマ区切り文字で複数して出します。指定順序は、キーと同じにしておいて下さい。例:keys="displayMsg,clear" vals="MSG0065,true" 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。こうしないとデータ自身にカンマを持っている場合に分解をミスの為です。</p>												
	<p>【TAG】 ラベルリソースのラベルIDを指定します。ラベルを変更するときに、lbl属性を使います。</p>												

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
6.lbl	ラベルID は、所定の language に基づく ResourceManager の getLabelData(id) を呼び出し、その結果のLabelInterfaceを使用します。 getMsgLbl() で取り出せます。 ラベルIDとメッセージIDは同時には登録できません。 ラベルとメッセージは統一されました。
7.href	【HTML】リンク先のURLを指定します。 リンク先のURLを指定します。
8.target	【HTML】リンク先の表示ターゲットを指定します(初期値:CONTENTS)。 リンク先の文書を表示させるフレーム、またはウィンドウの名前を指定します。 【HTML】要素に対して固有の名前(id)をつける場合に設定します。
9.id	特別な使用方法として、id="FOCUS" とすることで、フィールド系要素に フォーカスを移動させます。これは、そのページ内で唯一の id 属性として使用ください。
10.lang	【HTML】要素の内容と他の属性値の言語(lang,xml:lang)を指定します。 HTMLの言語属性に使われます。指定する値は、ISO 639で規定されている「言語コード」です。 [ja/en/zh/...]などのほかに、en-US:アメリカ英語、en-cockney:コックニー英語 など、 副言語を指定する方法も定められています。 ここでは、lang と xml:lang の両方に同じ値がセットされます。 タグの language 属性とは使用用途が異なります。lang セット
11.dir	【HTML】文字表記の方向(dir)を指定します。 当該要素の書字方向を指定する属性です。 ltr で、左から右に、rtl で、右から左に並べます。
12.title	【HTML】要素に対する補足的情報(title)を設定します。 title セットは、ボタンなどに適用すると、マウスオーバーによりこのメッセージが チップスのように表示されます。これを利用して、説明文を登録することが可能です。 ここに登録した文字列が、メッセージリソースに存在する場合は、そのメッセージを 存在しない場合は、そのままの値を設定します。
13.style	【HTML】この要素に対して適用させるスタイルシート(style)を設定します。 タグにstyle属性を設定します。これは、キー:値: のセットを複数記述できます。 通常は、class属性や、id属性で登録しておき、&lg:style type="text/css"> で 外部から指定する方がソースは読みやすくなります。
14.tabindex	【HTML】タブの移動順(tabindex)を指定します(0 ~ 32767)。 Tabキーを押したときに要素が選択される順番を指定します。 値には、選択させたい順番を数値で記述します。
15.accesskey	【HTML】アクセスキー(alt+キーで直接指定)を割り当てます。 アクセスキーは、マウスの使えない環境でも、リンクにジャンプする、ボタンを押す、入力フォームに フォーカスを移すなどの操作を簡単に行うことができるように考慮されたものです。 Windows の「ファイル(F)」メニューについての、F と同じような働きをします。
16.clazz	【HTML】要素に対して class 属性を設定します。 Javaの言語使用上、class で作成できないため、代用として clazz を使用しています。 html で作成される属性は、class で作成されます。
17.language	【TAG】タグ内部で使用する言語コード[ja/en/zh/...]を指定します。 この言語コードに基づいて、表示のラベルをリソースから作成します。
18.onClick	【HTML】JavaScriptのイベント onClick を設定します(例:onClick="renew('query.jsp','QUERY');")。 onClick をセットします。 例えば、<og:column name="KBSAKU" onClick="renew('query.jsp','QUERY');"/> のように指定することで、プルダウンメニューの絞り込み検索が可能になります。
19.onBlur	【HTML】JavaScriptのイベント onBlur を設定します(例:onBlur="this.value=value.toUpperCase();")。 onBlur は、フォーカスが離れたときに発生するイベントです。
20.onFocus	【HTML】JavaScriptのイベント onFocus を設定します。 onFocus は、フォーカスされたときに発生するイベントです。
21.ondblClick	【HTML】JavaScriptのイベント ondblClick を設定します。 ondblClick は、マウスでダブルクリックされたときに発生するイベントです。
22.onMouseDown	【HTML】JavaScriptのイベント onMouseDown を設定します。 onMouseDown は、マウスダウンされたときに発生するイベントです。
23.onMouseUp	【HTML】JavaScriptのイベント onMouseUp を設定します。 onMouseUp は、マウスアップされたときに発生するイベントです。
24.onMouseMove	【HTML】JavaScriptのイベント onMouseMove を設定します。 onMouseMove は、マウスが移動されたときに発生するイベントです。
25.onMouseOut	【HTML】JavaScriptのイベント onMouseOut を設定します。 onMouseOut は、マウスが離れたときに発生するイベントです。
26.onMouseOver	【HTML】JavaScriptのイベント onMouseOver を設定します。 onMouseOver は、マウスが重なったときに発生するイベントです。
27.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容															
形式サンプル																
5.5.4.2 og:bizLogic	【TAG】 実行する業務ロジック名を指定します。 実行する業務ロジック名を指定します。業務ロジック名は、クラス名を指定します。 クラス名については、クラス自身の名称のみを指定することができます。 (パッケージ名を含めた完全な形のクラス名を指定することもできます) また、カンマ区切りで、複数指定することもできます。 この場合、指定した順番に処理されます。															
Tag																
●形式：	<div><div><og:bizLog logics command scope dbid tableId selectedAll modifyType keys vals stopError quotCheck xssCheck debug ></div><div>= “業務ロジックのクラス名” = “ENTRY” = “session” = “DEFAULT” = “DEFAULT” = “false” = “A” = “SYSTEM_ID” = “{@SYSTEM_ID}” = “true” = “true” = “true” = “false”</div></div>															
●body：なし																
●Tag定義：	<div><div><og:bizLogic logics command scope dbid tableId selectedAll modifyType keys vals stopError quotCheck xssCheck multi debug ></div><div>○ 【TAG】 実行する業務ロジック名を指定します。(必須) 【TAG】 コマンドをセットします(初期値:ENTRY) 【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session) 【TAG】 (通常は使いません)Queryオブジェクトを作成する時のDB接続IDを指定します 【TAG】 (通常は使いません)結果のDBTableModelを、sessionに登録するときのキーを指定します 【TAG】 データを全件選択済みとして処理するかどうか[true/false]を指定します(初期値:false) 【TAG】 DB検索時の モディファイタイプを指定します[A:追加/C:更新/D:削除] 【TAG】 リンク先に渡すキーをCSV形式で複数指定します 【TAG】 リンク先に渡す値をCSV形式で複数指定します 【TAG】 処理エラーの時に処理を中止するかどうか[true/false]を設定します(初期値:true) 【TAG】 リクエスト情報の クォーティション(‘) 存在チェックを実施するかどうか[true/false]を設定します (初期値:USE_SQL_INJECTION_CHECK[=true]) 【TAG】 リクエスト情報の HTMLTag開始/終了文字(<>) 存在チェックを実施するかどうか[true/false]を設定します (初期値:USE_XSS_CHECK[=true]) 【TAG】 vals属性でパラメーターを取得する際、複数件存在する場合に、値を連結するかどうかを指定します(初期値:false) 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)</div></div>															
●使用例	<div><!-- 業務ロジックの呼び出しを行います --> <og:bizLogic logics=“org.opengion.logic.gf9110.BizLogic.0001” keys=“SYSTEM_ID” vals=“{@MEM.SYSTEM_ID}” /></div>															
1.logics	【TAG】 実行する業務ロジック名を指定します。 実行する業務ロジック名を指定します。業務ロジック名は、クラス名を指定します。 クラス名については、クラス自身の名称のみを指定することができます。 (パッケージ名を含めた完全な形のクラス名を指定することもできます) また、カンマ区切りで、複数指定することもできます。 この場合、指定した順番に処理されます。															
2.command	【TAG】 コマンドをセットします(初期値:ENTRY)。 command=ENTRY以外ではDBTableModelの処理を行いません。 コマンドは、HTMLから[get/post]指定されますので、CMD_xxx で設定される フィールド定数値のいづれかを、指定できます。															
3.scope	【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)。 “request”, “page”, “session”, “application” が指定できます。 JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、 主にテーブルモデルを管理するスコープを指します。															
	<table><tr><th>スコープ</th><th>変数の有効範囲</th><th></th></tr><tr><td>page</td><td>JSPページ内</td><td>そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会</td></tr><tr><td>request</td><td>HTTPリクエスト</td><td>リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やボ</td></tr><tr><td>session</td><td>HTTPセッション</td><td>初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。 内部的には、同じキーワー</td></tr><tr><td>application</td><td>Webアプリケーション</td><td>ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。</td></tr></table>	スコープ	変数の有効範囲		page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会	request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やボ	session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。 内部的には、同じキーワー	application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。
スコープ	変数の有効範囲															
page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会														
request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やボ														
session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。 内部的には、同じキーワー														
application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。														
4.dbid	【TAG】 (通常は使いません)Queryオブジェクトを作成する時のDB接続IDを指定します。 Queryオブジェクトを作成する時のDB接続IDを指定します。 これは、システムリソースで、DEFAULT_DB_URL 等で指定している データベース接続先 情報に、XX_DB_URL を定義することで、dbid=“XX” とすると、この 接続先を使用して データベースにアクセスできます。															
5.tableId	【TAG】 (通常は使いません)結果のDBTableModelを、sessionに登録するときのキーを指定します。 検索結果より、DBTableModelオブジェクトを作成します。これを、以下のview 等のタグに 渡す場合に、通常は、session を利用します。その場合の登録キーです。 query タグを同時に実行して、結果を求める場合、同一メモリに配置される為、 この tableId 属性を利用して、メモリ空間を分けます。 初期値は、HybsSystem.TBL_MD_L_KEY です。															
6.selectedAll	【TAG】 データを全件選択済みとして処理するかどうか[true/false]を指定します(初期値:false)。 全てのデータを選択済みデータとして扱って処理します。 全件処理する場合に、(true/false)を指定します。 初期値は false です。															
7.modifyType	【TAG】 DB検索時の モディファイタイプを指定します[A:追加/C:更新/D:削除]。 DB検索時に、そのデータをA(追加)、C(更新)、D(削除)のモディファイタイプを つけた状態にします。 その状態で、そのまま、update する事が可能になります。															
8.keys	【TAG】 リンク先に渡すキーをCSV形式で複数指定します。 リンク先に渡すキーを指定します。 Keysだけを指定して、Valsを指定しない場合、Keysで指定された項目名に対応するパラメーターを取得し、 Valsとして使用します。															
9.vals	【TAG】 リンク先に渡す値をCSV形式で複数指定します。 リンク先に渡す値を指定します。 Keysだけを指定して、Valsを指定しない場合、Keysで指定された項目名に対応するパラメーターを取得し、 Valsとして使用します。 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスの為です。															
10.stopError	【TAG】 処理エラーの時に処理を中止するかどうか[true/false]を設定します(初期値:true)。 false(中止しない)に設定する場合、後続処理では、{@DB.ERR_CODE}の値により、 PLSQL/SQLの異常/正常終了によって分岐処理は可能となります。 初期値は、true(中止する)です。															
	【TAG】 リクエスト情報の クォーティション(‘) 存在チェックを実施するかどうか[true/false]を設定します															

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
11.quotCheck	(初期値:USE_SQL_INJECTION_CHECK[=true])。 SQLインジェクション対策の一つとして、暫定的ではありますが、SQLのパラメータに渡す文字列にクォーティション(')を許さない設定にすれば、ある程度は防止できます。数字タイプの引数には、 or 5=5 などのクォーティションを使用しないコードを埋めても、数字チェックで検出可能です。文字タイプの場合は、必ず (')をはずして、 ' or 'A' like 'A' のような形式になる為、(')チェックだけでも有効です。 (') が含まれていたエラーにする(true)／かノーチェックか(false)を指定します。 (初期値:システム定数のUSE_SQL_INJECTION_CHECK[=@og.value.org.opengion.hayabusa.common.SystemData#USE_SQL_INJECTION_CHECK])。 【TAG】リクエスト情報の HTMLTag開始/終了文字(××) 存在チェックを実施するかどうか[true/false]を設定します (初期値:USE_XSS_CHECK[=true])。
12.xssCheck	クロスサイトスクリプティング(XSS)対策の一環としてless/greater than signについてのチェックを行います。 (××) が含まれていたエラーにする(true)／かノーチェックか(false)を指定します。 (初期値:システム定数のUSE_XSS_CHECK[=@og.value.org.opengion.hayabusa.common.SystemData#USE_XSS_CHECK])。 【TAG】vals属性でパラメーターを取得する際、複数件存在する場合に、値を連結するかどうかを指定します(初期値:false)。
13.multi	この属性がtrueに指定された場合、パラメーターが複数存在する場合に、カンマで連結します。 初期値は、false(連結しない)です。
14.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTML EXT BODY要素	内容
形式サンプル	
5.5.4.2 og:button 拡張 JSP	<p>【HTML】 ボタンのタイプ[submit/reset/button]を指定します。 submit: 提出ボタンを作成します。 reset: リセットボタンを作成します。 button: 押しボタンを作成します。</p>
<p>●形式 : <og:button type="..." name="..." value="..." lbl="..." /> ●body : あり</p>	
<p>●Tag定義 : <og:button type name value caseKey caseVal lbl id lang dir title style disabled tabindex accesskey clazz language onClick onBlur onFocus ondblClick onMouseDown onMouseUp onMouseMove onMouseOut onMouseOver roles debug > ... Body ... </og:button></p>	<p>○【HTML】 ボタンのタイプ[submit/reset/button]を指定します。(必須) 【HTML】 ボタンの名称を指定します 【HTML】 ボタンの値を指定します 【TAG】 このタグ自体を利用するかどうかの条件キーを指定します(初期値:null) 【TAG】 このタグ自体を利用するかどうかの条件値を指定します(初期値:null) 【TAG】 ラベルリソースのラベルIDを指定します 【HTML】 要素に対して固有の名前(id)をつける場合に設定します 【HTML】 要素の内容と他の属性値の言語(lang, xml:lang)を指定します 【HTML】 文字表記の方向(dir)を指定します 【HTML】 要素に対する補足的情報(title)を設定します 【HTML】 この要素に対して適用させるスタイルシート(style)を設定します 【TAG】 その部品に対して、選択や変更が出来ないように(disabled)指定します(サーバーに送信されない) 【HTML】 タブの移動順(tabindex)を指定します(0 ~ 32767) 【HTML】 アクセスキー(alt+キーで直接指定)を割り当てます 【HTML】 要素に対して class 属性を設定します 【TAG】 タグ内部で使用する言語コード[ja/en/zh/...]を指定します 【HTML】 JavaScriptのイベント onClick を設定します(例:onClick="renew('query.jsp','QUERY');") 【HTML】 JavaScriptのイベント onBlur を設定します(例:onBlur="this.value=value.toUpperCase();") 【HTML】 JavaScriptのイベント onFocus を設定します 【HTML】 JavaScriptのイベント ondblClick を設定します 【HTML】 JavaScriptのイベント onMouseDown を設定します 【HTML】 JavaScriptのイベント onMouseUp を設定します 【HTML】 JavaScriptのイベント onMouseMove を設定します 【HTML】 JavaScriptのイベント onMouseOut を設定します 【HTML】 JavaScriptのイベント onMouseOver を設定します 【TAG】 ロールをセットします 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)</p>
<p>●使用例 <og:button type="submit" name="test" value="1" lbl="test" /> LabelResource プロパティのラベル使用したいときはlbl属性に、LabelResourceを使います。 <og:button type="submit" name="test" value="1" msg="MSG0103" /> MessageResource プロパティのラベル使用したいときはmsg属性を使います。</p>	
1.type	<p>【HTML】 ボタンのタイプ[submit/reset/button]を指定します。 submit: 提出ボタンを作成します。 reset: リセットボタンを作成します。 button: 押しボタンを作成します。</p>
2.name	<p>【HTML】 ボタンの名称を指定します。</p>
3.value	<p>【HTML】 ボタンの値を指定します。</p>
4.caseKey	<p>【TAG】 このタグ自体を利用するかどうかの条件キーを指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。</p>
5.caseVal	<p>【TAG】 このタグ自体を利用するかどうかの条件値を指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。</p>
6.lbl	<p>【TAG】 ラベルリソースのラベルIDを指定します。 ラベルを変更するときに、lbl属性を使います。 ラベルID は、所定の language に基づく ResourceManager の getLabelData(id) を呼び出し、その結果のLabelInterfaceを使用します。 getMsgLbl() で取り出せます。 ラベルIDとメッセージIDは同時には登録できません。 ラベルとメッセージは統一されました。</p>
7.id	<p>【HTML】 要素に対して固有の名前(id)をつける場合に設定します。 特別な使用方法として、id="FOCUS" とすることで、フィールド系要素に フォーカスを移動させます。これは、そのページ内で唯一の id 属性として使用ください。</p>
8.lang	<p>【HTML】 要素の内容と他の属性値の言語(lang, xml:lang)を指定します。 HTMLの言語属性に使われます。指定する値は、ISO 639で規定されている「言語コード」です。 [ja/en/zh/...]などのほかに、en-US:アメリカ英語、en-cockney:コックニー英語 など、 副言語を指定する方法も定められています。 ここでは、lang と xml:lang の両方に同じ値がセットされます。 タグの language 属性とは使用用途が異なります。lang セット</p>
9.dir	<p>【HTML】 文字表記の方向(dir)を指定します。 当該要素の書字方向を指定する属性です。 ltr で、左から右に、rtl で、右から左に並べます。</p>
10.title	<p>【HTML】 要素に対する補足的情報(title)を設定します。 title セットは、ボタンなどに適用すると、マウスオーバーによりこのメッセージが チップスのように表示されます。これを利用して、説明文を登録することが可能です。 ここに登録した文字列が、メッセージリソースに存在する場合は、そのメッセージを 存在しない場合は、そのままの値を設定します。</p>
11.style	<p>【HTML】 この要素に対して適用させるスタイルシート(style)を設定します。 タグにstyle属性を設定します。これは、キー:値:のセットを複数記述できます。 通常は、class属性や、id属性で登録しておき、&lg:style type="text/css"> で 外部から指定する方がソースは読みやすくなります。</p>
12.disabled	<p>【TAG】 その部品に対して、選択や変更が出来ないように(disabled)指定します(サーバーに送信されない)。 BUTTON/INPUT/OPTGROUP/OPTION/SELECT/TEXTAREA 系に対して、指定可能です。 disabled="disabled" , disabled="true" が指定された場合は、有効です。 false も指定値としては、有効です。(大文字小文字の区別も不要) それ以外の指定は、エラーとします。</p>
13.tabindex	<p>【HTML】 タブの移動順(tabindex)を指定します(0 ~ 32767)。 Tabキーを押したときに要素が選択される順番を指定します。 値には、選択させたい順番を数値で記述します。</p>
14.accesskey	<p>【HTML】 アクセスキー(alt+キーで直接指定)を割り当てます。 アクセスキーは、マウスの使えない環境でも、リンクにジャンプする、ボタンを押す、入力フォームに フォーカスを移すなどの操作を簡単に行うことができるように考慮されたものです。 Windows の「ファイル(F)」メニューについての、F と同じような働きをします。</p>
15.clazz	<p>【HTML】 要素に対して class 属性を設定します。 Javaの言語使用上、class で作成できないため、代用として clazz を使用しています。 html で作成される属性は、class で作成されます。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
16.language	【TAG】タグ内部で使用する言語コード[ja/en/zh/...]を指定します。 この言語コードに基づいて、表示のラベルをリソースから作成します。
17.onClick	【HTML】JavaScriptのイベント onClick を設定します(例:onClick="renew('query.jsp','QUERY');")。 onClick をセットします。 例えば、<og:column name="KBSAKU" onClick="renew('query.jsp','QUERY');"/> のように指定することで、プルダウンメニューの絞込み検索が可能になります。
18.onBlur	【HTML】JavaScriptのイベント onBlur を設定します(例:onBlur="this.value=value.toUpperCase();")。 onBlur は、フォーカスが離れたときに発生するイベントです。
19.onFocus	【HTML】JavaScriptのイベント onFocus を設定します。 onFocus は、フォーカスされたときに発生するイベントです。
20.ondblClick	【HTML】JavaScriptのイベント ondblClick を設定します。 ondblClick は、マウスでダブルクリックされたときに発生するイベントです。
21.onMouseDown	【HTML】JavaScriptのイベント onMouseDown を設定します。 onMouseDown は、マウスダウンされたときに発生するイベントです。
22.onMouseUp	【HTML】JavaScriptのイベント onMouseUp を設定します。 onMouseUp は、マウスアップされたときに発生するイベントです。
23.onMouseMove	【HTML】JavaScriptのイベント onMouseMove を設定します。 onMouseMove は、マウスが移動されたときに発生するイベントです。
24.onMouseOut	【HTML】JavaScriptのイベント onMouseOut を設定します。 onMouseOut は、マウスが離れたときに発生するイベントです。
25.onMouseOver	【HTML】JavaScriptのイベント onMouseOver を設定します。 onMouseOver は、マウスが重なったときに発生するイベントです。
26.roles	【TAG】ロールをセットします。 ここで指定したカラムロールを元に、ユーザー毎のアクセス許可がチェックされます。 アクセス許可されないと、表示されません。 このロールを指定しない場合は、カラムリソースのロールが使用されます。
27.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
<div>5.5.4.2</div> <div>og:calendarParam</div> <div>Tag</div> <div>●形式 : <og:calendarParam</div> <div>●body : なし</div> <div>●Tag定義 :</div> <div><og:calendarParam</div> <div>viewKeys</div> <div>ymKey</div> <div>dayKey</div> <div>valueKey</div> <div>valueBRFlag</div> <div>firstWeek</div> <div>headerLocale</div> <div>columnSize</div> <div>debug</div> <div>>/></div> <div>●使用例</div> <div>ViewFormTag の viewFormType が、ViewCalendarParam の場合に使用します。</div> <div>useParam 属性を設定しておかないと、使用されません。</div> <div><og:view</div> <div>viewFormType = "HTMLCalendar"</div> <div>command</div> <div>startNo</div> <div>pageSize</div> <div>useParam</div> <div>></div> <div><og:calendarParam</div> <div>viewKeys</div> <div>ymKey</div> <div>dayKey</div> <div>valueKey</div> <div>valueBRFlag</div> <div>firstWeek</div> <div>headerLocale</div> <div>columnSize</div> <div>>/></div> <div></og:view ></div>	<div>【TAG】 行のキーとなるカラム名を複数指定します(事業所やWCなど)。</div> <div>エンジン標準カレンダー (GE13) では、年月 (YYYYMM) と事業所コード (CDJGS)</div> <div>でユニークキーになります。また、能力カレンダーなどでは、事業所、ワークセンタ</div> <div>で、ユニークになります。</div> <div>それらのキー項目を編集できるように、カラムをカンマ区切りで指定します。</div> <div>ymKey="YYYYMM" dayKey="DY" firstWeek="0" /></div> <div>【TAG】 行のキーとなるカラム名を複数指定します(事業所やWCなど)</div> <div>【TAG】 行のキーとなる年月を表すカラム名を指定します</div> <div>【TAG】 日付け(休日フラグ)のキーとなるカラム名の英字部分を指定します (DY1 ~ DY31)</div> <div>【TAG】 データの値となるカラム名の英字部分を指定します(初期値は使用しません)</div> <div>【TAG】 データの値を使用する場合に、日付けとの関係で、BR を入れるかどうか指定します</div> <div>【TAG】 カレンダー表示の週の初めを指定します (0 は、日曜日から、1 は月曜日から)</div> <div>【TAG】 週表示ヘッダーを表示する時の、Locale(language のみ) を指定します</div> <div>【TAG】 カレンダーを横に並べる場合の数量を指定します(初期値: 3)</div> <div>【TAG】 デバッグ情報を出力するかどうか [true/false] を指定します(初期値:false)</div> <div>【TAG】 行のキーとなるカラム名を複数指定します(事業所やWCなど)。</div> <div>エンジン標準カレンダー (GE13) では、年月 (YYYYMM) と事業所コード (CDJGS)</div> <div>でユニークキーになります。また、能力カレンダーなどでは、事業所、ワークセンタ</div> <div>で、ユニークになります。</div> <div>それらのキー項目を編集できるように、カラムをカンマ区切りで指定します。</div> <div>【TAG】 行のキーとなる年月を表すカラム名を指定します。</div> <div>現カレンダーは、年月毎に、3 1 個の日付けフィールドを持ったデータを</div> <div>前提としています。工場や事業所などの区分は、取得時には確定させておく必要があります。</div> <div>初期値は、ViewCalendarParam に依存します。(例: YYYYMM)</div> <div>【TAG】 日付け(休日フラグ)のキーとなるカラム名の英字部分を指定します (DY1 ~ DY31)。</div> <div>カレンダーを表示する場合の、各日付け欄のカラム名の英字部分を指定します。</div> <div>現カレンダーは、年月毎に、3 1 個の日付けフィールドを持ったデータを</div> <div>前提としています。よって、このカラム名(英字部分)+日付け(1~3 1)を</div> <div>動的に作成して、日付けカラムとして取得します。</div> <div>例えば、"DY" という文字を指定した場合、DY1 ~ DY31 のカラム名を</div> <div>使用して、検索結果の DBTableModel より値を取得します。</div> <div>初期値は、ViewCalendarParam に依存します。(例: DY)</div> <div>【TAG】 データの値となるカラム名の英字部分を指定します(初期値は使用しません)。</div> <div>カレンダーを表示する場合の、各日付け欄の値のカラム名の英字部分を指定します。</div> <div>現カレンダーは、年月毎に、3 1 個の値フィールドを持ったデータを</div> <div>前提としています。よって、このカラム名(英字部分)+日付け(1~3 1)を</div> <div>動的に作成して、値カラムとして取得します。</div> <div>例えば、"VAL" という文字を指定した場合、VAL1 ~ VAL31 のカラム名を</div> <div>使用して、検索結果の DBTableModel より値を取得します。</div> <div>初期値は、ViewCalendarParam に依存します。(例: 使用していません)</div> <div>【TAG】 データの値を使用する場合に、日付けとの関係で、BR を入れるかどうか指定します。</div> <div>データを表示する場合、日付けの横につけるか、日付けの下に付けるかを指定します。</div> <div>ただし、valueKey で、値カラムを指定した場合のみ、有効になります。</div> <div>初期値は、ViewCalendarParam に依存します。(例: true)</div> <div>【TAG】 カレンダー表示の週の初めを指定します (0 は、日曜日から、1 は月曜日から)。</div> <div>カレンダーの週表示で、どの曜日から表示を始めるかを指定します。</div> <div>0 は、日曜日から、1 は月曜日から表示を開始します。</div> <div>初期値は、ViewCalendarParam に依存します。(例: 0 日曜日から)</div> <div>【TAG】 週表示ヘッダーを表示する時の、Locale(language のみ) を指定します。</div> <div>現実装では、"ja" と "en" のみ指定可能です。</div> <div>将来的には、SimpleDateFormat を使用し、多言語対応しますが、現段階では、</div> <div>(日、月、火、水、木、金、土) と、(SUN, MON, TUE, WED, THU, FRI, SAT) の</div> <div>2 種類を切り替えているだけです。</div> <div>どの曜日から表示を始めるかは、firstWeek 属性で設定します。</div> <div>初期値は、ViewCalendarParam に依存します。(例: "en")</div> <div>【TAG】 カレンダーを横に並べる場合の数量を指定します(初期値: 3)。</div> <div>カレンダーを複数並べる場合の横方向の件数を指定します。</div> <div>1 年を表す場合、3 * 4 や 2 * 6 のカレンダーが一般的です。</div> <div>ここでは、横方向(3 や 2)の件数を指定することで、レイアウトを</div> <div>指定します。</div> <div>【TAG】 デバッグ情報を出力するかどうか [true/false] を指定します(初期値:false)。</div> <div>デバッグ情報を [true:出力する/false:しない] を指定します。</div> <div>出力形式自体は、個々のタグによって異なります。</div>

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:case	【TAG】 switch-case のマッチ条件(case_match)を指定します。 switch_key.match(case_match) でマッチした場合は、BODY が処理されます。
JSP	
●形式 : <og:switch key="..." /> <og:case match="A" /> ... </og:case> <og:case match="B" /> ... </og:case> <og:case match="C" /> ... </og:case> <og:case isDefault="true" /> ... </og:case> </og:switch>	
●body : あり	
●Tag定義 :	
<og:case match isDefault isBreak isNull debug > ... Body ... </og:case>	【TAG】 switch-case のマッチ条件(case_match)を指定します 【TAG】 どのcase にもマッチしなかった場合に処理する case 文かどうかを指定します(初期値:false) 【TAG】 マッチした以降に継続処理を行わない(ブレイクする)かどうかを指定(初期値:true) 【TAG】 switchのkeyが、null(またはゼロ文字列)の場合、マッチするかどうか[true/false]を設定します(初期値:false) 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)
●使用例	
<og:switch key="[@PARAM]" /> <og:case match="A" /> 処理A </og:case> <og:case match="B" /> 処理B </og:case> <og:case match="C" /> 処理C </og:case> <og:case isDefault="true" /> 処理X </og:case> </og:switch>	
	<ul style="list-style-type: none">・switch の key に対して、case の match に指定された値が、マッチ(switch_key.match(case_match))した場合に、case の BODY 部分が処理されます。・マッチしなければ、BODY部は、スキップされます。・isDefault="true" の場合は、どれとも マッチしなかった場合に、実行されます。・Javaの switch-case 文は、最初に処理された case 以降を処理します。通常は、break を入れて後続処理を実行されないようにしています。この、switch-case タグは、caseタグの isBreak 属性で制御します。初期値が isBreak="true" に、なっているため、通常は、どれかの case が実行された段階で、switchの処理は、終了されます。isBreak="false" にすると、switchから抜けずに、継続して case との match を実行します。この場合、Java等と異なるのは、直後のcase文が実行されるのではなく、あくまで match 作業が継続されるということです。つまり、複数の case で処理を行いたい場合は、isBreak="false" にすると同時に、match 条件もそれぞれで、マッチするように設定する必要があります。
	<og:switch key="[@PARAM]" /> <og:case match="[1]" isBreak="false" /> 処理A </og:case> <og:case match="[12]" isBreak="false" /> 処理B </og:case> <og:case match="[123]" isBreak="false" /> 処理C </og:case> <og:case isNull="true" /> 処理X </og:case> <og:case isDefault="true" /> 処理Y </og:case> </og:switch>
	<ul style="list-style-type: none">・上記指定では、isBreak="false" が指定されているため、マッチした後も継続して判定処理が実施されます。・上記例で言うと、PARAM が "1" の場合、上記3つともにマッチします。・isNull="true" は、switch の key が null の場合に成立します。(null とは、ゼロ文字列も含む)
1.match	【TAG】 switch-case のマッチ条件(case_match)を指定します。 switch_key.match(case_match) でマッチした場合は、BODY が処理されます。
2.isDefault	【TAG】 どのcase にもマッチしなかった場合に処理する case 文かどうかを指定します(初期値:false)。 trueに設定すると、どのcase にもマッチしなかった場合に処理されます このタグそのものがなにも出力しません。(つまり条件から消えます。) BODY 部に記述することが可能です。その場合は、value 属性になにも設定できません。 初期値は、false: default case 文ではない です。
3.isBreak	【TAG】 マッチした以降に継続処理を行わない(ブレイクする)かどうかを指定(初期値:true)。 true に設定すると、マッチした段階で、ブレイクします。 Javaの switch-case 文は、最初に処理された case 以降を処理します。通常は、break を入れて後続処理を実行されないようにしています。
	この、switch-case タグは、caseタグの break 属性で制御します。初期値が break="true" に、なっているため、通常は、どれかの case が実行された段階で、switchの処理は、終了されます。
	break="false" にすると、switchから抜けずに、継続して case との match を実行します。
	この場合、Java等と異なるのは、直後のcase文が実行されるのではなく、あくまで match 作業が継続されるということです。つまり、複数の case で処理を行いたい場合は、break="false" にすると同時に、match 条件もそれぞれで、マッチするように設定する必要があります。
	初期値は、true: ブレイクする です。
4.isNull	【TAG】 switchのkeyが、null(またはゼロ文字列)の場合、マッチするかどうか[true/false]を設定します(初期値:false)。 trueに設定すると、switchのkeyが、null(またはゼロ文字列)の場合、マッチします。 初期値のfalse にすると、キーが null でない場合だけ、マッチ処理を実行します。 case の条件判定で使用されます。 初期値は、 false (null でない) です。
5.debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:chartCreate JSP	【TAG】 チャートのタイトルをセットします。 チャートのタイトルをセットします。
<p>●形式 : <og:chartCreate title="..." ... /> ●body : あり (chartDataset)</p> <p>●Tag定義 :</p> <pre><og:chartCreate title 【TAG】 チャートのタイトルをセットします width 【TAG】 チャートの横幅をセットします (初期値:200) height 【TAG】 チャートの縦幅をセットします (初期値:200) domainLabel 【TAG】 チャートのドメインラベルを指定します showLegend 【TAG】 チャートの凡例の表示可否 [true/false] をセットします (初期値:true[表示する]) rectangleEdge 【TAG】 チャートの表示箇所を、[TOP/BOTTOM/RIGHT/LEFT]で指定します (初期値:BOTTOM) plotOrientation 【TAG】 チャートのプロット (Plot) 表示方向を、[VERTICAL (or V), HORIZONTAL (or H)]で指定します (初期値:VERTICAL) chartBackColor 【TAG】 チャートの背景色を指定します plotBackColor 【TAG】 チャートの描画領域の色を指定します rotationLabel 【TAG】 チャートのカテゴリラベルの方向を指定します (3:60度) domainMarker 【TAG】 チャートの横軸の値 (ドメイン) に合致する位置にマーカーラインを設定します useDomainLabel 【TAG】 横軸ラベルのラベルを表示するかどうか [true/false] を指定します (初期値:true) useMarkerLabel 【TAG】 マーカーラインに、その設定値を表示するかどうか [true/false] を指定します (初期値:true) useCache 【TAG】 JFreeChart オブジェクトをキャッシュするかどうか [true/false] を指定します (初期値:false) masterKey 【TAG】 キャッシュ用マスターキーを指定します seriesPickup 【TAG】 複数シリーズのピックアップを行う場合のシリーズ番号を指定します imageMapUrl 【TAG】 クリックابل・マップ用URLを指定します imageMapTarget 【TAG】 クリックابل・マップ用TARGETを指定します categoryMargin 【TAG】 カテゴリマージン (0.0~1.0) を指定します lowerMargin 【TAG】 下方マージン (0.0~1.0) を指定します upperMargin 【TAG】 上方マージン (0.0~1.0) を指定します useDomainLine 【TAG】 横軸のグリッド表示有無 (垂直線) を指定します (初期値:false) domainLineColor 【TAG】 横軸のグリッド線の色を指定します categorySkip 【TAG】 横軸ラベルをスキップする間隔を指定します categoryCutNo 【TAG】 横軸ラベルの文字位置指定のキーブレイクを指定します categoryAnchor 【TAG】 横軸のグリッド (垂直線) の書き出し位置 (START, MIDDLE, END) を指定します useRangeLine 【TAG】 縦軸のグリッド表示有無 (水平線) を指定します (初期値:true) useToolTip 【TAG】 ツールチップスの使用可否 [true:利用する/false:利用しない] を指定します (初期値:false) rangeLineColor 【TAG】 縦軸のグリッド線の色を指定します rangeSkip 【TAG】 縦軸のグリッド線 (水平線) をスキップする間隔を指定します debug 【TAG】 デバッグ情報を出力するかどうか [true/false] を指定します (初期値:false) > ... Body ... </og:chartCreate></pre> <p>●使用例</p> <pre><og:chartCreate title = "JFreeChart Test" チャートタイトル domainLabel = "ドメインラベル" 横軸ラベル width = "200" チャート表示幅 height = "200" チャート表示高さ showLegend = "[true/false]" 凡例の表示可否 [true/false] rectangleEdge = "[TOP BOTTOM RIGHT LEFT]" 凡例の表示箇所 plotOrientation = "[VERTICAL HORIZONTAL]" チャートの軸表示方向 chartBackColor = "WHITE" 背景色 plotBackColor = "LIGHT_GRAY" 描画領域色 rotationLabel = "3" 横軸ラベルの傾き domainMarker = "KING" 横軸のマーカーライン (縦棒) useMarkerLabel = "[true/false]" マーカーラインの文字有無 useDomainLabel = "[true/false]" 横軸ラベルの表示有無 categorySkip = "3" 横軸ラベルをスキップする間隔 categoryCutNo = "6" 横軸ラベルの文字位置指定のキーブレイク categoryAnchor = "[START MIDDLE END]" 横軸のグリッド (垂直線) の書き出し位置 useDomainLine = "[false true]" 横軸のグリッド表示有無 (垂直線) domainLineColor = "LIGHT_GRAY" 横軸のグリッド線の色 useRangeLine = "[true/false]" 縦軸のグリッド表示有無 (水平線) rangeLineColor = "LIGHT_GRAY" 縦軸のグリッド線の色 rangeSkip = "-5" 縦軸のグリッド (水平線) をスキップする間隔 useCache = "[false true]" キャッシュの有無 masterKey = "[@XXXX]" キャッシュ用マスターキー seriesPickup = "2" ビックアップするシリーズNo imageMapUrl = "link.jsp.BLOCK" クリックابل・マップ用URL imageMapTarget = "CONTENTS" クリックابل・マップ用TARGET categoryMargin = "[0.0~1.0]" カテゴリマージン (0.0~1.0) lowerMargin = "[0.0~1.0]" 下方マージン (0.0~1.0) upperMargin = "[0.0~1.0]" 上方マージン (0.0~1.0) > <og:chartDataset chartType = "[@chartType]" valueLabel = "[@valueLabel]" lowerBound = "[@lowerBound]" upperBound = "[@upperBound]" markValues = "[@markValues]" markColors = "[@markColors]" useGradient = "[@useGradient]" shapesVisible = "[@shapesVisible]" useDottedLine = "[@useDottedLine]" seriesColors = "[@seriesColors]" valueLabelsVisible = "[true false]" valueMarksVisible = "[true false]" > {@SQL} </og:chartDataset> </og:chartCreate></pre> <p>複数のグラフを重ね合わせる場合は、chartDataset タグを chartCreate のBODY部に複数記述します。</p> <pre><og:chartCreate title = "[@title]" domainLabel = "[@domainLabel]" width = "[@width]" height = "[@height]" ></pre>	

バージョン タグ名 HTMLTEXT BODY要素	内容
	<p>形式サンプル</p> <pre> <og:chartDataset chartType = "{@chartType1}" valueLabel = "{@valueLabel1}" lowerBound = "{@lowerBound1}" upperBound = "{@upperBound1}" markValues = "{@markValues1}" markColors = "{@markColors1}" > {@SQL1} </og:chartDataset> <og:chartDataset chartType = "{@chartType2}" valueLabel = "{@valueLabel2}" lowerBound = "{@lowerBound2}" upperBound = "{@upperBound2}" markValues = "{@markValues2}" markColors = "{@markColors2}" > {@SQL2} </og:chartDataset> </og:chartCreate> </pre> <p>rectangleEdge属性 は、凡例の表示箇所を示す、RectangleEdge クラスの値を設定します。 [rectangleEdge属性] TOP 上側 BOTTOM 下側 RIGHT 右側 LEFT 左側</p> <p>plotOrientation属性 は、チャートの軸表示方向を設定します。 [renderer属性] VERTICAL 縦方向 HORIZONTAL 横方向</p>
1.title	<p>【TAG】チャートのタイトルをセットします。 チャートのタイトルをセットします。</p>
2.width	<p>【TAG】チャートの横幅をセットします(初期値:200)。 タイトルや凡例も含んだ大きさです。データ領域は自動計算されます。</p>
3.height	<p>【TAG】チャートの縦幅をセットします(初期値:200)。 タイトルや凡例も含んだ大きさです。データ領域は自動計算されます。</p>
4.domainLabel	<p>【TAG】チャートのドメインラベルを指定します。 チャートのドメインラベルを指定します。</p>
5.showLegend	<p>【TAG】チャートの凡例の表示可否[true/false]をセットします(初期値:true[表示する])。 初期値は、表示する(true) です。</p>
6.rectangleEdge	<p>【TAG】チャートの表示箇所を、[TOP/BOTTOM/RIGHT/LEFT]で指定します(初期値:BOTTOM)。 表示箇所は、org.jfree.ui.RectangleEdge クラスの設定値を使用します。 指定できるのは、TOP、BOTTOM、RIGHT、LEFT で、各文字の頭一文字で判定してます。つまり、T,B,R,L で、判定を行い、それ以外はエラーになります。 初期値は、BOTTOM です。</p>
7.plotOrientation	<p>【TAG】チャートのプロット(Plot)表示方向を、[VERTICAL(or V),HORIZONTAL(or H)]で指定します(初期値:VERTICAL)。 軸表示方向は、org.jfree.chart.plot.PlotOrientation クラスの設定値を使用します。 指定できるのは、VERTICAL、HORIZONTALで、各文字の頭一文字で判定してます。つまり、V,H で、判定を行い、それ以外はエラーになります。 初期値は、VERTICAL です。</p>
8.chartBackColor	<p>【TAG】チャートの背景色を指定します。 指定文字列は、java.awt.Color クラスのstatic フィールド名で指定します。 BLACK, BLUE, CYAN, DARK_GRAY, GRAY, GREEN, LIGHT_GRAY, MAGENTA, ORANGE, PINK, RED, WHITE, YELLOW, (PURPLE) が指定できます。 また、#XXXXXX形式の16bitRGB表記 でも指定可能です。</p>
9.plotBackColor	<p>【TAG】チャートの描画領域の色を指定します。 指定文字列は、java.awt.Color クラスのstatic フィールド名で指定します。 BLACK, BLUE, CYAN, DARK_GRAY, GRAY, GREEN, LIGHT_GRAY, MAGENTA, ORANGE, PINK, RED, WHITE, YELLOW, (PURPLE) が指定できます。 また、#XXXXXX形式の16bitRGB表記 でも指定可能です。</p>
10.rotationLabel	<p>【TAG】チャートのカテゴリーラベルの方向を指定します(3:60度)。 方向は、上方向に対して、(PI / 指示数) で求まる値に設定します。 この指示数に相当する値を設定します。 1:180度、2:90度、3:60度、4:45度、6:30度 …… マイナスは、した方向に回転させます。 0 を指定した場合は、何も設定しません。 初期値は、3:60度です。</p>
11.domainMarker	<p>【TAG】チャートの横軸の値(ドメイン)に合致する位置にマーカーラインを設定します。 この属性には、マーカーラインを設定する値を記述します。</p>
12.useDomainLabel	<p>【TAG】横軸ラベルのラベルを表示するかどうか[true/false]を指定します(初期値:true)。 ドメイン(横軸)が、多数存在する場合、ドメインラベルが見えにくくなります。 そのようなケースで、横軸のラベルそのものを表示しない場合に、false を設定します。 初期値は、表示する(true)です。</p>
13.useMarkerLabel	<p>【TAG】マーカーラインに、その設定値を表示するかどうか[true/false]を指定します(初期値:true)。 ドメイン(横軸)が、多数存在する場合、ドメインラベルが見えない場合があります。 そのようなケースで、見たい値にマーカーラインを設定し、その横に、ドメインラベルを表示する事で、ピックアップしている軸の値を容易に知ることが可能です。 初期値は、表示する(true)です。</p>
14.useCache	<p>【TAG】JFreeChart オブジェクトをキャッシュするかどうか[true/false]を指定します(初期値:false)。 useCache="true" を設定すると、session に、HybsSystem.JF_CHART_CACHE_KEY キーで、キャッシュされます。 そのときに、domainMarker 属性と、seriesPikup 属性だけ、再設定可能になっています。</p>
15.masterKey	<p>初期値は、キャッシュしない(false)です。 【TAG】キャッシュ用マスタキーを指定します。 useCache="true" に設定した場合、キャッシュを使用できるか確認します。 この場合、seriesPikup 違いの場合は、JFreeChart オブジェクトそのものをキャッシュしておけば、データベースアクセスなしで、グラフを再描画させることが可能です。 この、同一 JFreeChart を指定するためのキーを、ここで設定します。 このキーの個数だけ、JFreeChart がキャッシュされます。 なお、キャッシュ保持時間は、5分固定です。</p>
16.seriesPikup	<p>【TAG】複数のシリーズのピックアップを行う場合のシリーズ番号を指定します。 複数シリーズ(検索時に複数項目を同時に検索する場合)では、チャート上に複数のグラフが表示されますが、その中の一つをピックアップする場合に、シリーズ番号を指定します。 シリーズ番号は、0 から始まる数字です。 ここでは、ピックアップされたシリーズは、赤色で表示されます。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
	それ以外は、グレー色での表示になります。 seriesPikup を使用すると、chartDataset タグの useValueVisible 属性が影響を受けます。この属性は、データの値(itemText)を表示しますが、seriesPikup が指定された場合は、そのシリーズのみにラベル表示します。 【TAG】 クリックابل・マップ用URLを指定します。 画像に、クリックابل・マップを作成する場合の、URL を指定します。 これは、画像上にエリア指定でリンク引数を作成することが可能です。 URL 自身は、? 付きで固定値の引数を連結することが可能です。 クリックしたエリアのカテゴリやインデックスの値(引数)は、自動的に設定されます。(指定しない場合はチャートによって異なります) ・Pie : category、pieIndex ・XY : series、item ・Category : series、category
17.imageMapUrl	この引数の URL の名称を変更したい場合は、URL に続けて、カンマ(,) で、名称を記述してください。 例: link.jsp_BLOCK
18.imageMapTarget	【TAG】 クリックابل・マップ用TARGETを指定します。 画像に、クリックابل・マップを作成する場合の、TARGET を指定します。 これは、画像上にエリア指定でリンクを作成する場合のフレーム指定です。
19.categoryMargin	【TAG】 カテゴリマージン(0.0~1.0)を指定します。 カテゴリ(グラフの横軸に相当)の表示間隔(マージン)の比率を指定します。 この比率は、% ではなく、数字(double)での設定になります。 何も指定しない場合は、デフォルトで自動調整されます。
20.lowerMargin	【TAG】 下方マージン(0.0~1.0)を指定します。 カテゴリ(グラフの横軸に相当)の下方側(左側)のマージンの比率を指定します。 この比率は、% ではなく、数字(double)での設定になります。 何も指定しない場合は、デフォルトで自動調整されます。
21.upperMargin	【TAG】 上方マージン(0.0~1.0)を指定します。 カテゴリ(グラフの横軸に相当)の上方側(右側)のマージンの比率を指定します。 この比率は、% ではなく、数字(double)での設定になります。 何も指定しない場合は、デフォルトで自動調整されます。
22.useDomainLine	【TAG】 横軸のグリッド表示有無(垂直線)を指定します(初期値:false)。 ドメイン(横軸)に対する、グリッドラインを表示するかどうか指定します。
23.domainLineColor	何も指定しない場合は、表示しません。(false) 【TAG】 横軸のグリッド線の色を指定します。 ドメイン(横軸)に対する、グリッドラインの表示色を指定します。 何も指定しない場合は、デフォルトで自動設定されます。
24.categorySkip	【TAG】 横軸ラベルをスキップする間隔を指定します。 横軸ラベル(カテゴリラベル表示)する際に、スキップする間隔を指定します。 “1” (初期値)では、1 つづつ表示(つまり、すべて表示する)します。 “2” とすると、1 つおきに、“3” とすると、2 つおきに表示します。 初期値は、“1” (すべて表示)です。 なお、先頭から表示を開始します。
25.categoryCutNo	注意: これとは別に、ラベル先頭に “ ” を付けた場合は、ラベルを表示しません。 また、categoryCutNo が指定された場合は、categorySkip は使用されません。 【TAG】 横軸ラベルの文字位置指定のキープレイクを指定します。 横軸ラベル(カテゴリラベル表示)する際に、ラベルの先頭から、この指定文字数だけカットして、表示します。 その際、前回作成したカットラベルと、同一ラベルの場合は、表示しません。 例えば、データは、年月日で、年と月のみ(先頭6文字)を指定すると、日のデータは、ラベルが表示されません。 指定される数字は、1 以上の整数としてください。 初期値は、すべて表示です。
26.categoryAnchor	注意: これとは別に、ラベル先頭に “ ” を付けた場合は、ラベルを表示しません。 【TAG】 横軸のグリッド(垂直線)の書き出し位置(START, MIDDLE, END)を指定します。 横軸のグリッド(垂直線)を、グラフのどの位置に記述するかを指定します。 具体的な値は、CategoryAnchor オブジェクトの値になります。 ここでは、文字列で(START, MIDDLE, END)を指定します。(先頭一文字で判定) 何も指定しない場合は、デフォルト(MIDDLE)です。
27.useRangeLine	【TAG】 縦軸のグリッド表示有無(水平線)を指定します(初期値:true)。 レンジ(縦軸)に対する、グリッドラインを表示するかどうか指定します。
28.useToolTip	何も指定しない場合は、表示しません。(false) 【TAG】 ツールチップの使用可否[true:利用する/false:利用しない]を指定します(初期値:false)。 ラベルを利用する際に、ラベルと図面、隣のパネル同士が重なることがあります。 この場合、ツールチップの利用をお勧めします。 初期値は false です。
29.rangeLineColor	【TAG】 縦軸のグリッド線の色を指定します。 レンジ(縦軸)に対する、グリッドラインの表示色を指定します。 何も指定しない場合は、デフォルトで自動設定されます。
30.rangeSkip	【TAG】 縦軸のグリッド線(水平線)をスキップする間隔を指定します。 縦軸のグリッド線(水平線)を表示する際に、スキップする間隔を指定します。 通常は、ラベルと同じだけのグリッド線が掛かれますが、ラベルよりも少ない数のグリッド線(例えば、2 つおき)を出す場合に、値を設定します。 “1” (初期値)では、1 つづつ表示(つまり、すべて表示する)します。 “2” とすると、1 つおきに、“3” とすると、2 つおきに表示します。 初期値は、“1” (すべて表示)です。 なお、先頭から表示を開始します。
31.debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

5.5.4.2
og:chartDataset

JSP

【TAG】 ChartDataset のデータタイプを指定します。

チャートタイプ は、外部からチャートを指定するのに便利のように、キー化されています。このキーに基づいて、ChartFactory クラスのチャートタイプ変換表に基づいて、レンダラーや、データセットを作成します。基本的には、チャートタイプ は、この、レンダラー名称から決定しています。

Area	Bar	Bar3D	BoxAndWhisker	CategoryStep
Gantt	GroupedStackedBar	IntervalBar	LayeredBar	Level
LineAndShape	Line3D	MinMax	StackedArea	StackedBar
StackedBar3D	WaterfallBar	CyclicXYItem	HighLow	StackedXYArea
StackedXYArea2	StandardXYItem	XYBubble	XYDifference	XYDot
XYError	XYLine3D	XYLineAndShape	XYStepArea	XYStep
Meter	MultiplePie	Pie	Pie3D	Ring
SpiderWeb	Thermometer			

●形式：<og:chartDataset renderer="..." ... />
●body：あり

●Tag定義：
<og:chartDataset
chartType
dbid
useTableData
tableId
scope
baseLegend
valueLabel
markValues
markColors
markOverColors
dynamicOCNo
useMarkAnchor
lowerBound
upperBound
tickSize
useGradient
shapesVisible
shapeColors
shapeScale
useDottedLine
seriesColors
valueLabelsVisible
valueMarksVisible
itemLabelVisible
useItemLabelSep
valueInset
barMaxWidth
barItemMargin
barWidth
visibleLimit
domainMargin
timeFormatType
useVerticalLabels
debug
>
... Body ...
</og:chartDataset>

○【TAG】 ChartDataset のデータタイプを指定します。(必須)
【TAG】 (通常は使いません)Datasetオブジェクトを作成する時のDB接続IDを指定します
【TAG】 グラフ化するデータを DBTableModel から作成するかどうか(初期値:false)
【TAG】 (通常使いません)sessionから所得する DBTableModelオブジェクトの ID (初期値:HybsSystem#TBL_MD_L_KEY[=h_tblmdl])
【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)
【TAG】 シリーズ単位の凡例 の表示可否を設定します
【TAG】 縦軸の表示名称を指定します
【TAG】 マーカーラインの設定値をカンマ区切り形式で複数指定します
【TAG】 マーカーラインの色をカンマ区切り形式で複数指定します
【TAG】 マーカーラインの超過時のShape色をカンマ区切り形式で複数指定します
【TAG】 動的なマーカーラインの基準シリーズ番号を設定します(初期値:null)
【TAG】 マーカーライン描画時に、その設定値を表示します
【TAG】 チャートの縦軸の最小値をセットします(初期値:自動計算)
【TAG】 チャートの縦軸の最大値をセットします(初期値:自動計算)
【TAG】 チャートの縦軸の目盛の幅をセットします(初期値:自動計算)
【TAG】 バーチャートのグラデーション処理を行うかどうか[true/false]をセットします(初期値:false)
【TAG】 ラインチャートのポイントを四角表示するかどうかを指定します(初期値:true)
【TAG】 データ毎にShapeを切り替える時の色の繰返しパターンを文字列配列で指定します
【TAG】 shapeの大きさを倍率指定で変更します(初期値:null)
【TAG】 ラインチャートの線をドットラインにするかどうか[true/false]を指定します(初期値:false 線分)
【TAG】 複数チャート描画時のチャート色の繰返しパターンをCVS形式で複数指定します
【TAG】 Value(縦軸)のラベルを表示するかどうかを指定します(初期値:true[表示する])
【TAG】 Value(縦軸)のマーカーを表示するかどうかを指定します(初期値:true[表示する])
【TAG】 データの値(itemText)を表示するかどうか[true/false/last]を指定します(初期値:false[表示しない])
【TAG】 データの値(itemText)の表示に、桁区切り文字を使用するかどうか[true/false]を指定します(初期値:true[使用する])
【TAG】 縦軸の表示領域INSET値(double)を指定します
【TAG】 BOXチャートのバー幅の最大値(0.0~1.0)を指定します
【TAG】 BOXチャートのバーアイテムのマージン(0.0~1.0)を指定します
【TAG】 BOXチャートのバー幅(double)を指定します
【TAG】 表示下限値(これ以下のデータは未表示)の値(double)を指定します
【TAG】 グラフの書き出し位置の調整比率を指定します
【TAG】 時刻を表す場合の表現の仕方[E1/E2/E3/H1/H2/H3]を指定します(初期値:null)
【TAG】 ラベルの表示向きを縦にするかどうか[false/true]を指定します(初期値:false)
【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)

●使用例
<og:chartCreate
command = "{@command}"
title = "{@title}"
domainLabel = "{@domainLabel}"
width = "{@width}"
height = "{@height}"
rectangleEdge = "{@rectangleEdge}"
plotOrientation = "{@plotOrientation}"
chartBackColor = "{@chartBackColor}"
plotBackColor = "{@plotBackColor}"
domainMarker = "{@domainMarker}"
useMarkerLabel = "{@useMarkerLabel}" >
<og:chartDataset
chartType = "[Bar|LineAndShape|...]"
baseLegend = "[true|false]"
valueLabel = "縦軸"
lowerBound = "0"
upperBound = "5000"
tickSize = "1000"
markValues = "2500, 2800"
markColors = "RED, GREEN"
markOverColors = "BLUE, YELLOW, RED"
dynamicOCNo = "2"
useMarkAnchor = "[true|false]"
useGradient = "[false|true]"
shapesVisible = "[true|false]"
shapeColors = "RED, GREEN, BLUE"
shapeScale = "0.8"
useDottedLine = "[false|true]"
seriesColors = "RED, GREEN, BLUE"
valueLabelsVisible = "[true|false]"
valueMarksVisible = "[true|false]"
itemLabelVisible = "[false|true|last]"
useItemLabelSep = "[true|false]"
valueInset = "15"
barMaxWidth = "[0.0~1.0]"
barItemMargin = "[0.0~1.0]"
barWidth = "10"
visibleLimit = "0"
domainMargin = "1.0"
useTableData = "[false|true]"
timeFormatType = "[1|2]"
useVerticalLabels = "[false|true]"

チャートの種類
シリーズ単位の凡例 の表示可否
縦軸のラベル
下限値
上限値
縦軸の目盛の幅
縦軸設定値(横棒)
縦軸設定ライン色
マーカーラインの超過時のShape色
動的なマーカーラインの基準シリーズ番号
縦軸設定値(横棒)を表示するかどうか
バーチャートのグラデーション処理
ラインの四角表示
データ毎のShapeの色
shapeの大きさを倍率(double)
点線使用有無
シリーズの色サイクル指定
縦軸のラベルの表示有無
縦軸のマーカーの表示有無
データの値の表示有無
データの桁区切り文字の使用有無
縦軸の表示領域INSET値
バー幅の最大値(0.0~1.0)
バーアイテムのマージン(0.0~1.0)
バー幅の実幅(double)
表示下限値(これ以下のデータは未表示))
グラフの書き出し位置の調整比率
SQLではなく、DBTableModelからグラフを作成するかどうか
時間軸の設定(00:00表示)
時間軸の設定(ラベルの縦書き)

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

```
        tableId      = "AAA"                useTableData="true"の場合のDBTableModelの取得先ID
        scope        = "[session|request]"   useTableData="true"の場合のDBTableModelの取得先scope
    >
        {@SQL}
    </og:chartDataset>
</og:chartCreate>
```

複数のグラフを重ね合わせる場合は、chartDataset タグを chartCreate のBODY部に複数記述します。

```
<og:chartCreate
    title      = "{@title}"
    domainLabel = "{@domainLabel}"
    width      = "{@width}"
    height     = "{@height}" >
    rectangleEdge = "{@rectangleEdge}" >
    plotOrientation = "{@plotOrientation}" >
    <og:chartDataset
        chartType = "{@chartType1}"
        valueLabel = "{@valueLabel1}"
        lowerBound = "{@lowerBound1}"
        upperBound = "{@upperBound1}"
        markValues = "{@markValues1}"
        markColors = "{@markColors1}"
    >
        {@SQL1}
    </og:chartDataset>
    <og:chartDataset
        chartType = "{@chartType2}"
        valueLabel = "{@valueLabel2}"
        lowerBound = "{@lowerBound2}"
        upperBound = "{@upperBound2}"
        markValues = "{@markValues2}"
        markColors = "{@markColors2}"
    >
        {@SQL2}
    </og:chartDataset>
</og:chartCreate>
```

chartType属性 は、チャートの種類を指定する、固有IDです。

[chartType属性]

Area	, Bar	, Bar3D	, BoxAndWhisker	, CategoryStep	,
Gantt	, GroupedStackedBar	, IntervalBar	, LayeredBar	, Level	,
LineAndShape	, Line3D	, MinMax	, StackedArea	, StackedBar	,
StackedBar3D	, WaterfallBar	, CyclicXYItem	, HighLow	, StackedXYArea	,
StackedXYArea2	, StandardXYItem	, XYBubble	, XYDifference	, XYDot	,
XYError	, XYLine3D	, XYLineAndShape	, XYStepArea	, XYStep	,
Meter	, MultiplePie	, Pie	, Pie3D	, Ring	,
SpiderWeb	, Thermometer				

markColors属性 は、マーカークラインの色を指定します
java.awt.Color クラスのstatic フィールド名の文字列で指定します。

[renderer属性]

BLACK	, BLUE	, CYAN	, DARK_GRAY	, GRAY	, GREEN	, LIGHT_GRAY	,
MAGENTA	, ORANGE	, PINK	, RED	, WHITE	, YELLOW		

#XXXXXX形式の16bitRGB表記

1.chartType

【TAG】 ChartDataset のデータタイプを指定します。
チャートタイプ は、外部からチャートを指定するのに便利ように、キー化されています。このキーに基づいて、ChartFactory クラスのチャートタイプ変換表に基づいて、レンダレーや、データセットを作成します。基本的には、チャートタイプ は、この、レンダレー名称から決定しています。

Area	, Bar	, Bar3D	, BoxAndWhisker	, CategoryStep	,
Gantt	, GroupedStackedBar	, IntervalBar	, LayeredBar	, Level	,
LineAndShape	, Line3D	, MinMax	, StackedArea	, StackedBar	,
StackedBar3D	, WaterfallBar	, CyclicXYItem	, HighLow	, StackedXYArea	,
StackedXYArea2	, StandardXYItem	, XYBubble	, XYDifference	, XYDot	,
XYError	, XYLine3D	, XYLineAndShape	, XYStepArea	, XYStep	,
Meter	, MultiplePie	, Pie	, Pie3D	, Ring	,
SpiderWeb	, Thermometer				

2.dbid

【TAG】（通常は使いません）Datasetオブジェクトを作成する時のDB接続IDを指定します。
Datasetオブジェクトを作成する時のDB接続IDを指定します。
これは、システムリソースで、DEFAULT_DB_URL 等で指定している データベース接続先情報に、XX_DB_URL を定義することで、 dbid="XX" とすると、この 接続先を使用してデータベースにアクセスできます。

3.useTableData

【TAG】 グラフ化するデータを DBTableModel から作成するかどうか(初期値:false)。
useTableData="false" (初期値) の場合は、BODY部に書かれた SQL 文を実行してデータを取得します。
useTableData="true" にすると、その前の query タグ等で作成した DBTableModel を利用してグラフ化します。その場合は、BODY部の SQL 文は、無視されます。
利用する DBTableModel は、tableId と scope 属性から取得します。

4.tableId

【TAG】（通常は使いません）sessionから所得する DBTableModelオブジェクトの ID (初期値:HybsSystem#TBL_MD_L_KEY[=h_tblmdl])。
表示処理後に、(内部ポインタを書き換えた)DBTableModelオブジェクトを同じキーで、sessionに登録します。
(初期値:システム定数のHybsSystem#TBL_MD_L_KEY[=@og.value org.opengion.hayabusa.common.HybsSystem#TBL_MD_L_KEY])。

5.scope

【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)。
"request", "page", "session", "applicaton" が指定できます。
JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、主にテーブルモデルを管理するスコープを指します。

スコープ	変数の有効範囲	
page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会
request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ
session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。 内部的には、同じキーワー
application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。

6.baseLegend

【TAG】 シリーズ単位の凡例 の表示可否を設定します。
シリーズ単位の凡例 の表示可否を設定します。

7.valueLabel

【TAG】 縦軸の表示名称を指定します。
縦軸の表示名称を指定します。

バージョン タグ名 HTMLTEXT BODY要素	内容
8.markValues	<p>【TAG】マーカーラインの設定値をカンマ区切り形式で複数指定します。 指定の値にマーカーラインを引きます。 色は、setMarkColors(String) で指定します。 markColors と、個数をあわせて設定する必要があります。 どちらかが、指定されていない場合は、マーカー表示されません。 dynamicOCNo を使用する場合は、そのValues の並びの箇所に、 “G” という文字列を設定して下さい。 例：2000_G 基準 1は、2000の直線、基準 2は、グラフ という意味。 【TAG】マーカーラインの色をカンマ区切り形式で複数指定します。 マーカーラインとは、縦軸設定のラインの事で、縦軸の設定値(markValues)に 横方向にラインを付与します。このラインの色を、カンマ区切り形式で 指定します。 markValues と、あわせて設定する必要があります。 個数が異なる場合は、markValues が優先され、無指定の場合は、Color.REDで 表示されます。</p> <p>指定文字列は、java.awt.Color クラスのstatic フィールド名で指定します。 BLACK , BLUE , CYAN , DARK_GRAY , GRAY , GREEN , LIGHT_GRAY , MAGENTA , ORANGE , PINK , RED , WHITE , YELLOW , (PURPLE) が指定できます。 また、#XXXXXX形式の16bitRGB表記 でも指定可能です。 BLACK , BLUE , CYAN , DARK_GRAY , GRAY , GREEN , LIGHT_GRAY , MAGENTA , ORANGE , PINK , RED , WHITE , YELLOW , (PURPLE) #XXXXXX形式の16bitRGB表記 でも指定可能です。</p>
9.markColors	<p>【TAG】マーカーラインの超過時のShape色をカンマ区切り形式で複数指定します。 HybsLine でのみ使用可能です。 マーカーラインを使用する場合に、そのラインを超えない色、超える色を指定します。 この色の指定は、マーカーラインの色やマーカーラインの設定値の個数+1に なります。つまり、色 1：設定値 1：色 2：設定値 2：色 3 となります。 色 1は、設定値 1 より小さい場合に使用され、色 2は、設定値 1 より大きく、 設定値 2 より小さい場合、色 3は、設定値 2 より大きい場合になります。</p> <p>指定文字列は、java.awt.Color クラスのstatic フィールド名で指定します。 BLACK , BLUE , CYAN , DARK_GRAY , GRAY , GREEN , LIGHT_GRAY , MAGENTA , ORANGE , PINK , RED , WHITE , YELLOW , (PURPLE) が指定できます。 また、#XXXXXX形式の16bitRGB表記 でも指定可能です。 markValues と、あわせて設定する必要があります。 どちらかが、指定されていない場合は、マーカー表示されません。 BLACK , BLUE , CYAN , DARK_GRAY , GRAY , GREEN , LIGHT_GRAY , MAGENTA , ORANGE , PINK , RED , WHITE , YELLOW , (PURPLE) #XXXXXX形式の16bitRGB表記 でも指定可能です。</p>
10.markOverColors	<p>【TAG】動的なマーカーラインの基準シリーズ番号を設定します(初期値:null)。 動的なマーカーラインを使用する場合は、基準値となるシリーズ番号を指定します。</p> <p>マーカーラインの最下位閾値に相当します。これは、グラフ化されますが、 Shape は自動的に削除されます。 この設定を使用する場合、最初のデータには、必ずShapeが付きます。それ以外の データに、Shape を付けるかどうかは、shapesVisible 属性で指定します。 この線の色は、markColors で指定した、最下位の色になります。また、 markValues で指定した、最下位の値は、使用されません。ただし、色指定の 関係上、設定しておく必要があります。 また、isValueVisible == true で、設定値の値表示を行う場合も、最下位の 値は表示しないようにします。 初期値は、使用しない(null)です。</p>
11.dynamicOCNo	<p>【TAG】マーカーライン描画時に、その設定値を表示します。 マーカーラインの縦軸の設定値(markValues)に、設定値をラベル表示します。 位置は、TextAnchor.BASELINE_LEFT 固定です。 初期値は、表示する(true)です。</p>
12.useMarkAnchor	<p>【TAG】チャートの縦軸の最小値をセットします(初期値:自動計算)。 何も指定しない場合は、データの最小値から、自動的に計算します。</p>
13.lowerBound	<p>【TAG】チャートの縦軸の最大値をセットします(初期値:自動計算)。 何も指定しない場合は、データの最大値から、自動的に計算します。</p>
14.upperBound	<p>【TAG】チャートの縦軸の目盛の幅をセットします(初期値:自動計算)。 何も指定しない場合は、データの範囲から、自動的に計算します。</p>
15.tickSize	<p>【TAG】バーチャートのグラデーション処理を行うかどうか[true/false]をセットします(初期値:false)。 通常のバーチャートは、単一色表示で表されますが、これにグラデーション効果を 付加するかどうかを指定します。 通常のバーチャートが、少しきれいに見えます。 初期値は、false(使用しない)です。</p>
16.useGradient	<p>【TAG】ラインチャートのポイントを四角表示するかどうかを指定します(初期値:true)。 ラインチャートは、通常、線で表され、各ポイントについても、線で接続されます。 shapesVisible を true に設定すると、各ポイントが、線上に四角く表示され、 そのポイントの位置を、容易に判断出来るようになります。 初期値は、true(LineAndShape)です。</p>
17.shapesVisible	<p>【TAG】データ毎にShapeを切り替える時の色の繰返しパターンを文字列配列で指定します。 HybsLine でのみ使用可能です。 これは、データそのものが、繰返し性のある場合に、その繰返し性に対応した 形状のShape を表示させる場合に使用します。 繰返しShapeの形状は、JFreeChart のシリーズ毎の繰返し標準形状を使用します。 現在のバージョンでは、10個までの繰返しに対応可能です。 繰返し色を、指定した分だけ、順に使用されていきます。</p> <p>指定文字列は、java.awt.Color クラスのstatic フィールド名で指定します。 BLACK , BLUE , CYAN , DARK_GRAY , GRAY , GREEN , LIGHT_GRAY , MAGENTA , ORANGE , PINK , RED , WHITE , YELLOW , (PURPLE) が指定できます。 また、#XXXXXX形式の16bitRGB表記 でも指定可能です。 【TAG】shapeの大きさを倍率指定で変更します(初期値:null)。 ラインチャートのShape(各グラフのポイントのマーカー)の大きさは、通常は、 自動設定されます。 この大きさを、倍率指定で、変更可能です。 指定は、double 型です。 初期値は、null は、スケール変更しません(自動設定のままの大きさ)。</p>
18.shapeColors	<p>【TAG】ラインチャートの線をドットラインにするかどうか[true/false]を指定します(初期値:false 線分)。 ラインチャートは、通常、線で表されます。 これに、点線で表すことで、グラフの違いを、色だけでなく形状でも識別 しやすくなることが可能です。 初期値は、false(線分)です。</p>
19.shapeScale	<p>【TAG】複数チャート描画時のチャート色の繰返しパターンをCVS形式で複数指定します。 通常、複数のチャートを同時に表示させる場合は、縦軸が共通であれば、 1回のSelect分で複数データを取得します。 この、データをシリーズと呼んでおり、これを区別する為に、色を分けます。</p>
20.useDottedLine	

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
21.seriesColors	<p>初期値は、JFreeChart が自動で割り振ります。</p> <p>これを、外部からCVS形式で、カンマ区切りで色コードを指定します。</p> <p>指定データが多い場合は、多い分の色は使用されません。少ない場合は、順番に繰り返して使用されます。</p> <p>例えば、1色だけ指定した場合は、すべてのシリーズが同じ色で表されます。</p> <p>指定文字列は、java.awt.Color クラスのstatic フィールド名で指定します。 BLACK, BLUE, CYAN, DARK_GRAY, GRAY, GREEN, LIGHT_GRAY, MAGENTA, ORANGE, PINK, RED, WHITE, YELLOW, (PURPLE) が指定できます。</p> <p>また、#XXXXXX形式の16bitRGB表記 でも指定可能です。</p> <p>【TAG】Value(縦軸)のラベルを表示するかどうかを指定します(初期値:true[表示する])。ValueAxis にて設定される、縦軸情報の、ラベルを表示するかどうか指定します。</p> <p>初期値は、true(表示する)です。</p>
22.valueLabelsVisible	<p>【TAG】Value(縦軸)のマーカーを表示するかどうかを指定します(初期値:true[表示する])。ValueAxis にて設定される、縦軸情報の、マーカーを表示するかどうか指定します。</p> <p>初期値は、true(表示する)です。</p>
23.valueMarksVisible	<p>【TAG】データの値(itemText)を表示するかどうか[true/false/last]を指定します(初期値:false[表示しない])。CategoryItemRenderer 関連のグラフの設定値をグラフ上に表示するかどうかを指定します。</p> <p>true に設定した場合、通常の場合は、すべてのシリーズにラベル表示されます。</p> <p>false に設定すると、表示されません。</p> <p>last を設定すると、各シリーズの最後の値のみ表示されます。</p> <p>ChartCreate クラスに、seriesPickup が設定されている場合は、指定のシリーズのみの設定値を表示し、他の値は、表示しません。</p> <p>同様に、dynamicOCNo が指定されている場合(動的なマーカーライン)</p> <p>最下位のシリーズは、閾値として使用されるため、設定値は表示されません。</p> <p>ラベルの表示位置は、表示する線グラフの傾きに応じてラベルの表示場所を変えます。山形、右坂、谷形、左坂 に応じて、上中、下右、下中、上右 に位置を設定します。右にずらすのは、10 ピクセル固定です。</p> <p>初期値は、false(表示しない)です。</p>
24.itemLabelVisible	<p>【TAG】データの値(itemText)の表示に、桁区切り文字を使用するかどうか[true/false]を指定します(初期値:true[使用する])。</p> <p>itemLabelVisible=true 時に、表示されるデータ値ラベルで、NumberFormat していますが、3 桁区切り文字(123,456,789.0) の区切り記号を表示するかどうかを指定します。</p> <p>true を指定すると、表示します。false では、表示しません。</p> <p>初期値は、true(使用する)です。</p>
25.useItemLabelSep	<p>【TAG】縦軸の表示領域INSET値(double)を指定します。</p> <p>縦軸文字表示領域(NumberAxis)の幅の追加値を設定します。</p> <p>これは、通常の表示領域の左側にスペースを挿入します。</p> <p>あくまで、追加する値なので、文字の長さは含まれません。</p> <p>何も指定しない場合は、設定しません。</p>
26.valueInset	<p>【TAG】BOXチャートのバー幅の最大値(0.0~1.0)を指定します。</p> <p>BOXチャートのバー幅の比率の最大値を指定します。</p> <p>表示領域を1として小数点以下の数値で棒の幅を設定します。</p> <p>設定した幅に無理がある時は適当なサイズに調整されます。</p> <p>(小さくしたときには棒が線になる)</p> <p>設定されるのは、org.jfree.chart.renderer.category.BarRenderer を使用した描画のみです。</p> <p>何も指定しない場合は、設定しません。</p>
27.barMaxWidth	<p>【TAG】BOXチャートのバーアイテムのマージン(0.0~1.0)を指定します。</p> <p>BOXチャートのバーアイテムのマージンの比率を指定します。</p> <p>棒の間を表示領域を1として小数点以下の数値で幅を設定します。</p> <p>無理がある時は適当なサイズに調整されます。</p> <p>barMaxWidth より優先されます。</p> <p>設定されるのは、org.jfree.chart.renderer.category.BarRenderer を使用した描画のみです。</p> <p>何も指定しない場合は、設定しません。</p>
28.barItemMargin	<p>【TAG】BOXチャートのバー幅(double)を指定します。</p> <p>BOXチャートのバー幅(double)を指定します。</p> <p>設定されるのは、org.jfree.chart.renderer.category.BarRenderer を使用した描画のみです。</p> <p>棒グラフのバー幅指定は、直接、CategoryItemRendererState に設定しています。</p> <p>通常は、barMaxWidth(0.0~1.0)とbarItemMargin(0.0~1.0)を用いて比率で指定します。</p> <p>何も指定しない場合は、設定しません。</p>
29.barWidth	<p>【TAG】表示下限値(これ以下のデータは未表示)の値(double)を指定します。</p> <p>HybsLine でのみ使用可能です。</p> <p>この設定値以下のデータは、存在しない扱いとします。</p> <p>Lineを引くとき、このデータと、存在しているデータ間にラインは引かれません。</p> <p>何も指定しない場合は、設定しません。</p>
30.visibleLimit	<p>【TAG】グラフの書き出し位置の調整比率を指定します。</p> <p>HybsStackedBar でのみ使用可能です。</p> <p>グラフを描画する場合の、書き出し位置を少しずらします。</p> <p>これは、グラフの幅に対して、比率で指定します。</p> <p>0.0(初期値)の場合は、初期描画位置である、CategoryAnchor.Middle と同じ箇所から、書き出されます。</p> <p>1.0 の場合、中心から、グラフ幅の半分が加算され、END位置に寄ります。</p> <p>同様に、-1.0 の場合は、グラフ幅の半分が減算され、START 位置になります。</p> <p>つまり、中心から、グラフ幅の半分単位で、前方/後方にずらす事が出来ます。</p> <p>書き出し位置 = 中心(Middle) + (domainMargin) * 幅/2</p> <p>初期値は、0.0(真ん中:MIDDLE)です。</p>
31.domainMargin	<p>【TAG】時刻を表す場合の表現の仕方[E1/E2/E3/H1/H2/H3]を指定します(初期値:null)。</p> <p>HybsNumberAxis にオーバーライドする 時間を表示する DecimalFormat の内部クラスを利用するに当たり、時刻の表示方法を指定します。</p> <p>外部から与える数字は、連続している必要があるため、1 0進数です。</p> <p>たとえば、1700 → 17:00, 2150 → 21:30 という感じです。</p> <p>2400 を超えると日付違いになります。</p> <p>英語表記(:)と日本語表記(時)の区別と、24時間を超える場合の表示方法によって、6種類のパターンが存在します。</p> <p>E1:そのまま、24:00 となり、加算されていく。</p> <p>E2:そのまま、0:00 に戻る。(日付は無視)</p> <p>E3:そのまま、1 00:00 と日付が付与される。</p> <p>H1:そのまま、24時00分 となり、加算されていく。</p> <p>H2:そのまま、00時00分 に戻る。(日付は無視)</p> <p>H3:そのまま、1日 00時00分 と日付が付与される。</p> <p>初期値は、使用しない(-1)です。</p>
32.timeFormatType	<p>【TAG】ラベルの表示向きを縦にするかどうか[false/true]を指定します(初期値:false)。</p> <p>ChartCreate の rotationLabel は、角度を指定できましたが、NumberAxis では、縦にするかどうかの指定しかできません。</p> <p>ここでは、true を指定するとラベルは、縦書きになります。</p> <p>初期値は、false(横書き)です。</p>
33.useVerticalLabels	<p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
34.debug	デバッグ情報を [true:出力する/false:しない] を指定します。 出力形式自体は、個々のタグによって異なります。 <div>形式サンプル</div>

バージョン タグ名 HTML EXT BODY要素	内容
形式サンプル	
5.5.4.2 og:column 拡張 JSP	【TAG】 カラムの名前を指定します。 <og:column name="PN" />
●形式：<og:column name=... value=... /> ●body：あり	
●Tag定義： <og:column name value defaultVal td addNoValue lbl must mustAny writable colspan id maxlength size renderer editor dbType optionAttributes useDefaultPropertieVal nextForm useRequestValue codeName codeList codeGroup clazz caseKey caseVal lang dir title style readonly disabled tabindex accessKey language onClick onChange onBlur onFocus onDbClick onMouseDown onMouseUp onMouseMove onMouseOut onMouseOver onSelect onKeyDown onKeyPress onKeyUp roles eventColumn param eventURL xssCheck useSLabel eventCallback debug > ... Body ... </og:column>	○【TAG】 カラムの名前を指定します。(必須) 【TAG】 設定値を指定します 【TAG】 value属性に値がセットされていないときに、初期値をセットします 【TAG】 テーブル形式の <td> タグを使用するかどうか[yes/no/false]を指定します(初期値:yes) 【TAG】 コンボボックスの場合、値なしのOptionを含めるかどうか[true/false]を指定します(初期値:true) 【TAG】 ラベルリソースのラベルIDを指定します 【TAG】 必須入力を表す色に変わるかどうか[true/false]を指定します(初期値:false) 【TAG】 選択必須入力(どれかひとつ必須)を表す色[true/mustAny/その他]を指定します(初期値:無指定) 【TAG】 書き込みタイプかどうか[true/false]を指定します(初期値:true) 【TAG】 フィールド部分の colspan を指定します 【HTML】 要素に対して固有の名前(id)をつける場合に設定します 【TAG】 最大入力文字数を指定します(入力可能文字数で、バイト数ではありません) 【TAG】 カラムの表示サイズ(横幅)を文字数でセットします(入力文字制限ではありません) 【TAG】 カラムの表示用レンダラー(Renderer_****)をセットします 【TAG】 カラムの編集用エディター(Editor_****)をセットします 【TAG】 DBタイプ(DBType_****)をセットします 【TAG】 JavaScript などの HTML基本タグ以外の属性を、そのままタグとして使用します 【TAG】 リソースファイルの初期値を使うかどうか[true/false]を指定します(初期値:false) 【TAG】 入力カーソルを、指定されたname属性のフィールドへ自動的に移動させます 【TAG】 value が null の場合に、name 属性から Request情報 を適用するかどうかを指定します(初期値:true) 【TAG】 コードセレクトのユニークキー(コード名称)をセットします 【TAG】 コードセレクトのサブセットとなるコードをCSV形式で指定します 【TAG】 コードセレクトのサブセットとなるグループを指定します 【HTML】 要素に対して class 属性を設定します 【TAG】 このタグ自体を利用するかどうかの条件キーを指定します(初期値:null) 【TAG】 このタグ自体を利用するかどうかの条件値を指定します(初期値:null) 【HTML】 要素の内容と他の属性値の言語(lang.xml:lang)を指定します 【HTML】 文字表記の方向(dir)を指定します 【HTML】 要素に対する補足情報(title)を設定します 【HTML】 この要素に対して適用させるスタイルシート(style)を設定します 【TAG】 その部品に対して変更が出来ないように(readonly)指定します(サーバーに送信される) 【TAG】 その部品に対して、選択や変更が出来ないように(disabled)指定します(サーバーに送信されない) 【HTML】 タブの移動順(tabindex)を指定します(0 ~ 32767) 【HTML】 アクセスキー(alt+キーで直接指定)を割り当てます 【TAG】 タグ内部で使用する言語コード[ja/en/zh/...]を指定します 【HTML】 JavaScriptのイベント onClick を設定します(例:onClick="renew(' query. jsp', ' QUERY');") 【HTML】 JavaScriptのイベント onChange を設定します(例:onChange="renew(' query. jsp', ' QUERY');") 【HTML】 JavaScriptのイベント onBlur を設定します(例:onBlur="this. value=value. toUpperCase();") 【HTML】 JavaScriptのイベント onFocus を設定します 【HTML】 JavaScriptのイベント onDbClick を設定します 【HTML】 JavaScriptのイベント onMouseDown を設定します 【HTML】 JavaScriptのイベント onMouseUp を設定します 【HTML】 JavaScriptのイベント onMouseMove を設定します 【HTML】 JavaScriptのイベント onMouseOut を設定します 【HTML】 JavaScriptのイベント onMouseOver を設定します 【HTML】 JavaScriptのイベント onSelect を設定します 【HTML】 JavaScriptのイベント onKeyDown を設定します 【HTML】 JavaScriptのイベント onKeyPress を設定します 【HTML】 JavaScriptのイベント onKeyUp を設定します 【TAG】 ロールをセットします 【TAG】 イベントカラム(親カラム)セット 【TAG】 パラメータをセットします 【TAG】 イベントURLのセット 【TAG】 リクエスト情報の HTMLTag開始/終了文字(<>) 存在チェックを実施するかどうか[true/false]を設定します (初期値:USE_XSS_CHECK[=true]) 【TAG】 MENU, DBMENUでSLABELを利用するかどうか[auto/true/false]を指定します(初期値:auto) 【TAG】 eventColumn実行後のcallback関数指定 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)
●使用例 <og:column name="PN" /> <og:column name="CDC" /> <og:column name="PN" value="syokichi" /> <og:column name="PN" lbl="PNL" /> <og:column name="CDC" /> <og:column name="CDC" addNoValue="false" /> <og:column name="CDC" addNoValue="false" must="true" /> <og:column name="PN" must="true" /> <og:column name="PN" td="false" /> <og:column name="PN" td="no" /> <og:column name="PN" writable="false" /> <og:column name="PN" size="20" /> <og:column name="A" nextForm="B" /> <og:column name="B" /> BODY 部分に記述した値は、カラムフィールドの直後にセットされます。 <og:column name="PN" > <button type="button" onclick="popup(...):return false;"> <og:message lbl="POPU" comment="▼" /> </button> </og:column> eventColumnを利用すると、親カラムの変更に従って子カラムの内容を変える事が可能です。 ・親子関係のプルダウンを作成する場合 <og:column name="OYA" /> <og:column name="KO" eventColumn="OYA" param="SELECT ... WHERE COL = '[@OYA]'" /> ・親子関係の入力欄(親のデータで子をDBから検索)を作成する場合 <og:column name="OYA" /> <og:column name="KO" eventColumn="OYA" param="SELECT A AS VALUE WHERE COL = [@OYA]" />	編集種別がTextの場合は、テキストフィールドが表示されます。 通常はNAME属性を指定するだけです。 編集種別がMenuの場合は、メニューが表示されます。 初期値をセットするときはvalue属性を使います。 ラベルを変更するときはlbl属性を使います。 リソースファイルからコンボボックスも自動作成されます。 NULL値を消すときはaddNoValue属性を使います。 属性は、複数組み合わせで使えます。 色を変更するときはmust属性を使います。 自動作成されるtdタグやラベルを外すときはtd属性を使います。 自動作成されるtdタグを外し、ラベルとフィールドは残します。 非入力にするときはwritable属性を使います。 強制的にSIZEの変更をするときはsize属性を使います。 最大桁数入力後、フォーム B にフォーカスが移動します。

バージョン タグ名 HTML EXT BODY要素	内容
	形式サンプル
	<p>・親子関係の要素をカスタム化されたJSPを利用して作成する場合 変更するカラムはJSPで記述されており、指定する事が可能です。 <og:column name="OYA" /> <og:column name="KO" eventColumn="OYA" param="@OYA" eventURL="custom_makeColumn.jsp" /></p>
1.name	【TAG】カラムの名前を指定します。 <og:column name="PN" />
2.value	【TAG】設定値を指定します。 <og:column name="PN" value="syokichi" />
3.defaultVal	【TAG】value属性に値がセットされていないときに、初期値をセットします。 value属性に値がセットされていないときに、初期値をセットします。 なお、commandがRENEWの場合は、このdefaultValは適用されません。 これは、defaultValの値が埋め込まれた項目が存在する画面に戻った際に、 defaultValの項目の値がnullとなっていた場合に、defaultValの値が復元されると、 検索結果との不一致が発生するためです。
4.td	【TAG】テーブル形式の <td> タグを使用するかどうか[yes/no/false]を指定します(初期値:yes)。 yes (初期値) とすると、ラベルとフィールドをテーブルのtdタグでそれぞれ記述します。 no とすると、tdタグを使用せず、ラベルとフィールドを表示します。 false は、ラベルを出さずに、フィールドのみを表示します。 <td><og:column name="PN" td="no" /> </td>
5.addNoValue	【TAG】コンボボックスの場合、値なしのOptionを含めるかどうか[true/false]を指定します(初期値:true)。 カラムがプルダウンメニュー形式の場合、選択肢には通常データベース登録できる候補が 並びます。しかし、検索時には、候補を絞りたくない(指定しない)場合があり、 その様な場合には、値なしのOptionを含める(true)ことで、対応可能です。 初期値:true(値なしのOptionを含める)なので、検索画面に利用する場合は、そのまま使用できます。 <og:column name="CDC" addNoValue="false" />
6.lbl	【TAG】ラベルリソースのラベルIDを指定します。 ラベルを変更するときに、lbl属性を使います。 ラベルID は、所定の language に基づく ResourceManager の getLabelData(id) を呼び出し、その結果のLabelInterfaceを使用します。 getMsglbl() で取り出せます。 ラベルIDとメッセージIDは同時には登録できません。 ラベルとメッセージは統一されました。
7.must	【TAG】必須入力を表す色に変わるかどうか[true/false]を指定します(初期値:false)。 初期値は、必須でない("false") です。 <og:input name="PN" must="true" />
8.mustAny	【TAG】選択必須入力(どれかひとつ必須)を表す色[true/mustAny/その他]を指定します(初期値:無指定)。 複数のカラムのうち、どれかひとつを必須とする選択必須入力を示す色を指定します。 true または、mustAny を設定すると、class属性に、mustAny がセットされます。 mustAny は、CSSファイルに初期設定されています。true または、mustAny 以外の値をセット すると、その値がそのまま、class属性にセットされますので、選択必須のグループ化が 可能です。 なお、実際の選択必須入力チェックは、ここではなく、columnCheck タグで指定が必要です。 自動処理は、mustAny="true" 指定の場合のみ有効です。 初期値は、無指定です。
9.writable	【TAG】書き込みタイプかどうか[true/false]を指定します(初期値:true)。 初期値は、書き込みタイプ("true") です。 <og:column name="PN" writable="false" />
10.colspan	【TAG】フィールド部分の colspan を指定します。 通常の出力では、ラベルとフィールドのtdが2つ使用されます。この colspan は、 フィールドに追加されます。 たとえば、3 を指定した場合、フィールドが、次のテーブルの、ラベル、フィールド部分まで 伸びるため、レイアウト的にすっきりと、長いフィールドを適用する事が可能です。 <og:column name="PN" colspan="3" />
11.id	【HTML】要素に対して固有の名前(id)をつける場合に設定します。 特別な使用方法として、id="FOCUS" とすることで、フィールド系要素に フォーカスを移動させます。これは、そのページ内で唯一の id 属性として使用ください。
12.maxlength	【TAG】最大入力文字数を指定します(入力可能文字数で、バイト数ではありません)。 <og:column name="PN" maxlength="20" />
13.size	【TAG】カラムの表示サイズ(横幅)を文字数でセットします(入力文字制限ではありません)。 <og:column name="PN" size="20" />
14.renderer	【TAG】カラムの表示用レンダラー(Renderer_****)をセットします。 これは、plugin.column 以下の Renderer_**** クラスの **** を 与えます。これらは、CellRenderer インターフェースを継承したサブクラスです。 属性クラス定義の [@link org.opengion.hayabusa.db.CellRenderer Renderer] を参照願います。
15.editor	【TAG】カラムの編集用エディター(Editor_****)をセットします。 これは、plugin.column 以下の Editor_**** クラスの **** を 与えます。これらは、CellEditor インターフェースを継承したサブクラスです。 属性クラス定義の [@link org.opengion.hayabusa.db.CellEditor Editor] を参照願います。
16.dbType	【TAG】DBタイプ(DBType_****)をセットします。 これは、org.opengion.hayabusa.db.column 以下の DBType_**** クラスの **** を 与えます。これらは、DBType インターフェースを継承したサブクラスです。 属性クラス定義の [@link org.opengion.hayabusa.db.DBType DBType] を参照願います。
17.optionAttributes	【TAG】JavaScript などの HTML基本タグ以外の属性を、そのままタグとして使用します。 JavaScript などの HTML基本タグ以外の属性を、そのまま タグとして使用します。 そのときに、()は、()に置き換えます。
18.useDefaultPropertieVal	【TAG】リソースファイルの初期値を使うかどうか[true/false]を指定します(初期値:false)。 初期値がセットされていないときに指定します。 使う場合はtrue、使わない場合は falseをセットします。 この初期値を、システムリソースの USE_DEFAULT_PROPRTIE_VALUE で設定可能とします。
19.nextForm	【TAG】入力カーソルを、指定されたname属性のフィールドへ自動的に移動させます。 JavaScript の onKeyup イベントを利用して、最大桁数まで入力すると、 次のフィールドに、自動的にカーソルが飛ぶように設定します。 引数は、次に飛ばすフィールドの name 属性です。 実行するには、JavaScript の nextForm 関数が必要です。(default.js で指定)
20.useRequestValue	【TAG】value が null の場合に、name 属性から Request情報 を適用するかどうかを指定します(初期値:true)。 name 属性に一致するリクエスト情報がある場合、value が null の場合や、 引数を設定していない場合は、この設定値に応じて、リクエスト情報の値を 設定します(初期値:true なので、通常はこの動きです) ごくまれに、ENTRY 系の削除を行う場合、リクエスト情報が残り、削除されずに 再表示(D B的には削除済み)されるケースでは、使用しない(false)設定が可能です。 useRequestValue true (初期値) false value A A A A _ _ _ _ _ _ _ _ name(Request情報) B B _ _ B B _ _ B B _ _

バージョン タグ名 HTML5 BODY要素	内容
	形式サンプル
	<div>defaultVal C _ C _ C _ C _ C _ C _</div> <div>=====</div> <div>設定値 A A A A B B C C C</div>
21.codeName	【TAG】コードセレクトのユニークキー(コード名称)をセットします。 このキーを元に、CodeData オブジェクトを構築します。
22.codeList	【TAG】コードセレクトのサブセットとなるコードをCSV形式で指定します。 この値に限定された、コードリソースを、既存の、CodeData のサブセットとして作成します。 codeName で指定されたリソースでも、サブセットを作成する事ができます。
23.codeGroup	【TAG】コードセレクトのサブセットとなるグループを指定します。 この値に限定された、コードリソースを、既存の、CodeData のサブセットとして作成します。 codeName で指定されたリソースでも、サブセットを作成する事ができます。
24.clazz	【HTML】要素に対して class 属性を設定します。 Javaの言語使用上、class で作成できないため、代用として clazz を使用しています。 html で作成される属性は、class で作成されます。
25.caseKey	【TAG】このタグ自体を利用するかどうかの条件キーを指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。
26.caseVal	【TAG】このタグ自体を利用するかどうかの条件値を指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。
27.lang	【HTML】要素の内容と他の属性値の言語(lang.xml::lang)を指定します。 HTMLの言語属性に使われます。指定する値は、ISO 639で規定されている「言語コード」です。 [ja/en/zh/...]などのほかに、en-US:アメリカ英語、en-cockney:コックニー英語 など、副言語を指定する方法も定められています。 ここでは、lang と xml:lang の両方に同じ値がセットされます。 タグの language 属性とは使用用途が異なります。lang セット
28.dir	【HTML】文字表記の方向(dir)を指定します。 当該要素の書字方向を指定する属性です。 ltr で、左から右に、rtl で、右から左に並べます。
29.title	【HTML】要素に対する補足的情報(title)を設定します。 title セットは、ボタンなどに適用すると、マウスオーバーによりこのメッセージがチップスのように表示されます。これを利用して、説明文を登録することが可能です。 ここに登録した文字列が、メッセージリソースに存在する場合は、そのメッセージを存在しない場合は、そのままの値を設定します。
30.style	【HTML】この要素に対して適用させるスタイルシート(style)を設定します。 タグにstyle属性を設定します。これは、キー:値: のセットを複数記述できます。 通常は、class属性や、id属性で登録しておき、&lg:style type="text/css"> で外部から指定する方がソースは読みやすくなります。
31.readonly	【TAG】その部品に対して変更が出来ないように(readonly)指定します(サーバーに送信される)。 INPUT/TEXTAREA 系に対して、指定可能です。 readonly="readonly" , readonly="true" が指定された場合は、有効です。 false も指定値としては、有効です。(大文字小文字の区別も不要) それ以外の指定は、エラーとします。
32.disabled	【TAG】その部品に対して、選択や変更が出来ないように(disabled)指定します(サーバーに送信されない)。 BUTTON/INPUT/OPTGROUP/OPTION/SELECT/TEXTAREA 系に対して、指定可能です。 disabled="disabled" , disabled="true" が指定された場合は、有効です。 false も指定値としては、有効です。(大文字小文字の区別も不要) それ以外の指定は、エラーとします。
33.tabindex	【HTML】タブの移動順(tabindex)を指定します(0 ~ 32767)。 Tabキーを押したときに要素が選択される順番を指定します。 値には、選択させたい順番を数値で記述します。
34.accesskey	【HTML】アクセスキー(alt+キーで直接指定)を割り当てます。 アクセスキーは、マウスの使えない環境でも、リンクにジャンプする、ボタンを押す、入力フォームにフォーカスを移すなどの操作を簡単に行うことができるように考慮されたものです。 Windows の「ファイル(F)」メニューについている、F と同じような動きをします。
35.language	【TAG】タグ内部で使用する言語コード[ja/en/zh/...]を指定します。 この言語コードに基づいて、表示のラベルをリソースから作成します。
36.onClick	【HTML】JavaScriptのイベント onClick を設定します(例:onClick="renew(' query. jsp',' QUERY');")。 onClick をセットします。 例えば、<og:column name="KBSAKU" onClick="renew(' query. jsp',' QUERY');"> /> のように指定することで、プルダウンメニューの絞込み検索が可能になります。
37.onChange	【HTML】JavaScriptのイベント onChange を設定します(例:onChange="renew(' query. jsp',' QUERY');")。 onChange をセットします。 例えば、<og:column name="KBSAKU" onChange="renew(' query. jsp',' QUERY');"> /> のように指定することで、プルダウンメニューの絞込み検索が可能になります。
38.onBlur	【HTML】JavaScriptのイベント onBlur を設定します(例:onBlur="this.value=value.toUpperCase();")。 onBlur は、フォーカスが離れたときに発生するイベントです。
39.onFocus	【HTML】JavaScriptのイベント onFocus を設定します。 onFocus は、フォーカスされたときに発生するイベントです。
40.ondblClick	【HTML】JavaScriptのイベント ondblClick を設定します。 ondblClick は、マウスでダブルクリックされたときに発生するイベントです。
41.onMouseDown	【HTML】JavaScriptのイベント onMouseDown を設定します。 onMouseDown は、マウスダウンされたときに発生するイベントです。
42.onMouseUp	【HTML】JavaScriptのイベント onMouseUp を設定します。 onMouseUp は、マウスアップされたときに発生するイベントです。
43.onMouseMove	【HTML】JavaScriptのイベント onMouseMove を設定します。 onMouseMove は、マウスが移動されたときに発生するイベントです。
44.onMouseOut	【HTML】JavaScriptのイベント onMouseOut を設定します。 onMouseOut は、マウスが離れたときに発生するイベントです。
45.onMouseOver	【HTML】JavaScriptのイベント onMouseOver を設定します。 onMouseOver は、マウスが重なったときに発生するイベントです。
46.onSelect	【HTML】JavaScriptのイベント onSelect を設定します。 onSelect は、テキストフィールド/テキストエリアのテキストが選択されたときに発生するイベントです。
47.onKeydown	【HTML】JavaScriptのイベント onkeydown を設定します。 onkeydown は、キーが押されたときに発生するイベントです。
48.onKeyPress	【HTML】JavaScriptのイベント onkeypress を設定します。 onkeypress は、キーが押され続けていたときに発生するイベントです。
49.onKeyUp	【HTML】JavaScriptのイベント onkeyup を設定します。 onkeyup は、キーが押された状態から離れたときに発生するイベントです。
50.roles	【TAG】ロールをセットします。 ここで指定したカラムロールを元に、ユーザー毎のアクセス許可がチェックされます。 アクセス許可されないと、表示されません。 このロールを指定しない場合は、カラムリソースのロールが使用されます。
	【TAG】イベントカラム(親カラム)セット。 親子関係を持った項目の動的な入れ替えを行う場合に利用します。 このカラムを変化させるトリガとなるカラムの名前をセットします。

バージョン タグ名 HTMLTEXT BODY要素	内容																										
51.eventColumn	<p>形式サンプル</p> <p>この属性がセットされた場合、親カラムが変更都度子カラム(自身のカラム)が再表示されます。 標準で再表示時に受け継がれる属性は以下の通りです。 (1)editor・renderer (2)writable (3)addNoValue (4)class(must・mustAnyや、DBTypeを含みます) (5)style(optionAttributes等でセットしたもの) (6)size (7)maxLength これら以外に特殊な設定を行ったカラムに関しては、eventURL属性でカスタムの置き換え用JSPを作成し、個別に対応して下さい。</p> <p>また、置き換えの定義は、param属性で定義します。 param属性に定義されたSQL文の結果により、各種エディター、レンダラーによって、以下のよう置き換えが行われます。</p> <p>(1)DBMENUの場合 paramに指定されたSQL文により動的にプルダウンを置き変えます。 SQL文の{@XXXX}の記述により、その時点での親カラムの値を参照することができます。</p> <p>例) SELECT A.CODE, B.LNAME FROM GEA04 A, GEA08 B WHERE A.CLM ' '. ' A.CODE = B.CLM AND A.SYSTEM_ID = B.SYSTEM_ID AND A.CLM like 'KBSAKU%' AND B.LANG like '{@LANG}' ← name=LANGを持つ項目のその時点での値に変換される AND B.SYSTEM_ID = '**'</p> <p>(2)DBMENU以外の場合 paramに指定されたSQL文により値(value値)を置き換えたカラムを作成し、置き換えます。 DBMENUの場合と同様、SQL文の{@XXXX}の記述により、その時点での親カラムの値を参照することができます。</p> <p>例) SELECT LABEL_NAME FROM GF41 WHERE SYSTEM_ID='{@SYS.SYSTEM_ID}' AND CLM='{@CLM}' AND LANG='{@LANG}'</p> <p>なお、いずれの場合も、{@XXXX}の部分については、\$1,\$2...の形式でも記述することができます。 この場合、\$1は1番目のeventColumnのカラム、\$2は2番目のeventColumnのカラムと指します。</p> <p>(注意点1) イベントカラムによるイベントは、画面表示時に1度実行されます。 但し、eventColumn属性で、カラム名の先頭に'_'をつけた場合、または親カラムの値がNULLの場合は実行されません。</p> <p>(注意点2) SQL文などのパラメーター中に記述された{@XXXX}の値の変換はJavaScriptにより行われるため、値が取得できる項目は、“画面に書き込み可能状態で表示されているカラム”に限ります。 テールモデル内の変数や表示のみの値については、ここでは利用することはできません。 {@USER.XXXX}等の予約語に関しては、5.5.4.0以降では処理されます。</p> <p>(注意点3) 現時点では、INMENU, INDBMENUについては、入れ替え後のJavaScriptが正常に動作しないため、イベントカラムが正常に動作しません。</p>																										
52.param	<p>【TAG】パラメータをセットします。 EditorParam等のパラメータを一括でセットします。 eventColumn利用時は{@XXXX}はリクエストパラメータではなくxxxカラムの実際の値が利用されます。</p>																										
53.eventURL	<p>【TAG】イベントURLのセット。 初期値はシステムリソースのEVENT_COLUMN_URLです。 イベントカラム指定時に部品を作成するJSPを指定します。 (設定例：eventURL="makeColumn_custom.jsp")</p> <p>置き換え対象のカラムに特殊な設定がされており、標準のカラム置き換えでは対応できない場合に、このURLを指定することで、個別に置き換え方法を定義する必要があります。</p> <p>なお、個別に置き換え方法を定義したJSPを作成する場合は、標準(jsp/common/eventColumnMaker.jsp)を参考にして作成して下さい。</p> <p>eventColumnから利用するJSPで利用可能なリクエスト変数</p> <table><tr><td data-bbox="368 1424 475 1447">{@EC.CNAME}</td><td data-bbox="512 1424 1027 1447">子カラム名(一覧表示の場合でも行番号__nは付加されません)</td></tr><tr><td data-bbox="368 1447 475 1469">{@EC.RENDERER}</td><td data-bbox="512 1447 660 1469">子カラムのレンダラー</td></tr><tr><td data-bbox="368 1469 475 1491">{@EC.EDITOR}</td><td data-bbox="512 1469 740 1491">子カラムのエディター</td></tr><tr><td data-bbox="368 1491 475 1514">{@EC.SQL}</td><td data-bbox="512 1491 1331 1514">子カラムのプルダウン、または値を出力するためのSQL文({@XXXX}をカラムのvalue値で置き換えたもの)</td></tr><tr><td data-bbox="368 1514 475 1536">{@EC.WRITABLE}</td><td data-bbox="512 1514 772 1536">子カラムが編集可能かどうか</td></tr><tr><td data-bbox="368 1536 475 1559">{@EC.CVALUE}</td><td data-bbox="512 1536 724 1559">子カラムの現在の値</td></tr><tr><td data-bbox="368 1559 475 1581">{@EC.NOVALUE}</td><td data-bbox="512 1559 1123 1581">子カラムの最上部Option値が空ならtrue。その他false(addNoValue属性値と同じです)</td></tr><tr><td data-bbox="368 1581 475 1603">{@EC.CLASS}</td><td data-bbox="512 1581 740 1603">子カラムのCLASS属性</td></tr><tr><td data-bbox="368 1603 475 1626">{@EC.SIZE}</td><td data-bbox="512 1603 724 1626">子カラムのSIZE属性</td></tr><tr><td data-bbox="368 1626 475 1648">{@EC.MAXLENGTH}</td><td data-bbox="512 1626 772 1648">子カラムのMAXLENGTH属性</td></tr><tr><td data-bbox="368 1648 475 1671">{@EC.OPTATTRS}</td><td data-bbox="512 1648 772 1671">子カラムのその他OPTION属性</td></tr><tr><td data-bbox="368 1671 475 1693">{@EC.PNAMEE}</td><td data-bbox="512 1671 1027 1693">親カラム名(一覧表示の場合でも行番号__nは付加されません)</td></tr><tr><td data-bbox="368 1693 475 1715">{@EC.PVALUE}</td><td data-bbox="512 1693 724 1715">親カラムの現在の値</td></tr></table>	{@EC.CNAME}	子カラム名(一覧表示の場合でも行番号__nは付加されません)	{@EC.RENDERER}	子カラムのレンダラー	{@EC.EDITOR}	子カラムのエディター	{@EC.SQL}	子カラムのプルダウン、または値を出力するためのSQL文({@XXXX}をカラムのvalue値で置き換えたもの)	{@EC.WRITABLE}	子カラムが編集可能かどうか	{@EC.CVALUE}	子カラムの現在の値	{@EC.NOVALUE}	子カラムの最上部Option値が空ならtrue。その他false(addNoValue属性値と同じです)	{@EC.CLASS}	子カラムのCLASS属性	{@EC.SIZE}	子カラムのSIZE属性	{@EC.MAXLENGTH}	子カラムのMAXLENGTH属性	{@EC.OPTATTRS}	子カラムのその他OPTION属性	{@EC.PNAMEE}	親カラム名(一覧表示の場合でも行番号__nは付加されません)	{@EC.PVALUE}	親カラムの現在の値
{@EC.CNAME}	子カラム名(一覧表示の場合でも行番号__nは付加されません)																										
{@EC.RENDERER}	子カラムのレンダラー																										
{@EC.EDITOR}	子カラムのエディター																										
{@EC.SQL}	子カラムのプルダウン、または値を出力するためのSQL文({@XXXX}をカラムのvalue値で置き換えたもの)																										
{@EC.WRITABLE}	子カラムが編集可能かどうか																										
{@EC.CVALUE}	子カラムの現在の値																										
{@EC.NOVALUE}	子カラムの最上部Option値が空ならtrue。その他false(addNoValue属性値と同じです)																										
{@EC.CLASS}	子カラムのCLASS属性																										
{@EC.SIZE}	子カラムのSIZE属性																										
{@EC.MAXLENGTH}	子カラムのMAXLENGTH属性																										
{@EC.OPTATTRS}	子カラムのその他OPTION属性																										
{@EC.PNAMEE}	親カラム名(一覧表示の場合でも行番号__nは付加されません)																										
{@EC.PVALUE}	親カラムの現在の値																										
54.xssCheck	<p>【TAG】リクエスト情報の HTMLTag開始/終了文字(<>) 存在チェックを実施するかどうか[true/false]を設定します (初期値:USE_XSS_CHECK[=true])。</p> <p>クロスサイトスクリプティング(XSS)対策の一環としてless/greater than signについてのチェックを行います。 <> が含まれていたエラーにする(true)／か／ーチェックか(false)を指定します。 (初期値:システム定数のUSE_XSS_CHECK[=(@og.value org.opengion.hayabusa.common.SystemData#USE_XSS_CHECK)])。</p>																										
55.useSLabel	<p>【TAG】MENU, DBMENUでSLABELを利用するかどうか[auto/true/false]を指定します(初期値:auto)。 通常はセレクトメニューは一覧とそれ以外で短縮ラベルの利用が自動で切り替わります。 強制的にSLABELを利用させたい場合にこの属性をtrue/falseにセットして下さい。 標準はauto(自動)です。 auto/true/false以外を指定した場合はfalse扱いとします。</p>																										
56.eventCallback	<p>【TAG】eventColumn実行後のcallback関数指定。 eventColumnの動作終了後に実行するcallback関数の指定を行います。 関数名のみを指定して下さい。 ex) sampleCallback() を実行する場合はeventCallback="sampleCallback" 実行する関数の第一引数には自動的に次の要素が入った配列が渡されます [0] 変更された子カラムのID属性値 [1] eventColumn実行後の子カラムの値 [2] eventColumn実行前の子カラムの値</p>																										
57.debug	<p>この属性は、optionAttributesへの設定と同様の動作を行います。 【TAG】デバッグ情報出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。</p>																										

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:columnCheck	<p>【TAG】 コマンド[NEW/RENEW/ENTRY]をセットします (初期値:NEW)。 コマンドは、HTMLから (get/post) 指定されますので、CMD_xxx で設定される フィールド定数値のいずれかを、指定できます。 何も設定されない、または、null の場合は、“NEW” が初期値にセットされます。</p>
JSP	<p>●形式 : <og:columnCheck command = “...” /> ●形式 : <og:columnCheck command = “...” >エラー時のみ処理 </og:columnCheck> ●body : なし/あり</p>
●Tag定義 : <og:columnCheck command columns nullCheck mustAnyCheck checkType maxRowCount minRowCount tableId matchKeys matchVals realTime useStrict useColumnLabel checkNames debug > ... Body ... </og:columnCheck>	<p>【TAG】 コマンド[NEW/RENEW/ENTRY]をセットします (初期値:NEW) 【TAG】 チェックすべきカラム列をカンマ区切り (CSV形式) で指定します 【TAG】 NULL チェックすべきカラム列をカンマ区切り (CSV形式) りで指定します 【TAG】 選択必須カラム (指定のカラムの内最低ひとつがNULLでない) を“AA BB CC, XX YY ZZ” 形式で指定します 【TAG】 チェック対象のデータ [AUTO/NEW/ENTRY] を指定します (初期値:AUTO) 【TAG】 ENTRY時にチェックで選択された行数の最大値を設定します 【TAG】 ENTRY時にチェックで選択された行数の最小値を設定します 【TAG】 (通常は使いません) sessionに登録されている DBTableModel を取り出すキーを指定します 【TAG】 正規表現でのマッチングを行うカラム列をカンマ区切り (CSV形式) で指定します 【TAG】 正規表現でのマッチングを行うカラム列に対する値 (正規表現) をカンマ区切り (CSV形式) で指定します 【TAG】 (通常使いません) リアルタイムチェックを行う場合に有効にします 【TAG】 NEWの場合に、厳密にチェックするかどうか [true/false] を指定します (初期値:false) 【TAG】 カラムラベルを使用するかどうか [true/false] を指定します (初期値:false) 【TAG】 リクエスト変数の正規化を行うカラムをCSV形式で複数指定します 【TAG】 デバッグ情報を出力するかどうか [true/false] を指定します (初期値:false)</p>
●使用例 <og:columnCheck command = “[@command]” columns = “AAA, BBB, CCC” maxRowCount = “1” minRowCount = “1” nullCheck = “AAA, CCC” > </og:columnCheck>	<p>DB定義 (DBColumnリソース) で定義した項目 (桁数, タイプ等) でチェックします。 チェックで選ばれた 最大選択件数 (設定値を含む) を指定。 チェックで選ばれた 最小選択件数 (設定値を含む) を指定。 NULLチェックを実行します。</p>
[エラー時に、BODY部に記述された内容出力する。正常時には、このBODY部の記述は出力されません。] <og:columnCheck command = “[@command]” columns = “AAA, BBB, CCC” nullCheck = “AAA, CCC” > <form method=“POST” action=“forward.jsp” target=“RESULT”> <og:input type=“button” onClick=“history.back()” msg=“MSG0049” accesskey=“R” td=“false” /> </form> </og:columnCheck>	<p>DB定義 (DBColumnリソース) で定義した項目 (桁数, タイプ等) でチェックします。 NULLチェックを実行します。</p>
mustAnyCheck 属性に、選択必須カラムを指定します。 例: mustAnyCheck=“AA BB CC” AA, BB, CC のカラムで選択必須 (すべてがnullならエラー) 例: mustAnyCheck=“AA BB CC, XX YY ZZ” AA, BB, CC のセットと、XX, YY, ZZのセットでそれぞれ選択必須。 例: mustAnyCheck=“AA XX, AA YY, AA ZZ” AA に値があればすべて成立。そうでない場合は、XX と YY と ZZ がすべて必須。 例: mustAnyCheck=“AA BB, BB CC, AA CC” AA, BB, CC の内、どれか 2 つが必須。AAが成立すればBBかCCが必須。同様に、 BBが成立すれば、AAかCCが必須。 例: query.jsp <og:column name=“AA” mustAny=“true” /> <og:column name=“BB” mustAny=“true” /> <og:column name=“XX” mustAny=“XYZ” /> <og:column name=“YY” mustAny=“XYZ” /> result.jsp <og:columnCheck mustAnyCheck=“AA BB, XX YY” /> custom/custom.css .XYZ { background-color: Green; }	
例:動的カラムで、entry.jsp でのチェックを行う場合。 entry.jsp <og:columnCheck command=“NEW” columns=“*” useColumnLabel=“true” /> useColumnLabel で、動的に作成されたカラム (SAVE=TRUEでキャッシュ済み) を使って チェックします。command=“NEW” なので、対象カラムを指定するのに、“*” ですべてを選択します。	
1.command	<p>【TAG】 コマンド[NEW/RENEW/ENTRY]をセットします (初期値:NEW)。 コマンドは、HTMLから (get/post) 指定されますので、CMD_xxx で設定される フィールド定数値のいずれかを、指定できます。 何も設定されない、または、null の場合は、“NEW” が初期値にセットされます。</p>
2.columns	<p>【TAG】 チェックすべきカラム列をカンマ区切り (CSV形式) で指定します。 columns=“AAA, BBB, CCC, DDD” 先に配列に分解してからリクエスト変数の値を取得するようにします。 こうする事で、リクエストにカンマ区切りの値を設定できるようになります。 分解方法は、通常のパラメータ取得後に、CSV分解します。 “*” を指定すると、command=“NEW” のときの判定にすべてのリクエスト文字列を チェック対象とします。通常は、必要分だけ指定しますが、動的カラムなどの 場合は、カラム名を予め指定できないため、“*” で指定できるようにします。 その場合、“h_”, “hX_”, maxRowCount, command, pageSize, pagePlus, GAMENID, BACK_JSPID, BACK_GAMENID の予約語は、対象から除きます。</p>
3.nullCheck	<p>【TAG】 NULL チェックすべきカラム列をカンマ区切り (CSV形式) りで指定します。 (must 属性のセットにより、自動処理されます) nullCheck=“AAA, BBB, CCC, DDD” 先に配列に分解してからリクエスト変数の値を取得するようにします。 こうする事で、リクエストにカンマ区切りの値を設定できるようになります。 分解方法は、通常のパラメータ取得後に、CSV分解します。</p>
	<p>【TAG】 選択必須カラム (指定のカラムの内最低ひとつがNULLでない) を“AA BB CC, XX YY ZZ” 形式で指定します。 (mustAny 属性のセットにより、自動処理されます) 複数のカラム属性の値のうち、どれかが null でない場合は正常とし、 すべてが null の場合を警告します。 例: mustAnyCheck=“AA BB CC”</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
4.mustAnyCheck	<p>AA, BB, CC のコラムで選択必須(すべてがnullならエラー)</p> <p>例: mustAnyCheck="AA BB CC, XX YY ZZ"</p> <p>AA, BB, CC のセットと、XX, YY, ZZのセットでそれぞれ選択必須。</p> <p>例: mustAnyCheck="AA XX, AA YY, AA ZZ"</p> <p>AA に値があればすべて成立。そうでない場合は、XX と YY と ZZ がすべて必須。</p> <p>例: mustAnyCheck="AA BB, BB CC, AA CC"</p> <p>AA, BB, CC の内、どれか2つが必須。AAが成立すればBBかCCが必須。同様に、BBが成立すれば、AAかCCが必須。</p> <p>選択必須は、must と同様に、色づけを行う(query.jsp)画面では、mustAny 属性をセットします。チェックを行う(result.jsp)画面では、columnCheck タグの mustAnyCheck 属性に、選択必須コラムを指定します。(上記参照)</p> <p>column タグ等の mustAny 属性に、mustAny="true" とセットすると、生成されるHTMLは、class="mustAny" が出力されます。エンジン標準では、default.css に .mustAny が定義されています。(background-color: #C0FFFF;)</p> <p>なお、mustAny 属性に、true 以外の値をセットした場合(mustAny="XYZ")は、生成されるHTMLに、class="XYZ" が出力されます。これは、複数のグループ間で色を変えて、選択必須を指定する場合に使用します。色は、custom/custom.css で指定します。</p> <p>分解方法は、通常のパラメータ取得後に、CSV分解します。</p>
5.checkType	<p>【TAG】 チェック対象のデータ[AUTO/NEW/ENTRY]を指定します(初期値:AUTO)。</p> <p>通常のリクエストデータは、キーそのものですが、エントリデータは表形式のデータを一括で登録する為、(キー+"_" + 行番号)形式を、バラす必要があります。</p> <p>ENTRY は、DBTableModelのリクエスト情報をチェックします。</p> <p>これは、全データが対象になります。(columns/nullCheck 属性は無効)</p> <p>AUTO は、command が、上記 NEW か ENTRY かを判断して処理を振り分けます。</p> <p>初期値は、AUTO です。</p>
6.maxRowCount	<p>【TAG】 ENTRY時にチェックで選択された行数の最大値を設定します。</p> <p>最大選択件数を超えた場合は、エラーメッセージを返します。</p> <p>例えば、1件のみを正常とする場合は、maxRowCount="1" とします。</p>
7.minRowCount	<p>【TAG】 ENTRY時にチェックで選択された行数の最小値を設定します。</p> <p>最小選択件数を超えない場合は、エラーメッセージを返します。</p> <p>例えば、1件のみを正常とする場合は、minRowCount="1" とします。</p>
8.tableId	<p>【TAG】 (通常は使いません) sessionに登録されている DBTableModel を取り出すキーを指定します。</p> <p>検索結果のDBTableModelを別メモリに分ける場合は、このキーを指定します。</p> <p>その場合、メモリ上から削除する方法がないため(session タイムアウトで破棄)大量のデータを分けた場合は、ご注意ください。</p> <p>初期値は、HybsSystem.TBL_MD_L_KEY です。</p>
9.matchKeys	<p>【TAG】 正規表現でのマッチングを行うコラム列をカンマ区切り(CSV形式)で指定します。</p> <p>matchKeys="AAA, BBB, CCC, DDD"</p> <p>matchKeys属性とmatchVals属性の個数は、同じでなければなりません。</p> <p>先に配列に分解してからリクエスト変数の値を取得するようにします。</p> <p>こうする事で、リクエストにカンマ区切りの値を設定できるようになります。</p>
10.matchVals	<p>【TAG】 正規表現でのマッチングを行うコラム列に対する値(正規表現)をカンマ区切り(CSV形式)で指定します。</p> <p>matchVals="AAA, BBB, CCC, DDD"</p> <p>matchKeys属性とmatchVals属性の個数は、同じでなければなりません。</p> <p>先に配列に分解してからリクエスト変数の値を取得するようにします。</p> <p>こうする事で、リクエストにカンマ区切りの値を設定できるようになります。</p>
11.realTime	<p>【TAG】 (通常使いません)リアルタイムチェックを行う場合に有効にします。</p> <p>リアルタイムチェックを行う場合に有効にする属性です。</p> <p>trueが指定された場合、通常のチェックと比較し、以下の差異があります。</p> <p>①エラー結果を簡易フォーマットで出力します。</p> <p>(ViewFormType="HTMLSimpleErrorList"で表示されます。)</p> <p>この簡易フォーマットで出力した場合は、ラベル(短)の定義で出力され、コラム名やデータ等は出力されません。</p> <p>②must, mustAny属性のチェックを行いません。</p> <p>必須及び選択必須は、視覚的に判別できるため、リアルタイムチェックの対象外とします。</p>
12.useStrict	<p>【TAG】 NEWの場合に、厳密にチェックするかどうか[true/false]を指定します(初期値:false)。</p> <p>データをチェックするに当たり、あいまい検索などで厳密にチェックするとエラーになるケースがあるため、command="NEW" のチェック時には、甘い目のチェックを行っています。DBCColumn#valueCheck(String , false)</p> <p>command="ENTRY" のチェックは、厳密なチェックです。</p> <p>このフラグは、検索時でも、厳密なチェックを行いたい場合に、true をセットします。</p>
13.useColumnLabel	<p>【TAG】 カラムラベルを使用するかどうか[true/false]を指定します(初期値:false)。</p> <p>カラムラベルはラベルローダーを利用せずにアプリケーション側で追加したラベルをResourceManagerにキャッシュしている所から取り出すかどうかを指定します。</p> <p>カラムラベルに存在しない場合は、通常のラベルから検索します。</p>
14.checkNames	<p>【TAG】 リクエスト変数の正規化を行うコラムをCSV形式で複数指定します。</p> <p>カラムチェックで正規化が行われていないプラグインに対して強制的に正規化を行います。</p> <p>この属性を利用したチェックを行った場合は、queryタグでPL/SQLを動作させる場合にも同名属性でカラムの指定を行う必要があります。</p> <p>(queryタグでcheckNamesを利用しない場合はこの属性はnullにしてください)</p> <p>この属性はチェックタイプがNEWの場合のみ有効です。</p> <p>リアルタイムチェックでこの機能を利用したい場合は、columnTag等のOptionAttributes属性にrtOption= checkNames=VIEW_DATE のように記述してください。</p>
15.debug	<p>【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。</p> <p>デバッグ情報を [true:出力する/false:しない]を指定します。</p> <p>出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
5.5.4.2 og:columnEditor 拡張 JSP ●形式: <og:ColumnEditor column="..." > ... </og:ColumnEditor > ●body: あり ●Tag定義: <og:columnEditor column columns label className size maxLength writable renderer editor dbType codeName codeList codeGroup defaultVal parameter roles must mustAny optionAttributes addNoValue useLabelMap scope tableId dbId caseKey caseVal lang style clazz onClick onChange onBlur onFocus ondblClick onMouseDown onMouseUp onMouseMove onMouseOut onMouseOver onSelect onKeyDown onKeyPress onKeyUp eventColumn eventURL eventCallback debug > ... Body ... </og:columnEditor>	【TAG】カラム名をセットします。 カラム名をセットします。 columns と同時にはセットできません。 値に"*"を指定することで、全カラムを対象とすることができます。 また、"PN*", CDK*"等のように前方一致でのワイルドカード指定も可能です。 【TAG】カラム名をカンマ区切りで複数セットします カラム名をカンマ区切りで複数セットします columns と同時にはセットできません。 値に"*"を指定することで、全カラムを対象とすることができます。 また、"PN*", CDK*"等のように前方一致でのワイルドカード指定も可能です。 【TAG】ラベルリソースに定義されたラベル I D をセットします ラベルリソースに定義されたラベル I D をセットします 【TAG】カラムのクラス名 (VARCHAR2, NUMBER など) をセットします カラムのクラス名 (VARCHAR2, NUMBER など) をセットします 【TAG】カラムの表示サイズをセットします (大ききだけで入力文字制限ではありません) カラムの表示サイズをセットします (大ききだけで入力文字制限ではありません) 【TAG】カラムの入力文字制限数をセットします カラムの入力文字制限数をセットします 【TAG】カラムが書き込み可能かどうか [true/false] をセットします (初期値:null) カラムが書き込み可能かどうか [true/false] をセットします (初期値:null) 【TAG】カラムの表示用レンダラー (Renderer_****) をセットします カラムの表示用レンダラー (Renderer_****) をセットします 【TAG】カラムの編集用エディター (Editor_****) をセットします カラムの編集用エディター (Editor_****) をセットします 【TAG】DBタイプ (DBType_****) をセットします DBタイプ (DBType_****) をセットします 【TAG】コードセレクトのユニークキー (コード名称) をセットします コードセレクトのユニークキー (コード名称) をセットします 【TAG】コードセレクトのサブセットとなるコードをCSV形式で指定します コードセレクトのサブセットとなるコードをCSV形式で指定します 【TAG】コードセレクトのサブセットとなるグループを指定します コードセレクトのサブセットとなるグループを指定します 【TAG】初期値をセットします 初期値をセットします 【TAG】パラメータをセットします パラメータをセットします 【TAG】ロールをセットします ロールをセットします 【TAG】必須入力を表す色に変えるかどうか [true/false] を指定します (初期値:false) 必須入力を表す色に変えるかどうか [true/false] を指定します (初期値:false) 【TAG】選択必須入力 (どれかひとつ必須) を表す色 [true/mustAny/その他] を指定します (初期値:無指定) 選択必須入力 (どれかひとつ必須) を表す色 [true/mustAny/その他] を指定します (初期値:無指定) 【TAG】JavaScript などの HTML基本タグ以外の属性を、そのままタグとして使用します JavaScript などの HTML基本タグ以外の属性を、そのままタグとして使用します 【TAG】コンボボックスの値なしのOptionの有無 [true/false] を指定します (初期値:falseと同等) コンボボックスの値なしのOptionの有無 [true/false] を指定します (初期値:falseと同等) 【TAG】SQL文を指定してラベル情報を動的に読み込むかどうか [true/false] 指定します (初期値:false) SQL文を指定してラベル情報を動的に読み込むかどうか [true/false] 指定します (初期値:false) 【TAG】キャッシュする場合のスコープ [request/page/session/applicaton] を指定します (初期値:session) キャッシュする場合のスコープ [request/page/session/applicaton] を指定します (初期値:session) 【TAG】(通常使いません) sessionから所得する DBTableModelオブジェクトの ID (通常使いません) sessionから所得する DBTableModelオブジェクトの ID 【TAG】データベースへ接続する場合のID データベースへ接続する場合のID 【TAG】このタグ自体を利用するかどうかの条件キーを指定します (初期値:null) このタグ自体を利用するかどうかの条件キーを指定します (初期値:null) 【TAG】このタグ自体を利用するかどうかの条件値を指定します (初期値:null) このタグ自体を利用するかどうかの条件値を指定します (初期値:null) 【HTML】要素の内容と他の属性値の言語 (lang.xml:lang) を指定します 要素の内容と他の属性値の言語 (lang.xml:lang) を指定します 【HTML】この要素に対して適用させるスタイルシート (style) を設定します この要素に対して適用させるスタイルシート (style) を設定します 【HTML】要素に対して class 属性を設定します 要素に対して class 属性を設定します 【HTML】JavaScriptのイベント onClick を設定します (例:onClick="renew(' query. jsp', ' QUERY');") JavaScriptのイベント onClick を設定します (例:onClick="renew(' query. jsp', ' QUERY');") 【HTML】JavaScriptのイベント onChange を設定します (例:onChange="renew(' query. jsp', ' QUERY');") JavaScriptのイベント onChange を設定します (例:onChange="renew(' query. jsp', ' QUERY');") 【HTML】JavaScriptのイベント onBlur を設定します (例:onBlur="this. value=value. toUpperCase();") JavaScriptのイベント onBlur を設定します (例:onBlur="this. value=value. toUpperCase();") 【HTML】JavaScriptのイベント onFocus を設定します JavaScriptのイベント onFocus を設定します 【HTML】JavaScriptのイベント ondblClick を設定します JavaScriptのイベント ondblClick を設定します 【HTML】JavaScriptのイベント onMouseDown を設定します JavaScriptのイベント onMouseDown を設定します 【HTML】JavaScriptのイベント onMouseUp を設定します JavaScriptのイベント onMouseUp を設定します 【HTML】JavaScriptのイベント onMouseMove を設定します JavaScriptのイベント onMouseMove を設定します 【HTML】JavaScriptのイベント onMouseOut を設定します JavaScriptのイベント onMouseOut を設定します 【HTML】JavaScriptのイベント onMouseOver を設定します JavaScriptのイベント onMouseOver を設定します 【HTML】JavaScriptのイベント onSelect を設定します JavaScriptのイベント onSelect を設定します 【HTML】JavaScriptのイベント onKeyDown を設定します JavaScriptのイベント onKeyDown を設定します 【HTML】JavaScriptのイベント onKeyPress を設定します JavaScriptのイベント onKeyPress を設定します 【HTML】JavaScriptのイベント onKeyUp を設定します JavaScriptのイベント onKeyUp を設定します 【TAG】イベントカラム (親カラム) のセット イベントカラム (親カラム) のセット 【TAG】eventColumn利用時に呼ばれるJSPのURL eventColumn利用時に呼ばれるJSPのURL 【TAG】eventColumn実行後のcallback関数指定 eventColumn実行後のcallback関数指定 【TAG】デバッグ情報を出力するかどうか [true/false] を指定します (初期値:false) デバッグ情報を出力するかどうか [true/false] を指定します (初期値:false) ●使用例: ・必須表示の場合 <og:columnEditor column="NOSYN" must="true" /> ・DBからのコンボボックスの場合 <og:columnEditor column="CDBK" editor="DBMENU"> select CDBK, CDBK '':' NMBK from DB02 order by 1 </og:columnEditor> ・親子関係のプルダウンを作る場合 ([@xxx. value] で同じ行のxxxカラムで選択した値を取得します) <og:columnEditor column="CDBK" editor="DBMENU" eventColumn="OYA"> select CDBK, CDBK '':' NMBK from DB02 where CLM = [@OYA. value] order by 1 </og:columnEditor>
1.column	【TAG】カラム名をセットします。 カラム名をセットします。 columns と同時にはセットできません。 値に"*"を指定することで、全カラムを対象とすることができます。 また、"PN*", CDK*"等のように前方一致でのワイルドカード指定も可能です。
2.columns	【TAG】カラム名をカンマ区切りで複数セットします。 カラム名をカンマ区切りで複数セットします。 column と同時にはセットできません。 分解方法は、通常のパラメータ取得後に、CSV分解します。 値に"*"を指定することで、全カラムを対象とすることができます。 また、"PN*", CDK*"等のように前方一致でのワイルドカード指定も可能です。
3.label	【TAG】ラベルリソースに定義されたラベル I D をセットします。
4.className	【TAG】カラムのクラス名 (VARCHAR2, NUMBER など) をセットします。
5.size	【TAG】カラムの表示サイズをセットします (大ききだけで入力文字制限ではありません)。
6.maxLength	【TAG】カラムの入力文字制限数をセットします。
7.writable	【TAG】カラムが書き込み可能かどうか [true/false] をセットします (初期値:null)。 カラムが書き込み可能かどうか [true/false] をセットします (初期値:null)。
8.renderer	【TAG】カラムの表示用レンダラー (Renderer_****) をセットします。 これは、plugin. column 以下の Renderer_**** クラスの **** を 与えます。これらは、CellRenderer インターフェースを継承したサブクラスです。 属性クラス定義の [link org. opengion. hayabusa. db. CellRenderer Rendererer] を参照願います。 【TAG】カラムの編集用エディター (Editor_****) をセットします。

バージョン タグ名 HTML5 BODY要素	内容															
	形式サンプル															
9.editor	これは、plugin.column 以下の Editor_**** クラスの **** を与えます。これらは、CellEditor インターフェースを継承したサブクラスです。 属性クラス定義の [@link org.opengion.hayabusa.db.CellEditor Editor] を参照願います。 【TAG】DBタイプ(DBType_****)をセットします。															
10.dbType	これは、org.opengion.hayabusa.db.column 以下の DBType_**** クラスの **** を与えます。これらは、DBType インターフェースを継承したサブクラスです。 属性クラス定義の [@link org.opengion.hayabusa.db.DBType DBType] を参照願います。 【TAG】コードセレクトのユニークキー(コード名称)をセットします。															
11.codeName	このキーを元に、CodeSelectionオブジェクトを構築します。 【TAG】コードセレクトのサブセットとなるコードをCSV形式で指定します。															
12.codeList	この値に限定された、コードリソースを、既存の、CodeSelection のサブセットとして作成します。 codeName で指定されたリソースでも、サブセットを作成する事ができます。															
13.codeGroup	【TAG】コードセレクトのサブセットとなるグループを指定します。 この値に限定された、コードリソースを、既存の、CodeData のサブセットとして作成します。 codeName で指定されたリソースでも、サブセットを作成する事ができます。															
14.defaultVal	【TAG】初期値をセットします。															
15.parameter	初期値をセットします。 【TAG】パラメータをセットします。 ここで指定したパラメータは、renderer、editor、dbType すべてのパラメータにセットされます。 パラメータをセットします。															
16.roles	【TAG】ロールをセットします。 ここで指定したカラムロールを元に、ユーザー毎のアクセス許可がチェックされます。 アクセス許可されないと、表示されません。 このロールを指定しない場合は、カラムリソースのロールが使用されます。															
17.must	【TAG】必須入力を表す色に変えるかどうか[true/false]を指定します(初期値:false)。 初期値は、必須でない("false") です。															
18.mustAny	<og:input name="PN" must="true" /> 【TAG】選択必須入力(どれかひとつ必須)を表す色[true/mustAny/その他]を指定します(初期値:無指定)。 複数のカラムのうち、どれかひとつを必須とする選択必須入力を示す色を指定します。 true または、mustAny を設定すると、class属性に、mustAny がセットされます。 mustAny は、CSSファイルに初期設定されています。true または、mustAny 以外の値をセットすると、その値がそのまま、class属性にセットされますので、選択必須のグループ化が可能です。 なお、実際の選択必須入力チェックは、ここではなく、columnCheck タグで指定が必要です。 自動処理は、mustAny="true" 指定の場合のみ有効です。 初期値は、無指定です。															
19.optionAttributes	【TAG】JavaScript などの HTML基本タグ以外の属性を、そのままタグとして使用します。 JavaScript などの HTML基本タグ以外の属性を、そのままタグとして使用します。そのときに、() は、()に置き換えます。															
20.addNoValue	【TAG】コンボボックスの値なしのOptionの有無[true/false]を指定します(初期値:falseと同等)。 コンボボックスの場合、値なしのOptionを選択に含めるかどうかを指定します。 ここでは、検索結果の登録画面に使用されるケースが多いため、なにも指定されない場合は、なにもセットしません。(Coderesourceの指定どおり) 動作としては、初期値は、含めない("false") と同等です。															
21.useLabelMap	<og:columnEditor column="CDBK" addNoValue="false" /> 【TAG】SQL文を指定してラベル情報を動的に読み込むかどうか[true/false]指定します(初期値:false)。 trueを指定すると、body部分に記述されたSQL文よりラベル情報を生成して、動的にラベル表示を切り替えます。 ワイルドカードを使用した、カラムの複数指定と組み合わせることで、外部から設定されたラベル情報で一括で変更することができます。 発行するクエリでは、第1カラムにラベルキーを、第2カラムにラベル名称を設定します。 第3カラムが存在する場合は、名称(短)として使用されます。(必須ではありません) ここで作成されたラベル情報は、カラムエディターを適用したDBTableModelのみで有効であり、内部的にもキャッシュされません。 初期値は、false(動的読込しない)です。															
22.scope	【TAG】キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)。 "request","page","session","applicaton" が指定できます。 JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、主にテーブルモデルを管理するスコープを指します。 <table><tr><th>スコープ</th><th>変数の有効範囲</th><th></th></tr><tr><td>page</td><td>JSPページ内</td><td>そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会</td></tr><tr><td>request</td><td>HTTPリクエスト</td><td>リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ</td></tr><tr><td>session</td><td>HTTPセッション</td><td>初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。内部的には、同じキーワー</td></tr><tr><td>application</td><td>Webアプリケーション</td><td>ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。</td></tr></table>	スコープ	変数の有効範囲		page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会	request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ	session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。内部的には、同じキーワー	application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。
スコープ	変数の有効範囲															
page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会														
request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ														
session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。内部的には、同じキーワー														
application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。														
23.tableId	【TAG】(通常使いません)sessionから所得する DBTableModelオブジェクトの ID。 表示処理後に、(内部ポインタを書き換えた)DBTableModelオブジェクトを同じキーで、sessionに登録します。 初期値は、HybsSystem.TBL_MDLY_KEY です。															
24.dbid	【TAG】データベースへ接続する場合のID。 データベース接続を使用するエディター、レンダラーを使用する場合に使用する接続先IDを指定します。 カラムオブジェクトは、query処理ではなく、view処理時に適用されるため、データベース接続先を指定する必要があります。 初期値は、null(DEFAULT 接続) です。															
25.caseKey	【TAG】このタグ自体を利用するかどうかの条件キーを指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。															
26.caseVal	【TAG】このタグ自体を利用するかどうかの条件値を指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。															
27.lang	【HTML】要素の内容と他の属性値の言語(lang.xml:lang)を指定します。 HTMLの言語属性に使われます。指定する値は、ISO 639で規定されている「言語コード」です。 [ja/en/zh/...]などのほかに、en-US:アメリカ英語、en-cockney:コックニー英語 など、副言語を指定する方法も定められています。 ここでは、lang と xml:lang の両方に同じ値がセットされます。 タグの language 属性とは使用用途が異なります。lang セット															
28.style	【HTML】この要素に対して適用させるスタイルシート(style)を設定します。 タグにstyle属性を設定します。これは、キー:値 のセットを複数記述できます。 通常は、class属性や、id属性で登録しておき、&lg:style type="text/css"> で															

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
29.clazz	<p>外部から指定する方がソースは読みやすくなります。</p> <p>【HTML】要素に対して class 属性を設定します。</p> <p>Javaの言語使用上、class で作成できないため、代用として clazz を使用しています。</p> <p>html で作成される属性は、 class で作成されます。</p>
30.onClick	<p>【HTML】JavaScriptのイベント onClick を設定します(例:onClick="renew('query.jsp','QUERY');")。</p> <p>onClick をセットします。</p> <p>例えば、<og:column name="KBSAKU" onClick="renew('query.jsp','QUERY');"/> のように指定することで、プルダウンメニューの絞り込み検索が可能になります。</p>
31.onChange	<p>【HTML】JavaScriptのイベント onChange を設定します(例:onChange="renew('query.jsp','QUERY');")。</p> <p>onChange をセットします。</p> <p>例えば、<og:column name="KBSAKU" onChange="renew('query.jsp','QUERY');"/> のように指定することで、プルダウンメニューの絞り込み検索が可能になります。</p>
32.onBlur	<p>【HTML】JavaScriptのイベント onBlur を設定します(例:onBlur="this.value=value.toUpperCase();")。</p> <p>onBlur は、フォーカスが離れたときに発生するイベントです。</p>
33.onFocus	<p>【HTML】JavaScriptのイベント onFocus を設定します。</p> <p>onFocus は、フォーカスされたときに発生するイベントです。</p>
34.ondblClick	<p>【HTML】JavaScriptのイベント ondblClick を設定します。</p> <p>ondblClick は、マウスでダブルクリックされたときに発生するイベントです。</p>
35.onMouseDown	<p>【HTML】JavaScriptのイベント onMouseDown を設定します。</p> <p>onMouseDown は、マウスダウンされたときに発生するイベントです。</p>
36.onMouseUp	<p>【HTML】JavaScriptのイベント onMouseUp を設定します。</p> <p>onMouseUp は、マウスアップされたときに発生するイベントです。</p>
37.onMouseMove	<p>【HTML】JavaScriptのイベント onMouseMove を設定します。</p> <p>onMouseMove は、マウスが移動されたときに発生するイベントです。</p>
38.onMouseOut	<p>【HTML】JavaScriptのイベント onMouseOut を設定します。</p> <p>onMouseOut は、マウスが離れたときに発生するイベントです。</p>
39.onMouseOver	<p>【HTML】JavaScriptのイベント onMouseOver を設定します。</p> <p>onMouseOver は、マウスが重なったときに発生するイベントです。</p>
40.onSelect	<p>【HTML】JavaScriptのイベント onSelect を設定します。</p> <p>onSelect は、テキストフィールド/テキストエリアのテキストが選択されたときに発生するイベントです。</p>
41.onKeyDown	<p>【HTML】JavaScriptのイベント onKeyDown を設定します。</p> <p>onKeyDown は、キーが押されたときに発生するイベントです。</p>
42.onKeyPress	<p>【HTML】JavaScriptのイベント onKeyPress を設定します。</p> <p>onKeyPress は、キーが押され続けていたときに発生するイベントです。</p>
43.onKeyUp	<p>【HTML】JavaScriptのイベント onKeyUp を設定します。</p> <p>onKeyUp は、キーが押された状態から離されたときに発生するイベントです。</p>
44.eventColumn	<p>【TAG】イベントカラム(親カラム)のセット。</p> <p>親子関係を持った項目の動的な入れ替えを行う場合に利用します。</p> <p>このカラムを変化させるトリガとなるカラムの名前をセットします。</p> <p>※詳細はcolumnTagのeventURL属性の説明をご覧ください。</p> <p>なお、columnEditorで、Viewに対して、SQL文の定義(置き換え定義)をする場合でも、他カラムの値の参照は、columnTagの場合と同様に、{@XXXX}または\${n}の変数により行います。</p> <p>この場合、これらの変数は、“自身と同一行のカラム”を意味します。</p>
45.eventURL	<p>【TAG】eventColumn利用時に呼ばれるJSPのURL。</p> <p>イベントカラム指定時に部品を作成するJSPを指定します。</p> <p>初期値はシステムリソースのEVENT_COLUMN_URLです。</p> <p>(例:eventURL="makeColumn_custom.jsp")</p> <p>※詳細はcolumnTagのeventURL属性の説明をご覧ください。</p>
46.eventCallback	<p>【TAG】eventColumn実行後のcallback関数指定。</p> <p>eventColumnの動作終了後に実行するCallback関数の指定を行います。</p> <p>関数名のみを指定して下さい。</p> <p>ex) sampleCallback() を実行する場合はeventCallback="sampleCallback"</p> <p>実行する関数の第一引数には自動的に次の要素が入った配列が渡されます</p> <p>[0] 変更された子カラムのID属性値</p> <p>[1] evnetColumn実行後の子カラムの値</p> <p>[2] eventColumn実行前の子カラムの値</p>
47.debug	<p>この属性は、optionAttributesへの設定と同様の動作を行います。</p> <p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。</p> <p>デバッグ情報を [true:出力する/false:しない]を指定します。</p> <p>出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:columnMarker	【TAG】 マーク処理を行うカラム名を指定します。 このカラム名のTableModelに対して、マークを処理します。
JSP	
●形式 : <og:columnMarker column="..." ... > ... </og:columnMarker >	
●body : あり	
●Tag定義 :	
<og:columnMarker	【TAG】 マーク処理を行うカラム名を指定します
column	【TAG】 マーク処理を行うカラム名をカンマ区切り (CSV形式) で複数指定します
columns	【TAG】 マークを処理する (true or 1) / 処理しない (false or 0) の設定を指定します (初期値: true)
onMark	【TAG】 処理するマークを含むような文字列を、" "区切りの文字列で指定します (初期値: "true TRUE 1")
markList	【TAG】 スペースで区切られた複数の値について、マークします
instrVals	【TAG】 デバッグ情報を出力するかどうか [true/false] を指定します (初期値: false)
debug	
> ... Body ...	
</og:columnMarker>	
●使用例	
商品CD 2 (CDSYHN02)は赤字で表示する。	
商品CD 3 (CDSYHN03)は——と表示する。	
<og:viewMarker command="@command">	
<og:columnMarker column="CDSYHN02" onMark="true" >	
[VCDSYHN02]	
</og:columnMarker>	
<og:columnMarker column="CDSYHN03" onMark="[XXXX]" markList="[YYYY]" >	
<CENTER><——</CENTER>	
</og:columnMarker>	
</og:viewMarker>	
1.column	【TAG】 マーク処理を行うカラム名を指定します。 このカラム名のTableModelに対して、マークを処理します。
2.columns	【TAG】 マーク処理を行うカラム名をカンマ区切り (CSV形式) で複数指定します。 この複数のカラム名のTableModelに対して、マークを処理します。 カラム名は、カンマ区切りで複数指定することができます。その場合は、 指定のカラムに対して、すべて同一の処理を行います。 先に配列に分解してからリクエスト変数の値を取得するようにします。 こうする事で、リクエストにカンマ区切りの値を設定できるようになります。 分解方法は、通常のパラメータ取得後に、CSV分解します。
3.onMark	【TAG】 マークを処理する (true or 1) / 処理しない (false or 0) の設定を指定します (初期値: true)。 マークを処理する場合は、"true" (または "1") 処理しない場合は、"true以外" (または "0") をセットします。 初期値は、"true" (マークを処理する) です。 さらに、[カラム名] で、動的にカラムの値で、マークをする、しないを 選ぶ事が可能になります。値は、"true" (または "1") で、マークします。 追記 3.5.2.0 (2003/10/20) : markList属性に、処理対象文字列部を指定できます。 これにより、マークを処理するかどうかの判断が、true, 1 以外にも使用できるようになりました。
4.markList	【TAG】 処理するマークを含むような文字列を、" "区切りの文字列で指定します (初期値: "true TRUE 1")。 markListで与えられた文字列に、onMark文字列 (大文字/小文字の区別あり) が 含まれていれば、処理します。 例えば、"A","B","C" という文字列が、onMark で指定された 場合に処理するようにしたい場合は、"A B C" をセットします。 markList には、[カラム名] 指定が可能です。 初期値は、"true TRUE 1" です。
5.instrVals	【TAG】 スペースで区切られた複数の値について、マークします。 最終的に作成された表示結果に対して、引数の文字列を含む箇所に、 引数文字列 文字列と置き換えます。 0 の部分は、引数文字列の現れた順番を指定します。 これにより、部分検索された箇所のみにマークすることが可能です。 部分文字列は、スペースで区切り、それぞれ独立した形でマーク されますので、検索エンジン等で検索したキャッシュ表示のような 効果を持たすことが可能になります。
6.debug	【TAG】 デバッグ情報を出力するかどうか [true/false] を指定します (初期値: false)。 デバッグ情報を [true:出力する/false:しない] を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
<div>5.5.4.2</div> <div>og:columnSet</div> <div>Tag</div> <div>●形式：<og:entry command="..."> <og:setColumn command ="..." : entryタグのcommandと一致する場合に実行 columnId ="..." : 処理を行うカラムIDを指定(必須、複数指定可) action ="..." : 処理方法を指定 value ="..." : 設定する値を指定(action="TBLSET"のみ[カラム名]使用可能) conditionKey ="..." : 条件判定するカラムIDを指定(初期値は columnId) conditionList ="..." : 条件判定する値のリストを、" "で区切って登録(初期値は、無条件) /> </og:entry> ●body：なし ●Tag定義： <og:columnSet command [TAG] コマンド(INSERT、COPY、MODIFY、DELETE、ENTRY、CHANGE、RESET、ALLACTION)を設定します columnId ○ [TAG] カラムIDをセットします(複数指定時は、カンマ区切り文字で設定する)。(必須) value [TAG] 値をセットします action [TAG] アクション(DEFAULT/CREAR/ADD/SET/NULLSET/LOWER/UPPER/COPY/TBLSET/WRTCTRL/DBMENU/REQSET)をセットします conditionKey [TAG] 条件判定するカラムIDを指定します(初期値は columnId) conditionList [TAG] 条件判定する値のリストを、" "で区切って登録します(初期値:無条件) debug [TAG] デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false) /> ●使用例 <og:entry command="@command]" "> <og:columnSet command="@command]" columnId="ECNO" action="CLEAR" /> <og:columnSet command="@command]" columnId="JYOKYO" action="SET" value="1" /> </og:entry> <og:entry command="MODIFY" rows="1" > <og:columnSet command="MODIFY" columnId="key" action="TBLSET" value ="[key][lang]"/> </og:entry> command属性 は、entryタグのcommand属性と同一の場合のみ、処理します。 [command属性] INSERT 新規 COPY 複写 MODIFY 変更 DELETE 削除 ENTRY エントリー CHANGE チェンジ RESET リセット ALLACTION オールアクション [action属性] DEFAULT カラムリソースで定義した初期値をセットします。 CLEAR 値をクリア(ゼロストリング "")します。 ADD 現在の値を +1 します。 0 ⇒ 1, A ⇒ B, 9 ⇒ 10 SET value で設定した値を 新しい値として登録します。 NULLSET 元の値が NULL の場合だけ、value で設定した新しい値を登録します。 LOWER 小文字に変換します。 UPPER 大文字に変換します。 COPY value にコピー元のカラムIDをセットすれば、その値を代入します。 TBLSET DBTableModel の内容を取り込んで指定の columnId カラムに設定します。 [カラム名] で指定できます。 また、これは文字列を解析して、value を作成しますので、文字列連結等に使用できます。 WRTCTRL writableControl を使用したカラムデータの先頭アンダーバーを削除します。 DBMENU DBMENUでパラメータ設定(コロン連結文字)を使用したカラムデータの先頭データのみにします。 REQSET valueで指定したカラムの値をキーに、リクエスト変数から値を取出し、セットします。 その他 カラムのDBType の valueAction メソッドを呼び出します。自由に設定可能です。</div>	<div>【TAG】 コマンド(INSERT、COPY、MODIFY、DELETE、ENTRY、CHANGE、RESET、ALLACTION)を設定します。 上位の EntryTag のコマンドと一致した場合のみ、処理されます。 ここでは、コマンド文字列の整合性チェックは行っていない。</div> <div>【TAG】 カラムIDをセットします(複数指定時は、カンマ区切り文字で設定する)。 処理を適用するカラムIDを指定します。 カラム名とカラムNoは同時には登録できません。 全く同じ内容を設定する場合には、カラム名をCSV形式(カンマ区切り)で 複数指定することが可能です。 【TAG】 値をセットします。 設定する値をセットします。 【TAG】 アクション(DEFAULT/CREAR/ADD/SET/NULLSET/LOWER/UPPER/COPY/TBLSET/WRTCTRL/DBMENU/REQSET)をセットします。 ここでは、アクション文字列の整合性チェックを行っておりません。</div> <div>DEFAULT カラムリソースで定義した初期値をセットします。 CREAR 値をクリア(ゼロストリング "")します。 ADD 現在の値を +1 します。 0 ⇒ 1, A ⇒ B, 9 ⇒ 10 SET value で設定した値を 新しい値として登録します。 NULLSET 元の値が NULL の場合だけ、value で設定した新しい値を登録します。 LOWER 小文字に変換します。 UPPER 大文字に変換します。 COPY value にコピー元のカラムIDをセットすれば、その値を代入します。 TBLSET DBTableModel の内容を取り込んで指定の columnId カラムに設定します。[カラム名] で指定できます。 WRTCTRL writableControl を使用したカラムデータの先頭アンダーバーを削除します。 DBMENU DBMENUでパラメータ設定(コロン連結文字)を使用したカラムデータの先頭データのみにします。 REQSET valueで指定したカラムの値をキーに、リクエスト変数から値を取出し、セットします。 その他 カラムのDBType の valueAction メソッドを呼び出します。自由に設定可能です。</div> <div>【TAG】 条件判定するカラムIDを指定します(初期値は columnId)。 指定のカラムIDの値と、conditionList の値を比較して、 存在する場合は、action処理を実行します。 例えば、初期値設定(action="DEFAULT")で、値が "0" の場合だけ適用するような ケースに使用できます。 指定しない場合は、columnId と同じキーになります。 注意：columnId は、CSV形式で複数同時に指定できます。その場合は、 conditionKey には、複数設定する機能はありませんので、全カラムに 同一カラムを使用する場合は、conditionKey に値を設定できます。 値を設定しない場合は、自動的に、columnId と同一になります。</div> <div>【TAG】 条件判定する値のリストを、" "で区切って登録します(初期値:無条件)。</div>

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
6.conditionList	conditionKey とペアで指定します。ここには、カラムの設定値のリストを指定することで、複数条件(OR結合)での比較を行い、リストにカラム値が存在する場合のみ、action処理を実行します。 設定しない場合は、無条件に実行します。
7.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:comment	【TAG】 コメントの設計変更要求コードを設定します。 コメントの設計変更要求コードを設定します。
JSP	
●形式 : <og:comment title="..." version="..." date="..." author="..." ... />	
●body : あり	
●Tag定義 :	
<og:comment	
rdcCode	【TAG】 コメントの設計変更要求コードを設定します
title	○ 【TAG】 コメントのタイトルを設定します。(必須)
version	○ 【TAG】 コメントのバージョンを設定します。(必須)
build	【TAG】 コメントのビルド番号を設定します
date	○ 【TAG】 コメントの変更日付を設定します。(必須)
author	○ 【TAG】 コメントの変更者を設定します。(必須)
text	【TAG】 コメントの内容を設定します
debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)
pgid	【TAG】 コメントのプログラムIDを設定します
system	【TAG】 コメントのシステム名称を設定します
> ... Body ...	
</og:comment>	
●使用例	
<og:comment	
title = "変更概要"	
version = "001"	
date = "YYYY/MM/DD"	
author = "Y. MATSUI (Hybs) "	
rdcCode = "RDC00001"	
text = "当処理中の変更内容を記述します。"	
/>	
<og:comment	
title = "変更概要"	
version = "001"	
date = "YYYY/MM/DD"	
author = "Y. MATSUI (Hybs) "	
rdcCode = "RDC00001"	
build = "ビルド番号" (必須属性では有りません。)	
>	
当処理中の変更内容を記述します。	
</og:comment>	
1.rdcCode	【TAG】 コメントの設計変更要求コードを設定します。 コメントの設計変更要求コードを設定します。
2.title	【TAG】 コメントのタイトルを設定します。 コメントのタイトルを設定します。
3.version	【TAG】 コメントのバージョンを設定します。 コメントのバージョンを設定します。
4.build	【TAG】 コメントのビルド番号を設定します。 コメントのビルド番号を設定します
5.date	【TAG】 コメントの変更日付を設定します。 コメントの変更日付を設定します
6.author	【TAG】 コメントの変更者を設定します。 コメントの変更者を設定します
7.text	【TAG】 コメントの内容を設定します。 コメントの内容を設定します。 ここでの内容は、BODY部に記述することも可能です。 両方に記述した場合は、text属性を優先します。
8.debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。
9.pgid	【TAG】 コメントのプログラムIDを設定します。 コメントのプログラムIDを設定します。
10.system	【TAG】 コメントのシステム名称を設定します。 コメントのシステム名称を設定します。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:commonForward Tag ●形式 : <og:commonForward /> ●body : なし ●Tag定義 : <og:commonForward dbkeys 【TAG】 DBキーをCSV 形式でセットします tableId 【TAG】 (通常使いません)sessionから所得する DBTableModelオブジェクトの ID debug 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false) useRedirectCheck 【TAG】 非選択状態の場合にforwardを許可するかどうか[true/false]を指定します(初期値:true) /> ●使用例 フォワードキャッシュによりページ転送します。 <og:commonForward dbkeys="@dbkeys" />	【TAG】 DBキーをCSV 形式でセットします。 URI の引数にセットするキーを CSV 形式でセットします。 この指定は, DBTableModel 上のデータを取り込みます。
1.dbkeys	【TAG】 DBキーをCSV 形式でセットします。 URI の引数にセットするキーを CSV 形式でセットします。 この指定は, DBTableModel 上のデータを取り込みます。
2.tableId	【TAG】 (通常使いません)sessionから所得する DBTableModelオブジェクトの ID。 表示処理後に、(内部ポインタを書き換えた)DBTableModelオブジェクトを 同じキーで、sessionに登録します。 初期値は、HybsSystem.TBL MDL KEY です。
3.debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。
4.useRedirectCheck	【TAG】 非選択状態の場合にforwardを許可するかどうか[true/false]を指定します(初期値:true)。 初期値はtrueが設定されています falseにすると許可されます

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

5.5.4.2
og:cookie

Tag

●形式：
<og:cookie
 action = "SAVE"
 keys = "AAA, BBB"
 vals = "VAL1, VAL2"
 path = "/ge"
 domain = ".foo.com"
 maxAge = "3600"
 useBase64 = "false"
>
●body：なし
●Tag定義：
<og:cookie
 action
 keys
 vals
 aliasNames
 path
 domain
 maxAge
 useBase64
 debug
>
●使用例
 例 1) 設定:複数キーを同時に書き込むことが可能です。
 <og:cookie
 action="SAVE" keys="CDJ, FG_NAME" vals="@CDJ, {@NAME}"
 />
 例 2) 取得:cookieタグで取得すると、それ以降では {@CDJ} や {@NAME} で扱えます。
 aliasNames 属性を使わない場合は、keys に指定した変数にセットされます。
 <og:cookie
 action="LOAD" keys="CDJ, FG_NAME" aliasNames="CDJ, NAME"
 />
 例 3) 取得:SYS パラメータでの取得も可能です。
 {@SYS.COOKIE.CDJ}
 例 4) QUERY画面では値の表示(LOAD)を行い、RESULT画面で値の設定(SAVE)を行うケース

 QUERY画面
 <og:cookie action="LOAD" useBase64="true"
 keys="CLM, NAME" aliasNames="CLM, LABEL_NAME" />

 <og:column name="CLM" defaultVal="@CLM" />
 <og:column name="LABEL_NAME" defaultVal="@LABEL_NAME"/>

 RESULT画面
 <og:cookie action="SAVE" maxAge="360000" useBase64="true"
 keys="CLM, NAME" vals="@CLM, {@LABEL_NAME}" />

 例 5) QUERY画面では、{@SYS.COOKIE.カラム名} で取得。
 RESULT画面では、ムラタ内全システム共通に使える値をセット。

 QUERY画面
 <og:column name="SYSTEM_ID" defaultVal="@SYS.COOKIE.SYSTEM_ID" />

 RESULT画面
 <og:cookie action="SAVE" maxAge="360000" domain="opengion.org"
 keys="SYSTEM_ID" vals="@SYSTEM_ID" />

1.action

2.keys

3.vals

4.aliasNames

【TAG】アクション(SAVE, LOAD, DELETE)をセットします。
アクションは、HTMLから(get/post)指定されますので、ACT_xxx で設定される
フィールド定数値のいつれかを、指定できます。
無指定の場合は、なにもしません。

	action	名称	機能
SAVE	登録	指定の keys のキーに vals の値をセットします。	
LOAD	取得	指定の keys のクッキーを(リクエスト中に)取得します。	
DELETE	削除	指定の keys のクッキーを削除します。	

【TAG】クッキーのキーをCSV形式で複数指定します。
クッキーにセットするときのキーを指定します。カンマ区切りで複数指定できます。
vals 属性には、キーに対応する値を、設定してください。
分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。
こうしないとデータ自身にカンマを持っている場合に分解をミスる為です。

【TAG】keys属性に対応する値をCSV形式で複数指定します。
キーに設定した値を、カンマ区切り文字で複数して出来ます。
指定順序は、キーと同じにしておいて下さい。
分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。
こうしないとデータ自身にカンマを持っている場合に分解をミスる為です。

【TAG】クッキーのキーの別名をCSV形式で複数指定します。
クッキーから値を取得する(action="LOAD")場合に、読み込みキー(keys)に対応する
別名を指定することで、別名の変数に読み込んだ値を登録することが出来ます。
別名を指定しない場合は、keys に指定された名前が、使用されます。
分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。
こうしないとデータ自身にカンマを持っている場合に分解をミスる為です。

【TAG】クライアントがこの Cookie を返さなくてはいけないパスを指定します(初期値: /+CONTEXT_NAME)。

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
5.path	<p>この値を指定すると Cookie が該当するディレクトリ内、さらに、サブディレクトリに存在する全てのページから参照できるようになります。Cookie のパスには Cookie をセットした Servlet が含まれていなければなりません。例えば、/catalog を指定したとします。このとき、サーバの /catalog 以下の全てのディレクトリから Cookie が見えるようになります。初期値は、"/" + CONTEXT_NAME です。</p>
6.domain	<p>Cookie のパス名指定についての詳細は RFC2109 を参照してください。</p> <p>【TAG】この Cookie がどこで生成されたかを表すドメインを指定します(初期値:付与したサーバ)。</p> <p>ドメイン名の形式は RFC2109 で指定されています。</p> <p>ドメイン名は (.foo.com のように) ドットで始まります。このように設定すると、Cookie は指定された Domain Name System (DNS) のゾーン内のサーバから見えるようになります(例えば、www.foo.com) からは見えるけれど、a.b.foo.com からは見えないというようにです)。デフォルトでは Cookie を付与したサーバにしか送り返しません。</p>
7.maxAge	<p>【TAG】Cookie の最長存続期間を秒単位で設定します(初期値: -1)。</p> <p>正の値が指定されると Cookie はある秒数が過ぎた後、削除されます。この値は、Cookie の有効期限が切れる 最長 存続期間であることに注意してください。Cookie の現在までの存続期間ではありません。</p> <p>負の値は Cookie が永続的に保存されないことを意味しています。この場合、Web ブラウザが終了すると Cookie も削除されます。0 という値を指定すると Cookie が削除されることになります。</p> <p>初期値は、-1(永続的に保存されない)です。</p>
8.useBase64	<p>【TAG】漢字等の文字を扱う場合に、BASE64で処理を行うかどうか[true/false]を設定します(初期値:false)。</p> <p>クッキーへの読み書きは、ASCII に限られます。漢字等のコードを書き込む場合は、BASE64でエンコードして書き込む必要があります。読み込む場合も同様です。ただし、一般のASCIIは、BASE64 ではエンコードしないため、外部で指定する必要があります。BASE64 で書き込んだ場合は、[@SYS.COOKIE.CDJ] での取得はできませんので、action="LOAD" で、取得してください。</p> <p>初期値は、false(使用しない)です。</p>
9.debug	<p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。</p> <p>デバッグ情報を [true:出力する/false:しない]を指定します。</p> <p>出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
<p>5.5.4.2 og:crossParam</p> <p>Tag</p> <p>●形式 : <og:crossParam breakColumn="ZZZ" noGroupColumns="AAA, BBB" sumNumber="2" /></p> <p>●body : なし</p> <p>●Tag定義 :</p> <pre><og:crossParam cubeXColumn 【TAG】 CUBE計算の 1 つ目 (X) カラムを指定します cubeYColumn 【TAG】 CUBE計算の 2 つ目 (Y) カラムを指定します sumNumber 【TAG】 合計値のカラム数を設定します breakColumn 【TAG】 ブレークによりヘッダー部を出力させるカラム名をセットします (初期値:ヘッダーN) noGroupColumns 【TAG】 カラム値を前段と比較して同じ場合でも表示させるカラム名をセットします shokeiLabel 【TAG】 列小計のカラムに表示するラベルIDを指定します (初期値:空文字列) gokeiLabel 【TAG】 列合計のカラムに表示するラベルIDを指定します (初期値:空文字列) cubeSortType 【TAG】 CUBE Y の列ヘッダーのソート方式を指定します (初期値:LOAD) gokeiSortDir 【TAG】 合計行のソート有無とその方向[true:正方向/false:逆方向/null:ソートしない]を指定します (初期値:null) useHeaderColumn 【TAG】 ヘッダーカラムにレンダラー、エディターを適用するかを指定します (初期値:false) useClassAdd 【TAG】 各列情報のclass属性に、カラム名などを付与するかを指定します (初期値:false) firstClimGokei 【TAG】 合計列をCUBEの先頭部分に出すかどうか[true/false]を指定します (初期値:false) saveTableId 【TAG】 クロス集計結果の DBTableModel をセーブするセッションキーワードを指定します saveScope 【TAG】 クロス集計結果の DBTableModel をセーブする scope を指定します debug 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します (初期値:false) /></pre> <p>●使用例</p> <p>ViewFormTag の viewFormType が、HTMLCrossTable の場合に使用します。 useParam 属性を設定しておかないと、使用されません。</p> <pre><og:view viewFormType = "HTMLCrossTable" command = "{@command}" startNo = "0" pageSize = "20" useParam = "true" ></pre> <pre><og:crossParam breakColumn = "ZZZ" : ブレークによりヘッダー部を出力させるカラム名 noGroupColumns = "AAA, BBB" : カラム値を前段と比較して同じ場合でも表示させるカラム名 cubeXColumn = "CUBE_X" : CUBE計算の 1 つ目 (X) カラムを指定 cubeYColumn = "CUBE_Y" : CUBE計算の 2 つ目 (Y) カラムを指定 shokeiLabel = "SHOKEI" : 列小計のカラムに表示するラベルID (初期値:SHOKEI) gokeiLabel = "GOKEI" : 列合計のカラムに表示するラベルID (初期値:GOKEI) sumNumber = "2" : 合計値のカラム数 (初期値 1) cubeSortType = "NUMBER" : CUBE Y の列ヘッダーのソート方式を指定[STRING/NUMBER/LOAD] gokeiSortDir = "false" : 合計行のソート有無とその方向 (正方向/逆方向)を指定[true/false/null] firstClimGokei = "true" : 合計列をCUBEの先頭部分に出すかどうかを指定[false/true/null] useHeaderColumn= "true" : ヘッダーカラムにレンダラー、エディターを適用するかを指定[false/true/null] saveTableId = "DEFAULT" : クロス集計結果の DBTableModel をセーブするセッションキーワードを指定 useClassAdd = "true" : String 各列情報のclass属性に、カラム名などを付与するかを指定[false/true/null] /></pre> <pre></og:view ></pre>	<p>【TAG】 CUBE計算の 1 つ目 (X) カラムを指定します。 列方向のキーとなるカラム名を指定します。 初期値は、互換性の関係より、sumNumber より逆計算します。</p> <p>【TAG】 CUBE計算の 2 つ目 (Y) カラムを指定します。 行方向のキーとなるカラム名を指定します。 初期値は、互換性の関係より、sumNumber より逆計算します。</p> <p>【TAG】 合計値のカラム数を設定します。(初期値: 1)。 CROSS集計で求めたフォーマットは、『ヘッダー1..N,縦,横,計1..N』です。</p> <p>合計は、複数並べることができますが、sumNumber で指定しておく必要があります。 初期値は、1 です。</p> <p>【TAG】 ブレークによりヘッダー部を出力させるカラム名をセットします (初期値:ヘッダーN)。 CROSS集計で求めたフォーマットは、『ヘッダー1..N,縦,横,計1..N』です。</p> <p>ヘッダー部は、複数指定できますが、デフォルトではヘッダーNがキーブレイクすると 合計用のヘッダーが挿入されます。 このヘッダーNそのものが、集計フィールドでなく、単なる属性の場合は、 キーブレイクして欲しくない為、breakColumn="ヘッダーN-1" を指定します。 初期値は、“ヘッダーN” です。</p> <p>【TAG】 カラム値を前段と比較して同じ場合でも表示させるカラム名をセットします。 CROSS集計で求めたフォーマットは、『ヘッダー1..N,縦,横,計1..N』です。</p> <p>ヘッダー部は、キーブレイクする都度、ヘッダーを出力します。それまでは、 各ヘッダーの値が、前段 (同一カラムの先の値) と同じ場合は、空白にします。 こうする事で、値のグルーピングが一目で判ります。(初期設定)</p> <p>このヘッダーが、集計フィールドでなく、単なる属性の場合は、 空白ではなく、値として表示したい為、グループ化しない事を指定します。 【TAG】 列小計のカラムに表示するラベルIDを指定します (初期値:空文字列)。 各列の小計のラベルIDを登録します。登録する文字列は、ラベルリソースに 定義しておいて下さい。 初期値は、空文字列(“)です。</p> <p>【TAG】 列合計のカラムに表示するラベルIDを指定します (初期値:空文字列)。 各列の合計のラベルIDを登録します。登録する文字列は、ラベルリソースに 定義しておいて下さい。 初期値は、空文字列(“)です。</p> <p>【TAG】 CUBE Y の列ヘッダーのソート方式を指定します (初期値:LOAD)。 CUBEのヘッダーに対応するカラム列をソート表示する場合の方式を指定します。 種類として、STRING, NUMBER, LOAD, があります。 初期値 (指定無し) は、LOAD (取り込み順にセット) です。</p> <p>【TAG】 合計行のソート有無とその方向[true:正方向/false:逆方向/null:ソートしない]を指定します (初期値:null)。 最も最後の合計カラムにソートを行うかどうか、その時の方向を指定します。 true/false 以外の文字列では、ソートを行いません。trueは、正方向 (昇順) で、 falseが逆方向 (降順) になります。 初期値 (指定無し) は、ソートしない (null) です。</p> <p>【TAG】 ヘッダーカラムにレンダラー、エディターを適用するかを指定します (初期値:false)。 ヘッダーカラムにレンダラー、エディターを適用するかを指定します。 trueが指定された場合は、ヘッダー部分の値そのものをカラム名として扱います。 リソースが存在しない場合は、ラベルのみを各カラムの値で置き換えます。 初期値 (指定無し) は、レンダラー、エディターを適用しない (false) です。</p>
1.cubeXColumn	<p>【TAG】 CUBE計算の 1 つ目 (X) カラムを指定します。 列方向のキーとなるカラム名を指定します。 初期値は、互換性の関係より、sumNumber より逆計算します。</p>
2.cubeYColumn	<p>【TAG】 CUBE計算の 2 つ目 (Y) カラムを指定します。 行方向のキーとなるカラム名を指定します。 初期値は、互換性の関係より、sumNumber より逆計算します。</p>
3.sumNumber	<p>【TAG】 合計値のカラム数を設定します。(初期値: 1)。 CROSS集計で求めたフォーマットは、『ヘッダー1..N,縦,横,計1..N』です。</p> <p>合計は、複数並べることができますが、sumNumber で指定しておく必要があります。 初期値は、1 です。</p>
4.breakColumn	<p>【TAG】 ブレークによりヘッダー部を出力させるカラム名をセットします (初期値:ヘッダーN)。 CROSS集計で求めたフォーマットは、『ヘッダー1..N,縦,横,計1..N』です。</p> <p>ヘッダー部は、複数指定できますが、デフォルトではヘッダーNがキーブレイクすると 合計用のヘッダーが挿入されます。 このヘッダーNそのものが、集計フィールドでなく、単なる属性の場合は、 キーブレイクして欲しくない為、breakColumn="ヘッダーN-1" を指定します。 初期値は、“ヘッダーN” です。</p>
5.noGroupColumns	<p>【TAG】 カラム値を前段と比較して同じ場合でも表示させるカラム名をセットします。 CROSS集計で求めたフォーマットは、『ヘッダー1..N,縦,横,計1..N』です。</p> <p>ヘッダー部は、キーブレイクする都度、ヘッダーを出力します。それまでは、 各ヘッダーの値が、前段 (同一カラムの先の値) と同じ場合は、空白にします。 こうする事で、値のグルーピングが一目で判ります。(初期設定)</p>
6.shokeiLabel	<p>このヘッダーが、集計フィールドでなく、単なる属性の場合は、 空白ではなく、値として表示したい為、グループ化しない事を指定します。 【TAG】 列小計のカラムに表示するラベルIDを指定します (初期値:空文字列)。 各列の小計のラベルIDを登録します。登録する文字列は、ラベルリソースに 定義しておいて下さい。 初期値は、空文字列(“)です。</p>
7.gokeiLabel	<p>【TAG】 列合計のカラムに表示するラベルIDを指定します (初期値:空文字列)。 各列の合計のラベルIDを登録します。登録する文字列は、ラベルリソースに 定義しておいて下さい。 初期値は、空文字列(“)です。</p>
8.cubeSortType	<p>【TAG】 CUBE Y の列ヘッダーのソート方式を指定します (初期値:LOAD)。 CUBEのヘッダーに対応するカラム列をソート表示する場合の方式を指定します。 種類として、STRING, NUMBER, LOAD, があります。 初期値 (指定無し) は、LOAD (取り込み順にセット) です。</p>
9.gokeiSortDir	<p>【TAG】 合計行のソート有無とその方向[true:正方向/false:逆方向/null:ソートしない]を指定します (初期値:null)。 最も最後の合計カラムにソートを行うかどうか、その時の方向を指定します。 true/false 以外の文字列では、ソートを行いません。trueは、正方向 (昇順) で、 falseが逆方向 (降順) になります。 初期値 (指定無し) は、ソートしない (null) です。</p>
10.useHeaderColumn	<p>【TAG】 ヘッダーカラムにレンダラー、エディターを適用するかを指定します (初期値:false)。 ヘッダーカラムにレンダラー、エディターを適用するかを指定します。 trueが指定された場合は、ヘッダー部分の値そのものをカラム名として扱います。 リソースが存在しない場合は、ラベルのみを各カラムの値で置き換えます。 初期値 (指定無し) は、レンダラー、エディターを適用しない (false) です。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
11.useClassAdd	<p>【TAG】各列情報のclass属性に、カラム名などを付与するかどうかを指定します(初期値:false)。 列情報の集計列に対して、class 属性を付与するかどうかを指定します。 class属性は、その列のオリジナルの属性名と、ラベル名の文字列を設定します。 例えば、集計行の計カラムが複数ある場合は、それぞれに色を指定して、ゼブラ模様を設定できます。また、ラベル(表示ヘッダー)も設定されるので、特別な列のみ指定することも可能になります。 ※ 特殊対応：cssなどで指定できるIDやCLASS属性は、先頭文字が数字の場合は、無効になります。(つまり、効きません。) 表示ヘッダーは、年月や、社員番号(数字)などのケースもあります。そこで、先頭が数字の場合は、“x”(小文字のx)を自動的に頭に追加します。この処理は、ViewForm_HTMLCrossTableで行います。</p>
12.firstClnGokei	<p>【TAG】合計列をCUBEの先頭部分に出すかどうか[true/false]を指定します(初期値:false)。 合計列を最終列に出力するか、CUBEの先頭列に出力するかを指定します。 trueが指定された場合はCUBEの先頭列に出力します。 初期値(false)は合計列を最終列に出力します。</p>
13.saveTableId	<p>【TAG】クロス集計結果の DBTableModel をセーブするセッションキーワードを指定します。 検索のみの場合は、何も設定しません。EXCEL等外部にクロス集計の形で取り出したい場合に、設定します。 “DEFAULT” という文字列を指定すると、内部では、HybsSystem.TBL_MDL_KEY が設定されます。(DEFAULT という文字列に設定されるわけではありません。) なお、DEFAULT を使用する場合は、検索結果の DBTbleModel をつづすことになりしますので、NEXT 等は使えません。DBTableModel のデータを利用した forward 等も使用できませんので、十分ご注意ください。 DEFAULT 以外の文字列の場合は、指定した文字列そのものがキーになります。 他のセッションキーと同じにすると動作が不安定になりますので、使用する場合は、“CROSS_TABLE_SAVE_KEY” を推奨致します。 指定しない場合は、セッションにセーブされません。(検索されたまま) 通常、EXCEL出力等を行う場合は、DBTableModel をセーブする必要がありますが、scope=“session” にセーブすると、PREV,NEXT が使えなくなります。これは、クロス集計時に元のカラムが表形式の別のカラムに置き換えられるためです。 scope=“request” では、エラーは発生しなくなりますが、外部に取り出せなくなります。</p>
14.saveScope	<p>【TAG】クロス集計結果の DBTableModel をセーブする scope を指定します。 スコープは (request,page,session,applicaton) がありますが、request か session が通常選択されます。 また、この設定が有効になるには、saveTableId を指定する必要があります。 saveTableId を指定しないと、そもそも書き出されないため、scope は無視されます。 scope=“session” にセーブすると、PREV,NEXT が使えなくなります。これは、クロス集計時に元のカラムが表形式の別のカラムに置き換えられるためです。 scope=“request” では、エラーは発生しなくなりますが、外部に取り出せなくなります。 何も指定しない場合は、viewタグに指定されたオリジナルのスコープに書き出されます。 そうで無い場合は、指定のスコープに書き出されます。</p>
15.debug	<p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:dataCheck JSP	【TAG】 コマンド(NEW or ENTRY)をセットします。 コマンドは、HTMLから(get/post) 指定されますので、CMD_xxx で設定される フィールド定数値のいつれかを、指定できます。
<p>●形式：</p> <pre><og:dataCheck command = "[@command]" exist = "[auto true false one notuse]" errRemove = "[true false]" msg = "[@msg]" msgParamKeys = "ZY03" : メッセージリソースのキーをカンマ区切りで指定。[2] 以降にセット sqlType = "[@sqlType]" execType = "INSERT COPY UPDATE MODIFY DELETE" : sqlType を含む場合、実行 conditionKey = "FGJ" : 条件判定するカラム I D を指定(初期値は columnId) conditionList = "0 1 8 9" : 条件判定する値のリストを、" "で区切って登録(初期値は、無条件) uniqCheckKeys = "CLM, LANG" : DBTableModel内でのユニークキーチェックを行うためのカラム ></pre> <p>●body：SQL文 又は JavaScript式</p> <p>●Tag定義：</p> <pre><og:dataCheck command 【TAG】 コマンド(NEW or ENTRY)をセットします exist 【TAG】 データベースのチェック方法[auto/true/false/one/notuse]を指定します(初期値:auto[自動]) tableId 【TAG】 (通常は使いません)結果をDBTableModelに書き込んで、sessionに登録するときのキーを指定します dbid 【TAG】 (通常は使いません)Queryオブジェクトを作成する時のDB接続IDを指定します lbl 【TAG】 ラベルリソースIDを指定します lblParamKeys 【TAG】 ラベルリソースの引数をカンマ区切りで指定します errRemove 【TAG】 エラー時の選択行を取り除いて継続処理を行うかどうか[true/false]を指定します(初期値:false) sqlType 【TAG】 このチェックを行う、SQLタイプ を指定します execType 【TAG】 このチェックを行う、実行タイプ を指定します conditionKey 【TAG】 条件判定するカラム I D を指定します conditionList 【TAG】 条件判定する値のリストを、" "で区切って登録します(初期値:無条件) uniqCheckClims 【TAG】 指定されたキーに従って、メモリ上のテーブルに対してユニークキーチェックを行います beforeErrorJsp 【TAG】 エラーが発生した際に、エラーメッセージの表示前にincludeするJSPを指定します afterErrorJsp 【TAG】 エラーが発生した際に、エラーメッセージの表示後にincludeするJSPを指定します selectedAll 【TAG】 データを全件選択済みとして処理するかどうか[true/false]を指定します(初期値:false) msg 【廃止】 メッセージIDを指定します(lbl 属性を使用してください) msgParamKeys 【廃止】 メッセージリソースの引数をカンマ区切りで指定します(lblParamKeys 属性を使用してください) debug 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false) > ... Body ... </og:dataCheck></pre> <p>●使用例</p> <pre><og:dataCheck command = "ENTRY" exist = "true" msg = "MSG0001" > select count(*) from GEA03 where clm = [CLM] </og:dataCheck></pre> <p>・exist 属性の値に応じて、チェック方法が異なります。 [auto , true , false , one , notuse が指定できます。]</p> <pre><og:dataCheck command = "ENTRY" msg = "MSG0001" > <![CDATA[[DYSTART] < [DY] && [DY] < [DYEND]]]> </og:dataCheck></pre> <pre><og:dataCheck command = "ENTRY" msg = "MSG0001" > <![CDATA[[GOKEI] < [TANKA] * [RITU]]]> </og:dataCheck></pre>	
1.command	【TAG】 コマンド(NEW or ENTRY)をセットします。 コマンドは、HTMLから(get/post) 指定されますので、CMD_xxx で設定される フィールド定数値のいつれかを、指定できます。
2.exist	【TAG】 データベースのチェック方法[auto/true/false/one/notuse]を指定します(初期値:auto[自動])。 exist 属性に指定された、「true:存在する」、「false:存在しない」、「one:ひとつ以下」、 の値は、いずれの場合も、成立時は、正常とみなします。 「auto:自動」は、DBTableModelのmodifyType(A,C,D)に応じて、チェックします。 A,C,D は、entryタグにコマンドを渡してデータを作成したときに、内部で作成されます。 (command="NEW"の場合は、trueと同じ動きになります。) notuse は、チェックを行いません。これは、このタグを共有使用する場合に、外部で チェックを行うかどうかを指定できるようにするために使用します。 (「true:存在する」には、データが存在した場合に、OKで、なければエラーです。) 初期値は、「auto:自動」です。
3.tableId	【TAG】 (通常は使いません)結果をDBTableModelに書き込んで、sessionに登録するときのキーを指定します。 初期値は、HybsSystem.TBL_MDL_KEY です。
4.dbid	【TAG】 (通常は使いません)Queryオブジェクトを作成する時のDB接続IDを指定します。 Queryオブジェクトを作成する時のDB接続IDを指定します。
5.lbl	【TAG】 ラベルリソースIDを指定します。 ラベルリソースIDを指定します。 各処理に応じた初期設定のラベルリソースIDは、以下の通りです。 exist="true" ERR0025=データ未登録エラー。キー=[0]、値=[1] のデータは、存在していません。 exist="false" ERR0026=データ登録済みエラー。キー=[0]、値=[1] のデータは、すでに存在しています。 exist="one" ERR0027=データ2重登録エラー。キー=[0]、値=[1] のデータは、重複して存在しています。 JavaScript ERR0030=入力したデータが不正です。key=[0] value=[1] 形式=[2] 引数のパラメータには、通常、チェックに使用した実データが、DBTableModel から取得されます。 引数を変更する場合は、msgParamKeys を使用してください。
	【TAG】 ラベルリソースの引数をカンマ区切りで指定します。 ラベルリソースのキーをカンマ区切りで指定することで、設定します。

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
6.lblParamKeys	ラベルに引数 ([0], [1] など) がある場合、ここで指定した値を順番に、[0], [1], [2] . . . に当てはめていきます。 キーワードは、カンマ区切りで指定し、それを分解後、ラベルリソースでリソース変換を行います。(つまり、記述された値そのものではありません) PL/SQL では、“{#PN}” などと指定していた分は、同様に “PN” と指定します。 内部的に、where 条件に指定されたキーと値は、@KEY と @VAL に、from と where の間の文字列は、@TBL に対応付けられます。 [@XXXX] 変数も使用できます。実データの値を取り出したい場合は、[PN] とすれば、DBTableModel の PN の値を取り出します。 なにも指定しない場合は、キー=[0]、値=[1]、from=[2] です。
7.errRemove	【TAG】エラー時の選択行を取り除いて継続処理を行うかどうか[true/false]を指定します(初期値:false)。 exist 属性に指定された、[true:存在する]、[false:存在しない]、[one:ひとつ以下]、 に対して、エラーが発生した選択行番号を、取り除いて以下の処理を継続するかどうかを指定します。 true に設定した場合は、エラーデータを削除し、継続処理を行うことができます。 false の場合は、エラーデータを表示して、継続処理を停止します。 初期値は、[false:エラー時停止]です。
8.sqlType	【TAG】このチェックを行う、SQLタイプ を指定します。 SQLタイプは、INSERT、COPY、UPDATE、MODIFY、DELETE などの記号を指定します。 一般には、result 画面から update 画面へ遷移するときの、command と同じにしておけばよいでしょう。 これは、execType とマッチした場合のみ、このチェックが処理されます。 簡易 equals タグの代役に使用できます。 なにも指定しない場合は、チェックは実行されます。
9.execType	【TAG】このチェックを行う、実行タイプ を指定します。 実行タイプは、sqlType とマッチした場合のみ、このチェックが処理されます。 簡易 equals タグの代役に使用できます。 execType は、複数指定が可能です。単純な文字列マッチで、sqlType を含めば、実行されます。 例えば、sqlType=[@sqlType] execType=“INSERT COPY” とすれば、sqlType に、INSERT または、COPY が登録された場合にチェックが掛かります。 なにも指定しない場合は、チェックは実行されます。
10.conditionKey	【TAG】条件判定するカラム I D を指定します。 指定のカラム I D の値と、conditionList の値を比較して、存在する場合は、check処理を実行します。 この処理が有効なのは、command=“ENTRY” の場合のみです。
11.conditionList	【TAG】条件判定する値のリストを、“ ”で区切って登録します(初期値:無条件)。 conditionKey とペアで指定します。ここには、カラムの設定値のリストを指定することで、複数条件(O R 結合)での比較を行い、リストにカラム値が存在する場合のみ、check処理を実行します。 この処理が有効なのは、command=“ENTRY” の場合のみです。 設定しない場合は、無条件に実行します。
12.uniqCheckClms	【TAG】指定されたキーに従って、メモリ上のテーブルに対してユニークキーチェックを行います。 ユニークキーチェックを行うキーを指定します。ここで、指定されたキーに対して、DBTableModelの値をチェックし、全てのキーに同じ値となっている行が存在すればエラーとなります。 このチェックは、command=“ENTRY”の場合のみ有効です。 また、このチェックは他のチェック(DB存在チェックなど)と同時に処理することはできません。 キーが指定され手いる場合は、ボディ部分に記述されている定義は無視されます。 errRemoveの属性がtrueに指定されている場合、重複行は、DBTableModelの並び順から見て、最初の行のみ処理され、2つめ以降の重複行は無視されます。 なお、キーはカンマ区切り(CSV形式)で複数指定が可能です。
13.beforeErrorJsp	【TAG】エラーが発生した際に、エラーメッセージの表示前にincludeするJSPを指定します。 エラーが発生した際に、エラーメッセージの表示前にincludeするJSPを指定します。 エラーが発生していない場合は、ここで指定されたJSPが処理されることはありません。 通常は、戻るリンクなどを指定します。
14.afterErrorJsp	指定の方法は、相対パス、絶対パスの両方で指定することができます。 但し、絶対パスで指定した場合、その基点は、コンテキストのルートディレクトリになります。 例) beforeErrorJsp = “/jsp/common/history_back.jsp” 【TAG】エラーが発生した際に、エラーメッセージの表示後にincludeするJSPを指定します。 エラーが発生した際に、エラーメッセージの表示前にincludeするJSPを指定します。 エラーが発生していない場合は、ここで指定されたJSPが処理されることはありません。
15.selectedAll	指定の方法は、相対パス、絶対パスの両方で指定することができます。 但し、絶対パスで指定した場合、その基点は、コンテキストのルートディレクトリになります。 例) afterErrorJsp = “/jsp/common/history_back.jsp” 【TAG】データを全件選択済みデータとして処理するかどうか[true/false]を指定します(初期値:false)。 全てのデータを選択済みデータとして扱って処理します。 全件処理する場合に、(true/false)を指定します。 初期値は false です。
16.msg	【廃止】メッセージIDを指定します(lbl 属性を使用してください)。 メッセージIDを指定します。 各処理に応じた初期設定のメッセージIDは、以下の通りです。 exist=“true” ERR0025=データ未登録エラー。キー=[0]、値=[1] のデータは、存在していません。 exist=“false” ERR0026=データ登録済みエラー。キー=[0]、値=[1] のデータは、すでに存在しています。 exist=“one” ERR0027=データ 2 重登録エラー。キー=[0]、値=[1] のデータは、重複して存在しています。 JavaScript ERR0030=入力したデータが不正です。key=[0] value=[1] 形式=[2] 引数のパラメータには、通常、チェックに使用した実データが、DBTableModel から取得されます。 引数を変更する場合は、msgParamKeys を使用してください。
17.msgParamKeys	【廃止】メッセージリソースの引数をカンマ区切りで指定します(lblParamKeys 属性を使用してください)。 メッセージリソースのキーをカンマ区切りで指定することで、設定します。 メッセージに引数 ([0], [1] など) がある場合、ここで指定した値を順番に、[0], [1], [2] . . . に当てはめていきます。 キーワードは、カンマ区切りで指定し、それを分解後、ラベルリソースでリソース変換を行います。(つまり、記述された値そのものではありません) PL/SQL では、“{#PN}” などと指定していた分は、同様に “PN” と指定します。 内部的に、where 条件に指定されたキーと値は、@KEY と @VAL に、from と where の間の文字列は、@TBL に対応付けられます。 [@XXXX] 変数も使用できます。実データの値を取り出したい場合は、[PN] とすれば、DBTableModel の PN の値を取り出します。 なにも指定しない場合は、キー=[0]、値=[1]、from=[2] です。
18.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:databaseMetaData	【TAG】（通常は使いません）Queryオブジェクトを作成する時のDB接続IDを指定します。 Queryオブジェクトを作成する時のDB接続IDを指定します。
Tag	
●形式： <og:databaseMetaData dbid = "[@DBID]" catalog = "[@catalog]" schema = "[@schema]" tableName = "[@tableName]" columnName = "[@columnName]" typeName = "[@typeName]" attributeName = "[@attriName]" procedureName = "[@procName]" maxRowCount = "[@maxRowCount]" likeKey = "[@likeKey]" />	
●body：なし	
●Tag定義： <og:databaseMetaData dbid = "[@DBID]" catalog = "[@catalog]" schema = "[@schema]" attributeName = "[@attriName]" columnName = "[@columnName]" procedureName = "[@procName]" tableName = "[@tableName]" typeName = "[@typeName]" maxRowCount = "[@maxRowCount]" likeKey = "[@likeKey]" debug = "[@debug]" />	【TAG】（通常は使いません）Queryオブジェクトを作成する時のDB接続IDを指定します 【TAG】（通常は使いません）Queryオブジェクトを作成する時のカタログを指定します 【TAG】（通常は使いません）Queryオブジェクトを作成する時のスキーマを指定します 【TAG】（通常は使いません）Queryオブジェクトを作成する時の属性名を指定します 【TAG】（通常は使いません）Queryオブジェクトを作成する時のカラム名を指定します 【TAG】（通常は使いません）Queryオブジェクトを作成する時のプロダクト名を指定します 【TAG】（通常は使いません）Queryオブジェクトを作成する時のテーブル名を指定します 【TAG】（通常は使いません）Queryオブジェクトを作成する時のタイプ名を指定します 【TAG】 maxRowCount ResultSetで検索するパラメータの最大検索件数を指定します（初期値：20） 【TAG】 実行するメソッドを選別するための部分一致するキーを指定します 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します（初期値：false）
●使用例 <og:databaseMetaData />	
1.dbid	【TAG】（通常は使いません）Queryオブジェクトを作成する時のDB接続IDを指定します。 Queryオブジェクトを作成する時のDB接続IDを指定します。
2.catalog	【TAG】（通常は使いません）Queryオブジェクトを作成する時のカタログを指定します。 Queryオブジェクトを作成する時のカタログを指定します。
3.schema	【TAG】（通常は使いません）Queryオブジェクトを作成する時のスキーマを指定します。 Queryオブジェクトを作成する時のスキーマを指定します。
4.attributeName	【TAG】（通常は使いません）Queryオブジェクトを作成する時の属性名を指定します。 Queryオブジェクトを作成する時の属性名を指定します。
5.columnName	【TAG】（通常は使いません）Queryオブジェクトを作成する時のカラム名を指定します。 Queryオブジェクトを作成する時のカラム名を指定します。
6.procedureName	【TAG】（通常は使いません）Queryオブジェクトを作成する時のプロダクト名を指定します。 Queryオブジェクトを作成する時のプロダクト名を指定します。
7.tableName	【TAG】（通常は使いません）Queryオブジェクトを作成する時のテーブル名を指定します。 Queryオブジェクトを作成する時のテーブル名を指定します。
8.typeName	【TAG】（通常は使いません）Queryオブジェクトを作成する時のタイプ名を指定します。 Queryオブジェクトを作成する時のタイプ名を指定します。
9.maxRowCount	【TAG】 maxRowCount ResultSetで検索するパラメータの最大検索件数を指定します（初期値：20）。 大量に検索されて、処理速度が低下するのを避けるため、最大件数を指定します。 他のタグの maxRowCount の初期値設定（初期値：DB_MAX_ROW_COUNT）は利用せず、独自に設定値を持っています。
10.likeKey	【TAG】 実行するメソッドを選別するための部分一致するキーを指定します。 DatabaseMetaData で取得できる各種情報をアクセスするにあたり、指定のキーで部分一致する メソッドだけ処理するように指定します。 メソッドの部分一致で判断します。 処理は、大文字、小文字を認識したうえで、部分一致で判定します。
11.debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します（初期値：false）。 デバッグ情報は [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:dbConfigEditor Tag	<p>【TAG】DBConfig.xmlに対する操作のコマンドを指定します(初期値:GET)。 GET → typeに応じた情報をDBConfig.xmlから読み出し、DBTableModelに登録します。 SET → typeに応じた情報をDBTableModelから読み出し、DBConfig.xmlに登録します。</p> <p>なお、保存(SET)を実行すると、元のDBConfig.xmlファイルと同じ場所に、タイプスタンプ付きのバックアップファイルが自動的に作成されます。</p> <p>また、xmlFile(DBConfig.xml)を指定しない状態で、保存(command="SET")を行うと、内部にキャッシュしている接続情報は、一旦全てクリアされます。</p>
●形式: <dbConfigEditor command="..." type="..." xmlFile="..." /> ●body: なし	
●Tag定義: <og:dbConfigEditor command type xmlFile debug > </>	<p>【TAG】DBConfig.xmlに対する操作のコマンドを指定します(初期値:GET)</p> <p>【TAG】DBConfig.xmlに対する操作のタイプを指定します(初期値:DBID)</p> <p>【TAG】編集を行うDBConfig.xmlのファイルのパスを指定します(初期値:自身のDBConfig.xml)</p> <p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)</p>
●使用例 ・取得 <dbConfigEditor command="GET" type="@TYPE" xmlFile="@DBCXML" /> ・保存 <dbConfigEditor command="SET" type="@MEM_TYPE" xmlFile="@MEM_DBCXML" />	
1.command	<p>【TAG】DBConfig.xmlに対する操作のコマンドを指定します(初期値:GET)。 GET → typeに応じた情報をDBConfig.xmlから読み出し、DBTableModelに登録します。 SET → typeに応じた情報をDBTableModelから読み出し、DBConfig.xmlに登録します。</p> <p>なお、保存(SET)を実行すると、元のDBConfig.xmlファイルと同じ場所に、タイプスタンプ付きのバックアップファイルが自動的に作成されます。</p> <p>また、xmlFile(DBConfig.xml)を指定しない状態で、保存(command="SET")を行うと、内部にキャッシュしている接続情報は、一旦全てクリアされます。</p>
2.type	<p>【TAG】DBConfig.xmlに対する操作のタイプを指定します(初期値:DBID)。 DBID → DBIDの各種情報を読み取り/書き込みします。 CLASS → ロードするDBドライバーの一覧を読み取り/書き込みします。</p>
3.xmlFile	<p>【TAG】編集を行うDBConfig.xmlのファイルのパスを指定します(初期値:自身のDBConfig.xml)。 何も指定されない場合は、自身のコンテキストのDBConfig.xmlに対する編集となります。</p>
4.debug	<p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。</p>

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

5.5.4.2
og:dbMetaDataQuery

【TAG】 コマンド[NEW/RENEW]をセットします (初期値:NEW)。
コマンドは、HTMLから (get/post) 指定されますので、CMD_xxx で設定される
フィールド定数値のいつれかを、指定できます。

Tag

●形式 : <og:dbMetaDataQuery action="..." />
●body : なし

●Tag定義 :
<og:dbMetaDataQuery
command
action
catalog
schema
tableName
columnName
procName
unique
approximate
scope
displayMsg
notfoundMsg
tableId
dbId
mainTrans
debug
>
●使用例
command属性 は、columnSetタグのcommand属性と同一の場合のみ、処理します。
[command属性]
NEW 新規
RENEW 再検索
[action属性]
SCHEMAS このデータベースで使用可能なスキーマ名を取得します。
TABLES 指定されたカタログで使用可能なテーブルに関する記述を取得します。
COLUMNS 指定されたカタログで使用可能なテーブル列の記述を取得します。
INDEXINFO 指定されたテーブルのインデックスと統計情報に関する記述を取得します。
PROCEDURES 指定されたカタログで使用可能なストアードプロシージャに関する記述を取得します。

1.command

【TAG】 コマンド[NEW/RENEW]をセットします (初期値:NEW)。
コマンドは、HTMLから (get/post) 指定されますので、CMD_xxx で設定される
フィールド定数値のいつれかを、指定できます。

2.action

【TAG】 アクション (SCHEMAS, TABLES, COLUMNS, INDEXINFO, PROCEDURES) を指定します。
アクションは、HTMLから (get/post) 指定されますので、ACT_xxx で設定される
フィールド定数値のいつれかを、指定できます。
SCHEMAS このデータベースで使用可能なスキーマ名を取得します。
TABLES 指定されたカタログで使用可能なテーブルに関する記述を取得します。
COLUMNS 指定されたカタログで使用可能なテーブル列の記述を取得します。
INDEXINFO 指定されたテーブルのインデックスと統計情報に関する記述を取得します。
PROCEDURES 指定されたカタログで使用可能なストアードプロシージャに関する記述を取得します。

3.catalog

【TAG】 カタログ名をセットします (初期値:null)。
データベースに格納されたカタログ名と一致しなければならない。
"" はカタログなしでカタログ名を検索する。
null は、カタログ名を検索の限定に使用してはならないことを意味する

4.schema

【TAG】 スキーマ名パターンをセットします (初期値:null)。
データベースに格納されたスキーマ名と一致しなければならない。
"" はスキーマなしでスキーマ名を検索する。
null は、スキーマ名を検索の限定に使用してはならないことを意味する

5.tableName

【TAG】 テーブル名パターンをセットします (初期値:null)。
データベースに格納されたテーブル名と一致しなければならない

6.columnName

【TAG】 列名パターンをセットします (初期値:null)。
データベースに格納された列名と一致しなければならない

7.procName

【TAG】 プロシージャ名パターンをセットします (初期値:null)。
データベースに格納されたプロシージャ名と一致しなければならない

8.unique

【TAG】 返すインデックスの種類 [true:ユニークのみ/false:非ユニーク含む] を指定します (初期値:false)。
true の場合は、一意の値のインデックスだけを返す。
false の場合は、一意であるかどうかにかかわらずインデックスを返す
初期値は、false:非ユニーク含む です。

9.approximate

【TAG】 統計情報の精度 [true:概数/false:正確] を指定します (初期値:true)。
指定されたテーブルのインデックスと統計情報に関する記述を取得する場合に
結果の精度を指定します。
true の場合は、結果は概数またはデータ値から外れることもある。
false の場合は、正確であることが要求される
初期値は、true:概数 です。

10.scope

【TAG】 キャッシュする場合のスコープ [request/page/session/applicaton] を指定します (初期値:session)。
"request", "page", "session", "applicaton" が指定できます。
JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、
主にテーブルモデルを管理するスコープを指します。

スコープ	変数の有効範囲	
page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会
request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ
session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。 内部的には、同じキーワー
application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。

11.displayMsg

【TAG】 検索結果を画面上に表示するメッセージリソースIDを指定します (初期値:MSG0033[件検索しました])。
ここでは、検索結果の件数や登録された件数をまず出力し、
その次に、ここで指定したメッセージをリソースから取得して
表示します。
表示させたくない場合は、displayMsg = "" をセットしてください。
初期値は、検索件数を表示します。

12.notfoundMsg

【TAG】 検索結果がゼロ件の場合に表示するメッセージリソースIDを指定します (初期値:MSG0077[対象データはありませんでした])。
ここでは、検索結果がゼロ件の場合のみ、特別なメッセージを表示させます。
従来は、displayMsg と兼用で、『0 件検索しました』という表示でしたが、
displayMsg の初期表示は、OFF になりましたので、ゼロ件の場合のみ別に表示させます。
表示させたくない場合は、notfoundMsg = "" をセットしてください。
初期値は、MSG0077[対象データはありませんでした] です。

13.tableId

【TAG】 (通常は使いません) 結果をDBTableModelに書き込んで、sessionに登録するときのキーを指定します。
初期値は、HybsSystem.TBL_MD_KEY です。

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
14.dbid	【TAG】（通常は使いません）Queryオブジェクトを作成する時のDB接続IDを指定します。 Queryオブジェクトを作成する時のDB接続IDを指定します。
15.mainTrans	【TAG】（通常は使いません）タグで処理される処理がメインとなるトランザクション処理かどうかを指定します（初期値：false）。 この値は、ファイルダウンロード処理に影響します。この値がtrueに指定された時にcommitされたDBTableModelが ファイルダウンロードの対象の表になります。 このパラメータは、通常、各タグにより実装され、ユーザーが指定する必要はありません。 但し、1つのJSP内でDBTableModelが複数生成される場合に、前に処理したDBTableModelについてファイルダウンロードをさせたい 場合は、後ろでDBTableModelを生成するタグで、明示的にこの値をfalseに指定することで、ファイルダウンロード処理の対象から 除外することができます。
16.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します（初期値：false）。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

<div> <div>バージョンタグ名 HTMLTEXT BODY要素</div> <div>内容</div> </div>	
	形式サンプル
<div>5.5.4.2</div> <div>og:directTableInsert</div> <div>JSP</div>	<div> <div>【TAG】読み取り元ディレクトリ名を指定します (初期値:FILE_URL [=filetemp/])。</div> <div>この属性で指定されるディレクトリより、ファイルを読み取ります。 指定方法は、通常の fileURL 属性と同様に、先頭が、'/' (UNIX) または、2 文字目が、 "." (Windows) の場合は、指定のURLそのままのディレクトリに、そうでない場合は、 fileURL = "~{@USER_ID}" と指定すると、FILE_URL 属性で指定のフォルダの下に、 さらに、各個人ID別のフォルダの下より、読み取ります。 (初期値:システム定数のFILE_URL [= {@og.value org.opengion.hayabusa.common.SystemData#FILE_URL}])。</div> </div> <div> <div>●形式: <og:directTableInsert filename="["..."] ... >INSERT INTO ... </og:directTableInsert ></div> <div>●body: あり</div> </div>
<div>●Tag定義:</div> <div><og:directTableInsert</div> <div>fileURL</div> <div>filename</div> <div>encode</div> <div>separator</div> <div>displayMsg</div> <div>columns</div> <div>commitBatch</div> <div>useColumnAdjust</div> <div>useColumnCheck</div> <div>nullCheck</div> <div>dbid</div> <div>debug</div> <div>> ... Body ...</div> <div></og:directTableInsert></div>	<div> <div>【TAG】読み取り元ディレクトリ名を指定します (初期値:FILE_URL [=filetemp/])</div> <div>【TAG】ファイルを作成するときのファイル名をセットします (初期値:FILE_FILENAME [=file.xls])</div> <div>【TAG】ファイルを作成するときのファイルエンコーディング名をセットします (初期値:FILE_ENCODE [=UnicodeLittle])</div> <div>【TAG】可変長ファイルを作成するときの項目区切り文字をセットします (初期値:タブ)</div> <div>【TAG】query の結果を画面上に表示するメッセージIDを指定します (初期値:MSG0040[件登録しました])</div> <div>【TAG】#NAME 属性の代わりとなるファイルのカラム名を CSV形式で指定します</div> <div>【TAG】指定数毎にコミットを発行します (初期値:0 終了までコミットしません)</div> <div>【TAG】カラム変換 (DBType変換) を行うかどうかを設定します (初期値:false)</div> <div>【TAG】カラムチェック (DBTypeチェック) を行うかどうかを設定します (初期値:false)</div> <div>【TAG】NULL チェックすべきカラム列をカンマ区切り (CSV形式) で指定します</div> <div>【TAG】 (通常は使いません) 検索時のDB接続IDを指定します (初期値:DEFAULT)</div> <div>【TAG】デバッグ情報を出力するかどうか [true/false] を指定します (初期値:false)</div> </div>
<div>●使用例</div> <div><og:directTableInsert</div> <div>dbid</div> <div>separator</div> <div>fileURL</div> <div>filename</div> <div>encode</div> <div>displayMsg</div> <div>columns</div> <div>commitBatch</div> <div>useColumnCheck</div> <div>useColumnAdjust</div> <div>nullCheck</div> <div>></div> <div>INSERT INTO GE41</div> <div>(CLM, NAME_JA, LABEL_NAME, KBSAKU, SYSTEM_ID, LANG,</div> <div>FGJ, DYSET, DYUPD, USRSET, USRUPD, PGUPD)</div> <div>VALUES</div> <div>([{CLM}], [NAME_JA], [LABEL_NAME], [KBSAKU], [SYSTEM_ID], [LANG],</div> <div>'1', '{@USER_YMDH}', '{@USER_YMDH}', '{@USER_ID}', '{@USER_ID}', '{@GUI_KEY}')</div> <div></og:directTableInsert ></div>	<div> <div>dbid = "ORCL" 接続データベースID (初期値:DEFAULT)</div> <div>separator = "," ファイルの区切り文字 (初期値:タブ)</div> <div>fileURL = "~{@USER_ID}" 読み取り元ディレクトリ名</div> <div>filename = "~{@filename}" 読み取り元ファイル名</div> <div>encode = "Shift_JIS" 読み取り元ファイルエンコード名</div> <div>displayMsg = "MSG0040" 登録完了後のメッセージ</div> <div>columns = "CLM, NAME_JA, LABEL_NAME, KBSAKU, SYSTEM_ID, LANG" #NAME の代わりに使用するカラム列名</div> <div>commitBatch = "100" この件数ずつコミットを発行 (初期値:無制限)</div> <div>useColumnCheck = "true" カラムチェックを行うかどうか (初期値:false)</div> <div>useColumnAdjust = "true" カラム変換を行うかどうか (初期値:false)</div> <div>nullCheck = "CLM, SYSTEM_ID" NULLチェックを実行します。</div> </div>
<div>1.fileURL</div> <div>2.filename</div> <div>3.encode</div> <div>4.separator</div> <div>5.displayMsg</div> <div>6.columns</div> <div>7.commitBatch</div> <div>8.useColumnAdjust</div> <div>9.useColumnCheck</div>	<div> <div>【TAG】読み取り元ディレクトリ名を指定します (初期値:FILE_URL [=filetemp/])。</div> <div>この属性で指定されるディレクトリより、ファイルを読み取ります。 指定方法は、通常の fileURL 属性と同様に、先頭が、'/' (UNIX) または、2 文字目が、 "." (Windows) の場合は、指定のURLそのままのディレクトリに、そうでない場合は、 fileURL = "~{@USER_ID}" と指定すると、FILE_URL 属性で指定のフォルダの下に、 さらに、各個人ID別のフォルダの下より、読み取ります。 (初期値:システム定数のFILE_URL [= {@og.value org.opengion.hayabusa.common.SystemData#FILE_URL}])。</div> </div> <div> <div>【TAG】ファイルを作成するときのファイル名をセットします (初期値:FILE_FILENAME [=file.xls])。</div> <div>ファイルを作成するときのファイル名をセットします。 (初期値:システム定数のFILE_FILENAME [= {@og.value org.opengion.hayabusa.common.SystemData#FILE_FILENAME}])。</div> </div> <div> <div>【TAG】ファイルを作成するときのファイルエンコーディング名をセットします (初期値:FILE_ENCODE [=UnicodeLittle])。</div> <div>初期値は、システムパラメータ の FILE_ENCODE 属性で、設定しています。 Shift_JIS, MS932, Windows-31J, UTF-8, ISO-8859-1, UnicodeLittle... (初期値:システム定数のFILE_ENCODE [= {@og.value org.opengion.hayabusa.common.SystemData#FILE_ENCODE}])。</div> </div> <div> <div>【TAG】可変長ファイルを作成するときの項目区切り文字をセットします (初期値:タブ)</div> <div>可変長ファイルを作成するときの項目区切り文字をセットします。</div> </div> <div> <div>【TAG】query の結果を画面上に表示するメッセージIDを指定します (初期値:MSG0040[件登録しました])。</div> <div>ここでは、検索結果の件数や登録された件数をまず出力し、 その次に、ここで指定したメッセージをリソースから取得して 表示します。 表示させたくない場合は、displayMsg = "" をセットしてください。 初期値は、検索件数を表示します。 ※ この属性には、リクエスト変数 ({@XXXX}) は使用できません。</div> </div> <div> <div>【TAG】#NAME 属性の代わりとなるファイルのカラム名を CSV形式で指定します。</div> <div>データファイルの先頭行に、#NAME 行があり、読み取るべきファイルの カラム名が記述されています。通常は、このカラム名を取り込んで、 各データ列のカラムを指定します。 この属性は、ファイルに#NAME 行が存在しない(他システムからの入力ファイル等) 場合に、#NAME 属性の代わりに、カラム名を外部より指定します。</div> </div> <div> <div>【TAG】指定数毎にコミットを発行します (初期値:0 終了までコミットしません)。</div> <div>通常は、全ての処理が正常に終了するか、なにもしないか(トランザクション) を判断すべきで、途中でのコミットはしません。 しかし、場合によって、件数が異常に多い場合や、再実行可能な場合は、 途中でコミットして、都度、処理できるものだけを処理してしまうという方法があります。 また、ロールバックエリアの関係などで、データ量が多い場合に、処理時間が異常に 長くなる事があり、指定件数ごとのコミット機能を用意しています。 0 に設定すると、終了までコミットしません。初期値は、0 です。</div> </div> <div> <div>【TAG】カラム変換 (DBType変換) を行うかどうかを設定します (初期値:false)。</div> <div>カラムの変換を行う場合、この属性を設定 (true) します。 初期値は、行わない (false) です。</div> </div> <div> <div>変換するカラムは、#NAME や columns で指定されたカラムではなく、 BODY部のSQL文で指定されたカラム名 [カラム名] です。これは、直接、SQL 文中に 記述している値や、{@XXXX} 文字等は、変換出来ない為です。</div> </div> <div> <div>【TAG】カラムチェック (DBTypeチェック) を行うかどうかを設定します (初期値:false)。</div> <div>カラムの整合性チェックを行う場合、この属性を設定 (true) します。 初期値は、行わない (false) です。</div> </div> <div> <div>チェックするカラムは、#NAME や columns で指定されたカラムではなく、 BODY部のSQL文で指定されたカラム名 ([カラム名]) です。これは、直接、SQL 文中に</div> </div>

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
10.nullCheck	記述している値や、{@XXX} 文字等は、チェック出来ない為です。 【TAG】 NULL. チェックすべきカラム列をカンマ区切り (CSV形式) で指定します。 nullCheck="AAA, BBB, CCC, DDD" 先に配列に分解してからリクエスト変数の値を取得するようにします。 こする事で、リクエストにカンマ区切りの値を設定できるようになります。
11.dbid	分解方法は、通常のパラメータ取得後に、CSV分解します。 【TAG】 (通常は使いません) 検索時のDB接続IDを指定します (初期値:DEFAULT)。 検索時のDB接続IDを指定します。初期値は、DEFAULT です。
12.debug	【TAG】 デバッグ情報を出力するかどうか [true/false] を指定します (初期値:false)。 デバッグ情報を [true:出力する/false:しない] を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
5.5.4.2 og:directWriteTable JSP	<p>【TAG】保存先ディレクトリ名を指定します (初期値:FILE_URL [=filetemp/])。</p> <p>この属性で指定されるディレクトリに、ファイルをセーブします。 指定方法は、通常の fileURL 属性と同様に、先頭が、'/' (UNIX) または、2 文字目が、 "." (Windows) の場合は、指定のURLそのままのディレクトリに、そうでない場合は、 fileURL = "[@USER_ID]" と指定すると、FILE_URL 属性で指定のフォルダの下に、 さらに、各個人ID別のフォルダを作成して、そこにセーブします。 (初期値:システム定数のFILE_URL [=[@og.value org.opengion.hayabusa.common.SystemData#FILE_URL]])。</p>
<p>●形式: <og:directWriteTable filename=" [...] " ... >SELECT * FROM ZYXX </og:directWriteTable ></p> <p>●body: あり</p> <p>●Tag定義:</p> <pre> <og:directWriteTable fileURL filename zipFilename encode fileAppend zip separator useHeader displayMsg notfoundMsg fetchSize names queryType dbid debug > ... Body ... </og:directWriteTable> </pre> <p>●使用例</p> <pre> <og:directWriteTable dbid = "ORCL" 接続データベースID(初期値:DEFAULT) separator = "," ファイルの区切り文字(初期値:タブ) fileURL = "[@USER_ID]" 保存先ディレクトリ名 filename = "[@filename]" 保存ファイル名 encode = "UnicodeLittle" 保存ファイルエンコード名 useHeader = "true" 保存ファイルにヘッダーを出力するかどうか zip = "true" ZIPファイルに圧縮するかどうか zipFilename = "Sample.zip" ZIPファイルのファイル名 fileAppend = "true" ファイルを追加モードで登録するかどうか displayMsg = "MSG0033" 実行後の表示メッセージ fetchSize = "200" DB検索する場合のフェッチするサイズ > SELECT * FROM ZYXX </og:directWriteTable> <og:directWriteTable fileURL = "[@USER_ID]" 保存先ディレクトリ名 filename = "[@filename]" 保存ファイル名 names = "AAA, BBB, CCC, ..." 指定のキーに対応するリクエスト値を ARG_ARRAY にセットします。 queryType = "JDBCErrMsg" JDBCErrMsg 形式のPL/SQL をコールします。 > { call PL/SQL(?, ?, ?, ?) } </og:directWriteTable> </pre>	
1.fileURL	<p>【TAG】保存先ディレクトリ名を指定します (初期値:FILE_URL [=filetemp/])。</p> <p>この属性で指定されるディレクトリに、ファイルをセーブします。 指定方法は、通常の fileURL 属性と同様に、先頭が、'/' (UNIX) または、2 文字目が、 "." (Windows) の場合は、指定のURLそのままのディレクトリに、そうでない場合は、 fileURL = "[@USER_ID]" と指定すると、FILE_URL 属性で指定のフォルダの下に、 さらに、各個人ID別のフォルダを作成して、そこにセーブします。 (初期値:システム定数のFILE_URL [=[@og.value org.opengion.hayabusa.common.SystemData#FILE_URL]])。</p>
2.filename	<p>【TAG】ファイルを作成するときのファイル名をセットします(初期値:システムパラメータのFILE_FILENAME)。</p> <p>ファイルを作成するときのファイル名をセットします。</p>
3.zipFilename	<p>【TAG】ZIPファイルを作成するときのZIPファイル名をセットします(初期値:filename + ".zip")。</p> <p>zip 属性に、true を指定した場合に、ZIPファイル化します。その場合のファイル名を指定します。 なにも指定しない場合は、filename + ".zip" になります。</p>
4.encode	<p>【TAG】ファイルを作成するときのファイルエンコーディング名をセットします (初期値:FILE_ENCODE [=UnicodeLittle])。</p> <p>"DEFAULT", "JISAutoDetect", "JIS", "EUC_JP", "MS932", "SJIS", "Windows-31J", "Shift_JIS" (初期値:システム定数のFILE_ENCODE [=[@og.value org.opengion.hayabusa.common.SystemData#FILE_ENCODE]])。</p>
5.fileAppend	<p>【TAG】追加モードで書き込むかどうか[true/false]を指定します(初期値:false[通常モード])。</p> <p>ファイルを書き込む場合、追加モードで書き込むかどうかをセットします。 新規モード(true)の場合、既存のファイルが存在し、かつ書き込み許可があれば、 上書きで新規に作成します。 初期値は、false(新規モード)です。</p>
6.zip	<p>【TAG】結果をファイルに出力するときに、ZIPで圧縮するかどうか[true/false]を指定します(初期値:false)。</p> <p>大量に抜き出す場合、そのまま、サーバーから取り出すだけでも大変です。 zip 属性を、true にすると、GZIP で圧縮したファイルを作成します。 初期値は、false(圧縮しない)です。</p>
7.separator	<p>【TAG】可変長ファイルを作成するときの項目区切り文字をセットします (初期値:TAB_SEPARATOR [=])</p> <p>可変長ファイルを作成するときの項目区切り文字をセットします。 (初期値:ローカル定義のTAB_SEPARATOR [=[@og.value #TAB_SEPARATOR]])。</p>
8.useHeader	<p>【TAG】ヘッダーを書き込むかどうか[true/false]を指定します(初期値:true)。</p> <p>#Name ヘッダーの書き込みを指定します。 通常は、書き込み(true)にしており、使用側でコメントと解釈するように 処理を行うべきです。コメントのため、append モードで途中に現れても 無視できます。また、エンジン標準でデータを取り込む場合に、データの配置が 変更されても取り込みプログラムはそのまま使用できます。 初期値は、true(書き込む)です。</p>
9.displayMsg	<p>【TAG】検索結果を画面上に表示するメッセージリソースIDを指定します(初期値:MSG0033[件検索しました])。</p> <p>ここでは、検索結果の件数や登録された件数をまず出力し、 その次に、ここで指定したメッセージをリソースから取得して 表示します。 表示させたくない場合は、displayMsg = "" をセットしてください。 初期値は、検索件数を表示します。</p>
	<p>【TAG】検索結果がゼロ件の場合に表示するメッセージリソースIDを指定します(初期値:MSG0077[対象データはありませんでした])。</p> <p>ここでは、検索結果がゼロ件の場合のみ、特別なメッセージを表示させます。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
10.notfoundMsg	従来は、displayMsg と兼用で、『0 件検索しました』という表示でしたが、displayMsg の初期表示は、OFF になりましたので、ゼロ件の場合のみ別に表示させます。 表示させたくない場合は、notfoundMsg = "" をセットしてください。 初期値は、MSG0077[対象データはありませんでした]です。
11.fetchSize	【TAG】（通常は使いません）データのフェッチサイズを指定します(初期値:100)。 より多くの行が必要なときに、データベースから取り出す必要がある行数についてのヒントを JDBC ドライバに提供します。 指定された行数は、この Statement を使って作成された結果セットにだけ影響します。 指定された値が 0 の場合、ヒントは無視されます。 初期値は、100 です。
12.names	【TAG】PL/SQLを利用する場合の引数にセットすべき データの名称をCSV形式で複数指定します。 複数ある場合は、カンマ区切り文字で渡します。 PL/SQL を使用しない場合は、無視されます。
13.queryType	【TAG】Query を発行する為のクラスID(JDBC, JDBCErrMsg)を指定します(初期値:JDBC)。 ストアドプロシージャ等を実行する場合に、queryType="JDBCErrMsg" を指定する必要があります。(それ以外の指定は、初期値の JDBC になります。) 初期値は、"JDBC" です。
14.dbid	【TAG】（通常は使いません）検索時のDB接続IDを指定します(初期値:DEFAULT)。 検索時のDB接続IDを指定します。初期値は、DEFAULT です。
15.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:directXMLSave	<p>【TAG】（通常は使いません）検索時のDB接続IDを指定します（初期値:DEFAULT）。検索時のDB接続IDを指定します。初期値は、DEFAULT です。</p> <p>Tag</p> <p>●形式: <og:directXMLSave filename=" [...] " ... /></p> <p>●body: なし</p> <p>●Tag定義:</p> <pre> <og:directXMLSave dbid fileURL filename displayMsg keys vals debug /> </pre> <p>【TAG】（通常は使いません）検索時のDB接続IDを指定します（初期値:DEFAULT）</p> <p>【TAG】 読み取り元ディレクトリ名を指定します（初期値:FILE_URL[=filetemp/]）</p> <p>【TAG】 ファイルを作成するときのファイル名をセットします（初期値:FILE_FILENAME[=file.xls]）</p> <p>【TAG】 query の結果を画面上に表示するメッセージIDを指定します（初期値:MSG0040[件登録しました]）</p> <p>【TAG】 XMLファイルを読み取った後で指定するキーをCSV形式で複数指定します</p> <p>【TAG】 XMLファイルを読み取った後で指定する値をCSV形式で複数指定します</p> <p>【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します（初期値:false）</p> <p>●使用例</p> <pre> <og:directXMLSave dbid = "ORCL" 接続データベースID(初期値:DEFAULT) fileURL = "{@USER_ID}" 読み取り元ディレクトリ名 filename = "{@filename}" 読み取り元ファイル名 displayMsg = "MSG0040" 登録完了後のメッセージ /> </pre>
1.dbid	<p>【TAG】（通常は使いません）検索時のDB接続IDを指定します（初期値:DEFAULT）。検索時のDB接続IDを指定します。初期値は、DEFAULT です。</p>
2.fileURL	<p>【TAG】 読み取り元ディレクトリ名を指定します（初期値:FILE_URL[=filetemp/]）。</p> <p>この属性で指定されるディレクトリより、ファイルを読み取ります。指定方法は、通常の fileURL 属性と同様に、先頭が、"/"（UNIX）または、2文字目が、"."（Windows）の場合は、指定のURLそのままのディレクトリに、そうでない場合は、fileURL = "{@USER_ID}" と指定すると、FILE_URL 属性で指定のフォルダの下に、さらに、各個人ID別のフォルダの下より、読み取ります。</p> <p>（初期値:システム定数のFILE_URL[= {@og.value.org.opengion.hayabusa.common.SystemData#FILE_URL}]）。</p>
3.filename	<p>【TAG】 ファイルを作成するときのファイル名をセットします（初期値:FILE_FILENAME[=file.xls]）。</p> <p>ファイルを作成するときのファイル名をセットします。</p> <p>（初期値:システム定数のFILE_FILENAME[= {@og.value.org.opengion.hayabusa.common.SystemData#FILE_FILENAME}]）。</p>
4.displayMsg	<p>【TAG】 query の結果を画面上に表示するメッセージIDを指定します（初期値:MSG0040[件登録しました]）。</p> <p>ここでは、検索結果の件数や登録された件数をまず出力し、その次に、ここで指定したメッセージをリソースから取得して表示します。</p> <p>表示させたくない場合は、displayMsg = "" をセットしてください。</p> <p>初期値は、検索件数を表示します。</p> <p>※ この属性には、リクエスト変数（{@XXXX}）は使用できません。</p>
5.keys	<p>【TAG】 XMLファイルを読み取った後で指定するキーをCSV形式で複数指定します。</p> <p>XMLファイルを読み取った後で、データを変更できます。</p> <p>変更するカラム名(キー)をCSV形式で指定します。</p> <p>XMLファイルにキーが存在していた場合は、vals で指定の値に書き換えます。</p> <p>キーが存在していない場合は、ここで指定するキーと値が、データとして追加されます。</p> <p>例えば、登録日や、登録者、または、テンプレートより各システムID毎に登録するなどの使い方を想定しています。</p> <p>分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。</p> <p>こうしないとデータ自身にカンマを持っている場合に分解をミスる為です。</p>
6.vals	<p>【TAG】 XMLファイルを読み取った後で指定する値をCSV形式で複数指定します。</p> <p>XMLファイルを読み取った後で、データを変更できます。</p> <p>変更する値をCSV形式で指定します。</p> <p>XMLファイルにキーが存在していた場合は、vals で指定の値に書き換えます。</p> <p>キーが存在していない場合は、ここで指定するキーと値が、データとして追加されます。</p> <p>例えば、登録日や、登録者、または、テンプレートより各システムID毎に登録するなどの使い方を想定しています。</p> <p>分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。</p> <p>こうしないとデータ自身にカンマを持っている場合に分解をミスる為です。</p>
7.debug	<p>【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します（初期値:false）。</p> <p>デバッグ情報は、[true:出力する/false:しない]を指定します。</p> <p>出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容												
形式サンプル													
5.5.4.2 og:docConvert	<p>【TAG】操作するファイルのディレクトリを指定します (初期値:FILE_URL[=filetemp/])。</p> <p>この属性で指定されるディレクトリのファイルを操作します。 指定方法は、通常の fileURL 属性と同様に、先頭が、'/' (UNIX) または、2 文字目が、 "." (Windows) の場合は、指定のURLそのままのディレクトリに、そうでない場合は、 (初期値:システム定数のFILE_URL[=@og.value org.opengion.hayabusa.common.SystemData#FILE_URL])。</p>												
Tag	<p>●形式: <og:docConvert fileURL="..." inputFile="..." outputFile="..." /> ●body: なし</p> <p>●Tag定義:</p> <table> <tr> <td><og:docConvert</td><td></td></tr> <tr> <td>fileURL</td><td>【TAG】操作するファイルのディレクトリを指定します (初期値:FILE_URL[=filetemp/])</td></tr> <tr> <td>inputFile</td><td>○【TAG】入力ファイル名を指定します。(必須)</td></tr> <tr> <td>outputFile</td><td>○【TAG】出力ファイル名を指定します。(必須)</td></tr> <tr> <td>delInput</td><td>【TAG】ドキュメント変換後、元のファイルを削除するかどうかを指定します(初期値:false[=削除しない])</td></tr> <tr> <td>debug</td><td>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)</td></tr> </table> <p></></p> <p>●使用例</p> <ul style="list-style-type: none"> ・Calc(ods) ファイルをPDFに変換 <og:docConvert inputFile="temp.ods" outputFile="out.pdf" /> ・Excel(xls) ファイルを連結 <og:docConvert inputFile="temp1.xls,temp2.xls" outputFile="out.xls" /> 	<og:docConvert		fileURL	【TAG】操作するファイルのディレクトリを指定します (初期値:FILE_URL[=filetemp/])	inputFile	○【TAG】入力ファイル名を指定します。(必須)	outputFile	○【TAG】出力ファイル名を指定します。(必須)	delInput	【TAG】ドキュメント変換後、元のファイルを削除するかどうかを指定します(初期値:false[=削除しない])	debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)
<og:docConvert													
fileURL	【TAG】操作するファイルのディレクトリを指定します (初期値:FILE_URL[=filetemp/])												
inputFile	○【TAG】入力ファイル名を指定します。(必須)												
outputFile	○【TAG】出力ファイル名を指定します。(必須)												
delInput	【TAG】ドキュメント変換後、元のファイルを削除するかどうかを指定します(初期値:false[=削除しない])												
debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)												
1.fileURL	<p>【TAG】操作するファイルのディレクトリを指定します (初期値:FILE_URL[=filetemp/])。</p> <p>この属性で指定されるディレクトリのファイルを操作します。 指定方法は、通常の fileURL 属性と同様に、先頭が、'/' (UNIX) または、2 文字目が、 "." (Windows) の場合は、指定のURLそのままのディレクトリに、そうでない場合は、 (初期値:システム定数のFILE_URL[=@og.value org.opengion.hayabusa.common.SystemData#FILE_URL])。</p>												
2.inputFile	<p>【TAG】入力ファイル名を指定します。</p> <p>入力ファイルは、カンマ区切りで複数指定することができます。 この場合、複数の入力ファイルをマージして出力を行います。</p> <p>※現状は、ファイルのマージは、入力ファイルがExcelまたはCalcの場合のみ対応しています。 また、マージを行う場合、印刷範囲など、ドキュメントに関連する情報は、1つ目のファイルの 情報が使用されます。</p>												
3.outputFile	<p>【TAG】出力ファイル名を指定します。</p> <p>出力ファイル名を指定します。 出力ファイルには、拡張子のみ(xls, ods等)を指定することもできます。 この場合、出力ファイル名は、入力ファイル名と同じになります。(拡張子のみが変換される)</p>												
4.delInput	<p>【TAG】ドキュメント変換後、元のファイルを削除するかどうかを指定します(初期値:false[=削除しない])。</p> <p>ドキュメント変換後、元のファイルを削除するかどうかを指定します。 (初期値:false(削除しない))</p>												
5.debug	<p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。</p> <p>デバッグ情報を [true:出力する/false:しない] を指定します。 出力形式自体は、個々のタグによって異なります。</p>												

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:engine Tag ●形式 : <og:engine connectionClear="[true false]" ... /> ●body : なし ●Tag定義 : <og:engine allClear connectionClear resourceClear guiClear userClear calendarClear debug > ●使用例 <og:engine connectionClear = "true" 初期値:false resourceClear = "true" 初期値:false guiClear = "true" 初期値:false userClear = "true" 初期値:false calendarClear = "true" 初期値:false > <og:engine allClear = "true" >	【TAG】すべてのキャッシュデータをクリアします(初期値:false)。 connectionClear, resourceClear, guiClear, calendarClearの すべてを、true に設定することと同じです。 【TAG】すべてのキャッシュデータをクリアします(初期値:false) 【TAG】データベースコネクションプールのコネクションを削除(開放)します(初期値:false) 【TAG】リソースデータをクリアします(初期値:false) 【TAG】GUI情報のキャッシュをクリアします(初期値:false) 【TAG】User情報のパラメータデータをクリアします(初期値:false) 【TAG】カレンダーオブジェクトのキャッシュをクリアします(初期値:false) 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false) 初期値:false 初期値:false 初期値:false 初期値:false 初期値:false 初期値:false 初期値:false 初期値:false 初期値:false 初期値:false 初期値:false 初期値:false
1.allClear	【TAG】すべてのキャッシュデータをクリアします(初期値:false)。 connectionClear, resourceClear, guiClear, calendarClearの すべてを、true に設定することと同じです。
2.connectionClear	【TAG】データベースコネクションプールのコネクションを削除(開放)します(初期値:false)。 データベース操作がおかしい場合は、クリアしてみてください。
3.resourceClear	【TAG】リソースデータをクリアします(初期値:false)。 ユーザーリソースとシステムリソースを除くリソースキャッシュ データをクリアします。
4.guiClear	【TAG】GUI情報のキャッシュをクリアします(初期値:false)。 GUI情報の追加/更新/削除が発生したときは、キャッシュをクリアする必要があります。
5.userClear	【TAG】User情報のパラメータデータをクリアします(初期値:false)。 User情報のパラメータデータ (GE16から読み込み)に追加/更新/削除が発生したときは、 キャッシュをクリアする必要があります。 User情報のパラメータデータは、{@USER: XXXX} でアクセスできる値ではなく、 システムパラメータ {@SYS: XXXX} としてアクセスできるユーザー個別情報のことです。
6.calendarClear	【TAG】カレンダーオブジェクトのキャッシュをクリアします(初期値:false)。 カレンダーオブジェクトは、事業所カレンダー(例: GE13)毎にオブジェクトを 作成します。キャッシュに持っているため、データベースを更新するたびに、 キャッシュをクリアしなおす必要があります。
7.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

5.5.4.2
og:entry

【TAG】 コマンド (INSERT, COPY, MODIFY, DELETE, ENTRY, CHANGE, ALLACTION, RESET) をセットします。
コマンドは、HTMLから (get/post) 指定されますので、CMD_xxx で設定される
フィールド定数値のいつれかを、指定できます。

JSP

●形式：

<og:entry command="...">
 <og:columnSet command="..." />
</og:entry>
<og:entry command="..." />
 ...columnSetを使わない場合でもresult.jspから次画面(insert,modify,copy.jsp等)に
 にDBTableModelをもっていく場合には、必ず2を書いてください。
 (取消のとき、エンジン内でDBTableModelを操作するのに使用する為)

●body：あり (columnSet)

●Tag定義：

<og:entry
 command
 repeatCount
 tableId
 useConsistency
 selectedAll
 strictCheck
 noTransition
 debug
> ... Body ...
</og:entry>

○ 【TAG】 コマンド (INSERT, COPY, MODIFY, DELETE, ENTRY, CHANGE, ALLACTION, RESET) をセットします。(必須)
【TAG】 キャッシュする場合のスコープ [request/page/session/applicaton] を指定します (初期値:session)
【TAG】 指定の回数分だけ、繰り返し処理を行う回数を指定します (初期値:1)
【TAG】 (通常使いません) sessionから所得する DBTableModelオブジェクトの ID
【TAG】 Consistency キー による整合性チェックを行うかどうかを指定します (初期値:true)
【TAG】 データを全件選択済みとして処理するかどうか [true/false] を指定します (初期値:false)
【TAG】 カラム I D の存在チェックを行うかどうか [true/false] を指定します (初期値:true)
【TAG】 (通常は使いません) 画面遷移を行わない形式の登録方法を使用するかを指定します
【TAG】 デバッグ情報を出力するかどうか [true/false] を指定します (初期値:false)

●使用例

<og:entry command="@command">
 <og:columnSet command="@command" columnId="ECNO" action="CLEAR" />
 <og:columnSet command="@command" columnId="JYOKYO" action="SET" value="1" />
</og:entry>

<og:entry command="MODIFY" rows="1">
 <og:columnSet command="MODIFY" columnId="key" action="TBLSET" value="[key][lang]" />
</og:entry>

<og:entry command="@command" repeatCount="5">
 <og:columnSet command="@command" columnId="YKNO" action="ADD" />
</og:entry>

command属性 は、columnSetタグのcommand属性と同一の場合のみ、処理します。
[command属性]
INSERT 新規
COPY 複写
MODIFY 変更
DELETE 削除
ENTRY エントリー
CHANGE チェンジ
RESET リセット
ALLRESET 全件リセット
ALLACTION オールアクション
RESETDATA データリセット
INSERTONE 新規 (1行のみ)
REALDELETE 物理削除

[action属性]
DEFAULT カラムリソースで定義した初期値をセットします。
CLEAR 値をクリア(ゼロストリング "") します。
ADD 現在の値を +1 します。 0 ⇒ 1, A ⇒ B, 9 ⇒ 10
SET value で設定した値を 新しい値として登録します。
NULLSET 元の値が NULL の場合だけ、value で設定した新しい値を登録します。
LOWER 小文字に変換します。
UPPER 大文字に変換します。
COPY value にコピー元のカラムIDをセットすれば、その値を代入します。
TBLSET DBTableModel の内容を取り込んで指定の columnId カラムに設定します。[カラム名] で指定できます。
また、これは文字列を解析して、value を作成しますので、文字列連結等に使用できます。
WRTCTRL writableControl を使用したカラムデータの先頭アンダーバーを削除します。
DBMENU DBMENUでパラメータ設定(コロン連結文字)を使用したカラムデータの先頭データのみにします。
REQSET valueで指定したカラムの値をキーに、リクエスト変数から値を取り出し、セットします。

[strictCheck属性]は、カラム I D の存在チェックを行うかどうかを指定します (初期値:true)
true カラム I D がDBTableModel に存在しない場合は、エラーになる。
false カラム I D がDBTableModel に存在しない場合は、無視する。

1.command

【TAG】 コマンド (INSERT, COPY, MODIFY, DELETE, ENTRY, CHANGE, ALLACTION, RESET) をセットします。
コマンドは、HTMLから (get/post) 指定されますので、CMD_xxx で設定される
フィールド定数値のいつれかを、指定できます。

2.scope

【TAG】 キャッシュする場合のスコープ [request/page/session/applicaton] を指定します (初期値:session)。
"request", "page", "session", "application" が指定できます。
JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、
主にテーブルモデルを管理するスコープを指します。

スコープ	変数の有効範囲	
page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会
request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ
session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。 内部的には、同じキーワー
application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。

3.repeatCount

【TAG】 指定の回数分だけ、繰り返し処理を行う回数を指定します (初期値:1)。
追加や複写時に、指定の回数分だけ、処理を繰り返して、新規に行を
作成します。
繰り返しは、指定の行に対して行われ、繰り返し毎に、直前に作成された
行を元に処理します。これは、例えば、columnSet で、action="ADD"の場合に、
繰り返し毎に、ADD処理が実行されることを意味します。
行が指定されていない場合は、先頭空行に追加します。
初期値は、1 回です。

4.tableId

【TAG】 (通常使いません) sessionから所得する DBTableModelオブジェクトの ID。
表示処理後に、(内部ポインタを書き換えた) DBTableModelオブジェクトを

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
5.useConsistency	同じキーで、sessionに登録します。 初期値は、HybsSystem.TBL_MDL_KEY です。 【TAG】Consistency キー による整合性チェックを行うかどうかを指定します(初期値:true)。 検索結果を DBTableModel にセットする時に、整合性キーの Consistency キーを 作成します。これを、Viewタグでhidden出力しておき、Entryタグでデータ書き換え時に 整合性チェックを行います。これは、IEの戻るボタンで戻った場合に、画面の キーと検索結果の DBTableModel の内容が一致しない場合のエラーチェックに なります。 この属性は、何らかのケースで、このエラーチェックを行いたくない場合に、 false に設定することで、整合性チェックを行いません。 初期値は、true(整合性チェックを行う)です。
6.selectedAll	【TAG】データを全件選択済みデータとして処理するかどうか[true/false]を指定します(初期値:false)。 全てのデータを選択済みデータとして扱って処理します。 全件処理する場合に、(true/false)を指定します。 初期値は false です。
7.strictCheck	【TAG】カラム I D の存在チェックを行うかどうか[true/false]を指定します(初期値:true)。 true の場合、カラム I D がDBTableModel に存在しない場合は、エラーになります。 false の場合、カラム I D がDBTableModel に存在しない場合は、無視します。 これは、検索条件によって、設定されるカラムが異なる場合でも、entryタグを 正常に動作させたい場合に、使用します。 初期値は true (チェックを行う) です。
8.noTransition	【TAG】(通常は使いません)画面遷移を行わない形式の登録方法を使用するかを指定します。 画面遷移なしの登録を行うかどうかを指定します。 trueが指定された場合、entryタグでは、行の追加・複写時にDBTableModel上の最終行にデータを 追加します。 画面遷移なしモードの場合、途中行に挿入された場合、既にクライアントに出力されている チェックボックスの行番号や各入力フィールドの変数名との整合性を合わせるためには、 編集行以降の各変数値を全て再計算する必要があります。 この処理は、レスポンス悪化に繋がるため、DBTableModel上は、中間に行の挿入を行いません。 但し画面表示上は、通常通り選択行の直下に行が挿入されるため、DBTableModelの順番と標準順が 異なります。(エンジン側では、各チェックボックスの値で行を識別しているため、問題は発生しません)
9.debug	この値は、og:headタグで設定値、または前画面からの値を継承するため、通常、この属性ではセットしません。 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容															
形式サンプル																
5.5.4.2 og:entryQuery	【TAG】Query を発行する為のクラスIDを指定します(初期値:JDBKeyEntry)。引数指定のINSERT/UPDATE文を実行する場合の、queryType 属性を使用します。このタグでは、execute(String[] , String[])を実行します。代表的なクラスとして、“JDBKeyEntry” が標準で用意されています。															
JSP	タグにより使用できる／出来ないがありますが、これは、org.opengion.hayabusa.db 以下の Query_**** クラスの **** を与えます。これらは、Query インターフェースを継承したサブクラスです。															
●形式： <og:entryQuery queryType = “JDBKeyEntry” command = “[@command]” > { call DYNAMIC_PKG.DYNAMIC_TEST(?,?,?) } </og:query> queryType : Queryオブジェクトを作成する時のキー(オプション) QueryFactory参照 command : 処理コマンド(NEWのみ)(オプション) skipRowCount : データの読み始めの初期値を指定(オプション) maxRowCount : データの最大読み込み件数を指定(オプション)																
●body : あり																
●Tag定義： <og:entryQuery queryType ○ 【TAG】Query を発行する為のクラスIDを指定します(初期値:JDBKeyEntry)。(必須) command 【TAG】コマンド(NEW, RENEW)をセットします(PlsqlUpdateTag, UpdateTag の場合は、ENTRY) scope 【TAG】キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session) maxRowCount 【TAG】(通常は使いません)データの最大読み込み件数を指定します(初期値:DB_MAX_ROW_COUNT[=1000]) skipRowCount 【TAG】(通常は使いません)データの読み始めの初期値を指定します displayMsg 【TAG】検索結果を画面上に表示するメッセージリソースIDを指定します(初期値:VIEW_DISPLAY_MSG[=]) notfoundMsg 【TAG】検索結果がゼロ件の場合に表示するメッセージリソースIDを指定します(初期値:MSG0077[対象データはありませんでした]) names 【TAG】PL/SQLを利用する場合の引数にセットすべき データの名称をCSV形式で複数指定します tableId 【TAG】(通常は使いません)結果のDBTableModelを、sessionに登録するときのキーを指定します dbId 【TAG】(通常は使いません)Queryオブジェクトを作成する時のDB接続IDを指定します debug 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false) > ... Body ... </og:entryQuery>																
●使用例 例 1) <table><tr><og:input name=“PN” value=“[@PN]” /></tr> <tr><og:input name=“CD” value=“[@CD]” /></tr> </table> 例 2) <table><tr><og:input name=“PN” value=“[@PN]” >部品入力フィールド</og:input></tr> <tr><og:input name=“CD” value=“[@CD]” >コードフィールド</og:input></tr> </table> BODY 部分は、無視されます。コメント等に使用できます。 HTMLファイルには、コメント部分は出力されません。																
1.queryType	【TAG】Query を発行する為のクラスIDを指定します(初期値:JDBKeyEntry)。引数指定のINSERT/UPDATE文を実行する場合の、queryType 属性を使用します。このタグでは、execute(String[] , String[])を実行します。代表的なクラスとして、“JDBKeyEntry” が標準で用意されています。 タグにより使用できる／出来ないがありますが、これは、org.opengion.hayabusa.db 以下の Query_**** クラスの **** を与えます。これらは、Query インターフェースを継承したサブクラスです。															
2.command	【TAG】コマンド(NEW, RENEW)をセットします(PlsqlUpdateTag, UpdateTag の場合は、ENTRY)。コマンドは、HTMLから(get/post)指定されますので、CMD_xxx で設定されるフィールド定数値のいつれかを、指定できます。															
3.scope	【TAG】キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)。“request”, “page”, “session”, “applicaton” が指定できます。JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、主にテーブルモデルを管理するスコープを指します。 <table><tr><th>スコープ</th><th>変数の有効範囲</th><th></th></tr><tr><td>page</td><td>JSPページ内</td><td>そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会</td></tr><tr><td>request</td><td>HTTPリクエスト</td><td>リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ</td></tr><tr><td>session</td><td>HTTPセッション</td><td>初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。内部的には、同じキーワー</td></tr><tr><td>application</td><td>Webアプリケーション</td><td>ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。</td></tr></table>	スコープ	変数の有効範囲		page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会	request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ	session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。内部的には、同じキーワー	application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。
スコープ	変数の有効範囲															
page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会														
request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ														
session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。内部的には、同じキーワー														
application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。														
4.maxRowCount	【TAG】(通常は使いません)データの最大読み込み件数を指定します(初期値:DB_MAX_ROW_COUNT[=1000])。データベース自体の検索は、指定されたSQLの全件を検索しますが、DBTableModelのデータとして登録する最大件数をこの値に設定します。サーバーのメモリ資源と応答時間の確保の為です。0 をセットすると、無制限(Integer.MAX_VALUE)になります。(初期値:ユーザー定数のDB_MAX_ROW_COUNT[=[@og.value org.opengion.hayabusa.common.SystemData#DB_MAX_ROW_COUNT]])。															
5.skipRowCount	【TAG】(通常は使いません)データの読み始めの初期値を指定します。データベース自体の検索は、指定されたSQLの全件を検索しますが、DBTableModelのデータとしては、スキップ件数分は登録されません。サーバーのメモリ資源と応答時間の確保の為です。															
6.displayMsg	【TAG】検索結果を画面上に表示するメッセージリソースIDを指定します(初期値:VIEW_DISPLAY_MSG[=])。ここでは、検索結果の件数や登録された件数をまず出力し、その次に、ここで指定したメッセージをリソースから取得して表示します。件数を表示させたい場合は、displayMsg = “MSG0033[件検索しました]” をセットしてください。表示させたくない場合は、displayMsg = “” をセットしてください。(初期値:システム定数のVIEW_DISPLAY_MSG[=[@og.value org.opengion.hayabusa.common.SystemData#VIEW_DISPLAY_MSG]])。															
7.notfoundMsg	【TAG】検索結果がゼロ件の場合に表示するメッセージリソースIDを指定します(初期値:MSG0077[対象データはありませんでした])。ここでは、検索結果がゼロ件の場合のみ、特別なメッセージを表示させます。従来は、displayMsg と兼用で、『0 件検索しました』という表示でしたが、displayMsg の初期表示は、OFF になりましたので、ゼロ件の場合のみ別に表示させます。表示させたくない場合は、notfoundMsg = “” をセットしてください。初期値は、MSG0077[対象データはありませんでした]です。															
	【TAG】PL/SQLを利用する場合の引数にセットすべき データの名称をCSV形式で複数指定します。複数ある場合は、カンマ区切り文字で渡します。names 属性は、queryType に応じて設定可否が異なりますので、ご注意ください。															

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
8.names	names なし : JDBC, JDBCUpdate names あり : JDBCCallable, JDBCErrMsg, JDBCUpdate (JDBCUpdateは、names 属性のあり/なし両方に対応しています。)
9.tableId	【TAG】 (通常は使いません) 結果のDBTableModelを、sessionに登録するときのキーを指定します。 検索結果より、DBTableModelオブジェクトを作成します。これを、以下のview 等のタグに 渡す場合に、通常は、session を利用します。その場合の登録キーです。 query タグを同時に実行して、結果を求める場合、同一メモリに配置される為、 この tableId 属性を利用して、メモリ空間を分けます。 初期値は、HybsSystem.TBL_MDL_KEY です。
10.dbid	【TAG】 (通常は使いません) Queryオブジェクトを作成する時のDB接続IDを指定します。 Queryオブジェクトを作成する時のDB接続IDを指定します。 これは、システムリソースで、DEFAULT_DB_URL 等で指定している データベース接続先 情報に、XX_DB_URL を定義することで、 dbid="XX" とすると、この 接続先を使用して データベースにアクセスできます。
11.debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTML5 BODY要素	内容
形式サンプル	
<div>5.5.4.2</div> <div>og:equals</div> <div>JSP</div> <div>●形式：<og:equals val1="..." val2="..." ...></og:equals></div> <div>●body：あり</div> <div>●Tag定義：</div> <div><og:equals</div> <div>val1</div> <div>val2</div> <div>ignoreCase</div> <div>startsWith</div> <div>nullCheck</div> <div>notEquals</div> <div>contains</div> <div>endsWith</div> <div>matches</div> <div>useStop</div> <div>operator</div> <div>compareTo</div> <div>debug</div> <div>></div> <div>... Body ...</div> <div></og:equals></div> <div>●使用例</div> <div>・<og:equals val1="ABCD" val2="@value" ...></div> <div>val1.equals(val2) 時に実行したい内容</div> <div></og:equals></div> <div>・<og:equals val1="@value" nullCheck="true" ...></div> <div>val1がnullの時に実行したい内容</div> <div></og:equals></div> <div>val1</div> <div>val2</div> <div>ignoreCase</div> <div>startsWith</div> <div>nullCheck</div> <div>notEquals</div> <div>contains</div> <div>endsWith</div> <div>matches</div> <div>operator</div> <div>compareTo</div> <div>debug</div>	<div>【TAG】 equals で比較するときの、第 1 引数 (左辺) を指定します。</div> <div>equals で比較するときの、第 1 引数を指定します。</div> <div>val1 が null の場合は、無条件で非成立になります。</div> <div>(val1 == val2) はチェックしないというイメージです。</div> <div>val1 が null かどうか判断する場合は、nullCheck="true" を使用してください。</div> <div>○【TAG】 equals で比較するときの、第 1 引数 (左辺) を指定します。(必須)</div> <div>【TAG】 equals で比較するときの、第 2 引数 (右辺) を指定します</div> <div>【TAG】 大文字/小文字を区別しないかどうか[true/false]を指定します (初期値:false (区別する))</div> <div>【TAG】 この文字列が、指定された接頭辞で始まるかどうか[true/false]を判定します (初期値:false)</div> <div>【TAG】 null チェックを行うかどうか[true/false]を指定します (初期値:false)</div> <div>【TAG】 判定結果を反転させるかどうか[true/false]を指定します (初期値:false)</div> <div>【TAG】 文字列が含まれているかどうか[true/false]の判定します (初期値:false)</div> <div>【TAG】 指定された接尾辞で終わるかどうか[true/false]を判定します (初期値:false)</div> <div>【TAG】 指定された正規表現と一致するかどうか[true/false]を判定します (初期値:false)</div> <div>【TAG】 BODYを処理後に停止するかどうか[true/false]を指定します (初期値:false)</div> <div>【TAG】 比較する操作を EQ, LT, LE, GT, GE から指定します (初期値:EQ)</div> <div>【TAG】 大小比較する方法 (STRING: 前方比較、NUMBER: 数値比較) を指定します (初期値:STRING)</div> <div>【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します (初期値:false)</div> <div>【TAG】 equals で比較するときの、第 1 引数 (左辺) を指定します。</div> <div>equals で比較するときの、第 1 引数を指定します。</div> <div>val1 が null の場合は、無条件で非成立になります。</div> <div>(val1 == val2) はチェックしないというイメージです。</div> <div>val1 が null かどうか判断する場合は、nullCheck="true" を使用してください。</div> <div>【TAG】 equals で比較するときの、第 2 引数 (右辺) を指定します。</div> <div>equals で比較するときの、第 2 引数を指定します。</div> <div>【TAG】 大文字/小文字を区別しないかどうか[true/false]を指定します (初期値:false (区別する))。</div> <div>startsWith, contains, endsWith, equalsIgnoreCase での比較時に、比較対象の</div> <div>大文字/小文字を区別しないかどうかを指定します。</div> <div>区別しない ("true") 場合、aaa と AAA は、一致したとみなされます。</div> <div>初期値は、区別する ("false") です。</div> <div>【TAG】 この文字列が、指定された接頭辞で始まるかどうか[true/false]を判定します (初期値:false)。</div> <div>val1.startsWith(val2) の書式で判定されます。この場合、"ABCDEFGG".startsWith("ABC")</div> <div>の場合に、条件成立します。(つまり、val1 に対して、val2 で始まっているかどうか問合せる)</div> <div>初期値は、判定しない ("false")</div> <div>【TAG】 null チェックを行うかどうか[true/false]を指定します (初期値:false)。</div> <div>チェックを行うように指定 ("true") した場合に、第一引数が null の場合は、</div> <div>条件成立して、タグのBody は実行されます。</div> <div>初期値は、行わない (true 以外) です。</div> <div>【TAG】 判定結果を反転させるかどうか[true/false]を指定します (初期値:false)。</div> <div>通常の成立条件において、不成立の場合に、BODY を実行します。</div> <div>通常の処理結果を求めて、最後に、反転処理を行います。</div> <div>初期値は、通常 (true 以外) です。</div> <div>【TAG】 文字列が含まれているかどうか[true/false]の判定します (初期値:false)。</div> <div>val1.indexOf(val2) >= 0 の書式で判定されます。この場合、"ABCDEFGG".indexOf("CDE")</div> <div>の場合に、条件成立します。(つまり、val1 に対して、val2 が含まれているかどうか問合せる)</div> <div>初期値は、判定しない ("false")</div> <div>【TAG】 指定された接尾辞で終わるかどうか[true/false]を判定します (初期値:false)。</div> <div>val1.endsWith(val2) の書式で判定されます。この場合、"ABCDEFGG".endsWith("EFG")</div> <div>の場合に、条件成立します。(つまり、val1 に対して、val2 で終わっているかどうか問合せる)</div> <div>初期値は、判定しない ("false")</div> <div>【TAG】 指定された正規表現と一致するかどうか[true/false]を判定します (初期値:false)。</div> <div>val1.matches(val2) の書式で判定されます。val2 に指定された正規表現で、</div> <div>一致するかどうかを判定します。ここでの正規表現とは、</div> <div>java.util.regex.Pattern.matches(val1, val2) での判定結果と同じです。</div> <div>初期値は、判定しない ("false")</div> <div>【TAG】 BODYを処理後に停止するかどうか[true/false]を指定します (初期値:false)。</div> <div>処理結果などに応じて、以下の処理を停止したい場合に、使用します。</div> <div>ここでは、条件を判定後、true の場合に、BODY部を出力 (処理) した後に、</div> <div>ここにセットされた値に応じて、以下のページ処理を行うかどうかを指定します。</div> <div>true を指定すると、以下の処理は、行われません。</div> <div>初期値は、停止しない ("false")</div> <div>【TAG】 比較する操作を EQ, LT, LE, GT, GE から指定します (初期値:EQ)。</div> <div>比較方法として、EQ(==)、LT(<)、LE(<=)、GT(>)、GE(>=) があります。</div> <div>初期値は、EQ(同じかどうか) です。</div> <div>比較は、val1 に対して行われます。val1 または val2 が null の場合は、常にfalseが</div> <div>返されます。通常の A < B ならば、not(B >= A) の関係は成立しません。</div> <div>val1 が null でない場合は、val1(5) LT vla2(8) は、true を返します。</div> <div>ignoreCase属性 (大文字/小文字の区別) を指定することで、比較する文字列の</div> <div>大文字小文字を統一させることが可能です。</div> <div>なお、比較時には、compareTo 属性にもご注意ください。これは、文字列比較か</div> <div>数値比較を指定します。</div> <div>【TAG】 大小比較する方法 (STRING: 前方比較、NUMBER: 数値比較) を指定します (初期値:STRING)。</div> <div>operator 属性で大小を比較する場合、比較方法として、前方比較と数値比較を指定できます。</div>

<div>バージョン タグ名 HTMLTEXT BODY要素</div>	内容
12.compareType	<div>形式サンプル</div> <p>STRING (前方比較) とは、お互いの文字列を前方から順に比較していきます。例えば、ABC と AABBC や、123 と 112233 では、AABBC や 112233 が小さいと判断されます。 NUMBER (数値比較) では、123 と 112233 では、123 が小さいと判断されます。 NUMBER は、数字に変換できる値である必要があります。 STRING は、ignoreCase属性 (大文字/小文字の区別) を指定することで、比較する文字列の大文字小文字を統一させることが可能です。 初期値は、STRING (前方比較) です。</p>
13.debug	<p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します (初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
5.5.4.2 og:error JSP	<p>【TAG】メール送信可否を指定します(初期値:true)。 エラー発生時に管理者にメールを送信するかどうかを指定します。 メールは、システムパラメータの COMMON_MAIL_SERVER に、ERROR_MAIL_TO_USERS に送信します。 ERROR_MAIL_TO_USERS が未設定の場合は、送信しません。 初期値は、true(送信する)です。</p>
●形式 :	<pre><og:error useMail = "[true false]" メール送信可否を指定します(初期値:true) logMsgType = "[LONG MEDIUM SHORT NONE]" ログに書き込むメッセージの形式を指定(初期値:MEDIUM) viewMsgType = "[LONG MEDIUM SHORT NONE ALLNONE TABLE]" 画面に表示するメッセージの形式を指定(初期値:SHORT) /></pre>
●body : あり	
●Tag定義 :	<pre><og:error useMail [TAG] メール送信可否を指定します(初期値:true) logMsgType [TAG] ログに書き込むメッセージの形式を指定(初期値:MEDIUM) viewMsgType [TAG] 画面に書き込むメッセージの形式を指定(初期値:MEDIUM) debug [TAG] デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false) skipPage [TAG] エラーが発生した時に、以降の処理をスキップするか(初期値:false[=スキップしない]) > ... Body ... </og:error></pre>
●使用例 <og:error />	
1.useMail	<p>【TAG】メール送信可否を指定します(初期値:true)。 エラー発生時に管理者にメールを送信するかどうかを指定します。 メールは、システムパラメータの COMMON_MAIL_SERVER に、ERROR_MAIL_TO_USERS に送信します。 ERROR_MAIL_TO_USERS が未設定の場合は、送信しません。 初期値は、true(送信する)です。</p>
2.logMsgType	<p>【TAG】ログに書き込むメッセージの形式を指定(初期値:MEDIUM)。 ログ、および、メール送信時のメッセージの形式を指定します。 エラー時のExceptionは、階層構造になっており、ルートまでさかのぼることが可能です。また、通常は、スタックとレース情報より、エラーのプログラムを特定することで、早く対応することが可能になります。 メッセージの形式には、LONG MEDIUM SHORT NONE が指定できます。 ボディ一部分に記述されたメッセージは全ての場合で出力されます。</p> <ul style="list-style-type: none"> LONG :すべてのスタックトレース情報を取得します。 MEDIUM:org. opengion以下のパッケージのみスタックトレース情報を取得します。 SHORT :メッセージ部分のみ情報を取得します。 NONE :取得しません。 <p>初期値は、MEDIUM です。</p>
3.viewMsgType	<p>【TAG】画面に書き込むメッセージの形式を指定(初期値:MEDIUM)。 画面に表示するメッセージの形式を指定します。 エラー時のExceptionは、階層構造になっており、ルートまでさかのぼることが可能です。また、通常は、スタックとレース情報より、エラーのプログラムを特定することで、早く対応することが可能になります。 メッセージの形式には、LONG MEDIUM SHORT NONE ALLNONE TABLE が指定できます。 ボディ一部分に記述されたメッセージは全ての場合で出力されます。</p> <ul style="list-style-type: none"> LONG :すべてのスタックトレース情報を取得します。 MEDIUM :org. opengion以下のパッケージのみスタックトレース情報を取得します。 SHORT :メッセージ部分のみ情報を取得します。 NONE :取得しません。 ALLNONE:ヘッダも表示しません。 TABLE :テーブル形式でエラーメッセージのみを表示します。 <p>初期値は、SHORT です。</p>
4.debug	<p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。</p>
5.skipPage	<p>【TAG】エラーが発生した時に、以降の処理をスキップするか(初期値:false[=スキップしない])。 エラーが発生した時に、以降の処理をスキップするかを設定します。 trueが設定された場合は、以降の処理をスキップします。</p> <p>初期値は、false(スキップしない) です。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:errorMessage Tag ●形式： <og:errorMessage command="@command" clear="@clear" /> ●body：なし ●Tag定義： <og:errorMessage command 【TAG】 コマンド(NEW, RENEW, RESET, REVIEW)をセットします clear 【TAG】 メッセージを初期化するかどうか[true/false]を指定します(初期値:false) viewType 【TAG】 表示形式『表題(TITLE)か、内容(BODY)』を指定します(初期値:TITLE) displayMsg 【TAG】 plsqlUpdate の結果を画面上に表示するメッセージIDを指定します(初期値:MSG0059『登録しました』) warningMsg 【TAG】 登録処理実行後のワーニング結果を画面上に表示するメッセージIDを指定します(初期値:ERR0020) debug 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false) /> ●使用例 result.jsp 等のSQL登録実行後の戻り画面に、上記タグを配置すれば、エラーメッセージが存在すれば、リンクとなって現れ、無ければ、なにも現れません。 リンクのとび先は自動的に設定されます。 なお、clear="true" または、command="NEW" の場合に、エラーメッセージは、クリアされます。 [entry.jsp] <% String forwardPage="result.jsp"; %> <jsp:forward page="<%= response.encodeRedirectURL(forwardPage) %>" > <jsp:param name="command" value="REVIEW" /> <jsp:param name="clear" value="false" /> </jsp:forward> [result.jsp] <og:errorMessage command="@command" clear="@clear" />	【TAG】 コマンド(NEW, RENEW, RESET, REVIEW)をセットします。 コマンドは、HTMLから(get/post)指定されますので、CMD_xxx で設定されるフィールド定数値のいつれかを、指定できます。 【TAG】 メッセージを初期化するかどうか[true/false]を指定します(初期値:false)。 メッセージは、一般には、エラーメッセージかワーニングです。 最終処理でメッセージが無ければ、標準でクリアします。 また、command が NEW の場合も、メッセージは自動でクリアされます。 初期値は、クリアしない (true 以外) です。 【TAG】 表示形式『表題(TITLE)か、内容(BODY)』を指定します(初期値:TITLE)。 一般には、表題(TITLE)を表示しておきます。 表題表示時には、リンクを張り、共通エラー表示画面をオープン出来る様になっています。 【TAG】 plsqlUpdate の結果を画面上に表示するメッセージIDを指定します(初期値:MSG0059『登録しました』)。 指定したメッセージをリソースから取得して表示します。 表示させたくない場合は、displayMsg = "MSG0065" をセットしてください。 初期値は、MSG0059『登録しました。』を表示します。 【TAG】 登録処理実行後のワーニング結果を画面上に表示するメッセージIDを指定します(初期値:ERR0020)。 指定したメッセージをリソースから取得して表示します。 表示させたくない場合は、warningMsg = "" をセットしてください。 初期値は、ERR0020『データ登録時にワーニングが発生しました。』を表示します。 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。
1.command	【TAG】 コマンド(NEW, RENEW, RESET, REVIEW)をセットします。 コマンドは、HTMLから(get/post)指定されますので、CMD_xxx で設定されるフィールド定数値のいつれかを、指定できます。
2.clear	【TAG】 メッセージを初期化するかどうか[true/false]を指定します(初期値:false)。 メッセージは、一般には、エラーメッセージかワーニングです。 最終処理でメッセージが無ければ、標準でクリアします。 また、command が NEW の場合も、メッセージは自動でクリアされます。 初期値は、クリアしない (true 以外) です。
3.viewType	【TAG】 表示形式『表題(TITLE)か、内容(BODY)』を指定します(初期値:TITLE)。 一般には、表題(TITLE)を表示しておきます。 表題表示時には、リンクを張り、共通エラー表示画面をオープン出来る様になっています。
4.displayMsg	【TAG】 plsqlUpdate の結果を画面上に表示するメッセージIDを指定します(初期値:MSG0059『登録しました』)。 指定したメッセージをリソースから取得して表示します。 表示させたくない場合は、displayMsg = "MSG0065" をセットしてください。 初期値は、MSG0059『登録しました。』を表示します。
5.warningMsg	【TAG】 登録処理実行後のワーニング結果を画面上に表示するメッセージIDを指定します(初期値:ERR0020)。 指定したメッセージをリソースから取得して表示します。 表示させたくない場合は、warningMsg = "" をセットしてください。 初期値は、ERR0020『データ登録時にワーニングが発生しました。』を表示します。
6.debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:favoriteLink JSP ●形式 : <og:favoriteLink direct="true" target="_blank" method="GET" /> ●body : あり ●Tag定義 : <og:favoriteLink target direct method href lbl linkCache lastQueryRedirect debug useIcon > ... Body ... </og:favoriteLink> ●使用例 : 直接お気に入りのリンクを作成する。 デフォルト属性を使用(direct="true" target="_blank" method="GET") <og:favoriteLink > Favorite Link </og:favoriteLink >	【TAG】リンク先の文書を表示させるフレーム、またはウィンドウの名前を指定します(初期値:_blank)。 リンク先のフレーム名(ターゲット属性)を設定します。 【TAG】リンク先の文書を表示させるフレーム、またはウィンドウの名前を指定します(初期値:_blank) 【TAG】直接アクセスできる形式のリンクを作成するかどうか[true/false]を指定します(初期値:false) 【TAG】リンクの作成元となるメソッド[POST/GET/ALL]を指定します(初期値:GET) 【TAG】リンクを作成する時の転送先アドレスを指定します(初期値:index.jsp) 【TAG】ラベルリソースのラベルIDを指定します 【TAG】リンクをキャッシュするかどうか[true/false]を指定します(初期値:false) 【TAG】キャッシュされたリンク先に転送するかどうか[true/false]を指定します(初期値:false) 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false) 【TAG】お気に入りアイコンリンクを作成するかどうか[true/false]を指定します(初期値:false)
1.target	【TAG】リンク先の文書を表示させるフレーム、またはウィンドウの名前を指定します(初期値:_blank)。 リンク先のフレーム名(ターゲット属性)を設定します。
2.direct	【TAG】直接アクセスできる形式のリンクを作成するかどうか[true/false]を指定します(初期値:false)。 trueでは、ユーザーパスワードの入力なしで直接アクセスできる形式のリンクが作成されます。falseの場合は、通常のリンクが作成されます。 trueは、指定の画面のフレームレベルでの指定になります。false は、トップフレームを含む形なので、通常の登録画面と同じ形式になります。 初期値は、false(直接リンクしない)です。
3.method	【TAG】リンクの作成元となるメソッド[POST/GET/ALL]を指定します(初期値:GET)。 ここで指定したメソッドでリクエストされた場合のみ、リンクを作成します。 初期値は、GET です。(つまり GET のみリンクを作成します。) これは、POST では、引数が付かない為です。(実際は付ける事ができますが・・・) ALL は、どちらの場合でもリンクを作成しますが、先に述べたように POST では引数がつきません。 初期値は、GETです。
4.href	【TAG】リンクを作成する時の転送先アドレスを指定します(初期値:index.jsp)。 direct="false"(初期値)に使用されるリンクの転送先アドレスを指定します。 初期値は、index.jspです。
5.lbl	【TAG】ラベルリソースのラベルIDを指定します。 ラベルを変更するときに、lbl属性を使います。 ラベルID は、所定の language に基づく ResourceManager の getLabelData(id) を呼び出し、その結果のLabelInterfaceを使用します。 getMsgLbl() で取り出せます。 ラベルIDとメッセージIDは同時には登録できません。 ラベルとメッセージは統一されました。
6.linkCache	【TAG】リンクをキャッシュするかどうか[true/false]を指定します(初期値:false)。 この、favoriteLink で指定された画面を、ユーザー毎にキャッシュします。 キャッシュされた画面は、lastQuery を指定することで、取り出し(転送)することが出来ます。 ここでのキャッシュは、direct="true" を指定した場合のアドレスです。 direct="false" は、index.jsp からのフレーム形式の為、メール等で送り、後ほど再開するような使い方(または、ワークフロー的な使い方)を想定していますが、direct="true" により単独フレームデータを、リアルタイムで使用するケース(EXCELのWebクエリーなど)で使います。 初期値は、falseです。
7.lastQueryRedirect	【TAG】キャッシュされたリンク先に転送するかどうか[true/false]を指定します(初期値:false)。 この、favoriteLink で指定された画面を、キャッシュします。 キャッシュされた画面は、lastQuery を指定することで、取り出し(転送)することが出来ます。 ここでのキャッシュは、direct="true" を指定した場合のアドレスです。 direct="false" は、index.jsp からのフレーム形式の為、メール等で送り、後ほど再開するような使い方(または、ワークフロー的な使い方)を想定していますが、direct="true" により単独フレームデータを、リアルタイムで使用するケース(EXCELのWebクエリーなど)で使います。 初期値は、falseです。
8.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。
9.useIcon	【TAG】お気に入りアイコンリンクを作成するかどうか[true/false]を指定します(初期値:false)。 初期値は、falseです。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:fieldset JSP	<p>【TAG】ラベルリソースのラベルIDを指定します。 ラベルを変更するときに、lbl属性を使います。</p> <p>ラベルID は、所定の language に基づく ResourceManager の getLabelData(id) を呼び出し、その結果のLabelInterfaceを使います。 getMsgLbl() で取り出せます。 ラベルIDとメッセージIDは同時には登録できません。 ラベルとメッセージは統一されました。</p>
<p>●形式：<og:fieldset lbl="..." or msg="..." ラベルかメッセージリソースのキー [useDisplayHide="true"] 表示 ON/OFF機能を使用する(true)かどうか [useDisplayHide="true"] 初期値を表示ON(false)にするかOFF(true)にするか > <input ..."/> <input ..."/> </og:fieldset></p> <p>●body：あり</p> <p>●Tag定義： <og:fieldset lbl 【TAG】ラベルリソースのラベルIDを指定します useDisplayHide 【TAG】表示 ON/OFF機能を使用するかどうか[true/false]を指定します(初期値:true) defaultNone 【TAG】表示ON/OFF機能を使用する場合の初期値を、隠し(none)にするかどうか[true/false]を指定します(初期値:true) debug 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false) > ... Body ... </og:fieldset></p> <p>●使用例 例 1：通常の状態。表示ON/OFF機能を使用し、初期値は表示OFF 状態 <og:fieldset lbl="INSERT_GEA11" > <og:submit value="insertGEA11" lbl="COPY" command="COPY" /> <og:column name="SYSTEM_ID" useRequestValue="false" must="true" td="no" /> </og:fieldset></p> <p>例 2：通常の状態。表示ON/OFF機能を使用し、初期値は表示ON 状態 <og:fieldset msg="GEM0001" defaultNone="false" > <og:column name="SYSTEM_ID" useRequestValue="false" must="true" td="no" /> </og:fieldset></p>	
1.lbl	<p>【TAG】ラベルリソースのラベルIDを指定します。 ラベルを変更するときに、lbl属性を使います。</p> <p>ラベルID は、所定の language に基づく ResourceManager の getLabelData(id) を呼び出し、その結果のLabelInterfaceを使います。 getMsgLbl() で取り出せます。 ラベルIDとメッセージIDは同時には登録できません。 ラベルとメッセージは統一されました。</p>
2.useDisplayHide	<p>【TAG】表示 ON/OFF機能を使用するかどうか[true/false]を指定します(初期値:true)。 表示 ON/OFF機能を使用する場合は、true / 通常の fieldset を使用する場合は、 false を指定します。 初期値は、true (ON/OFF機能を使用する) です。</p>
3.defaultNone	<p>【TAG】表示ON/OFF機能を使用する場合の初期値を、隠し(none)にするかどうか[true/false]を指定します(初期値:true)。 表示 ON/OFF機能を使用する場合にのみ設定値は有効に機能します。 隠し(none)にする場合は、true を、表示にする場合は、false をセットします。 初期値は、true (隠し(none)にする) です。</p>
4.debug	<p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTML5 BODY要素	内容
形式サンプル	
5.5.4.2 og:file JSP ●形式: <og:file action="..." fileURL="...">...</og:file> ●body: あり ●Tag定義: <og:file fileURL file1 file2 action notEquals useStop encode debug > ... Body ... </og:file> ●使用例 ・ファイルの存在チェック-存在しなければエラーメッセージを表示。 <og:file action="exists" fileURL="N:/CIR/" file1="@USER.LK[SB]/{@USER.LDN01KA1}.cir/001.sht"> <og:message msg="RKE_0157" comment="回路図が存在しません。"/> </og:file> ・N:/Filetemp/にユーザーディレクトリが存在しなければ作成。→失敗した場合エラーメッセージを表示。 <og:file action="mkdir" fileURL="N:/Filetemp/{@USER.ID}"> <og:message comment="エラーが発生しました。システム管理者に連絡してください。"/> </og:file> ・N:/Filetemp/test.txt ファイルの削除。ファイルが存在しなくても処理を続ける。 <og:file action="delete" fileURL="N:/Filetemp/" file1="test.txt" useStop="false"> <og:message comment="ファイルは存在しませんでした。"/> </og:file>	【TAG】 操作するファイルのディレクトリを指定します (初期値:FILE_URL[=filetemp/])。 この属性で指定されるディレクトリのファイル进行操作します。 指定方法は、通常の fileURL 属性と同様に、先頭が、'/' (UNIX) または、2 文字目が、 ":" (Windows) の場合は、指定のURLそのままのディレクトリに、そうでない場合は、 (初期値:システム定数のFILE_URL[=@og.value org.opengion.hayabusa.common.SystemData#FILE_URL])。
1.fileURL	【TAG】 操作するファイルのディレクトリを指定します (初期値:FILE_URL[=filetemp/])。 この属性で指定されるディレクトリのファイル进行操作します。 指定方法は、通常の fileURL 属性と同様に、先頭が、'/' (UNIX) または、2 文字目が、 ":" (Windows) の場合は、指定のURLそのままのディレクトリに、そうでない場合は、 (初期値:システム定数のFILE_URL[=@og.value org.opengion.hayabusa.common.SystemData#FILE_URL])。
2.file1	【TAG】 基準となるファイル名を指定します(コマンドの左辺のファイル名です)。
3.file2	コマンドの左辺のファイル名です。 【TAG】 処理結果となるファイル名を指定します(コマンドの右辺のファイル名です)。 コマンドの右辺のファイル名です。
4.action	【TAG】 アクション(canRead, canWrite, createNewFile, delete, exists, isDirectory, isFile, isHidden, mkdir, makedirs)を指定します。 アクションは、HTMLから(get/post)指定されますので、ACT_xxx で設定される フィールド定数値のいつれかを、指定できます。 canRead 読み込めるかどうかを判定。 canWrite 変更できるかどうか判定。 createNewFile 空の新しいファイルを不可分 (atomic) に生成。(そのファイルがまだ存在しない場合だけ) delete ファイルまたはディレクトリを削除。 renameTo ファイルまたはディレクトリ名を変更。 exists ファイルが存在するかどうか判定。 isDirectory ファイルがディレクトリであるかどうか判定。 isFile ファイルが普通のファイルかどうか判定。 isHidden ファイルが隠しファイルかどうか判定。 mkdir ディレクトリを生成。 makedirs ディレクトリを複数生成。 read ファイルを読み込んでjspWriterに出力 existsLength ファイルサイズがOByte以上のファイルが存在するかどうか判定。 copy ファイルまたはディレクトリをコピー。
5.notEquals	【TAG】 判定結果を反転させるかどうか[true/false]を指定します(初期値:false)。 通常の判定結果において、不成立(false)の場合に、BODY を実行します。 通常の処理結果の正反対の処理を行います。 初期値は、通常 (true 以外) です。
6.useStop	【TAG】 エラー時BODYを処理後に停止するかどうか[true/false]を指定します(初期値:true)。 処理結果などに応じて、以下の処理を停止したい場合に、使用します。 通常は、条件を判定後、false の場合に、BODY部を出力(処理)した後に、 処理を停止します。(useStop="true") false を指定すると、判定結果に無関係に、以下の処理を実行します。 処理は継続したい、警告表示する場合に、useStop="false" を指定します。 初期値は、停止する ("true")です。
7.encode	【TAG】 ファイルを読み込む(action="READ")際のエンコードを指定します(初期値:OS依存文字コード)。 ファイルを読み込む(action="READ")際のエンコードを指定します。 action="READ"以外場合には、この属性値は利用されません。 指定しない場合は、OS依存文字コードで読み込まれます。
8.debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:fileOption	<p>【TAG】ファイルの検索元となるディレクトリを指定します (初期値:FILE_URL[=filetemp/])。</p> <p>ファイルの検索元となるディレクトリを指定します。 (初期値:システム定数のFILE_URL[=@og.value org.opengion.hayabusa.common.SystemData#FILE_URL])。</p>
JSP	<p>●形式: <og:fileOption from="..." value="..." ...></og:fileOption></p> <p>●body: あり</p> <p>●Tag定義:</p> <pre><og:fileOption from value orderBy desc debug > ... Body ... </og:fileOption></pre> <p>●使用例</p> <pre><og:fileOption val1="ABCD" val2="@value" > <og:fileWhere startsWith="ABCD" ... /> </og:fileOption></pre>
1.from	<p>【TAG】ファイルの検索元となるディレクトリを指定します (初期値:FILE_URL[=filetemp/])。</p> <p>ファイルの検索元となるディレクトリを指定します。 (初期値:システム定数のFILE_URL[=@og.value org.opengion.hayabusa.common.SystemData#FILE_URL])。</p>
2.value	<p>【TAG】Optionの初期値で選ばれる値を指定します。</p> <p>キーになるのは、ファイル属性の NAME です。(ディレクトリなしのファイル名) ここで value属性に指定した場合、このファイル名と(大文字小文字を無視して) 一致する場合に、プルダウンの初期値に表示されます。(selected 属性が設定される。)</p>
3.orderBy	<p>【TAG】検索した結果を表示する表示順をファイル属性名で指定します(初期値:自然順序)。</p> <p>ファイルをソートする順(Comparator)を指定します。ソートに指定できる ファイル属性名は、"NAME", "LASTMODIFIED", "FILE_LENGTH" の内のどれかひとつです。 何も指定しない場合は、Fileオブジェクトの自然順序でのソートになります。 (※ 下位互換性のため、LENGTH も残しますが、廃止予定です。)</p>
4.desc	<p>【TAG】表示順を逆転するかどうか[true/false]を指定します(初期値:false)。</p> <p>orderBy 属性で指定した表示順を、逆順にするかどうかを指定できます。 初期値は、false (昇順) です。</p>
5.debug	<p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。</p> <p>デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
<p>5.5.4.2 og:fileQuery</p> <p>JSP</p> <p>●形式 : <og:fileQuery from="..." multi="true/false" > ... <og:fileWhere ... /> ... </og:fileQuery></p> <p>●body : あり</p> <p>●Tag定義 :</p> <p><og:fileQuery from multi tableId level orderBy desc addCims useUpdateCims fileType addFrom command maxRowCount displayMsg notfoundMsg scope debug > ... Body ... </og:fileQuery></p> <p>●使用例</p> <ul style="list-style-type: none">一般的な属性でファイルの検索を行います。 <og:fileQuery from = "d:/webapps/dbdef/jsp/" multi = "true" command = "[@command]" > <og:fileWhere endWith=".jsp" /> </og:fileQuery>最終変更日で逆順ソートする。対象は、2002/10/01 以降に変更されたファイル。 <og:fileQuery from = "d:/webapps/dbdef/jsp/" multi = "true" orderBy = "LASTMODIFIED" desc = "true" command = "[@command]" > <og:fileWhere lastModified="20021001000000" /> </og:fileQuery>	<p>【TAG】 ファイルの検索元となるディレクトリを指定します。 ファイルの検索元となるディレクトリを指定します。</p> <p>○ 【TAG】 ファイルの検索元となるディレクトリを指定します。(必須) 【TAG】 多段階展開するか、1 レベル展開するかどうか[true/false]を指定します(初期値:false) 【TAG】 (通常は使いません) 結果のDBTableModelを、sessionに登録するときのキーを指定します 【TAG】 多段階展開するレベルを指定します(初期値:100) 【TAG】 ソートするカラム名を指定します(一つのみ) 【TAG】 表示順を逆転するかどうか[true/false]を指定します(初期値:false) 【TAG】 検索結果のカラム列に追加するカラム名を、カンマ区切り文字で指定します 【TAG】 TO_PARENT、TO_NAMEカラムを追加するかどうか[true/false]を指定します(初期値:false) 【TAG】 ファイル名が、指定されたファイルタイプと一致した場合、スルー(選択)されます 【TAG】 from属性で指定された基準ファイル/フォルダ自体をリストに追加するかどうか[true/false]を指定します(初期値:true) 【TAG】 コマンド(NEW, RENEW)をセットします(PisqlUpdateTag, UpdateTag の場合は、ENTRY) 【TAG】 (通常は使いません) データの最大読み込み件数を指定します (初期値:DB_MAX_ROW_COUNT[=1000]) 【TAG】 検索結果を画面上に表示するメッセージリソースIDを指定します (初期値:VIEW_DISPLAY_MSG[=]) 【TAG】 検索結果がゼロ件の場合に表示するメッセージリソースIDを指定します(初期値:MSG0077[対象データはありませんでした]) 【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session) 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)</p>
1.from	【TAG】 ファイルの検索元となるディレクトリを指定します。 ファイルの検索元となるディレクトリを指定します。
2.multi	【TAG】 多段階展開するか、1 レベル展開するかどうか[true/false]を指定します(初期値:false)。 初期値は、false (1 レベル) です。
3.tableId	【TAG】 (通常は使いません) 結果のDBTableModelを、sessionに登録するときのキーを指定します。 検索結果より、DBTableModelオブジェクトを作成します。これを、以下のview 等のタグに渡す場合に、通常は、session を利用します。その場合の登録キーです。 query タグを同時に実行して、結果を求める場合、同一メモリに配置される為、この tableId 属性を利用して、メモリ空間を分けます。 初期値は、HybsSystem.TBL_MD_L_KEY です。
4.level	【TAG】 多段階展開するレベルを指定します(初期値:100)。
5.orderBy	【TAG】 ソートするカラム名を指定します(一つのみ)。 ソートするカラム名を、“LEVEL”、“FILE_TYPE”、“PARENT”、“NAME”、“LASTMODIFIED”、“FILE_LENGTH”、“RWH”から一つ選びます。 これは、複数カラムでのソートはできません。 逆順にソートする場合は、desc属性を true にセットください。 + をつけても、無効(カラム名がないということでエラーになります)。 【TAG】 表示順を逆転するかどうか[true/false]を指定します(初期値:false)。
6.desc	orderBy 属性で指定した表示順を、逆順にするかどうかを指定できます。 初期値は、false (昇順) です。
7.addCims	【TAG】 検索結果のカラム列に追加するカラム名を、カンマ区切り文字で指定します。 デフォルトのカラム名、“LEVEL”、“FILE_TYPE”、“PARENT”、“NAME”、“LASTMODIFIED”、“FILE_LENGTH”、“RWH”以外に、指定のカラム名を追加することが可能です。 これは、ファイル検索結果以外の項目を追加して、データベースに書き込む場合に、利用できます。 並び順は、デフォルトカラムの後ろに、指定のカラムの順番で付きます。 ここで追加したカラムには、値はセットされていません。よって、ソート対象のカラムにも指定できません。
8.useUpdateCims	【TAG】 TO_PARENT、TO_NAMEカラムを追加するかどうか[true/false]を指定します(初期値:false)。 fileUpdateタグでは、ファイルのコピーやMOVEが出来ますが、そのコピー先、移動先のファイルを行ごとに指定する場合、TO_PARENT、TO_NAMEカラムという固定名のカラムが必要で、これを、addCims 属性で指定する代わりに、この属性で、true をセットすることで、自動的に追加されます。 初期値は、false (追加しない) です。
9.fileType	【TAG】 ファイル名が、指定されたファイルタイプと一致した場合、スルー(選択)されます。 大文字小文字は区別しません。 ファイルタイプ は、DIR, FILE が指定できます。 DIR は、ディレクトリのみ検索します。(階層下がりも行います) FILEは、ファイルのみ検索します。(階層下がりも行います) 引数が null の場合は、追加しません。(つまり、すべてスルーされます。)
10.addFrom	【TAG】 from属性で指定された基準ファイル/フォルダ自体をリストに追加するかどうか[true/false]を指定します(初期値:true)。 初期値はtrue(追加する)です。
11.command	【TAG】 コマンド(NEW, RENEW)をセットします(PisqlUpdateTag, UpdateTag の場合は、ENTRY)。 コマンドは、HTMLから(get/post) 指定されますので、CMD_xxx で設定されるフィールド定数値のいずれかを、指定できます。
12.maxRowCount	【TAG】 (通常は使いません) データの最大読み込み件数を指定します (初期値:DB_MAX_ROW_COUNT[=1000])。 データベース自体の検索は、指定されたSQLの全件を検索しますが、DBTableModelのデータとして登録する最大件数をこの値に設定します。 サーバーのメモリ資源と応答時間の確保の為です。 0 をセットすると、無制限(Integer, MAX_VALUE)になります。 (初期値:ユーザー定数のDB_MAX_ROW_COUNT[=@og.value.org.opengion.hayabusa.common.SystemData#DB_MAX_ROW_COUNT])。
	【TAG】 検索結果を画面上に表示するメッセージリソースIDを指定します

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

13.displayMsg

(初期値:VIEW_DISPLAY_MSG[=])。

ここでは、検索結果の件数や登録された件数をまず出力し、その次に、ここで指定したメッセージをリソースから取得して表示します。

件数を表示させたい場合は、displayMsg = "MSG0033"[件検索しました] をセットしてください。

表示させたくない場合は、displayMsg = "" をセットしてください。

(初期値:システム定数のVIEW_DISPLAY_MSG[=@og.value org.opengion.hayabusa.common.SystemData#VIEW_DISPLAY_MSG])。

【TAG】 検索結果がゼロ件の場合に表示するメッセージリソースIDを指定します(初期値:MSG0077[対象データはありませんでした])。

ここでは、検索結果がゼロ件の場合のみ、特別なメッセージを表示させます。

従来は、displayMsg と兼用で、『0 件検索しました』という表示でしたが、displayMsg の初期表示は、OFF になりましたので、ゼロ件の場合のみ別に表示させます。

表示させたくない場合は、notfoundMsg = "" をセットしてください。

初期値は、MSG0077[対象データはありませんでした]です。

14.notfoundMsg

【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)。

"request","page","session","applicaton" が指定できます。

JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、主にテーブルモデルを管理するスコープを指します。

15.scope

スコープ	変数の有効範囲	
page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会
request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ
session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。 内部的には、同じキーワー
application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。

16.debug

【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。

デバッグ情報を [true:出力する/false:しない]を指定します。

出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
5.5.4.2 og.fileUD JSP	<p>【TAG】command を指定します。 command を指定します。 ダウンロードは、“NEW RENEW FIRST LAST PREV NEXT VIEW” のみ有効です。 アップロードは、“COPY INSERT” のみ有効です。 それ以外のコマンドでは、アイコンは表示されません。 例えば、NEXT/PREV など。</p> <p>●形式：一般ユーザーが直接組み込むことはありません。 ●body：あり</p> <p>●Tag定義： <pre> <og:fileUD command downloadImg downloadJsp uploadImg uploadJsp imgWidth imgHeight filename roles target errorMsg typeDownload editImg debug > ... Body ... </og:fileUD> </pre> </p> <p>【TAG】command を指定します 【TAG】ダウンロード時のアイコンを指定します(初期値:“{@SYS. JSP}/image/fileDownload. gif”) 【TAG】ダウンロード時の呼び出しJSPを指定します(初期値:“{@SYS. JSP}/common/fileDownload. jsp”) 【TAG】アップロード時のアイコンを指定します(初期値:“{@SYS. JSP}/image/fileUpload. gif”) 【TAG】アップロード時の呼び出しJSPを指定します(初期値:“{@SYS. JSP}/common/fileUpload. jsp”) 【TAG】アイコン幅サイズ(px)を指定します(初期値:20px) 【TAG】アイコン高さサイズ(px)を指定します(初期値:20px) 【TAG】ダウンロード時ファイル名を指定します(初期値:“{@GUI. LABEL}. xls”) 【TAG】ロールズをセットします 【TAG】ダウンロード文書を表示させるフレーム、またはウィンドウの名前を指定します 【TAG】ダウンロード出来ない場合のエラーメッセージIDを指定します(初期値:ERROR028) 【TAG】Download 処理を行うかどうかを指定します(初期値:false) 【TAG】編集アイコンを指定します(初期値:“{@SYS. JSP}/image/edit. png”) 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)</p> <p>●使用例 <pre> <og:fileUD command=“{@command}” /> <og:fileUD command = command設定 (ダウンロード=“NEW RENEW FIRST LAST PREV NEXT VIEW”、 アップロード=“COPY INSERT”) [downloadImg] = “{@SYS. JSP}/image/fileDownload. gif” ダウンロード時のアイコン [downloadJsp] = “{@SYS. JSP}/common/fileDownload. jsp” ダウンロード時の呼び出しJSP [uploadImg] = “{@SYS. JSP}/image/fileUpload. gif” アップロード時のアイコン [uploadJsp] = “{@SYS. JSP}/common/filePopop. jsp” アップロード時の呼び出しJSP [imgWidth] = “20px” アイコン幅サイズ(px) [imgHeight] = “20px” アイコン高さサイズ(px) [filename] = “{@GUI. LABEL}. xls” ダウンロード時ファイル名 [roles] = “GE DOC” アクセスロールズ [target] = “_blank” ダウンロードファイルを開くターゲット [errorMsg] = “ERROR028” エラーメッセージID [typeDownload] = “false/true” Download 処理を行うかどうか(初期値:false) /> </pre> </p>
1.command	<p>【TAG】command を指定します。 command を指定します。 ダウンロードは、“NEW RENEW FIRST LAST PREV NEXT VIEW” のみ有効です。 アップロードは、“COPY INSERT” のみ有効です。 それ以外のコマンドでは、アイコンは表示されません。 例えば、NEXT/PREV など。</p>
2.downloadImg	<p>【TAG】ダウンロード時のアイコンを指定します(初期値:“{@SYS. JSP}/image/fileDownload. gif”) ダウンロード時に表示されるアイコンを指定します。 初期値:“{@SYS. JSP}/image/fileDownload. gif” です。</p>
3.downloadJsp	<p>【TAG】ダウンロード時の呼び出しJSPを指定します(初期値:“{@SYS. JSP}/common/fileDownload. jsp”) ダウンロード時に呼び出されるJSPファイル名を指定します。 初期値:“{@SYS. JSP}/common/fileDownload. jsp” です。</p>
4.uploadImg	<p>【TAG】アップロード時のアイコンを指定します(初期値:“{@SYS. JSP}/image/fileUpload. gif”) アップロード時に表示されるアイコンを指定します。 初期値:“{@SYS. JSP}/image/fileUpload. gif” です。</p>
5.uploadJsp	<p>【TAG】アップロード時の呼び出しJSPを指定します(初期値:“{@SYS. JSP}/common/fileUpload. jsp”) アップロード時に呼び出されるJSPファイル名を指定します。 初期値:“{@SYS. JSP}/common/fileUpload. jsp” です。</p>
6.imgWidth	<p>【TAG】アイコン幅サイズ(px)を指定します(初期値:20px)。 アップロード/ダウンロードのアイコンを表示する際の幅サイズ(px)を指定します。 ファイルサイズは、アップロード/ダウンロード共通です。 初期値は、20px です。</p>
7.imgHeight	<p>【TAG】アイコン高さサイズ(px)を指定します(初期値:20px)。 アップロード/ダウンロードのアイコンを表示する際の高さサイズ(px)を指定します。 ファイルサイズは、アップロード/ダウンロード共通です。 初期値は、20px です。</p>
8.filename	<p>【TAG】ダウンロード時ファイル名を指定します(初期値:“{@GUI. LABEL}. xls”) ダウンロード時のファイル名を指定します 通常は、画面のラベル(言語対応)+XLS拡張子です。 初期値は、“{@GUI. LABEL}. xls” です。</p>
9.roles	<p>【TAG】ロールズをセットします。 ここで指定したロールを元に、ユーザー毎のアクセス許可がチェックされます。 アクセス許可されないと、表示されません。 このロールを指定しない場合は、画面のロールが使用されます。</p>
10.target	<p>【TAG】ダウンロード文書を表示させるフレーム、またはウィンドウの名前を指定します。 フレーム名(ターゲット属性)を設定します。 【TAG】ダウンロード出来ない場合のエラーメッセージIDを指定します(初期値:ERROR028)。 DBTableModel が、最後に作成された画面と、メモリ中の DBLastSql オブジェクトが 管理している情報が異なる場合は、ダウンロード出来ません。 同様に、DBLastSql オブジェクト 自身が作成されていない場合も出来ません。 そのようなケースに表示するエラーメッセージのメッセージIDを指定します。 引数は、指定できません。 初期値は、『ERROR028: 選択されていません。もう一度、選択しなおして下さい。』です。</p>
11.errorMsg	
12.typeDownload	<p>【TAG】Download 処理を行うかどうかを指定します(初期値:false)。 Download 処理を行うには、指定の画面(呼び出し元)と、抜き出した時の DBTableModel がマッチしているか判定します。 また、DBTableModel が存在しない、または、不備な場合(オーバーフロー)は、 最終SQLを実行する必要がある、そもそも、抜出処理が出来ない状態では、 エラーを返す必要があります。 それら、一連のDownload 処理を行うかどうかを指定します。 true の場合、session の HybsSystem. DB. LAST. SQL. KEY に格納された DBLastSql オブジェクトを取得し、scope が request または、overflow が true の場合に、BODY部を実行します。(通常、queryタグを記述します。) query の出力先は、scope を request にします。 それ以外は、tableId から取り出した DBTableModel を、scope = “request” に(アドレスのみ)コピーします。 writeTable タグでは、scope = “request” から、DBTableModel を取り出します。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
13.editImg	初期値は、false(抜き出しでは無くリンク作成)です。 【TAG】編集アイコンを指定します(初期値:"{@SYS. JSP}/image/edit.png")。 編集アイコンを指定します。 初期値:"{@SYS. JSP}/image/edit.png" です。
14.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

5.5.4.2
og:fileUpdate

【TAG】アクション(SAVE, LOAD, DELETE)をセットします。
アクションは、HTMLから(get/post)指定されますので、ACT_xxx で設定される
フィールド定数値のいつれかを、指定できます。
無指定の場合は、なにもしません。

action	名称	機能
SAVE	登録	指定の keys のキーに vals の値をセットします。
LOAD	取得	指定の keys のクッキーを(リクエスト中に)取得します。
DELETE	削除	指定の keys のクッキーを削除します。

●body : なし
●形式 :

・<og:fileUpdate
 action = "COPY|MOVE|MODIFY|DELETE" アクション属性(必須)
 command = "[ENTRY]" ENTRY 時のみ実行します(初期値:ENTRY)
 targetDir = "[指定フォルダ]" ターゲットとなるフォルダ
 createDir = "[true/false]" ターゲットとなるフォルダがなければ作成する(true)かどうか(初期値:true)
 tableId = [HybsSystem.TBL_MDL_KEY] DBTableModel を取り出すキー
 outMessage = "[true/false]" 検索結果のメッセージを表示する(true)かどうかを指定(初期値:true)
 displayMsg = "MSG0040"; 処理結果を表示します(初期値:「 件登録しました。」)
 selectedAll = "[false/true]" データを全件選択済みとして処理する(true)かどうか指定(初期値:false)
 keepTimeStamp = "[false/true]" COPY, 親違いMOVE(.MODIFY)の時にオリジナルのタイムスタンプを使用するかどうか(初期値:false)
/>

[action属性(必須)]
COPY オリジナルファイルを、ターゲットにコピーします。
MOVE オリジナルファイルを、ターゲットに移動(COPY+DELETE)/名称変更(RENAME)します。
MODIFY (MOVE と同じ。エンジンの command を利用するための簡易action)
DELETE オリジナルファイルを、削除します。(フォルダ、ファイルに関わらず)

●Tag定義 :

<og:fileUpdate
 action
 command
 targetDir
 createDir
 tableId
 outMessage
 displayMsg
 selectedAll
 keepTimeStamp
 scope
 debug
>

○【TAG】アクション(SAVE, LOAD, DELETE)をセットします。(必須)
【TAG】 コマンド(ENTRY)をセットします
【TAG】 ターゲットとなるフォルダを指定します
【TAG】 ターゲットとなるフォルダがなければ、作成するかどうかを指定します(初期値:true)
【TAG】 (通常は使いません)結果のDBTableModelを、sessionに登録するときのキーを指定します
【TAG】 検索結果のメッセージを表示する/しない[true/false]を指定します(初期値:true)
【TAG】 処理結果を画面上に表示するメッセージリソースIDを指定します(初期値:MSG0040[件登録しました])
【TAG】 データを全件選択済みとして処理するかどうか[true/false]を指定します(初期値:false)
【TAG】 オリジナルのタイムスタンプを利用するかどうかを指定します(初期値:false)
【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)
【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)

●使用例
・<og:fileUpdate command="@[command]" action="COPY" />
 TO_PARENT または、 TO_NAME(両方指定も可)による行単位 COPY 処理
 fileQuery の useUpdateCln="true"を設定し、検索結果に、TO_PARENT、 TO_NAMEカラムを追加します。
 TO_PARENT または、 TO_NAME は、columnSet などて値をセットしておきます。
・<og:fileUpdate command="@[command]" action="MODIFY" targetDir="AAA_DIR" />
 fileQuery の検索結果を、AAA_DIR フォルダに移動します。
 ファイル名は、そのままオリジナルの値が使用されます。

1.action

【TAG】アクション(SAVE, LOAD, DELETE)をセットします。
アクションは、HTMLから(get/post)指定されますので、ACT_xxx で設定される
フィールド定数値のいつれかを、指定できます。
無指定の場合は、なにもしません。

action	名称	機能
SAVE	登録	指定の keys のキーに vals の値をセットします。
LOAD	取得	指定の keys のクッキーを(リクエスト中に)取得します。
DELETE	削除	指定の keys のクッキーを削除します。

2.command

【TAG】コマンド(ENTRY)をセットします。
このタグは、command="ENTRY"でのみ実行されます。
コマンドは、HTMLから(get/post)指定されますので、CMD_xxx で設定される
フィールド定数値のいつれかを、指定できます。
初期値は、ENTRY なので、何も指定しなければ、実行されます。
【TAG】ターゲットとなるフォルダを指定します。
targetDir 属性を利用する場合は、引数のファイル、またはフォルダが指定されたことにな
ります。COPY、MOVE(.MODIFY)の場合は、targetDir 属性にフォルダを指定することで一括処理可能です。
指定先のフォルダが存在しない場合は、createDir属性の値により処理が異なります。
createDir="true"(初期値)で、ターゲットフォルダが存在しない場合は、自動作成します。
【TAG】ターゲットとなるフォルダがなければ、作成するかどうかを指定します(初期値:true)。
COPY、MOVE(.MODIFY)などの処理で、ターゲットフォルダが存在しないときに、作成するか、エラーにするかを
createDir属性 で指定できます。
これは、COPY先やMOVE(.MODIFY)先が存在している前提のシステムで、不要な箇所に間違っ
てフォルダを自動作成されると困る場合に、false:作成しない とすれば、間違いに気づく確率が上がります。
初期値は true:作成する です。

3.targetDir

【TAG】(通常は使いません)結果のDBTableModelを、sessionに登録するときのキーを指定します。
検索結果より、DBTableModelオブジェクトを作成します。これを、以下のview 等のタグに
渡す場合に、通常は、session を利用します。その場合の登録キーです。
query タグを同時に実行して、結果を求める場合、同一メモリに配置される為、
この tableId 属性を利用して、メモリ空間を分けます。
初期値は、HybsSystem.TBL_MDL_KEY です。

4.createDir

【TAG】検索結果のメッセージを表示する/しない[true/false]を指定します(初期値:true)。
初期値は、表示する : true です。
【TAG】処理結果を画面上に表示するメッセージリソースIDを指定します(初期値:MSG0040[件登録しました])。
ここでは、検索結果の件数や登録された件数をまず出力し、
その次に、ここで指定したメッセージをリソースから取得して表示します。
表示させたくない場合は、displayMsg = "" をセットしてください。
displayMsg の初期値は、MSG0040[件登録しました]です。
【TAG】データを全件選択済みとして処理するかどうか[true/false]を指定します(初期値:false)。
全てのデータを選択済みデータとして扱って処理します。
全件処理する場合に、(true/false)を指定します。
初期値は false です。

5.tableId

【TAG】オリジナルのタイムスタンプを利用するかどうかを指定します(初期値:false)。
COPYや親違いMOVE(.MODIFY)の時に、オリジナルのタイムスタンプをそのままコピー先のファイルにも
適用するかどうかを指定します。

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

タイムスタンプを初期化されたくない場合に、true に設定します。
初期値は 利用しない:false です。
【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)。
"request", "page", "session", "applicaton" が指定できます。
JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、
主にテーブルモデルを管理するスコープを指します。

スコープ	変数の有効範囲	
page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会
request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ
session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。 内部的には、同じキーワー
application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。

10.scope

【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。
デバッグ情報を [true:出力する/false:しない]を指定します。
出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:fileWhere	【TAG】 指定された接頭辞で始まる場合、スルー（選択）されます。 引数が null の場合は、追加しません。
Tag	
●形式 : <og:fileWhere name="..." startsWith="..." endsWith="..." ... />	
●body : なし	
●Tag定義 :	
<og:fileWhere	
startsWith	【TAG】 指定された接頭辞で始まる場合、スルー（選択）されます
endsWith	【TAG】 指定された接頭辞で終わる場合、スルー（選択）されます
instr	【TAG】 指定された文字列がファイル名に含まれる場合、スルー（選択）されます
equals	【TAG】 ファイル名が一致する場合、スルー（選択）されます
matches	【TAG】 ファイル名が、指定された正規表現と一致する場合、スルー（選択）されます
unMatches	【TAG】 ファイル名が、指定された正規表現と一致しない場合、スルー（選択）されます
lastModified	【TAG】 指定のタイムスタンプ以後に変更されている場合、スルー（選択）されます
isLarger	【TAG】 指定の大きさより大きいファイルの場合、スルー（選択）されます
isSmaller	【TAG】 指定の大きさより小さいファイルの場合、スルー（選択）されます
useDir	【TAG】 フィルタによる対象の絞込みをディレクトリにも適用するかどうかを指定します
debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)
/>	
●使用例	
<og:fileOption select="NAME" from="." /jsp" >	
<og:fileWhere	
startsWith = "File"	:指定された接頭辞で始まる場合、選択される
endsWith = ".java"	:指定された接頭辞で終わる場合、選択される
instr = "Tag"	:指定された文字列がファイル名に含まれる場合、選択される
fileEquals = "FileWhereTag.java"	:ファイル名が一致する場合、選択される(大文字小文字は区別しない)
matches = "File*Tag"	:ファイル名が、指定された正規表現と一致する場合、選択される(大文字小文字は区別しない)
unMatches = "File*Tag"	:ファイル名が、指定された正規表現と一致しない場合、選択される(大文字小文字は区別しない)
lastModified = "20050101"	:指定のタイムスタンプ以後に変更された場合、選択される 日付けの指定は、YYYYMMDD 形式、TODAY , YESTERDAY が使用できます。
isLarger = "1024"	:指定の大きさ(バイト単位)より大きいファイルの場合、選択される
isSmaller = "1024"	:指定の大きさ(バイト単位)より小さいファイルの場合、選択される
useDir = "false"	:フィルタによる対象の絞込みをディレクトリにも適用するかどうか
/>	
</og:fileOption>	
・検索条件が入力された時	
条件を満足して全てのファイルを列挙します。	
・検索条件が入力されなかった時	
該当ディレクトリ内の全てのファイルを列挙します。	
1.startsWith	【TAG】 指定された接頭辞で始まる場合、スルー（選択）されます。 引数が null の場合は、追加しません。
2.endsWith	【TAG】 指定された接頭辞で終わる場合、スルー（選択）されます。 引数が null の場合は、追加しません。
3.instr	【TAG】 指定された文字列がファイル名に含まれる場合、スルー（選択）されます。 引数が null の場合は、追加しません。
4.equals	【TAG】 ファイル名が一致する場合、スルー（選択）されます。 大文字小文字は区別しません。 引数が null の場合は、追加しません。
5.matches	【TAG】 ファイル名が、指定された正規表現と一致する場合、スルー（選択）されます。 大文字小文字は区別しません。 Pattern.compile(str.Pattern.CASE_INSENSITIVE) ; pattern.matcher(pathname.getName()).find() == true と同じ結果が得られます。 引数が null の場合は、追加しません。
6.unMatches	【TAG】 ファイル名が、指定された正規表現と一致しない場合、スルー（選択）されます。 大文字小文字は区別しません。 Pattern.compile(str.Pattern.CASE_INSENSITIVE) ; pattern.matcher(pathname.getName()).find() == false と同じ結果が得られます。 引数が null の場合は、追加しません。
7.lastModified	【TAG】 指定のタイムスタンプ以後に変更されている場合、スルー（選択）されます。 ディレクトリは、ここの判定では無視します。(必ず true を返します) 日付けの指定に、YYYYMMDD 形式の 8 文字数字文字列以外に、 TODAY や YESTERDAY なども使用できます。 TODAY は、実行日の 00:00:00 を基準時刻とし、YESTERDAY は、その前日になります。 引数が null の場合は、追加しません。
8.isLarger	【TAG】 指定の大きさより大きいファイルの場合、スルー（選択）されます。 ファイルの大きさ(バイト単位)は同値を含みます。 引数が 0以下(マイナス) の場合は、追加しません。
9.isSmaller	【TAG】 指定の大きさより小さいファイルの場合、スルー（選択）されます。 ファイルの大きさ(バイト単位)は同値を含みません。 引数が 0以下(マイナス) の場合は、追加しません。
10.useDir	【TAG】 フィルタによる対象の絞込みをディレクトリにも適用するかどうかを指定します。 初期値は、 false (ディレクトリはフィルタによる絞込みをしない) です。
11.debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

5.5.4.2
og:forward

Tag

【TAG】 gameId 属性を登録します。
gameId 属性は、画面IDをキーに、実アドレスを求めるのに使用します。
画面IDが指定された場合は、実アドレスに変換する。指定されない場合は、
page 属性をそのままダイレクトアドレスとします。
gameId を指定した場合は、このuseRedirect属性に関係なく、Redirect されます。

●形式 : <og:forward page=“...” gameId=“[...]” />
●body : なし

●Tag定義 :
<og:forward
gameId
page
keys
vals
dbkeys
scope
useRedirect
target
filename
noTransition
useAjaxSubmit
debug
>

●使用例
<og:forward
page
dbkeys
keys
vals
scope
gameId
useRedirect
filename
target
noTransition
>

1.gameId

2.page

3.keys

4.vals

5.dbkeys

6.scope

7.useRedirect

8.target

9.filename

10.noTransition

11.useAjaxSubmit

12.debug

【TAG】 gameId 属性を登録します。
gameId 属性は、画面IDをキーに、実アドレスを求めるのに使用します。
画面IDが指定された場合は、実アドレスに変換する。指定されない場合は、
page 属性をそのままダイレクトアドレスとします。
gameId を指定した場合は、このuseRedirect属性に関係なく、Redirect されます。

【TAG】 転送先URLを指定します。
転送先URLを指定します。

【TAG】 キーを CSV 形式でセットします。
URI の引数にセットするキーを CSV 形式でセットします。

【TAG】 キーに対応する値を CSV 形式でセットします。
URI の引数にセットする値を CSV 形式でセットします。
分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。
こうしないとデータ自身にカンマを持っている場合に分解をミスの為です。

【TAG】 DBキーをCSV 形式でセットします。
URI の引数にセットするキーを CSV 形式でセットします。
この指定は、DBTableModel 上のデータを取り込みます。
【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)。
“request”, “page”, “session”, “applicaton” が指定できます。
JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、
主にテーブルモデルを管理するスコープを指します。

スコープ	変数の有効範囲	
page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会
request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ
session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。 内部的には、同じキーワー
application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。

【TAG】 response.sendRedirect するかどうか[true/false]を指定します(初期値:false)。
useRedirect 属性は、URLの転送方式を指定できます。
通常は、pageContext.forward しますが、この属性に true を
設定すると、response.sendRedirect します。
gameId を指定した場合は、この属性に関係なく、Redirect されます。
初期値は、false (forward) です。

【TAG】 サブミット先の文書を表示させるフレーム、またはウィンドウの名前を指定します。
サブミット先のフレーム名(ターゲット属性)を設定します。

【TAG】 ファイル転送時にファイル名の別名を指定します(初期値:null)。
ファイルを転送する場合に、オープンさせるファイル名を指定します。
指定しない場合は、フォワード前に呼び出されたアドレスがそのまま使用されます。
【TAG】 (通常は使いません)画面遷移を行わない形式の登録方法を使用するかを指定します。
画面遷移なしの登録を行うかどうかを指定します。
trueが指定された場合、forwardタグでは、実際の画面遷移は行わずに、forward先の
URLを
[URL]
の形でHTMLとして出力し、JavaScript側で
実際の画面遷移を発生させます。target属性は
[TARGET]
で
出力されます。

この値は、og:headタグで設定値、または前画面からの値を継承するため、通常、この属性ではセットしません。
【TAG】 (通常は使いません) ajaxを利用したsubmitを利用するかどうかを指定します。
ajaxを利用したSubmit処理を行うかどうかを指定します。
setNoTransition=trueにした場合と同様の処理を行います。
(forwardせずにURLをDIVタグで出力する)

この値は、og:headタグで設定値、または前画面からの値を継承するため、通常、この属性ではセットしません。
【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。
デバッグ情報を [true:出力する/false:しない]を指定します。
出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTML EXT BODY要素	内容
形式サンプル	
5.5.4.2 og:frame 拡張 Tag ●形式: <og:frame src="..." name="..." /> ●body: なし ●Tag定義: <og:frame src name changeGamen keys dbTable longdesc marginwidth marginheight noresize scrolling frameborder id clazz title style debug > ●使用例: ・一般的な例: フレーム分割する構文は、HTML準拠。リクエスト変数は各フレームまで転送されます。 <frameset> <og:frame marginheight="2" marginwidth="2" src="query.jsp" name="QUERY" /> <og:frame marginheight="2" marginwidth="2" src="forward.jsp" name="RESULT" /> </frameset> ・DBTableModel の値(例ではPN)を、取り出して、リクエスト変数として利用します。 現状では、commonForward タグ の useTableData="true" dbkeys="@dbkeys" 属性を利用します。 <frameset> <og:frame marginheight="2" marginwidth="2" src="query.jsp" name="QUERY" keys="PN" dbTable="true" /> <og:frame marginheight="2" marginwidth="2" src="forward.jsp" name="RESULT" keys="PN" dbTable="true" /> </frameset> ・changeGamen 属性を利用して、ソース指定の画面を切り替えます。 たとえば、jsp/index.jsp では、GAMENID属性がURLに存在する場合、直接その画面を表示させることができます。 <frameset cols="160,*" frameborder="1" framespacing="1"> <og:frame marginheight="2" marginwidth="2" src="menu/menu.jsp" name="MENU" /> <og:frame marginheight="2" marginwidth="2" src="GE0000/index.jsp" name="CONTENTS" changeGamen="@GAMENID" /> </frameset>	【HTML】フレームに表示するソースファイルを指定します。 フレームに表示するソースファイルを指定します。 ○【HTML】フレームに表示するソースファイルを指定します。(必須) ○【HTML】フレーム名を指定します(例: QUERY, RESULT, CONTENTS など)。(必須) 【TAG】ソース指定の画面を変更します 【TAG】引数を指定します 【TAG】前ページで選択したデータ列の情報を次のページに渡すかどうか[true/false]を指定します(初期値:false) 【HTML】フレームに関する詳しい説明のあるURL (lobgdesc)を指定します 【HTML】フレームの左右余白サイズ (marginwidth)を指定します 【HTML】フレームの上下余白サイズ (marginheight)を指定します 【HTML】フレームサイズを変更できないよう (noresize)に指定します 【HTML】スクロールバー (scrolling)の表示/非表示 [auto/yes/no]を指定します(初期値:auto) 【HTML】フレームの境界線 (frameborder)の [0:非表示/1:表示]を指定します(初期値:1) 【HTML】要素に対して固有の名前 (id)をつける場合に設定します 【HTML】要素に対して class 属性を設定します 【HTML】要素に対する補足的情報 (title)を設定します 【HTML】この要素に対して適用させるスタイルシート (style)を設定します 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false) 【HTML】フレームに表示するソースファイルを指定します。 フレームに表示するソースファイルを指定します。 【TAG】ソース指定の画面を変更します。 src 指定がデフォルト設定で、changeGamen属性が設定されている場合には、この値が優先されます。 changeGamen は、画面IDのみ指定してください。src には、このID+"/index.jsp" が追加されます。つまり、changeGamen="@GAMENID" という指定をしておけば、FavoriteLinkTag 等で引数に GAMENID が指定された場合のみ、この属性が有効になり、src="実画面ID/index.jsp" が指定されたことと同じ結果になります。 【TAG】引数を指定します。 URL の引数にセットするキーを カンマ区切りでセットします。 【TAG】前ページで選択したデータ列の情報を次のページに渡すかどうか[true/false]を指定します(初期値:false)。 ただし、1行分のデータのみです。(複数選択時は、最初の1件目) true: 渡す false: 渡さない。 初期値は、false: 渡さない です。 【HTML】フレームに関する詳しい説明のあるURL (lobgdesc)を指定します。 lobgdescを指定します。 【HTML】フレームの左右余白サイズ (marginwidth)を指定します。 フレームの左右余白サイズを指定します。 【HTML】フレームの上下余白サイズ (marginheight)を指定します。 フレームの上下余白サイズを指定します。 【HTML】フレームサイズを変更できないよう (noresize)に指定します。 フレームサイズを変更できないように指定します。 【HTML】スクロールバー (scrolling)の表示/非表示 [auto/yes/no]を指定します(初期値:auto)。 auto: 必要に応じてスクロールバーを表示(初期値) yes: 常にスクロールバーを表示 no: 常にスクロールバーを表示しない 【HTML】フレームの境界線 (frameborder)の [0:非表示/1:表示]を指定します(初期値:1)。 0: 枠を表示しない 1: 枠を表示する。 初期値は、1: 枠を表示する です。 【HTML】要素に対して固有の名前 (id)をつける場合に設定します。 特別な使用方法として、id="FOCUS" とすることで、フィールド系要素にフォーカスを移動させます。これは、そのページ内で唯一の id 属性として使用ください。 【HTML】要素に対して class 属性を設定します。 Javaの言語使用上、class で作成できないため、代用として clazz を使用しています。 html で作成される属性は、class で作成されます。 【HTML】要素に対する補足的情報 (title)を設定します。 title セットは、ボタンなどに適用すると、マウスオーバーによりこのメッセージがチップスのように表示されます。これを利用して、説明文を登録することが可能です。ここに登録した文字列が、メッセージリソースに存在する場合は、そのメッセージを存在しない場合は、そのままの値を設定します。 【HTML】この要素に対して適用させるスタイルシート (style)を設定します。 タグにstyle属性を設定します。これは、キー:値: のセットを複数記述できます。 通常は、class属性や、id属性で登録しておき、&lg:style type="text/css"> で外部から指定する方がソースは読みやすくなります。 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。
1.src	【HTML】フレームに表示するソースファイルを指定します。 フレームに表示するソースファイルを指定します。
2.name	【HTML】フレーム名を指定します(例: QUERY, RESULT, CONTENTS など)。 フレーム名を指定します。
3.changeGamen	【TAG】ソース指定の画面を変更します。 src 指定がデフォルト設定で、changeGamen属性が設定されている場合には、この値が優先されます。 changeGamen は、画面IDのみ指定してください。src には、このID+"/index.jsp" が追加されます。つまり、changeGamen="@GAMENID" という指定をしておけば、FavoriteLinkTag 等で引数に GAMENID が指定された場合のみ、この属性が有効になり、src="実画面ID/index.jsp" が指定されたことと同じ結果になります。
4.keys	【TAG】引数を指定します。 URL の引数にセットするキーを カンマ区切りでセットします。
5.dbTable	【TAG】前ページで選択したデータ列の情報を次のページに渡すかどうか[true/false]を指定します(初期値:false)。 ただし、1行分のデータのみです。(複数選択時は、最初の1件目) true: 渡す false: 渡さない。 初期値は、false: 渡さない です。
6.longdesc	【HTML】フレームに関する詳しい説明のあるURL (lobgdesc)を指定します。 lobgdescを指定します。
7.marginwidth	【HTML】フレームの左右余白サイズ (marginwidth)を指定します。 フレームの左右余白サイズを指定します。
8.marginheight	【HTML】フレームの上下余白サイズ (marginheight)を指定します。 フレームの上下余白サイズを指定します。
9.noresize	【HTML】フレームサイズを変更できないよう (noresize)に指定します。 フレームサイズを変更できないように指定します。
10.scrolling	【HTML】スクロールバー (scrolling)の表示/非表示 [auto/yes/no]を指定します(初期値:auto)。 auto: 必要に応じてスクロールバーを表示(初期値) yes: 常にスクロールバーを表示 no: 常にスクロールバーを表示しない
11.frameborder	【HTML】フレームの境界線 (frameborder)の [0:非表示/1:表示]を指定します(初期値:1)。 0: 枠を表示しない 1: 枠を表示する。 初期値は、1: 枠を表示する です。
12.id	【HTML】要素に対して固有の名前 (id)をつける場合に設定します。 特別な使用方法として、id="FOCUS" とすることで、フィールド系要素にフォーカスを移動させます。これは、そのページ内で唯一の id 属性として使用ください。
13.clazz	【HTML】要素に対して class 属性を設定します。 Javaの言語使用上、class で作成できないため、代用として clazz を使用しています。 html で作成される属性は、class で作成されます。
14.title	【HTML】要素に対する補足的情報 (title)を設定します。 title セットは、ボタンなどに適用すると、マウスオーバーによりこのメッセージがチップスのように表示されます。これを利用して、説明文を登録することが可能です。ここに登録した文字列が、メッセージリソースに存在する場合は、そのメッセージを存在しない場合は、そのままの値を設定します。
15.style	【HTML】この要素に対して適用させるスタイルシート (style)を設定します。 タグにstyle属性を設定します。これは、キー:値: のセットを複数記述できます。 通常は、class属性や、id属性で登録しておき、&lg:style type="text/css"> で外部から指定する方がソースは読みやすくなります。
16.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og.ganttHeader	【TAG】 表示開始日付けを設定します (yyyyMMdd または、yyyyMMddHHmm 形式)。 この値は、ヘッダーの先頭に表示する日付けを指定します。 通常の日付ヘッダーの場合は、 yyyyMMdd 形式ですが、時間ヘッダーの場合は、 yyyyMMddHHmm 形式で指定します。
Tag	●形式 : <og.ganttHeader startDate="..." endDate="..." zoom="..." /> ●body : なし
●Tag定義 :	●Tag定義 : <og.ganttHeader startDate 【TAG】 表示開始日付けを設定します (yyyyMMdd または、yyyyMMddHHmm 形式) endDate 【TAG】 表示最終日を設定します (yyyyMMdd または、yyyyMMddHHmm 形式) zoom 【TAG】 ズーム変数 (0 ~ 9) を設定します (初期値:2) calDB 【TAG】 DB検索するDBを指定します arg1 【TAG】 DB検索する場合の第1のキーを指定します arg2 【TAG】 DB検索する場合の第2のキーを指定します arg3 【TAG】 DB検索する場合の第3のキーを指定します arg4 【TAG】 DB検索する場合の第4のキーを指定します daySpan 【TAG】 1日の開始終了時間を24時間形式 (HHmm-HHmm) で設定します (初期値:0700-2000) breakTimes 【TAG】 時間軸 (zoom=20) での休憩時間の開始-終了時刻をカンマ区切りで複数指定します breakTimeStyle 【TAG】 時間軸 (zoom=20) での休憩時間の表示スタイルを設定します debug 【TAG】 デバッグ情報を出力するかどうか [true/false] を指定します (初期値: false) skipHoliday 【TAG】 時間ガントで休日を表示する/しないを設定します (初期値: USE_GANTTHEADER_SKIP_HOLIDAY [=true]) </>
●使用例	参照 : jsp/GANTT06 フォルダ
	<pre><og.ganttHeader startDate = "[@VIEW_DATE]" endDate = "[@DYMAX]" zoom = "[@VIEW_ZOOM]" calDB = "GE13" arg1 = "[@CDJGS]" arg2 = "" </og.ganttHeader> <og.ganttHeader startDate = "[@VIEW_DATE]" endDate = "[@DYMAX]" daySpan = "0700-2000" breakTimes = "1200-1245,1645-1650,200508200700-1200:" breakTimeStyle = "top:20px;background-color:yellow;" zoom = "20" </og.ganttHeader></pre> <p>HHmm-HHmm 形式で1日の時間範囲を指定 休憩時間は、HHmm-HHmm形式、または、yyyyMMddHHmm-HHmm形式、または、yyyyMMddHHmm-yyyyMMddHHmm形式です 休憩時間のスタイルに追加します。</p>
●定義・解説	PL/SQL の VIEW_ZOOM メソッドを用いて、実際の日付けから画面表示ピクセルへ 変換する割合を求めます。これと、jsp¥GANTT¥image の ganttBgx.gif (x は ZOOM変数) を対応付けて、日付けのヘッダーを求めます。 jsp¥custom¥custom.css の th.zoomx (x は ZOOM変数) により、日付けヘッダー文字の 表示間隔を制御していますので、これらの関係を正確に把握しておいて下さい。 日付けヘッダーは、イメージ変数を用いている関係上、固定の日数を加算して、 求めます。(週:7日, 半月:15日, 月:30日, 4半期:90日, 年:360日)
	ヘッダー部のカレンダーに、事業所カレンダーを使用することが出来ます。 使用する場合は、事業所コード (arg1) 属性のセットと、システムパラメータ の USE_CALENDAR_DATABASE 属性を true に設定しておく必要があります。 このフラグは、事業所カレンダーテーブル (GE13) を使用するために必要です。 zoom="10" (週単位) を使用の週数の計算は、年初の最初の7日間が含まれる週を、第1週と数えます。
	zoom="20" (時間) では、1日の時間範囲をdaySpanでHHmm-HHmm形式で指定します。 開始終了は、startDate、endDateで指定しますが、yyyyMMdd または、yyyyMMddHHmm 形式とします。 calDB で指定された カレンダーテーブルを見て、休日の場合は、表示をスキップします。 休憩時間は、breakTimes で指定します。(HHmm-HHmm形式、または、yyyyMMddHHmm-HHmm、 または、yyyyMMddHHmm-yyyyMMddHHmm形式) カンマ区切りで複数指定できます。日付指定がない場合は、毎日、日付指定がある場合は、 その日のみ有効です。休憩時間は、日をまたぐ事は可能です。 breakTimeStyle として、休憩時間の指定に追加するスタイル属性を追加します。 通常は、バックカラーと上からの位置です。初期値は、"top:20px;background-color:yellow;" です。 ヘッダー表示は、1時間単位です。
	zoom="21" (時間) では、一時間のピクセルを"20" (時) の半分に変更します。
	システムリソースに USE_GANTTHEADER_MULTIPLE_COLUMN パラメータを設定すると、 ガントヘッダーにおいて、時間ヘッダーの上段に日付を、下段の時間軸の (h) の削除を行います。 日付は、表示開始時刻の上と、その6時間ごとに表示します。 2段組を使用するかどうかを、システムリソースの USE_GANTTHEADER_MULTIPLE_COLUMN で指定します。
	システムリソースに USE_GANTTHEADER_MONTH_DATE パラメータを設定すると、 ガントヘッダーにおいて、30日、90日、180日のヘッダー表示で、日付まで 表示するか、月のみ表示するかを指定します。 ガントヘッダーの繰返し表示でバックグラウンドイメージは固定日なので、一月表示を 30日単位で表示していますが、先頭日付が1日でないため、判りにくいと苦情がありました。 そこで、30日、90日、180日のヘッダー表示時に、日付を表示しないようにするフラグです。 月表示は、その月が20日以降の場合、翌月を表示します。 初期値は、互換性の為、false です。
1.startDate	【TAG】 表示開始日付けを設定します (yyyyMMdd または、yyyyMMddHHmm 形式)。 この値は、ヘッダーの先頭に表示する日付けを指定します。 通常の日付ヘッダーの場合は、 yyyyMMdd 形式ですが、時間ヘッダーの場合は、 yyyyMMddHHmm 形式で指定します。
2.endDate	【TAG】 表示最終日を設定します (yyyyMMdd または、yyyyMMddHHmm 形式)。 表示開始日から終了日までを、zoom 変数の値に応じてヘッダーラベルを 作成します。 通常の日付ヘッダーの場合は、 yyyyMMdd 形式ですが、時間ヘッダーの場合は、 yyyyMMddHHmm 形式で指定します。
	【TAG】 ズーム変数 (0 ~ 9) を設定します (初期値:2)。 CSSファイルでの定義と、背景画像との関係で、作成します。 ズーム変数は、CSSファイルや、JavaScript、PL/SQLと連動して、日付間隔 あたりのピクセルを計算して、ヘッダーを作成しています。 また、現在の標準的なクライアントでの換算になる為、高解像度のディスプレイや 特殊設定された環境では、表示間隔にずれが発生する可能性があります。

バージョン タグ名 HTMLTEXT BODY要素	内容					
	形式サンプル					
3.zoom	ZOOM	ラベル	日数ピクセル変換率	ヘッダ上	ヘッダ下	1文字 (CSS)
	0	: 1週間	日数 * 12 * 4	48	48px = 1日 [02/]	[23 (EE)] 8px
	1	: 2週間	日数 * 12 * 2	24	24px = 1日 [02/]]	[23] 8px
	2	: 1ヶ月	日数 * 12	12	12px = 1日 [2]	[3] 12px
	3	: 2ヶ月	日数 * 12 / 1.5	8	56px = 7日 [2004/]	[02/03] 8px
	4	: 3ヶ月	日数 * 12 / 3	4	56px = 14日 [2004/]	[02/03] 8px
	5	: 4ヶ月	日数 * 12 / 4	3	90px = 30日 [2004/]	[02/03] 9px
	6	: 6ヶ月	日数 * 12 / 6	2	60px = 30日 [2004/]	[02/03] 10px
	7	: 1年	日数 * 12 / 12	1	90px = 90日 [2004/]	[02/03] 9px
	8	: 2年	日数 * 12 / 24	0.5	90px =180日 [2004/]	[02/03] 9px
	9	: 3年	日数 * 12 / 36	0.33	60px =180日 [2004/]	[02/03] 10px
	10	: 週単位	日数 * 12 / 3.5	3.43	24px = 7日 [19W]	[12] 8px
	11	予約済み (他物件使用済み)				
	12	予約済み (他物件使用済み)				
	20	: 1時間	時間 * 12 * 4	48	48px = 1時間 [02日]	[07h] 12px
	21	: 1時間2	時間 * 12 * 2	24	24px = 1時間 [02]	[07] 12px
	22	: 2時間	時間 * 12 * 4 / 2	24	48px = 2時間 [02日]	[07h] 12px
	23	: 2時間2	時間 * 12 * 2 / 2	12	24px = 2時間 [02]	[07] 12px
	24	: 4時間	時間 * 12 * 4 / 4	12	48px = 4時間 [02日]	[07h] 12px
	25	: 4時間2	時間 * 12 * 2 / 4	6	24px = 4時間 [02]	[07] 12px
	26	: 6時間	時間 * 12 * 4 / 6	8	48px = 6時間 [02日]	[07h] 12px
	27	: 6時間2	時間 * 12 * 2 / 6	4	24px = 6時間 [02]	[07] 12px
	28	: 8時間	時間 * 12 * 4 / 8	6	48px = 8時間 [02日]	[07h] 12px
	29	: 8時間2	時間 * 12 * 2 / 8	3	24px = 8時間 [02]	[07] 12px
	30	: 12時間	時間 * 12 * 4 / 12	4	48px =12時間 [02日]	[07h] 12px
	31	: 12時間2	時間 * 12 * 2 / 12	2	24px =12時間 [02]	[07] 12px
4.calDB	【TAG】DB検索するDBを指定します。 カレンダーデータは、システムパラメータのCalendarQuery_****で指定する、CalendarQuery クラスのQUERY文で、実際に取得します。上記の****部分に、ここで指定するキーワードを設定して置きます。通常は、データベースID (GEI3, ZY01 など)を指定します。このキーワードに対する実装クラスを、先のシステムパラメータのキーワード以下に記述しておきます。					
5.arg1	【TAG】DB検索する場合の第1のキーを指定します。 例えば、GEI3, ZY01 では、事業所コード(CDJGS)を設定します。この値は、システムパラメータのCalendarQuery_****で指定する、CalendarQuery クラスのQUERY文に依存します。データベース定義の統一を望むところです。					
6.arg2	【TAG】DB検索する場合の第2のキーを指定します。 例えば、TP652 では、事業部コード(CDJGS)と物件工程コード(GDKTEI)の2つのキーで、カレンダーを取得します。(十年月で、行を決定)引数が、2つまで対応していますこの値は、システムパラメータのCalendarQuery_****で指定する、CalendarQuery クラスのQUERY文に依存します。データベース定義の統一を望むところです。					
7.arg3	【TAG】DB検索する場合の第3のキーを指定します。 この値は、システムパラメータのCalendarQuery_****で指定する、CalendarQuery クラスのQUERY文に依存します。データベース定義の統一を望むところです。					
8.arg4	【TAG】DB検索する場合の第4のキーを指定します。 この値は、システムパラメータのCalendarQuery_****で指定する、CalendarQuery クラスのQUERY文に依存します。データベース定義の統一を望むところです。					
9.daySpan	【TAG】1日の開始終了時間を24時間形式(HHmm-HHmm)で設定します(初期値:0700-2000)。この値は、1日の開始終了時間を指定します。 開始時刻より終了時刻が小さい場合は、翌日への日またぎとして計算します。 たとえば、夜勤等で、17:00-07:00 という指定が可能です。 初期値は、朝の7:00から夜の20:00までです。(0700-2000)です。					
10.breakTimes	【TAG】時間軸(zoom=20)での休憩時間の開始-終了時刻をカンマ区切りで複数指定します。 休憩時間は、HHmm-HHmm形式、または、yyyyMMddHHmm-HHmm、または、yyyyMMddHHmm-yyyyMMddHHmm形式で複数指定できます。 これは、開始-終了というハイフン形式でつなげます。 このセットを、カンマで複数指定できます。 例えば、200508201200-1245, 200508201300-1315 という感じです。 -の前後が、4桁の場合は、HHmm-HHmm形式として、日付に関係なく、常にこの時分で休憩が入るという認識をします。つまり、開始は、12桁か、4桁の日付形式、終了も、12桁か、4桁の日付形式になります。					
11.breakTimeStyle	【TAG】時間軸(zoom=20)での休憩時間の表示スタイルを設定します。 通常は、ヘッダーのトップからの位置と、色です。 それ以外にも、スタイルシート属性(キー:値:)というセットを指定できます。 初期値は、top:20px;background-color:yellow; です。 設定の後ろに、: を付ける様にしてください。					
12.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を[true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。					
13.skipHoliday	【TAG】時間ガントで休日を表示する/しないを設定します (初期値:USE_GANTTHEADER_SKIP_HOLIDAY[=true])。 この値をfalseにすると、時間ガントでも休日が赤色で表示されるようになります。trueにした場合は、休日は表示されません。 (初期値:システム定数のUSE_GANTTHEADER_SKIP_HOLIDAY[=@og.value org.opengion.hayabusa.common.SystemData#USE_GANTTHEADER_SKIP_HOLIDAY])。					

バージョン タグ名 HTMLTEXT BODY要素	内容
5.5.4.2 og:ganntParam Tag	<p>【TAG】各タスクを 1 行にまとめる時のカラム名(カンマ区切り)をセットします (初期値:ViewGanttTableParam#GROUP_COLUMNS_VALUE[=PRJCODE, RESOURCE_ID])。</p> <p>ガントチャートで、同一要件のタスクを 1 行にまとめて表示します。 その場合のグルーピングのキーとなるカラム名を指定します。 複数ある場合は、カンマ区切りで指定できます。</p> <p>日付けでの重複は考慮されていませんので、データ側でチェックしておいて下さい。 (初期値:ViewGanttTableParam#GROUP_COLUMNS_VALUE[=@og.value ViewGanttTableParam#GROUP_COLUMNS_VALUE])。</p>
●形式 : <og:ganntParam groupColumns="PRJCODE, RESOURCE_ID" minDuration="0.5" /> ●body : なし	
●Tag定義 : <og:ganntParam groupColumns durationColumn dystartColumn dystartFormat minDuration headerDuration headerLocale useSeqDay startDay endDay debug />	<p>【TAG】各タスクを 1 行にまとめる時のカラム名(カンマ区切り)をセットします (初期値:ViewGanttTableParam#GROUP_COLUMNS_VALUE[=PRJCODE, RESOURCE_ID])</p> <p>【TAG】期間欄のカラム名をセットします</p> <p>【TAG】開始日付けのカラム名をセットします (初期値:ViewGanttTableParam#DYSTART_COLUMN_VALUE[=DYSTART])</p> <p>【TAG】開始日付けのフォーマットを指定します (初期値:ViewGanttTableParam#DYSTART_FORMAT_VALUE[=yyyyMMdd])</p> <p>【TAG】最小期間(少数可)をセットします(初期値:1)</p> <p>【TAG】ヘッダーの表示期間(少数可)をセットします(初期値:最小期間(minDuration)と同じ)</p> <p>【TAG】ヘッダーを表示する時の、Locale(language のみ)を指定します(初期値:ViewGanttTableParam#HEADER_LOCALE_VALUE[=ja])</p> <p>【TAG】ヘッダー日付けの連続表示を行うかどうか[true/false]を指定します(初期値:false)</p> <p>【TAG】ヘッダー日付けの開始日を指定します</p> <p>【TAG】ヘッダー日付けの終了日を指定します</p> <p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)</p>
●使用例 ViewFormTag の viewFormType が、ViewForm_HTMLGanttTable の場合に使用します。 <og:view viewFormType = "HTMLGanttTable" command = "[command]" startNo = "0" pageSize = "20" useParam = "true" > <og:ganntParam groupColumns = "PRJCODE, RESOURCE_ID" durationColumn = "DURATION" dystartColumn = "DYSTART" dystartFormat = "yyyyMMdd" minDuration = "0.5" headerDuration = "1" headerLocale = "en" useSeqDay = "true" startDay = "20041020" endDay = "20041028" /> /> </og:view >	
1.groupColumns	<p>【TAG】各タスクを 1 行にまとめる時のカラム名(カンマ区切り)をセットします (初期値:ViewGanttTableParam#GROUP_COLUMNS_VALUE[=PRJCODE, RESOURCE_ID])。</p> <p>ガントチャートで、同一要件のタスクを 1 行にまとめて表示します。 その場合のグルーピングのキーとなるカラム名を指定します。 複数ある場合は、カンマ区切りで指定できます。</p> <p>日付けでの重複は考慮されていませんので、データ側でチェックしておいて下さい。 (初期値:ViewGanttTableParam#GROUP_COLUMNS_VALUE[=@og.value ViewGanttTableParam#GROUP_COLUMNS_VALUE])。</p>
2.durationColumn	<p>【TAG】期間欄のカラム名をセットします。</p> <p>ガントを表示する場合の、各タスクの幅(期間)のカラム名を指定します。 各タスクは、開始日と期間より、ガントの指定のロケーションに割り当てられます。 初期値は null です。</p>
3.dystartColumn	<p>【TAG】開始日付けのカラム名をセットします (初期値:ViewGanttTableParam#DYSTART_COLUMN_VALUE[=DYSTART])。</p> <p>ガントを表示する場合の、各タスクの開始日付けのカラム名を指定します。 各タスクは、開始日と期間より、ガントの指定のロケーションに割り当てられます。 (初期値:ViewGanttTableParam#DYSTART_COLUMN_VALUE[=@og.value ViewGanttTableParam#DYSTART_COLUMN_VALUE])。</p>
4.dystartFormat	<p>【TAG】開始日付けのフォーマットを指定します (初期値:ViewGanttTableParam#DYSTART_FORMAT_VALUE[=yyyyMMdd])。</p> <p>ガントの開始日付 データを取り込むときの日付けフォーマットを指定します。 この形式の日付け文字列を、内部で、日付けデータとして管理します。 Java. text. SimpleDateFormat で指定できる形式を指定します。 (初期値:ViewGanttTableParam#DYSTART_FORMAT_VALUE[=@og.value ViewGanttTableParam#DYSTART_FORMAT_VALUE])。</p>
5.minDuration	<p>【TAG】最小期間(少数可)をセットします (初期値:ViewGanttTableParam#MIN_DURATION_VALUE[=1.0])。</p> <p>Duration で指定する期間の最小単位を指定します。 例えば、1 日単位の精度でタスクを割り当てる場合、期間の最小単位は、1 になります。表示する場合も、この数字が最小単位となり、HTML のテーブルの 1 カラムに相当します。例えば、最小単位を、0.5 にすると、半日単位でタスクを割り当てることができます。1.0 にすると、1 日単位になります。</p> <p>(初期値:ViewGanttTableParam#MIN_DURATION_VALUE[=@og.value ViewGanttTableParam#MIN_DURATION_VALUE])。</p>
6.headerDuration	<p>【TAG】ヘッダーの表示期間(少数可)をセットします(初期値:最小期間(minDuration)と同じ)。 ヘッダー部の日付け欄で指定する表示間隔を指定します。</p> <p>例えば、最小期間(minDuration)を、0.5 日単位の精度で指定した場合、そのままでは、ヘッダーも最小期間単位で表示されますが、1 日単位で表示させたい場合は、1 を設定します。 また、1 週間単位では、7 を指定します。 1 ヶ月単位などの、不定長での指定はできません。 初期値は最小期間(minDuration)と同じです。</p>
7.headerLocale	<p>【TAG】ヘッダーを表示する時の、Locale(language のみ)を指定します (初期値:ViewGanttTableParam#HEADER_LOCALE_VALUE[=ja])。</p> <p>ヘッダー部の日付け欄で SimpleDateFormat を用いて解析していますが、この Locale (language のみ)を指定できます。</p> <p>language は、有効な ISO 言語コードです。 これらのコードは、ISO-639 で定義される 2 桁の小文字です。 コードの完全なリストは、 http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt を始めいくつかの場所で入手できます。 SimpleDateFormat の Locale を指定するコンストラクタでは、全ての</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
8.useSeqDay	<p>ロケールをサポートするわけではありませんのでご注意ください。 (初期値:ViewGanttTableParam#HEADER_LOCALE_VALUE[=@og.value ViewGanttTableParam#HEADER_LOCALE_VALUE])。</p> <p>【TAG】ヘッダー日付けの連続表示を行うかどうか[true/false]を指定します (初期値:ViewGanttTableParam#USE_SEQ_DAY_VALUE[=false])。</p> <p>ヘッダー部の日付け欄で開始日と終了日の範囲の連続日付け表示を行う場合に、“true” に、設定します。開始日と終了日 の設定方法は、各属性を参照してください。 true:連続表示を行う / false:dystartColumn カラムで使用した分のみ (初期値:ViewGanttTableParam#USE_SEQ_DAY_VALUE[=@og.value ViewGanttTableParam#USE_SEQ_DAY_VALUE])。</p> <p>【TAG】ヘッダー日付けの開始日を指定します。</p>
9.startDay	<p>ヘッダー部の日付け欄で連続日付け表示使用時(seqDay=“true”)に、この開始日より、終了日までの日付けヘッダーを連続表示します。 seqDay=“true” で startDayを指定しない場合は、dystartColumn で指定された、最小日付けを使用します。逆に、この最小日付けよりも大きな日付けは、セットできません。データとヘッダーの整合性が取れなくなります。</p> <p>この時の日付けフォーマットは、dystartFormat の指定と同じです。</p>
10.endDay	<p>【TAG】ヘッダー日付けの終了日を指定します。</p> <p>ヘッダー部の日付け欄で連続日付け表示使用時(seqDay=“true”)に、開始日より、この終了日までの日付けヘッダーを連続表示します。 seqDay=“true” で endDayを指定しない場合は、dystartColumn で指定された、最大日付けを使用します。逆に、この最大日付けよりも小さな日付けは、セットできません。データとヘッダーの整合性が取れなくなります。</p> <p>この時の日付けフォーマットは、dystartFormat の指定と同じです。</p>
11.debug	<p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容	
形式サンプル		
5.5.4.2 og:guiQuery	【TAG】画面リソースの条件となるロールズを指定します。 ロールズ判定は、ダミーユーザーを作成して通常の処理と同様の判定方式で有効な画面リソースをピックアップします。	
JSP	●形式 : <og:guiQuery command="..." roles="..." /> ●body : なし	
●Tag定義 :	<og:guiQuery roles level rwmode command scope displayMsg notfoundMsg tableId debug > ... Body ... </og:guiQuery>	
●使用例	<og:guiQuery command = "NEW" roles = "AA BB CC" > </>	
1.roles	【TAG】画面リソースの条件となるロールズを指定します。 ロールズ判定は、ダミーユーザーを作成して通常の処理と同様の判定方式で有効な画面リソースをピックアップします。	
2.level	【TAG】画面リソースの条件となるレベルを指定します。 レベル判定は、文字列レベルの判定を行います。 画面リソースの階層番号(レベル)は、 0:グループ分類メニュー(class="GUI_GRP"のtdタグで囲われます) 1:トップ階層(【分類名称】) 2:選択階層(通常の折りたたみメニュー) 3:選択非表示(通常は、隠してあります) です。 ロール等の他の条件でピックアップされたリソースと AND 処理されます。 何も指定しなければ、すべてを対象とします。	
3.rwmode	【TAG】画面リソースの条件となるRWモードを指定します。 RWモード判定は、文字列レベルの判定を行います。(通常のRW判定と異なります) RWモードの代表的な記述は、mw,mr,-w,-r です。ここでは、特殊な検索は出来ないため、上記文字列そのまま、一致するか、m,- の前方一致で判断するかのどちらかです。 ロール等の他の条件でピックアップされたリソースと AND 処理されます。 何も指定しなければ、すべてを対象とします。	
4.command	【TAG】コマンド(NEW,RENEW)をセットします(PisqlUpdateTag,UpdateTag の場合は、ENTRY)。 コマンドは、HTMLから(get/post)指定されますので、CMD_xxx で設定されるフィールド定数値のいずれかを、指定できます。	
5.scope	【TAG】キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)。 "request","page","session","applicaton" が指定できます。 JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、主にテーブルモデルを管理するスコープを指します。	
	スコープ	変数の有効範囲
	page	JSPページ内
	request	HTTPリクエスト
	session	HTTPセッション
	application	Webアプリケーション
6.displayMsg	【TAG】検索結果を画面上に表示するメッセージリソースIDを指定します (初期値:VIEW_DISPLAY_MSG[=])。 ここでは、検索結果の件数や登録された件数をまず出力し、 その次に、ここで指定したメッセージをリソースから取得して表示します。 件数を表示させたい場合は、displayMsg = "MSG0033"[件検索しました] をセットしてください。 表示させたくない場合は、displayMsg = "" をセットしてください。 (初期値:システム定数のVIEW_DISPLAY_MSG[=@og.value org.opengion.hayabusa.common.SystemData#VIEW_DISPLAY_MSG])。	
7.notfoundMsg	【TAG】検索結果がゼロ件の場合に表示するメッセージリソースIDを指定します(初期値:MSG0077[対象データはありませんでした])。 ここでは、検索結果がゼロ件の場合のみ、特別なメッセージを表示させます。 従来は、displayMsg と兼用で、『0 件検索しました』という表示でしたが、 displayMsg の初期表示は、OFF になりましたので、ゼロ件の場合のみ別に表示させます。 表示させたくない場合は、notfoundMsg = "" をセットしてください。 初期値は、MSG0077[対象データはありませんでした]です。	
8.tableId	【TAG】(通常は使いません)結果のDBTableModelを、sessionに登録するときのキーを指定します。 検索結果より、DBTableModelオブジェクトを作成します。これを、以下のview 等のタグに渡す場合に、通常は、session を利用します。その場合の登録キーです。 query タグを同時に実行して、結果を求める場合、同一メモリに配置される為、この tableId 属性を利用して、メモリ空間を分けます。 初期値は、HybsSystem.TBL_MDL_KEY です。	
9.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。	

バージョン タグ名 HTML5 BODY要素	内容
形式サンプル	
5.5.4.2 og:head JSP	<p>【TAG】HTML の タイトルを設定します(初期値:session に“GUI_TITLE” キーで登録されている値)。 head タグのBody部に、title タグを設定します。 <title>タイトル </title> を追加します。 設定は、このタグでの指定が最優先されます。次に、リクエスト情報の GUI_TITLE があれば、使用されます。最後に、初期値である、session の “GUI_TITLE” キーで登録されている値を使用します。 session に登録されている値とは、GUIリソースで定義された、名前(長)です。</p> <p>初期値は、session に“GUI_TITLE” キーで登録されている値です。 トップメニューのタイトルは、システムパラメータ の、GUI_TOP_TITLE キー で登録されている値です。</p>
<p>●形式 : <og:head /> ●body : あり</p> <p>●Tag定義 :</p> <pre><og:head title headType useScrollBar useTabstrip useGantt useTreeBOM useGuest useAjaxTreeTable noTransition useTabLink useRealTimeCheck useIE7Header useAjaxSubmit useFocusHiligh debug > ... Body ... </og:head></pre> <p>●使用例</p> <pre><og:head headType title // useTimeSet // useToggleFrame // useInputFocus // useHilighRow useScrollBar // usePrintLine useTabstrip useTabLink useGantt useTreeBOM useAjaxTreeTable= noTransition useTabLink // useRealTimeCheck= // useAutoConvert // useFocusHiligh // useFocusFirstElm= // useAdjustFrameHeight // 5.4.5.0 (2012/03/01) 追加(復活) useRealTimeCheck= useFocusHiligh > ... Body ... </og:head></pre>	<p>【TAG】HTML の タイトルを設定します(初期値:session に“GUI_TITLE” キーで登録されている値)</p> <p>【TAG】ヘッダータイプを設定します(初期値:null)</p> <p>【TAG】ScrollBar.js JavaScript を使用するかどうか[true/false]を指定します(初期値:true)</p> <p>【TAG】tabstrip.js JavaScript を使用するかどうか[true/false]を指定します(初期値:false)</p> <p>【TAG】option/gantt.js JavaScript を使用するかどうか[true/false]を指定します(初期値:false)</p> <p>【TAG】option/treeBOM.js JavaScript を使用するかどうか[true/false]を指定します(初期値:false)</p> <p>【TAG】ゲストユーザを利用するかどうか[true/false]を指定します(初期値:false)</p> <p>【TAG】Ajaxを利用したツリー形式のテーブル表示を使用するかを指定します</p> <p>【TAG】画面遷移を行わない形式の登録方法を使用するかを指定します</p> <p>【TAG】og:tabLinkタグで使用される common/tagLink.js をインクルードするかどうか[true/false]を指定します(初期値:false)</p> <p>【TAG】カラムチェックをリアルタイムで行うかどうか[true/false]を指定します(初期値:true)</p> <p>【TAG】IE8でIE7エミュレートモードで動作させるタグを出力するかどうか[true/false]を指定します (初期値:USE_IE7_HEADER[=true])</p> <p>【TAG】フォームのSubmit処理をJavaScriptを利用して行うか(ポップアップエラー表示)</p> <p>【TAG】フォーカスが当たっている入力ボックスの色を変更するかどうか[true/false]を指定します(初期値:true)</p> <p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)</p>
1.title	<p>【TAG】HTML の タイトルを設定します(初期値:session に“GUI_TITLE” キーで登録されている値)。 head タグのBody部に、title タグを設定します。 <title>タイトル </title> を追加します。 設定は、このタグでの指定が最優先されます。次に、リクエスト情報の GUI_TITLE があれば、使用されます。最後に、初期値である、session の “GUI_TITLE” キーで登録されている値を使用します。 session に登録されている値とは、GUIリソースで定義された、名前(長)です。</p> <p>初期値は、session に“GUI_TITLE” キーで登録されている値です。 トップメニューのタイトルは、システムパラメータ の、GUI_TOP_TITLE キー で登録されている値です。</p>
2.headType	<p>【TAG】ヘッダータイプを設定します(初期値:null)。 ヘッダータイプ とは、個別の設定をしなくても、基本的なバリエーションを 簡易的に設定する場合に使用するキーワードです。 ここでは、null, xml, init, query, result, menu, simple の7つが指定できます。 query を指定すると、query.jsp で使用する基本的な設定を再現できます。 timeSet.js , toggleFrame.js , inputFocus.js が標準で設定されます。 result も同様で、result.jsp で使用する基本的な設定を再現できます。 hilighRow.js , ScrollBar.js , printLine.js , timeCheck.js , inputFocus.js が標準で設定されます。 inputFocus.js に関しては、query, result 両タイプとも設定されます。 init の場合は、通常はユーザーに見えない画面ですが、登録時エラーで 処理が停止する場合があります、CSSファイルのみ組み込む必要があります。 menu はmultiMenu.jsp等のメニュー画面用の設定です。 なにも指定しない場合(null)では、index.jsp , forward.jsp などの、 直接画面に表示されないタイプのヘッダーになります。 simple は何もインクルードしないタイプです。携帯電話用画面等でdefault.css すらインクルードしたくないが、JSPの作りは同じにしておきたい場合に利用します。 xml は、Webサービスや、Ajax などで使用する XML レスポンス用の 特別なヘッダーを作成します。</p> <p>初期値は、 null です。</p>
3.useScrollBar	<p>【TAG】ScrollBar.js JavaScript を使用するかどうか[true/false]を指定します(初期値:true)。 標準組み込みJavaScriptである、ScrollBar.js を使用するかどうかを指定します。 (従来のTYPE2です。) result タイプで有効です。 初期値は、使用する(true)です。</p>
4.useTabstrip	<p>【TAG】tabstrip.js JavaScript を使用するかどうか[true/false]を指定します(初期値:false)。 標準組み込みJavaScriptである、tabstrip.js を使用するかどうかを指定します。 init, query, result タイプで有効です。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
5.useGantt	この機能は特別な為、初期値は、使用しない(false)です。 【TAG】 option/gantt.js JavaScript を使用するかどうか[true/false]を指定します(初期値:false)。 標準組み込みJavaScriptである、option/gantt.js を使用するかどうかを指定します。 result タイプで有効です。 この機能は特別な為、初期値は、使用しない(false)です。
6.useTreeBOM	【TAG】 option/treeBOM.js JavaScript を使用するかどうか[true/false]を指定します(初期値:false)。 標準組み込みJavaScriptである、option/treeBOM.js を使用するかどうかを指定します。 result タイプで有効です。 この機能は特別な為、初期値は、使用しない(false)です。
7.useGuest	【TAG】 ゲストユーザを利用するかどうか[true/false]を指定します(初期値false)。 ユーザIDが指定されていなかった場合にユーザのIDをQUESTとしてログインさせるかを指定します。 TOMCATの認証機能を外した領域でユーザを限定せずにログインさせたい場合等に利用します。 なお、システム定数RESOURCE_USER_SQLで帰ってくる結果に「QUEST」IDのユーザが必要です。 初期値はfalse(利用しない)となります。
8.useAjaxTreeTable	【TAG】 Ajaxを利用したツリー形式のテーブル表示を使用するかを指定します。 Ajaxを利用したツリー形式のテーブル表示を使用するかを指定します。 viewタグのviewFormType="HTMLAjaxTreeTable"を使用する場合は、必ずtrueを指定して下さい。 また、ツリー形式のテーブルに対して登録を行う場合は、noTransition属性も同時にtrueにしておく必要があります。
9.noTransition	初期値は、false(使用しない)です。 【TAG】 画面遷移を行わない形式の登録方法を使用するかを指定します。 画面遷移なしの登録を行うかどうかを指定します。 trueが指定された場合、画面遷移を行わず選択行のみを動的に切り替えることで、編集、登録を行います。サーバーへのデータ送信には、jQueryのFormPluginを利用しています。 また、データを送信する対象のフォームには、id="noTransition"を指定して下さい。 (該当のIDのフォームが存在しない場合は、JSPの1番目のフォームが送信されます。) 初期値は、システムリソースのNO_TRANSITION_MODEで設定することが可能です。 システムパラメーターまたは、このタグの属性でtrueに指定された場合でも、submitされたcommandがINSERT, COPY, MODIFY, DELETE, ENTRY, UNDOの場合のみ、バックグラウンドで実行されます。 それ以外のコマンドでは通常通り、画面遷移を伴うsubmitとなります。 また、システムパラメーターまたは、このタグの属性でtrueに指定された場合、バックグラウンドで実行されるJSPには、その値が引き継がれるため、これらのJSPの各タグでtrueを指定する必要はありません。
10.useTabLink	【TAG】 og:tabLinkタグで使用される common/tagLink.js をインクルードするかどうか[true/false]を指定します(初期値:false)。 og:tabLinkタグで使用される common/tagLink.js をインクルードするかを指定します。 trueが指定された場合、該当のJavaScriptをインクルードします。 og:tabLinkを使用する場合は、必ずtrueを指定して下さい。 初期値は、false(インクルードしない)です。
11.useRealTimeCheck	【TAG】 カラムチェックをリアルタイムで行うかどうか[true/false]を指定します(初期値:true)。 common/realTimeCheck.jsをインクルードするかの指定です。 インクルードするとカラムチェックがリアルタイムで行われます。 (フォーカスが離れた時に動作) リアルタイムチェックイベントを無効にするかどうかを決めます。 falseを指定すると、リアルタイムチェックを行いません。 query, result 両タイプで有効です。 このチェックはDBTypeを用いた簡易的なチェックです。 チェックにかかった場合はバルーンで警告が表示されます。 初期値は、システムリソースのUSE_REAL_TIME_CHECKで設定することが可能です。 初期値はtrue(利用する)です。
12.useIE7Header	【TAG】 IE8でIE7エミュレートモードで動作させるタグを出力するかどうか[true/false]を指定します (初期値:USE_IE7_HEADER[=true])。 IE8でIE7エミュレートモード動作をさせるためのMETAタグを出力するか指定します。 (初期値:システム定数のUSE_IE7_HEADER[=@og.value.org.opengion.hayabusa.common.SystemData#USE_IE7_HEADER])。 【TAG】 フォームのSubmit処理をJavaScriptを利用して行うか(ポップアップエラー表示)。 この属性をtrueにセットする事で、ajaxを利用したsubmitを行います。 submit後のページでエラーが発生した場合は画面が切り替わらずにポップアップ形式でエラー表示されます(command=ENTRYの場合のみ)。 動的カラム(eventColumn属性)をRESULT部分で利用する場合は、戻るボタンが正常に動作しないため、この属性をtrueにする方が望ましいです。 データを送信する対象のフォームには、id="ajaxSubmit"を指定して下さい。 (該当のIDのフォームが存在しない場合は、JSPの1番目のフォームが送信されます。)
13.useAjaxSubmit	システムパラメーターまたは、このタグの属性でtrueに指定された場合でも、submitされたcommandがENTRYの場合のみ、バックグラウンドで実行されます。 それ以外のコマンドでは通常通り、画面遷移を伴うsubmitとなります。 また、システムパラメーターまたは、このタグの属性でtrueに指定された場合、バックグラウンドで実行されるJSPには、その値が引き継がれるため、これらのJSPの各タグでtrueを指定する必要はありません。 初期値はシステムパラメータの「USE_AJAX_SUBMIT」です。
14.useFocusHilight	【TAG】 フォーカスが当たっている入力ボックスの色を変更するかどうか[true/false]を指定します(初期値:true)。 標準組み込みJavaScriptである、focusHilight.jsをインクルードするかの指定です。 フォーカスが当たっている入力ボックスの背景色を変更します。 インクルードすると自動的に有効になります。 入力ボックス色変更JavaScriptのイベントを無効にするフラグの制御を行います。 falseを指定するとイベントを無効にします。 ポップアップ時にプルダウンが正常動作しない場合はこれをfalseとしてください。 query, result 両タイプで有効です。 初期値は使用する(true)です。
15.debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:help Tag ●形式：一般ユーザーが直接組み込むことはありません。 ●body：なし ●Tag定義： <og:help guiInfoKey extension lbl target viewMsg iconURL faqIconURL useFaq debug /> ●使用例 <og:help guiInfoKey="@[GUI.KEY]" msg="HELP" /> <og:help guiInfoKey extension msg target viewMsg iconURL />	【TAG】GUIInfo のキーを指定します。 GUIInfo のキーを指定します。 【TAG】GUIInfo のキーを指定します 【TAG】拡張子を指定します(初期値:html) 【TAG】ラベルリソースのラベルIDを指定します 【TAG】TARGET 属性を指定します(初期値:_blank) 【TAG】メッセージを常時表示させるかどうか[true/false]を指定します(初期値:false) 【TAG】ヘルプリンクをアイコンで指定する場合のアイコンURLを指定します (初期値:DEFAULT_HELP_ICON[=/image/help2.png]) 【TAG】FAQリンクをアイコンで指定する場合のアイコンURLを指定します (初期値:DEFAULT_FAQ_ICON[=/image/qaicon.png]) 【TAG】FAQ表示の機能を利用するかどうか[true/false]を指定します (初期値:USE_GUI_FAQ[=false]) 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false) = "GUIInfo のキーを指定します。(必須)" = "拡張子を指定します(初期値:html)。" = "メッセージリソースのメッセージIDを指定します。" = "TARGET 属性を指定します(初期値:_blank)。" = "メッセージを常時表示させるかどうか[true/false]を指定します(初期値:false)。" = "ヘルプアイコンのURL(初期値:/image/help.png)": // 5.3.8.0 (2011/08/01)
1.guiInfoKey	【TAG】GUIInfo のキーを指定します。 GUIInfo のキーを指定します。
2.extension	【TAG】拡張子を指定します(初期値:html)。 なにも設定されていない場合は、“html”が初期値となります。 ここでは、ビリオドは、含める必要はありません。
3.lbl	【TAG】ラベルリソースのラベルIDを指定します。 ラベルを変更するときに、lbl属性を使います。 ラベルID は、所定の language に基づく ResourceManager の getLabelData(id) を呼び出し、その結果のLabelInterfaceを使用します。 getMsgLbl() で取り出せます。 ラベルIDとメッセージIDは同時には登録できません。 ラベルとメッセージは統一されました。
4.target	【TAG】TARGET 属性を指定します(初期値:_blank)。 初期値は、“_blank”として、新規に画面を立ち上げます。 CONTENTS 等を指定すれば、コンテンツフレーム(メニューの右側全面)に、 RESULT を指定すれば、リザルトフレーム(メニュー右下側)に表示します。
5.viewMsg	【TAG】メッセージを常時表示させるかどうか[true/false]を指定します(初期値:false)。 “true”の場合は、常時表示させます。 ファイルが、存在した場合は、リンクが張られ、存在しない場合は、リンクが 張られません。 “false”の場合は、ファイルが、存在した場合は、リンクが張られ、存在しない場合は、 なにも表示されません。
6.iconURL	初期値は、“false”(メッセージを常時表示しない)です。 【TAG】ヘルプリンクをアイコンで指定する場合のアイコンURLを指定します (初期値:DEFAULT_HELP_ICON[=/image/help2.png])。 ヘルプリンクをアイコンで指定する場合、そのアイコン画像のURLを指定します。 URLは、/[CONTEXT_PATH]/jspを基準として指定します。 例) /ge/jsp/image/help.pngに存在する画像を指定する場合、iconURL=/image/help.pngを指定します。 このURLが指定されない場合、ヘルプリンクは、msgLbl属性で指定されたテキストで表示されます。 (初期値:システム定数のDEFAULT_HELP_ICON[=@og.value org.opengion.hayabusa.common.SystemData#DEFAULT_HELP_ICON])。
7.faqIconURL	【TAG】FAQリンクをアイコンで指定する場合のアイコンURLを指定します (初期値:DEFAULT_FAQ_ICON[=/image/qaicon.png])。 FAQリンクをアイコンで指定する場合、そのアイコン画像のURLを指定します。 URLは、/[CONTEXT_PATH]/jspを基準として指定します。 例) /ge/jsp/image/help.pngに存在する画像を指定する場合、iconURL=/image/help.pngを指定します。 (初期値:システム定数のDEFAULT_FAQ_ICON[=@og.value org.opengion.hayabusa.common.SystemData#DEFAULT_FAQ_ICON])。
8.useFaq	【TAG】FAQ表示の機能を利用するかどうか[true/false]を指定します (初期値:USE_GUI_FAQ[=false])。 trueを指定すると、FAQ画面へのリンクが表示されます。(GE80にデータが存在するかは無関係) リンク先はfaqGUIでセットした画面に対して画面IDを引数としてわたします。 (初期値:システム定数のUSE_GUI_FAQ[=@og.value org.opengion.hayabusa.common.SystemData#USE_GUI_FAQ])。
9.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:hidden JSP	【TAG】名前を指定します (name属性とkeys属性は同時には登録できません)。 名前を指定します。 name属性とkeys属性は同時には登録できません。
●形式： ・ <code><og:hidden name="..." value="..." /></code> ・ <code><og:hidden keys="..." vals="..." /></code> ・ <code><og:hidden name="..." /></code> ... <code></og:hidden></code> ●body：あり ●Tag定義： <code><og:hidden</code> name value keys vals defaultVal caseKey caseVal debug > ... Body ... <code></og:hidden></code> ●使用例 ・次画面にPNという名前で{@PN}に入っている値を渡します。 <code><input:hidden name="PN" value="{@PN}" /></code> ・次画面で使用するPLSQLと引数を渡します。 <code><input:hidden name="name" value="UNIQ, LKIS, KBLOYA, KBLKO, KBNGK, HJO, KBLKNM" /></code> <code><input:hidden name="SQL" value="[call RKPO011B. RK0011B(?, ?, ?, ?, ?)]" /></code> ↓ <code><og:plsqlUpdate</code> command = "{@command}" names = "{@names}" dbType = "RK0011ARG" queryType = "JDBCPLSQL" > {@SQL} <code></og:plsqlUpdate></code> ・次画面にSQLという名前でBodyに記述した値を渡します。 <code><og:hidden name="SQL"</code> SELECT AAA FROM RK01 WHERE UNIQ = ? <code></og:hidden></code>	
1.name	【TAG】名前を指定します (name属性とkeys属性は同時には登録できません)。 名前を指定します。 name属性とkeys属性は同時には登録できません。
2.value	【TAG】値を指定します (この属性を使うとBODY部は無視されます)。 ここで値を設定した場合は、BODY 部の値は使用しません。 value="{@value}" などとリクエスト変数を使用した場合に、その値が 設定されていないケースでも、この属性を使用したとみなされます。
3.keys	【TAG】名前をCSV形式で複数指定します (name属性とkeys属性は同時には登録できません)。 名前をCSV形式で複数指定します。 name属性とkeys属性は同時には登録できません。
4.vals	【TAG】keys属性に対応する値をCSV形式で複数指定します。 keys属性に対応する値をCSV形式で複数指定します。 keys属性が設定されている場合のみ、使用されます。 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスの為です。
5.defaultVal	【TAG】value属性に値がセットされていないときに、初期値をセットします。 注意：互換性維持の為、defaultVal を使用した場合の設定では null がセット されないようにしています。使用しない場合は、null のままです。
6.caseKey	【TAG】このタグ自体を利用するかどうかの条件キーを指定します (初期値:null)。 caseKey.matches (caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。
7.caseVal	【TAG】このタグ自体を利用するかどうかの条件値を指定します (初期値:null)。 caseKey.matches (caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。
8.debug	【TAG】デバッグ情報を出力するかどうか [true/false] を指定します (初期値:false)。 デバッグ情報を [true:出力する/false:しない] を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:hideMenu JSP	【TAG】 このタグのBODY部分にIDを指定します。 開閉される部分を囲っているDIVにIDを指定します。 特定のhideMenuにスタイルを指定したい場合等に利用して下さい。
●形式： <og:hideMenu [id = "uniqHideId"] タグで囲われた部分に付けるID [buttonId = "targetId"] ボタンを出力する場所のIDを指定 [defaultNone = "[true false]"] 初期状態の開閉指定 [openButton = "open.png"] 開く(+)ボタンの画像 [closeButton = "close.png"] 閉じる(ー)ボタンの画像 [lbl = "MSG0076"] 画像のalt属性に付けるメッセージ(初期値:MSG0076) > ●body：あり ●Tag定義： <og:hideMenu id 【TAG】 このタグのBODY部分にIDを指定します buttonId 【TAG】 画像ボタン出力先のIDを指定します defaultNone 【TAG】 初期状態の開閉指定[[true:閉じている/false:開いている]を指定します(初期値:true) openButton 【TAG】 開くボタンの画像を指定します closeButton 【TAG】 閉じるボタンの画像を指定します lbl 【TAG】 ラベルリソースのラベルIDを指定します debug 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false) > ... Body ... </og:hideMenu> ●使用例 例 1：通常のQUERY画面での利用 <og:hideMenu> <table><tr> <og:column name="CLM"/> </tr></table> </og:hideMenu> 例 2：queryButtonタグが存在しない状況での利用(buttonIdの明示的な指定) <og:hideMenu> </og:hideMenu>	
1.id	【TAG】 このタグのBODY部分にIDを指定します。 開閉される部分を囲っているDIVにIDを指定します。 特定のhideMenuにスタイルを指定したい場合等に利用して下さい。
2.buttonId	【TAG】 画像ボタン出力先のIDを指定します。 初期値は"query-button-append"です。 queryButtonタグが存在する画面では通常指定する必要はありません。 queryButtonタグが存在しない場合は、別途用意したspanタグのIDを指定する事でボタンが表示されます。
3.defaultNone	【TAG】 初期状態の開閉指定[[true:閉じている/false:開いている]を指定します(初期値:true)。 初期状態が開いているか、閉じているかを指定します。 初期値は閉じる(true)です。
4.openButton	【TAG】 開くボタンの画像を指定します。 ボタンの画像を指定します。 絶対パスの場合は、システムの先頭から、相対パスの場合、基底は各画面のフォルダとなります。 初期値はjsp/imageフォルダ内のシステムパラメータHIDE_OPEN_BUTTONのファイルです。 (/システムID/jsp/image/expand_u.png)
5.closeButton	【TAG】 閉じるボタンの画像を指定します。 ボタンの画像を指定します。 絶対パスの場合は、システムの先頭から、相対パスの場合、基底は各画面のフォルダとなります。 初期値はjsp/imageフォルダ内のシステムパラメータHIDE_CLOSE_BUTTONのファイルです。 (/システムID/jsp/image/shrink_u.png)
6.lbl	【TAG】 ラベルリソースのラベルIDを指定します。 ラベルを変更するときに、lbl属性を使います。 ラベルID は、所定の language に基づく ResourceManager の getLabelData(id) を呼び出し、その結果のLabelInterfaceを使用します。 getMsgLbl() で取り出せます。 ラベルIDとメッセージIDは同時には登録できません。 ラベルとメッセージは統一されました。
7.debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:img 拡張 Tag ●形式: <og:img src="..." alt="..."; /> ●body: なし ●Tag定義: <og:img name src alt longdesc width height usemap ismap clazz id title style lang dir onClick onBlur onFocus ondblClick onMouseDown onMouseUp onMouseMove onMouseOut onMouseOver roles debug > ●使用例 <og:img src="{@URL}" alt="sample" /> src : 表示させる画像のURL (必須) alt : 画像が表示されない場合、代わりに表示させるテキスト (必須)	【HTML】名前を指定します。 名前を指定します。 【HTML】名前を指定します ○【HTML】表示させる画像のURLを指定します。(必須) ○【HTML】画像が表示されない場合、代わりに表示させるテキストを指定します。(必須) 【HTML】画像部分についての詳しい情報があるURL (longdesc) を指定します 【HTML】横幅をピクセル数または%で指定します 【HTML】縦幅をピクセル数または%で指定します 【HTML】クライアントサイドクリッカブルを指定します 【HTML】サーバーサイドクリッカブルを指定します 【HTML】要素に対して class 属性を設定します 【HTML】要素に対して固有の名前(id)をつける場合に設定します 【HTML】要素に対する補足的情報(title)を設定します 【HTML】この要素に対して適用させるスタイルシート(style)を設定します 【HTML】要素の内容と他の属性値の言語(lang, xml:lang)を指定します 【HTML】文字表記の方向(dir)を指定します 【HTML】JavaScriptのイベント onClick を設定します(例:onClick="renew(' query. jsp', ' QUERY');") 【HTML】JavaScriptのイベント onBlur を設定します(例:onBlur="this. value=value. toUpperCase();") 【HTML】JavaScriptのイベント onFocus を設定します 【HTML】JavaScriptのイベント ondblClick を設定します 【HTML】JavaScriptのイベント onMouseDown を設定します 【HTML】JavaScriptのイベント onMouseUp を設定します 【HTML】JavaScriptのイベント onMouseMove を設定します 【HTML】JavaScriptのイベント onMouseOut を設定します 【HTML】JavaScriptのイベント onMouseOver を設定します 【TAG】ロールを設定します 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false) </> ●使用例 <og:img src="{@URL}" alt="sample" /> src : 表示させる画像のURL (必須) alt : 画像が表示されない場合、代わりに表示させるテキスト (必須)
1.name	【HTML】名前を指定します。 名前を指定します。
2.src	【HTML】表示させる画像のURLを指定します。 表示させる画像のURLを指定します。
3.alt	【HTML】画像が表示されない場合、代わりに表示させるテキストを指定します。 画像が表示されない場合、代わりに表示させるテキストを指定します。
4.longdesc	【HTML】画像部分についての詳しい情報があるURL (longdesc) を指定します。 longdescを指定します。
5.width	【HTML】横幅をピクセル数または%で指定します。 横幅を指定します。
6.height	【HTML】縦幅をピクセル数または%で指定します。 縦幅を指定します。
7.usemap	【HTML】クライアントサイドクリッカブルを指定します。 クライアントサイドクリッカブルを指定します。
8.ismap	【HTML】サーバーサイドクリッカブルを指定します。 サーバーサイドクリッカブルを指定します。
9.clazz	【HTML】要素に対して class 属性を設定します。 Javaの言語使用上、class で作成できないため、代用として clazz を使用しています。 html で作成される属性は、class で作成されます。
10.id	【HTML】要素に対して固有の名前(id)をつける場合に設定します。 特別な使用方法として、id="FOCUS" とすることで、フィールド系要素に フォーカスを移動させます。これは、そのページ内で唯一の id 属性として使用ください。
11.title	【HTML】要素に対する補足的情報(title)を設定します。 title セットは、ボタンなどに適用すると、マウスオーバーによりこのメッセージが チップスのように表示されます。これを利用して、説明文を登録することが可能です。 ここに登録した文字列が、メッセージリソースに存在する場合は、そのメッセージを 存在しない場合は、そのままの値を設定します。
12.style	【HTML】この要素に対して適用させるスタイルシート(style)を設定します。 タグにstyle属性を設定します。これは、キー:値: のセットを複数記述できます。 通常は、class属性や、id属性で登録しており、&lg:style type="text/css"> で 外部から指定する方がソースは読みやすくなります。
13.lang	【HTML】要素の内容と他の属性値の言語(lang, xml:lang)を指定します。 HTMLの言語属性に使われます。指定する値は、ISO 639で規定されている「言語コード」です。 [ja/en/zh/...]などのほかに、en-US:アメリカ英語、en-cockney:コックニー英語 など、 副言語を指定する方法も定められています。 ここでは、lang と xml:lang の両方に同じ値がセットされます。 タグの language 属性とは使用用途が異なります。lang セット
14.dir	【HTML】文字表記の方向(dir)を指定します。 当該要素の書字方向を指定する属性です。 ltr で、左から右に、rtl で、右から左に並べます。
15.onClick	【HTML】JavaScriptのイベント onClick を設定します(例:onClick="renew(' query. jsp', ' QUERY');")。 onClick をセットします。 例えば、<og:column name="KBSAKU" onClick="renew(' query. jsp', ' QUERY');"> /> のように指定することで、プルダウンメニューの絞込み検索が可能になります。
16.onBlur	【HTML】JavaScriptのイベント onBlur を設定します(例:onBlur="this. value=value. toUpperCase();")。 onBlur は、フォーカスが離れたときに発生するイベントです。
17.onFocus	【HTML】JavaScriptのイベント onFocus を設定します。 onFocus は、フォーカスされたときに発生するイベントです。
18.ondblClick	【HTML】JavaScriptのイベント ondblClick を設定します。 ondblClick は、マウスでダブルクリックされたときに発生するイベントです。
19.onMouseDown	【HTML】JavaScriptのイベント onMouseDown を設定します。 onMouseDown は、マウスダウンされたときに発生するイベントです。
20.onMouseUp	【HTML】JavaScriptのイベント onMouseUp を設定します。 onMouseUp は、マウスアップされたときに発生するイベントです。
21.onMouseMove	【HTML】JavaScriptのイベント onMouseMove を設定します。 onMouseMove は、マウスが移動されたときに発生するイベントです。
22.onMouseOut	【HTML】JavaScriptのイベント onMouseOut を設定します。 onMouseOut は、マウスが離れたときに発生するイベントです。
23.onMouseOver	【HTML】JavaScriptのイベント onMouseOver を設定します。 onMouseOver は、マウスが重なったときに発生するイベントです。

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
24.roles	【TAG】 ロールをセットします。 ここで指定したカラムロールを元に、ユーザー毎のアクセス許可がチェックされます。 アクセス許可されないと、表示されません。 このロールを指定しない場合は、カラムリソースのロールが使用されます。
25.debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTML EXT BODY要素	内容
	形式サンプル
5.5.4.2 og:input 拡張 JSP	<pre>【HTML】 表示形式[text/password/checkbox/radio/submit/reset/button/image/file]を指定します(初期値:text)。 <og:input type="text" /> テキストボックスを表示 <og:input type="password" /> パスワード専用テキストボックスを表示 <og:input type="checkbox" /> チェックボックスを表示 <og:input type="radio" /> ラジオボタンを表示 <og:input type="submit" /> 送信実行ボタンを表示 <og:input type="reset" /> 入力したフィールド内容を全てリセット <og:input type="button" /> ボタンを表示 <og:input type="image" /> イメージを表示してクリックした座標を返す <og:input type="file" /> ファイル名入力ボックスを表示 <og:input type="hidden" /> 表示させずにサーバーに送信するデータ</pre>
	<p>●形式: <og:input name=... value=... /> ●body: あり</p> <p>●Tag定義:</p> <pre><og:input type name value caseKey caseVal lbl lbls td colspan nextForm optionAttributes must mustAny size maxLength checked src alt accept usemap ismap id lang dir title style readonly disabled tabindex accesskey clazz language onClick onChange onBlur onFocus ondblClick onMouseDown onMouseUp onMouseMove onMouseOut onMouseOver onSelect onKeydown onKeyPress onKeyUp roles debug > ... Body ... </og:input></pre> <p>【HTML】 表示形式[text/password/checkbox/radio/submit/reset/button/image/file]を指定します(初期値:text) 【HTML】 名前を指定します 【HTML】 このフィールドに設定する値を指定します 【TAG】 このタグ自体を利用するかどうかの条件キーを指定します(初期値:null) 【TAG】 このタグ自体を利用するかどうかの条件値を指定します(初期値:null) 【TAG】 ラベルリソースのラベルIDを指定します 【TAG】 ラベルをCSV形式で複数指定します 【TAG】 テーブル形式の <td> タグを使用するかどうか[yes/no/false]を指定します(初期値:yes) 【TAG】 フィールド部分の colspan を指定します 【TAG】 入力カーソルを指定されたname属性のフィールドへ自動的に移動します 【TAG】 JavaScript などの HTML基本タグ以外の属性を、そのままタグとして使用します 【TAG】 必須入力を表す色に変えるかどうか[true/false]を指定します(初期値:false) 【TAG】 選択必須入力(どれかひとつ必須)を表す色[true/mustAny/その他]を指定します(初期値:無指定) 【HTML】 横幅を指定します(typeがtextまたはpasswordの場合は文字数で、それ以外はピクセル) 【HTML】 最大入力文字数を指定します(typeがtextまたはpasswordの場合) 【HTML】 type が checkbox か radio のとき、初めから選択された状態で表示します(“checked”のみ指定可) 【HTML】 type が image の場合、送信ボタンとして利用する画像のURLをセットします 【HTML】 type が image の場合で、画像が表示できない場合の代替テキストをセットします 【HTML】 type が file の場合に、MIMEタイプをCSV形式で複数指定出来ます 【HTML】 type が image の場合、クライアントサイド・イメージマップのURLを指定します 【HTML】 type が image の場合に、サーバーサイド・イメージマップを使用する場合に指定します 【HTML】 要素に対して固有の名前(id)をつける場合に設定します 【HTML】 要素の内容と他の属性値の言語(lang.xml::lang)を指定します 【HTML】 文字表記の方向(dir)を指定します 【HTML】 要素に対する補足的情報(title)を設定します 【HTML】 この要素に対して適用させるスタイルシート(style)を設定します 【TAG】 その部品に対して変更が出来ないように(readonly)指定します(サーバーに送信される) 【TAG】 その部品に対して、選択や変更が出来ないように(disabled)指定します(サーバーに送信されない) 【HTML】 タブの移動順(tabindex)を指定します(0 ~ 32767) 【HTML】 アクセスキー(alt+キーで直接指定)を割り当てます 【HTML】 要素に対して class 属性を設定します 【TAG】 タグ内部で使用する言語コード[ja/en/zh/...]を指定します 【HTML】 JavaScriptのイベント onClick を設定します(例:onClick="renew(' query. jsp',' QUERY');") 【HTML】 JavaScriptのイベント onChange を設定します(例:onChange="renew(' query. jsp',' QUERY');") 【HTML】 JavaScriptのイベント onBlur を設定します(例:onBlur="this.value=value.toUpperCase();") 【HTML】 JavaScriptのイベント onFocus を設定します 【HTML】 JavaScriptのイベント ondblClick を設定します 【HTML】 JavaScriptのイベント onMouseDown を設定します 【HTML】 JavaScriptのイベント onMouseUp を設定します 【HTML】 JavaScriptのイベント onMouseMove を設定します 【HTML】 JavaScriptのイベント onMouseOut を設定します 【HTML】 JavaScriptのイベント onMouseOver を設定します 【HTML】 JavaScriptのイベント onSelect を設定します 【HTML】 JavaScriptのイベント onKeydown を設定します 【HTML】 JavaScriptのイベント onKeyPress を設定します 【HTML】 JavaScriptのイベント onKeyUp を設定します 【TAG】 ロールをセットします 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)</p>
	<p>●使用例</p> <pre><og:input name="OYA" /> lbl属性が指定されていない場合は、name属性をキーにしてLabelResourceを読み取ります。 <og:input name="PN" lbl="K0" /> lbl属性を指定することで、ラベルだけを付替えることができます。 <og:input name="PN" size="13" maxlength="11" /> 強制的にサイズの変更をするときは、size属性、maxlength属性を使用できます。 <og:input name="OYA" td="false" /> 自動作成されるtdタグやラベルを外すときはtd属性を使います。 <og:input name="OYA" td="no" /> 自動作成されるtdタグを外し、ラベルとフィールドは残します。 <og:input name="PN" optionAttributes="onBlur='blr()'
onFocus='fcs()'"/> JavaScriptのイベントをコーディングするときは、optionAttributes属性を使います。 <og:input type="radio" name="OYA" /> ラベル部分と入力フィールド部分がテーブルタグの<td>により左右に分割されます。 <table>
 <tr><og:input name="PN" value="{@PN}" /></tr>
 <tr><og:input name="CD" value="{@CD}" /></tr>
</table>
<table>
 <tr><og:input name="PN" value="{@PN}" />部品入力フィールド</og:input></tr>
 <tr><og:input name="CD" value="{@CD}" />コードフィールド</og:input></tr>
</table>

HTML 表示時は、前後に<tr>タグで囲って、整形できます。 <og:input name="A" nextForm="B" /> 最大桁数入力後、フォーム B にフォーカスが移動します。 <og:input name="B" />

BODY 部分に記述した値は、入力フィールドの直後にセットされます。 <og:input name="PN" />
 <button type="button" onclick="popup(...):return false;">
 <og:message lbl="POPUP" comment="▼" />
 </button>
</og:input></pre>
	<pre>【HTML】 表示形式[text/password/checkbox/radio/submit/reset/button/image/file]を指定します(初期値:text)。 <og:input type="text" /> テキストボックスを表示 <og:input type="password" /> パスワード専用テキストボックスを表示</pre>

バージョン タグ名 HTML5 BODY要素	内容
	形式サンプル
1.type	<code><og:input type="checkbox" /></code> チェックボックスを表示 <code><og:input type="radio" /></code> ラジオボタンを表示 <code><og:input type="submit" /></code> 送信実行ボタンを表示 <code><og:input type="reset" /></code> 入力したフィールド内容を全てリセット <code><og:input type="button" /></code> ボタンを表示 <code><og:input type="image" /></code> イメージを表示してクリックした座標を返す <code><og:input type="file" /></code> ファイル名入力ボックスを表示 <code><og:input type="hidden" /></code> 表示させずにサーバーに送信するデータ
2.name	【HTML】名前を指定します。 <code><og:input name="PN" /></code>
3.value	【HTML】このフィールドに設定する値を指定します。 type属性の値が"radio"か"checkbox"以外の場合、この属性は省略可能です。 <code><og:input name="PN" value="syokichi" /></code>
4.caseKey	【TAG】このタグ自体を利用するかどうかの条件キーを指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。
5.caseVal	【TAG】このタグ自体を利用するかどうかの条件値を指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。
6.lbl	【TAG】ラベルリソースのラベルIDを指定します。 ラベルを変更するときに、lbl属性を使います。
7.lbls	ラベルID は、所定の language に基づく ResourceManager の getLabelData(id) を呼び出し、その結果のLabelInterfaceを使用します。 getMsgLbl() で取り出せます。 ラベルIDとメッセージIDは同時には登録できません。 ラベルとメッセージは統一されました。 【TAG】ラベルをCSV形式で複数指定します。 シングルラベルの lbl 属性との違いは、ここではラベルを複数 カンマ区切りで渡すことが可能であることです。これにより、"A,B,C" という値に対して、 "Aのラベル表示,Bのラベル表示,Cのラベル表示" という具合に文字列を再合成します。 これは、SQL文のOrder By 句で、ソート順を指定する場合などに便利です。
8.td	<code><og:orderBy columns="MKNMJP,MKCD,MKNMEN" lbls="MKNMJP,MKCD,MKNMEN" /></code> 【TAG】テーブル形式の <td> タグを使用するかどうか[yes/no/false]を指定します(初期値:yes)。 通常(初期値:yes)は、テーブル形式の<td> タグを使用して、ラベルとフィールドを整列させます。 no の場合は、tdタグを使用せず、ラベルとフィールドを連続して出力します。 false は、ラベルを出力せず、フィールドのみ出力します。 <code><og:input name="PN" td="false" /></code>
9.colspan	【TAG】フィールド部分の colspan を指定します。 通常は、ラベルとフィールドが、別のtdで囲われます。しかし、場合によっては、フィールド部の長いカラムがある場合、上下のカラムとの位置関係がおかしくなり、ラベルとフィールドが離れて表示されるケースがあります。 この場合、長いフィールド部を持つカラムに、colspan 属性を指定すると、その分のカラムを割り当てることにより、上下のカラムとの位置関係を調節することが可能になります。 通常は、3 を指定して、フィールド、ラベル、フィールドの組み合わせ部分に、長いフィールドが適用されるように使用します。
10.nextForm	<code><og:input name="PN" colspan="3" /></code> 【TAG】入力カーソルを指定されたname属性のフィールドへ自動的に移動します。 JavaScript の onKeyUp イベントを利用して、最大桁数まで入力すると、次のフィールドに、自動的に飛ぶように設定します。 引数は、次に飛ばすフィールドの name 属性です。 実行するには、JavaScript の nextForm フังก์ションが必要です。(default.js で指定)
11.optionAttributes	【TAG】JavaScript などの HTML基本タグ以外の属性を、そのままタグとして使用します。 JavaScript などの HTML基本タグ以外の属性を、そのままタグとして使用します。 そのときに、()は、()に置き換えます。
12.must	【TAG】必須入力を表す色に変わるかどうかが[true/false]を指定します(初期値:false)。 初期値は、必須でない("false") です。
13.mustAny	<code><og:input name="PN" must="true" /></code> 【TAG】選択必須入力(どれかひとつ必須)を表す色[true/mustAny/その他]を指定します(初期値:無指定)。 複数のカラムのうち、どれかひとつを必須とする選択必須入力を示す色を指定します。 true または、mustAny を設定すると、class属性に、mustAny がセットされます。 mustAny は、CSSファイルに初期設定されています。true または、mustAny 以外の値をセットすると、その値がそのまま、class属性にセットされますので、選択必須のグループ化が可能です。 なお、実際の選択必須入力チェックは、ここではなく、columnCheck タグで指定が必要です。 自動処理は、mustAny="true" 指定の場合のみ有効です。 初期値は、無指定です。
14.size	【HTML】横幅を指定します(typeがtextまたはpasswordの場合は文字数で、それ以外はピクセル)。 <code><og:input name="PN" size="20" /></code>
15.maxlength	【HTML】最大入力文字数を指定します(typeがtextまたはpasswordの場合)。 最大文字数はsize指定を超え得るが、超えた場合ユーザエージェントはスクロール機構を提供する必要があります。 この属性の初期値は「無制限」です。 <code><og:input name="PN" maxlength="20" /></code>
16.checked	【HTML】type が checkbox か radio のとき、初めから選択された状態で表示します("checked"のみ指定可)。 <code><og:input name="PN" type="checkbox" checked="checked" /></code>
17.src	【HTML】type が image の場合、送信ボタンとして利用する画像のURLをセットします。 src セット
18.alt	【HTML】type が image の場合で、画像が表示できない場合の代替テキストをセットします。 alt セット
19.accept	【HTML】type が file の場合に、MIMEタイプをCSV形式で複数指定出来ます。 accept セット
20.usemap	【HTML】type が image の場合、クライアントサイド・イメージマップのURLを指定します。 usemap セット
21.ismap	【HTML】type が image の場合に、サーバーサイド・イメージマップを使用する場合に指定します。 ismap セット
22.id	【HTML】要素に対して固有の名前(id)をつける場合に設定します。 特別な使用方法として、id="FOCUS" とすることで、フィールド系要素にフォーカスを移動させます。これは、そのページ内で唯一の id 属性として使用ください。
23.lang	【HTML】要素の内容と他の属性値の言語(lang.xml:lang)を指定します。 HTMLの言語属性に使われます。指定する値は、ISO 639で規定されている「言語コード」です。 [ja/en/zh/...]などのほかに、en-US:アメリカ英語、en-cockney:コックニー英語 など、副言語を指定する方法も定められています。 ここでは、lang と xml:lang の両方に同じ値がセットされます。 タグの language 属性とは使用用途が異なります。lang セット
24.dir	【HTML】文字表記の方向(dir)を指定します。 当該要素の書字方向を指定する属性です。

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
25.title	<p>ltr で、左から右に、rtl で、右から左に並べます。</p> <p>【HTML】要素に対する補足的情報(title)を設定します。</p> <p>title セットは、ボタンなどに適用すると、マウスオーバーによりこのメッセージがチップスのように表示されます。これを利用して、説明文を登録することが可能です。ここに登録した文字列が、メッセージリソースに存在する場合は、そのメッセージが存在しない場合は、そのままの値を設定します。</p>
26.style	<p>【HTML】この要素に対して適用させるスタイルシート(style)を設定します。</p> <p>タグにstyle属性を設定します。これは、キー:値: のセットを複数記述できます。</p> <p>通常は、class属性や、id属性で登録しておき、&lg:style type="text/css"> で外部から指定する方がソースは読みやすくなります。</p>
27.readonly	<p>【TAG】その部品に対して変更が来ないように(readonly)指定します(サーバーに送信される)。</p> <p>INPUT/TEXTAREA 系に対して、指定可能です。</p> <p>readonly="readonly"、readonly="true" が指定された場合は、有効です。</p> <p>false も指定値としては、有効です。(大文字小文字の区別も不要)</p> <p>それ以外の指定は、エラーとします。</p>
28.disabled	<p>【TAG】その部品に対して、選択や変更が来ないように(disabled)指定します(サーバーに送信されない)。</p> <p>BUTTON/INPUT/OPTGROUP/OPTION/SELECT/TEXTAREA 系に対して、指定可能です。</p> <p>disabled="disabled"、disabled="true" が指定された場合は、有効です。</p> <p>false も指定値としては、有効です。(大文字小文字の区別も不要)</p> <p>それ以外の指定は、エラーとします。</p>
29.tabindex	<p>【HTML】タブの移動順(tabindex)を指定します(0 ~ 32767)。</p> <p>Tabキーを押したときに要素が選択される順番を指定します。</p> <p>値には、選択させたい順番を数値で記述します。</p>
30.accesskey	<p>【HTML】アクセスキー(alt+キーで直接指定)を割り当てます。</p> <p>アクセスキーは、マウスの使えない環境でも、リンクにジャンプする、ボタンを押す、入力フォームにフォーカスを移すなどの操作を簡単に行うことができるように考慮されたものです。</p> <p>Windows の「ファイル(F)」メニューについている、F と同じような動きをします。</p>
31.clazz	<p>【HTML】要素に対して class 属性を設定します。</p> <p>Javaの言語使用上、class で作成できないため、代用として clazz を使用しています。</p> <p>html で作成される属性は、class で作成されます。</p>
32.language	<p>【TAG】タグ内部で使用する言語コード[ja/en/zh/...]を指定します。</p> <p>この言語コードに基づいて、表示のラベルをリソースから作成します。</p>
33.onClick	<p>【HTML】JavaScriptのイベント onClick を設定します(例:onClick="renew('query.jsp','QUERY');")。</p> <p>onClick をセットします。</p> <p>例えば、<og:column name="KBSAKU" onClick="renew('query.jsp','QUERY');"/> のように指定することで、プルダウンメニューの絞込み検索が可能になります。</p>
34.onChange	<p>【HTML】JavaScriptのイベント onChange を設定します(例:onChange="renew('query.jsp','QUERY');")。</p> <p>onChange をセットします。</p> <p>例えば、<og:column name="KBSAKU" onChange="renew('query.jsp','QUERY');"/> のように指定することで、プルダウンメニューの絞込み検索が可能になります。</p>
35.onBlur	<p>【HTML】JavaScriptのイベント onBlur を設定します(例:onBlur="this.value=value.toUpperCase();")。</p> <p>onBlur は、フォーカスが離れたときに発生するイベントです。</p>
36.onFocus	<p>【HTML】JavaScriptのイベント onFocus を設定します。</p> <p>onFocus は、フォーカスされたときに発生するイベントです。</p>
37.ondblClick	<p>【HTML】JavaScriptのイベント ondblClick を設定します。</p> <p>ondblClick は、マウスでダブルクリックされたときに発生するイベントです。</p>
38.onMouseDown	<p>【HTML】JavaScriptのイベント onMouseDown を設定します。</p> <p>onMouseDown は、マウスダウンされたときに発生するイベントです。</p>
39.onMouseUp	<p>【HTML】JavaScriptのイベント onMouseUp を設定します。</p> <p>onMouseUp は、マウスアップされたときに発生するイベントです。</p>
40.onMouseMove	<p>【HTML】JavaScriptのイベント onMouseMove を設定します。</p> <p>onMouseMove は、マウスが移動されたときに発生するイベントです。</p>
41.onMouseOut	<p>【HTML】JavaScriptのイベント onMouseOut を設定します。</p> <p>onMouseOut は、マウスが離れたときに発生するイベントです。</p>
42.onMouseOver	<p>【HTML】JavaScriptのイベント onMouseOver を設定します。</p> <p>onMouseOver は、マウスが重なったときに発生するイベントです。</p>
43.onSelect	<p>【HTML】JavaScriptのイベント onSelect を設定します。</p> <p>onSelect は、テキストフィールド/テキストエリアのテキストが選択されたときに発生するイベントです。</p>
44.onKeydown	<p>【HTML】JavaScriptのイベント onKeydown を設定します。</p> <p>onKeydown は、キーが押されたときに発生するイベントです。</p>
45.onKeyPress	<p>【HTML】JavaScriptのイベント onKeyPress を設定します。</p> <p>onKeyPress は、キーが押され続けていたときに発生するイベントです。</p>
46.onKeyUp	<p>【HTML】JavaScriptのイベント onKeyUp を設定します。</p> <p>onKeyUp は、キーが押された状態から離されたときに発生するイベントです。</p>
47.roles	<p>【TAG】ロールをセットします。</p> <p>ここで指定したカラムロールを元に、ユーザー毎のアクセス許可がチェックされます。</p> <p>アクセス許可されないと、表示されません。</p> <p>このロールを指定しない場合は、カラムリソースのロールが使用されます。</p>
48.debug	<p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。</p> <p>デバッグ情報を [true:出力する/false:しない]を指定します。</p> <p>出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:itd	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。
JSP	
●形式 : <og:itd> ... Body ... </og:itd>	
●body : あり	
●Tag定義 :	
<og:itd debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)
> ... Body ... </og:itd>	
●使用例	
<og:thead rowspan="1"> <tr> <og:itd>M/d</og:itd> </tr> </og:thead> <og:tbody rowspan="1" > <tr> <og:itd> <span class="cGntBar[!FGTASK]" title="[NMMCN]:
:[NMKT]" val1="[GVZOOM]" val2="[CDJGS]" val3="[WC]" val4="[NOMCN]" val5="[CDKT]" val6="[FGTASK]" val7="[DYSTART]" val8="[DYDELAY]" val9="[DURATION]" > </og:itd> </tr> </og:tbody>	
1.debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

5.5.4.2
og.jspCreate

【TAG】 プログラムIDを指定します。
作成先のプログラムIDを指定します。
ただし、作成先の実際のフォルダは、useUserDef の設定によって異なります。

Tag

●形式 : <og.jspCreate outdir="..." pgid="..." />
●body : なし

●Tag定義 :
<og.jspCreate
pgid
outdir
hpgid
useUserDef
tableId
scope
debug
>

●使用例
・先に、query タグでDBTableModel が作成済み
<og.jspCreate
outdir = "[@SYS.REAL_PATH] jsp/" : 出力先フォルダ
pgid = "[@PGID]" : 作成画面ID
hpgid = "[@HPGID]" : 雛形画面ID
useUserDef = "true/false" : 仮環境の使用可否 (初期値:true 使用する)
>

1.pgid

【TAG】 プログラムIDを指定します。
作成先のプログラムIDを指定します。
ただし、作成先の実際のフォルダは、useUserDef の設定によって異なります。

2.outdir

【TAG】 作成先のディレクトリを指定します。
作成先のディレクトリを指定します。
初期値は、実際に実行しているアプリケーションの REAL_PATH + jsp フォルダ以下です。
作成先のフォルダも、useUserDef の設定によって異なります。

3.hpgid

【TAG】 雛形のプログラムIDを指定します。
雛形のプログラムIDをバースして、実際のプログラムを作成します。
ただし、作成先の実際のフォルダは、useUserDef の設定によって異なります。
また、バースするのは、作成先の画面IDのフォルダが存在しない場合のみです。
すでに、存在している場合は、元の画面IDのフォルダを読み取って、バースを行います。基本的に、作成先のソースを手で修正した場合でも、バースと無関係な箇所の修正はそのまま反映のこされます。

4.useUserDef

【TAG】 仮環境を使用するかどうか[true/false]を指定します (初期値:true;使用する)。
true:使用する を設定すると、“customUserDef” フォルダの下に、画面IDのフォルダを作成します。
false:使用しない を設定すると、実際の リアルパス (REAL_PATH/jsp)の下に、画面IDのフォルダを作成します。こちらは、実際の画面と同様に、画面リソース等を作成してアクセスすることになります。

5.tableId

【TAG】 (通常は使いません)DBTableModel が登録されているメモリのキーを指定します。
初期値は、HybsSystem.TBL_MDL_KEY です。
この設定値から、DBTableModel を取り出します。リクエストかセッションかは、scope 属性で指定します。

6.scope

【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します (初期値:session)。
“request”、“page”、“session”、“applicaton” が指定できます。
JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、主にテーブルモデルを管理するスコープを指します。

スコープ	変数の有効範囲	
page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会
request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やボ
session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。内部的には、同じキーワー
application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。

7.debug

【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します (初期値:false)。
デバッグ情報を [true:出力する/false:しない]を指定します。
出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:LdapQuery	<p>【TAG】初期コンテキストファクトリを指定します (初期値:LDAP_INITIAL_CONTEXT_FACTORY[=com.sun.jndi.Ldap.LdapCtxFactory])。 例)com.sun.jndi.Ldap.LdapCtxFactory (初期値:システム定数のLDAP_INITIAL_CONTEXT_FACTORY[=@og.value org.opengion.hayabusa.common.SystemData#LDAP_INITIAL_CONTEXT_FACTORY])。</p>
Tag	
●形式:	<pre><og:LdapQuery debug="false" command = "NEW" initctx = "com.sun.jndi.Ldap.LdapCtxFactory" providerURL = "ldap://ldap.opengion.org:389" entrydn = "cn=inquiry-sys,o=opengion,c=JP" password = "xxxxxx" searchbase = "soouid=employeeuser,o=opengion,c=JP" filter = "(objectClass=person)" attributes = "uid,cn,officeName,ou,mail,belongOUID" searchScope = "SUBTREE" maxRowCount = "0" /></pre> <p>NEW, RENEW が使用可能です。 初期コンテキストファクトリ サービスプロバイダの構成情報 属性の取得元のオブジェクトの名前 属性の取得元のオブジェクトのパスワード 検索するコンテキストまたはオブジェクトのベース名 検索に使用するフィルタ式。null は不可 属性を識別する属性 IDをカンマ区切りで指定 検索範囲を設定(『OBJECT』『ONELEVEL』『SUBTREE』のどれか) 検索時の最大検索数(0は、無制限: デフォルト)</p>
●body: なし	
●Tag定義:	<pre><og:LdapQuery initctx providerURL entrydn password searchbase filter searchScope timeLimit derefLinkFlag maxRowCount returningObjFlag attributes columns orderBy tableId command displayMsg notfoundMsg overflowMsg scope mainTrans debug</pre> <p>【TAG】初期コンテキストファクトリを指定します (初期値:LDAP_INITIAL_CONTEXT_FACTORY[=com.sun.jndi.Ldap.LdapCtxFactory]) 【TAG】サービスプロバイダの構成情報を指定します (初期値:LDAP_PROVIDER_URL[=ldap://ldap.opengion.org:389]) 【TAG】属性の取得元のオブジェクトの名前を設定します (初期値:LDAP_ENTRYDN[=cn=inquiry-sys,o=opengion,c=JP]) 【TAG】属性の取得元のオブジェクトのパスワードを設定します (初期値:LDAP_PASSWORD[=password]) 【TAG】検索するコンテキストまたはオブジェクトの名前を設定します (初期値:LDAP_SEARCH_BASE[=soouid=employeeuser,o=opengion,c=JP]) ○ 【TAG】検索に使用するフィルタ式(例: belongOUID=61200)。(必須) 【TAG】検索範囲(OBJECT/ONELEVEL/SUBTREE)を設定します (初期値:LDAP_SEARCH_SCOPE[=SUBTREE]) 【TAG】これらの SearchControls の時間制限をミリ秒単位で設定します (初期値:0[無制限]) 【TAG】属性中のリンクへの間接参照を有効または無効[true/false]にします (初期値:false) 【TAG】レジストリの最大検索件数をセットします (初期値:0[無制限]) 【TAG】結果の一部としてオブジェクトを返すことを有効または無効[true/false]にします (初期値:false) 【TAG】検索の一部として返される属性をCSVで複数指定します 【TAG】作成する DBTableModel のカラム名をCSV形式で指定します 【TAG】検索した結果を表示する表示順をファイル属性名で指定します 【TAG】(通常は使いません) 結果をDBTableModelに書き込んで、sessionに登録するときのキーを指定します 【TAG】コマンド(NEW, RENEW)をセットします (初期値:NEW) 【TAG】検索結果を画面上に表示するメッセージリソースIDを指定します (初期値は検索件数) 【TAG】検索結果がゼロ件の場合に表示するメッセージリソースIDを指定します (初期値:MSG0077[対象データはありませんでした]) 【TAG】オーバーフロー時に画面上に表示するメッセージIDを指定します (初期値:MSG0007) 【TAG】キャッシュする場合のスコープ[request/page/session/application]を指定します (初期値:session) 【TAG】(通常使いません) タグで処理される処理がメインとなるトランザクション処理かどうかを指定します (初期値:false) 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します (初期値:false)</p>
●使用例	<p>システムパラメータ に、必要なLDAP定数を定義しておいた場合は、下記のタグで検索できます。 例では、filter にあいまい検索(*)とAND条件(&(条件A)(条件B))を指定した例を挙げておきます。 filter の形式と解釈は、RFC 2254 に準拠しています。 (RFC 2254 で定義されている演算子のすべてをすべての属性に適用できるわけではありません。)</p> <pre><og:LdapQuery debug="false" command = "NEW" filter = "(&(cnphoneticlangjp=OPENGION*)(businessName=*COMMON*))" attributes = "uid,cn,officeName,ou,mail,belongOUID" /></pre> <p>・queryOption タグの内部に記述すれば、ブルダウメニューを作成することが可能です。</p> <pre><og:select name="UID"> <og:queryOption> <og:LdapQuery filter = "(&(belongOUID=60000)(belongOUID=70000))(uid=C*)" attributes = "uid,uid,sn" orderBy = "uid" /> </og:queryOption> </og:select></pre>
1.initctx	<p>【TAG】初期コンテキストファクトリを指定します (初期値:LDAP_INITIAL_CONTEXT_FACTORY[=com.sun.jndi.Ldap.LdapCtxFactory])。 例)com.sun.jndi.Ldap.LdapCtxFactory (初期値:システム定数のLDAP_INITIAL_CONTEXT_FACTORY[=@og.value org.opengion.hayabusa.common.SystemData#LDAP_INITIAL_CONTEXT_FACTORY])。</p>
2.providerURL	<p>【TAG】サービスプロバイダの構成情報を指定します (初期値:LDAP_PROVIDER_URL[=ldap://ldap.opengion.org:389])。 プロトコルとサーバーとポートを指定します。 例)『ldap://ldap.opengion.org:389』 (初期値:システム定数のLDAP_PROVIDER_URL[=@og.value org.opengion.hayabusa.common.SystemData#LDAP_PROVIDER_URL])。</p>
3.entrydn	<p>【TAG】属性の取得元のオブジェクトの名前を設定します (初期値:LDAP_ENTRYDN[=cn=inquiry-sys,o=opengion,c=JP])。 例)『soOUID=opengion-sys,o=opengion,c=JP』 (初期値:システム定数のLDAP_ENTRYDN[=@og.value org.opengion.hayabusa.common.SystemData#LDAP_ENTRYDN])。</p>
4.password	<p>【TAG】属性の取得元のオブジェクトのパスワードを設定します (初期値:LDAP_PASSWORD[=password])。 (初期値:システム定数のLDAP_PASSWORD[=@og.value org.opengion.hayabusa.common.SystemData#LDAP_PASSWORD])。</p>
5.searchbase	<p>【TAG】検索するコンテキストまたはオブジェクトの名前を設定します (初期値:LDAP_SEARCH_BASE[=soouid=employeeuser,o=opengion,c=JP])。 例)『soOUID=employeeuser,o=opengion,c=JP』 (初期値:システム定数のLDAP_SEARCH_BASE[=@og.value org.opengion.hayabusa.common.SystemData#LDAP_SEARCH_BASE])。</p>
6.filter	<p>【TAG】検索に使用するフィルタ式(例: belongOUID=61200)。 例)『belongOUID=61200』</p>
7.searchScope	<p>【TAG】検索範囲(OBJECT/ONELEVEL/SUBTREE)を設定します (初期値:LDAP_SEARCH_SCOPE[=SUBTREE])。 検索範囲を OBJECT_SCOPE、ONELEVEL_SCOPE、SUBTREE_SCOPE のどれか 1 つです。 指定文字列は、それぞれ『OBJECT』『ONELEVEL』『SUBTREE』です。 (初期値:システム定数のLDAP_SEARCH_SCOPE[=@og.value org.opengion.hayabusa.common.SystemData#LDAP_SEARCH_SCOPE])。</p>
8.timeLimit	<p>【TAG】これらの SearchControls の時間制限をミリ秒単位で設定します (初期値:0[無制限])。 値が 0 の場合、無制限に待つことを意味します。</p>
9.derefLinkFlag	<p>【TAG】検索中のリンクへの間接参照を有効または無効[true/false]にします (初期値:false)。 検索中のリンクへの間接参照を有効または無効にします。</p>
10.maxRowCount	<p>【TAG】レジストリの最大検索件数をセットします (初期値:0[無制限])。 DBTableModelのデータとして登録する最大件数をこの値に設定します。 サーバーのメモリ資源と応答時間の確保の為に。</p>

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

11.returningObjFlag

初期値は、0 は、無制限です。
【TAG】結果の一部としてオブジェクトを返すことを有効または無効[true/false]にします(初期値:false)。
無効にした場合、オブジェクトの名前およびクラスだけが返されます。
有効にした場合、オブジェクトが返されます。

12.attributes

【TAG】検索の一部として返される属性をCSVで複数指定します。
null は属性が何も返されないことを示します。
このメソッドからは、空の配列をセットすることは出来ません。

13.columns

【TAG】作成する DBTableModel のカラム名をCSV形式で指定します。
なにも指定しない場合は、attributes 属性が使用されます。
LDAP検索結果に、別名をつけるイメージで使用します。

14.orderBy

【TAG】検索した結果を表示する表示順をファイル属性名で指定します。
attributes 属性で指定するキー、または、LDAPから返されたキーについて
その属性でソートします。逆順を行う場合は、DESC を指定のカラム名の後ろに
付けて下さい。

15.tableId

【TAG】(通常は使いません)結果をDBTableModelに書き込んで、sessionに登録するときのキーを指定します。
初期値は、HybsSystem.TBL_MDL_KEY で定義した値です。

16.command

【TAG】コマンド(NEW, RENEW)をセットします(初期値:NEW)。
コマンドは、HTMLから(get/post)指定されますので、CMD_xxx で設定される
フィールド定数値のいつれかを、指定できます。
何も設定されない、または、null の場合は、“NEW” が初期値にセットされます。

17.displayMsg

【TAG】検索結果を画面上に表示するメッセージリソースIDを指定します(初期値は検索件数)。
ここでは、検索結果の件数や登録された件数をまず出力し、
その次に、ここで指定したメッセージをリソースから取得して
表示します。
表示させたくない場合は、displayMsg = "" をセットしてください。
初期値は、検索件数を表示します。

18.notfoundMsg

【TAG】検索結果がゼロ件の場合に表示するメッセージリソースIDを指定します(初期値:MSG0077[対象データはありませんでした])。
ここでは、検索結果がゼロ件の場合のみ、特別なメッセージを表示させます。
従来は、displayMsg と兼用で、『0 件検索しました』という表示でしたが、
displayMsg の初期表示は、OFF になりましたので、ゼロ件の場合のみ別に表示させます。
表示させたくない場合は、notfoundMsg = "" をセットしてください。
初期値は、MSG0077[対象データはありませんでした]です。

19.overflowMsg

【TAG】オーバーフロー時に画面上に表示するメッセージIDを指定します(初期値:MSG0007)。
表示させたくない場合は、overflowMsg = "" をセットしてください。

20.scope

【TAG】キャッシュする場合のスコープ[request/page/session/applcation]を指定します(初期値:session)。
“request”, “page”, “session”, “applicaton” が指定できます。
JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、
主にテーブルモデルを管理するスコープを指します。

スコープ	変数の有効範囲	
page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会
request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やボ
session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。内部的には、同じキーワー
application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。

21.mainTrans

【TAG】(通常使いません)タグで処理される処理がメインとなるトランザクション処理かどうかを指定します(初期値:false)。
この値は、ファイルダウンロード処理に影響します。この値がtrueに指定された時にcommitされたDBTableModelが
ファイルダウンロードの対象の表になります。

このパラメーターは、通常、各タグにより実装され、ユーザーが指定する必要はありません。
但し、1つのJSP内でDBTableModelが複数生成される場合に、前に処理したDBTableModelについてファイルダウンロードをさせたい
場合は、後ろでDBTableModelを生成するタグで、明示的にこの値をfalseに指定することで、ファイルダウンロード処理の対象から
除外することができます。

22.debug

【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。
デバッグ情報を [true:出力する/false:しない]を指定します。
出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTML EXT BODY要素	内容
形式サンプル	
5.5.4.2 og:link 拡張 JSP	【HTML】 リンク先のMIMEタイプを指定します。 リンク先のMIMEタイプを指定します。
●形式 : <og:link href="..."> ... </og:link>	
●body : あり	
●Tag定義 : <og:link type name gamenId useGamenId column keys vals onLink markList href target caseKey caseVal lbl xlink fileCheck filePath hreflang rel rev charset shape coords id lang dir title style tabindex accesskey clazz language useDirectJump useRequestCache hrefOnly delKeys onClick onBlur onFocus ondblClick onMouseDown onMouseUp onMouseMove onMouseOut onMouseOver roles useURLCheck urlCheckUser urlCheckTime debug > ... Body ... </og:link>	【HTML】 リンク先のMIMEタイプを指定します 【HTML】 この要素をリンクの対象とするための名前を指定します 【TAG】 画面ID属性を登録します 【TAG】 GAMENID, BACK_GAMENID 属性を付加するかどうか[true/false]を設定します (初期値:true) 【TAG】 リンクをつきたいカラムを指定します (ViewLink タグと併用) 【TAG】 リンク先に渡すキーをCSV形式で複数指定します 【TAG】 リンク先に渡す値をCSV形式で複数指定します 【TAG】 リンクを張る (true or 1)/張らない(false or 0) の設定を指定します (初期値:true) 【TAG】 処理するマークを、“ ”区切りの文字列で指定します 【HTML】 リンク先のURLを指定します 【HTML】 リンク先の表示ターゲットを指定します 【TAG】 このタグ自体を利用するかどうかの条件キーを指定します (初期値:null) 【TAG】 このタグ自体を利用するかどうかの条件値を指定します (初期値:null) 【TAG】 ラベルリソースのラベルIDを指定します 【TAG】 href ではなく、xlink:href を使用するかどうか[true/false]を指定します (初期値:false) 【TAG】 ファイルの存在チェックを行うかどうか[true/false]を指定します (初期値:false) 【TAG】 アドレスにファイルのディレクトリ直接のアドレスを指定するかどうか[true/false]設定します (初期値:false) 【HTML】 リンク先の基本となる言語コードを指定します 【HTML】 この文書から見た、href属性で指定されるリンク先との関係を指定します 【HTML】 href属性で指定されるリンク先から見た、この文書との関係をしてします 【HTML】 リンク先の文字コードセットを指定します 【HTML】 リンクボタンの形状 (イメージマップ rect/circle/poly/default) を指定します 【HTML】 リンクボタンの座標を指定します 【HTML】 要素に対して固有の名前(id)をつける場合に設定します 【HTML】 要素の内容と他の属性値の言語(lang,xml::lang)を指定します 【HTML】 文字表記の方向(dir)を指定します 【HTML】 要素に対する補足的情報(title)を設定します 【HTML】 この要素に対して適用させるスタイルシート(style)を設定します 【HTML】 タブの移動順(tabindex)を指定します (0 ~ 32767) 【HTML】 アクセスキー(alt+キーで直接指定)を割り当てます 【HTML】 要素に対して class 属性を設定します 【TAG】 タグ内部で使用する言語コード[ja/en/zh/...]を指定します 【TAG】 ダイレクトジャンプを使用するかどうか[true/false]を指定します (初期値:false) 【TAG】 リクエストキャッシュ情報を使用するかどうか[true/false]を指定します (初期値:true) 【TAG】 リンク (アンカータグ)ではなく、href 属性をそのまま返すかどうか[true/false]を指定します (初期値:false) 【TAG】 リクエストキャッシュ情報使用時に使用しない(削除する) キーをCSV形式で複数指定します 【HTML】 JavaScriptのイベント onClick を設定します (例:onClick="renew('query.jsp','QUERY');") 【HTML】 JavaScriptのイベント onBlur を設定します (例:onBlur="this.value=value.toUpperCase();") 【HTML】 JavaScriptのイベント onFocus を設定します 【HTML】 JavaScriptのイベント ondblClick を設定します 【HTML】 JavaScriptのイベント onMouseDown を設定します 【HTML】 JavaScriptのイベント onMouseUp を設定します 【HTML】 JavaScriptのイベント onMouseMove を設定します 【HTML】 JavaScriptのイベント onMouseOut を設定します 【HTML】 JavaScriptのイベント onMouseOver を設定します 【TAG】 ロールをセットします 【TAG】 リンクアドレスにURLチェック用の確認キーを付加するかどうか[true/false]を指定します (初期値:false) 【TAG】 URLチェック用の確認キーに対して、リンクを開くことのできるユーザーIDを指定します (初期値:ログインユーザー) 【TAG】 URLチェック用の確認キーに対して、リンクの有効期限を指定します (初期値:60分) 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します (初期値:false)
●使用例 ・ 普通のリンク <og:link gamenId="32ExcelIn" href="direct.jsp" keys="location,filename,modifyType" value="[@GUI.ADDRESS],[@GUI.ADDRESS].xls,A" target="RESULT" > <og:message msg="MSG0057" /> </og:link> ・ 検索結果テーブル内でのリンク <og:viewLink command="[@command]"> <og:link column="KMZ" gamenId="RK0251" href="index.jsp" target="CONTENTS" keys="KMZ,ORDER_BY,command" vals="[KMZ],'LKIS','NEW' /> </og:viewLink> ・ ダイレクトジャンプ (指定の画面へ直接飛ばす) URL gamenId を指定しなければこの画面へのURLを作成します。 このURLは、その画面の index.jsp に直接ジャンプするためのURLです。 この場合、useRequestCache="true" が初期値なので、必要なければ、false に設定願います。 <og:link useDirectJump="true" delKeys="pageSize" keys="location,filename,modifyType" value="[@GUI.ADDRESS],[@GUI.ADDRESS].xls,A" />	
1.type	【HTML】 リンク先のMIMEタイプを指定します。 リンク先のMIMEタイプを指定します。
2.name	【HTML】 この要素をリンクの対象とするための名前を指定します。 この要素をリンクの対象とするための名前を指定します。
3.gamenId	【TAG】 画面ID属性を登録します。 gamenId 属性は、画面IDをキーに、実アドレスを求めるのに使用します。 画面IDが指定された場合は、実アドレスに変換する。指定されない場合は、 href 属性をそのままリンク情報を作成します。 画面IDを指定する場合の href は、JSPファイル名 (例えば、index.jsp 等) です。 【TAG】 GAMENID, BACK_GAMENID 属性を付加するかどうか[true/false]を設定します (初期値:true)。 "true"の場合は、GAMENID, BACK_GAMENID 属性を href に追加します。(?KEY1=VAL1&KEY2=VAL2...) "false"の場合は、付加しません。 JavaScript や、通常のファイル(.html や .pdf 等)に、GAMENID, BACK_GAMENID 属性を付けないように する事が出来ます。 初期値は、“true” (GAMENID, BACK_GAMENID 属性を href に追加する) です。
4.useGamenId	【TAG】 リンクをつきたいカラムを指定します (ViewLink タグと併用)。 このカラム名のTableModelに対して、ViewLink タグが存在する場合は、 リンクを張ります。
5.column	【TAG】 リンク先に渡すキーをCSV形式で複数指定します。 リンク先に渡すキーを指定します。 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。
6.keys	

バージョン タグ名 HTML5 TEXT BODY要素	内容
	形式サンプル
7.vals	こうしないとデータ自身にカンマを持っている場合に分解をミスの為です。 [TAG] リンク先に渡す値をCSV形式で複数指定します。 リンク先に渡す値を指定します。 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスの為です。
8.onLink	【TAG】リンクを張る(true or 1)/張らない(false or 0)の設定を指定します(初期値:true)。 リンクを張る場合は、“true”(または“1”)リンクを張らない場合は、“true以外”(または“0”)をセットします。 初期値は、“true”(リンクを張る)です。 さらに、[カラム名]で、動的にカラムの値で、リンクを張る、張らないを選ぶ事が可能になります。値は、“true”(または“1”)で、マークします。 追記 3.5.2.0 (2003/10/20): markList属性に、処理対象文字列郡を指定できます。 これにより、リンクを処理するかどうかの判断が、true, 1 以外にも使用できるようになりました。
9.markList	【TAG】処理するマークを、“ ”区切りの文字列で指定します。 markListで与えられた文字列に、onLink文字列(大文字/小文字の区別あり)が含まれていれば、処理します。 例えば、“A”, “B”, “C” という文字列が、onLink で指定された場合に処理するようにしたい場合は、“A B C” をセットします。 markList には、[カラム名]指定が可能です。 初期値は、“true TRUE 1”です。
10.href	【HTML】リンク先のURLを指定します。 リンク先のURLを指定します。
11.target	【HTML】リンク先の表示ターゲットを指定します。 リンク先の文書を表示させるフレーム、またはウィンドウの名前を指定します。
12.caseKey	【TAG】このタグ自体を利用するかどうかの条件キーを指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。
13.caseVal	【TAG】このタグ自体を利用するかどうかの条件値を指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。
14.lbl	【TAG】ラベルリソースのラベルIDを指定します。 ラベルを変更するときに、lbl属性を使います。 ラベルID は、所定の language に基づく ResourceManager の getLabelData(id) を呼び出し、その結果のLabelInterfaceを使用します。 getMsgLbl() で取り出せます。 ラベルIDとメッセージIDは同時には登録できません。 ラベルとメッセージは統一されました。
15.xlink	【TAG】href ではなく、xlink:href を使用するかどうか[true/false]を指定します(初期値:false)。 xlink:href 属性は、XMLでのリンクを指定します。その場合、href 属性で指定した内容は、xlink:href に置き換えられます。 また、URL文字列の接続文字列(&)を、(&)で接続します。これは、xlink として、XMLファイルに埋め込む場合に、置き換えが必要の為です。 初期値は、“false”(通常の href)です。
16.fileCheck	【TAG】ファイルの存在チェックを行うかどうか[true/false]を指定します(初期値:false)。 “true”の場合は、ファイルの存在チェックを行い、ファイルがあればリンクを張ります。ファイルが存在しない場合は、何も表示されません。 初期値は、“false”(ファイルチェックを行わない)です。 注意：現状、viewLink タグからの使用は出来ませんので、ご了承願います。
17.filePath	【TAG】アドレスにファイルのディレクトリ直接のアドレスを指定するかどうか[true/false]設定します(初期値:false)。 “true”の場合は、ファイルの物理ディレクトリをリンクします。 初期値は、“false”(ファイルの直接アドレスを行わない)です。
18.hreflang	【HTML】リンク先の基本となる言語コードを指定します。 リンク先の基本となる言語コードを指定します。
19.rel	【HTML】この文書から見た、href属性で指定されるリンク先との関係を指定します。 この文書から見た、href属性で指定されるリンク先との関係を指定します。
20.rev	【HTML】href属性で指定されるリンク先から見た、この文書との関係をしてします。 href属性で指定されるリンク先から見た、この文書との関係をしてします。
21.charset	【HTML】リンク先の文字コードセットを指定します。 リンク先の文字コードセットを指定します。
22.shape	【HTML】リンクボタンの形状(イメージマップ rect/circle/poly/default)を指定します。 リンクボタンの形状(イメージマップ rect/circle/poly/default)を指定します。
23.coords	【HTML】リンクボタンの座標を指定します。 coords セット
24.id	【HTML】要素に対して固有の名前(id)をつける場合に設定します。 特別な使用方法として、id="FOCUS" とすることで、フィールド系要素にフォーカスを移動させます。これは、そのページ内で唯一の id 属性として使用ください。
25.lang	【HTML】要素の内容と他の属性値の言語(lang, xml:lang)を指定します。 HTMLの言語属性に使われます。指定する値は、ISO 639で規定されている「言語コード」です。 [ja/en/zh/...]などのほかに、en-US:アメリカ英語、en-cockney:コックニー英語 など、副言語を指定する方法も定められています。 ここでは、lang と xml:lang の両方に同じ値がセットされます。 タグの language 属性とは使用用途が異なります。lang セット
26.dir	【HTML】文字表記の方向(dir)を指定します。 当該要素の書字方向を指定する属性です。 ltr で、左から右に、rtl で、右から左に並べます。
27.title	【HTML】要素に対する補足的情報(title)を設定します。 title セットは、ボタンなどに適用すると、マウスオーバーによりこのメッセージがチップスのように表示されます。これを利用して、説明文を登録することが可能です。 ここに登録した文字列が、メッセージリソースに存在する場合は、そのメッセージを存在しない場合は、そのままの値を設定します。
28.style	【HTML】この要素に対して適用させるスタイルシート(style)を設定します。 タグにstyle属性を設定します。これは、キー:値: のセットを複数記述できます。 通常は、class属性や、id属性で登録しておき、&lg:style type="text/css" で外部から指定する方がソースは読みやすくなります。
29.tabindex	【HTML】タブの移動順(tabindex)を指定します(0 ~ 32767)。 Tabキーを押したときに要素が選択される順番を指定します。 値には、選択させたい順番を数値で記述します。
30.accesskey	【HTML】アクセスキー(alt+キーで直接指定)を割り当てます。 アクセスキーは、マウスの使えない環境でも、リンクにジャンプする、ボタンを押す、入力フォームにフォーカスを移すなどの操作を簡単に行うことができるように考慮されたものです。 Windows の「ファイル(F)」メニューについている、F と同じような動きをします。
31.clazz	【HTML】要素に対して class 属性を設定します。 Javaの言語使用上、class で作成できないため、代用として clazz を使用しています。 html で作成される属性は、class で作成されます。
32.language	【TAG】タグ内部で使用する言語コード[ja/en/zh/...]を指定します。 この言語コードに基づいて、表示のラベルをリソースから作成します。
	【TAG】ダイレクトジャンプを使用するかどうか[true/false]を指定します(初期値:false)。 ダイレクトジャンプ(指定の画面へ直接飛ばす)は、簡易ワークフロー的に、画面へのリンクを作成する場合に使用します。一般には、メール本文に記述して、指定の画面まで

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
33.useDirectJump	<p>直接飛ばす場合に使用します。 このリンクは、その画面の index.jsp に直接ジャンプします。</p> <p>gameId を指定しなければ、この画面が指定されたと考えます。 target を指定しなければ、“_blank” を指定します。 msg がなければ(または、BODY部が空の場合)、画面名にこのリンクを作成します。 初期値は、“false”(使用しない) です。</p>
34.useRequestCache	<p>【TAG】 リクエストキャッシュ情報を使用するかどうか[true/false]を指定します(初期値:true)。 リクエストキャッシュ情報は、戻るリンクで画面を差表示(command=RENEW)する場合に最後に検索に利用された情報をキャッシュしておき、再利用する場合に利用する値です。 ダイレクトジャンプ機能(useDirectJump=true)の場合のみ有効です。 戻るリンクによる再表示は、各画面自身が持っている情報を利用しますが、ダイレクトジャンプ等で、すでにキャッシュが存在していない(セッションが異なっている)場合に、リンク情報に直接紐付けて、再検索させる必要がある場合に利用します。 このキャッシュは、keys,vals による値の再設定で、書き換えることが可能です。</p> <p>初期値は、“true”(使用する) です。</p>
35.hrefOnly	<p>【TAG】 リンク(アンカータグ)ではなく、href 属性をそのまま返すかどうか[true/false]を指定します(初期値:false)。 通常のリンクは、アンカータグ(aタグ)の状態で出力されます。その場合、HTML上ではリンクとして表示されます。このフラグをtrue にセットすると、リンク文字列そのものが出力されます。これは、リンク文字列を作成してDBに登録する場合などに使用します。</p> <p>初期値は、“false”(リンク) です。</p>
36.delKeys	<p>【TAG】 リクエストキャッシュ情報使用時に使用しない(削除する)キーをCSV形式で複数指定します。 リクエストキャッシュ情報には、一般に検索時に使用されたすべての引数が使用されます。この引数を、転送先画面では、使用しないケースがあり、その場合に、指定のキー情報を、削除します。 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスる為です。</p>
37.onClick	<p>【HTML】 JavaScriptのイベント onClick を設定します(例:onClick="renew('query.jsp','QUERY');")。 onClick をセットします。 例えば、<og:column name="KBSAKU" onClick="renew('query.jsp','QUERY');"/>のように指定することで、プルダウンメニューの絞込み検索が可能になります。</p>
38.onBlur	<p>【HTML】 JavaScriptのイベント onBlur を設定します(例:onBlur="this.value=value.toUpperCase();")。 onBlur は、フォーカスが離れたときに発生するイベントです。</p>
39.onFocus	<p>【HTML】 JavaScriptのイベント onFocus を設定します。 onFocus は、フォーカスされたときに発生するイベントです。</p>
40.ondblClick	<p>【HTML】 JavaScriptのイベント ondblClick を設定します。 ondblClick は、マウスでダブルクリックされたときに発生するイベントです。</p>
41.onMouseDown	<p>【HTML】 JavaScriptのイベント onMouseDown を設定します。 onMouseDown は、マウスダウンされたときに発生するイベントです。</p>
42.onMouseUp	<p>【HTML】 JavaScriptのイベント onMouseUp を設定します。 onMouseUp は、マウスアップされたときに発生するイベントです。</p>
43.onMouseMove	<p>【HTML】 JavaScriptのイベント onMouseMove を設定します。 onMouseMove は、マウスが移動されたときに発生するイベントです。</p>
44.onMouseOut	<p>【HTML】 JavaScriptのイベント onMouseOut を設定します。 onMouseOut は、マウスが離れたときに発生するイベントです。</p>
45.onMouseOver	<p>【HTML】 JavaScriptのイベント onMouseOver を設定します。 onMouseOver は、マウスが重なったときに発生するイベントです。</p>
46.roles	<p>【TAG】 ロールをセットします。 ここで指定したカラムロールを元に、ユーザー毎のアクセス許可がチェックされます。 アクセス許可されないと、表示されません。 このロールを指定しない場合は、カラムリソースのロールが使用されます。</p>
47.useURLCheck	<p>【TAG】 リンクアドレスにURLチェック用の確認キーを付加するかどうか[true/false]を指定します(初期値:false)。 この属性は、URLCheckFilter(org. opengion. hayabusa. filter. URLCheckFilter)と合わせて使用します。 trueが指定された場合、リンクURLのパラメーターに、URL確認用のパラメーターが出力されます。 このパラメーターを、URLCheckFilterが復号化し、正式なアクセスであるかどうかの判定を行います。</p> <p>初期値は、“false”(確認キーを出力しない) です。</p>
48.urlCheckUser	<p>【TAG】 URLチェック用の確認キーに対して、リンクを開くことのできるユーザーIDを指定します(初期値:ログインユーザー)。 URLチェック用の確認キーに対して、リンクを開くことのできるユーザーIDをカンマ区切りで指定します。 “*”が指定された場合、ユーザーチェックは行われません。</p> <p>初期値は、 ログインユーザーです。</p>
49.urlCheckTime	<p>【TAG】 URLチェック用の確認キーに対して、リンクの有効期限を指定します(初期値:60分)。 URLチェック用の確認キーに対して、リンクが生成されてからの有効期限を分単位で設定します。 属性値の先頭に'D'をつけることで、日付指定を行うことができます。 (D20090612 -> 20090612まで(当日を含む)は有効)</p> <p>初期値は、現在時間を基準として+ “1440”(1日) です。</p>
50.debug	<p>【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:localResource	【TAG】ローカルリソース設定用のシステムID をセットします。 このシステムIDを利用して、ResourceManager を構築します。 指定のない場合は、システムプロパティの SYSTEM_ID パラメータ値です。
Tag	
●形式 : <og:localResource systemId="GE" lang="zh" initLoad="false" />	
●body : なし	
●Tag定義 :	
<og:localResource	
systemId	【TAG】ローカルリソース設定用のシステムID をセットします
lang	【TAG】ローカルリソースを構築する場合の、言語(lang)を指定します
initLoad	【TAG】リソース情報の先読み込みの可否をセットします(初期値:false)
remove	【TAG】ページスコープ内のResourceManager を削除します(初期値:false)
debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)
/>	
●使用例	
この設定以降のページにローカルリソースを適用します。	
・<og:localResource	
systemId = "GE"	システムID
lang = "zh"	言語(ja, en, zh)
initLoad = "false"	初期読み込み可否(true:先読み込みする/false:しない)
/>	
ページの途中でローカルリソースを破棄します。	
・<og:localResource	
remove = "true"	ページスコープ内のResourceManager を削除します。
/>	
1.systemId	【TAG】ローカルリソース設定用のシステムID をセットします。 このシステムIDを利用して、ResourceManager を構築します。 指定のない場合は、システムプロパティの SYSTEM_ID パラメータ値です。
2.lang	【TAG】ローカルリソースを構築する場合の、言語(lang)を指定します。 ここで言語を指定しない場合は、language 属性が使用されます。
3.initLoad	【TAG】リソース情報の先読み込みの可否をセットします(初期値:false)。 リソース情報は、FGLoadフラグが'1'の情報を先読み込みします。 ローカルリソースは、全リソースを使用することはないため、通常、 先読み込みする必要はありません。また、一度読み込んだリソースは、 内部でキャッシュされるため、最初の使用時のみ、DBアクセスが発生します。 (先読み込みは、Tomcat起動時の、初めてのアクセス時に行われます。) 初期値は、先読みしない(false)です。
4.remove	【TAG】ページスコープ内のResourceManager を削除します(初期値:false)。 ページスコープ内のResourceManager を削除すると、初期化されます。 これは、ページ内の部分的にリソースを適用する場合に、使用します。 初期値は、削除しない(false)です。
5.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

<div>バージョン タグ名 HTMLTEXT BODY要素</div>	内容
形式サンプル	
5.5.4.2 og:mailSender	【TAG】タイトルを指定します。 タイトルを指定します。
JSP	
●形式：<og:mailSender from="..." to="..." subject="..." />	
●body：あり	
●Tag定義： <og:mailSender subject from to cc bcc replyTo host charset fileURL filename useLDAP ldapKey ldapVals debug > ... Body ... </og:mailSender>	【TAG】タイトルを指定します 【TAG】送信元(F R O M)アドレスを指定します 【TAG】送信先(T O)アドレスをCSV形式で指定します 【TAG】送信先(C C)アドレスをCSV形式で指定します 【TAG】送信先(B C C)アドレスをCSV形式で指定します 【TAG】返信先(replyTo)アドレスをCSV形式で指定します 【TAG】メールサーバーを指定します (初期値:COMMON_MAIL_SERVER [=]) 【TAG】文字エンコーディングを指定します (初期値:MAIL_DEFAULT_CHARSET [=ISO-2022-JP]) 【TAG】添付ファイルのセーブディレクトリを指定します (初期値:FILE_URL [=filetemp/]) 【TAG】添付ファイル名をCSV形式で指定します 【TAG】メールアドレスをLDAPを利用して取得するかどうか[true/false]を指定します(初期値:false) 【TAG】アドレスを検索するキーを指定します(初期値:uid C社員番号) 【TAG】LDAP検索時の属性名を指定します(初期値:cn.mail) 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)
●使用例	
<og:mailSender from="C00000" to="C99999" subject="@[SUBJECT]" > メール内容 </og:mailSender >	
charset = キャラクタセット (例: Windows-31J , ISO-2022-JP) from = From 送信元 to = To 送信先(複数登録可能) cc = Cc 送信先(複数登録可能) bcc = Bcc 送信先(複数登録可能) replyTo = ReplyTo 返信先(複数登録可能) useLDAP = false/true メールアドレスをLDAPを利用して取得するかどうか ldapKey = "uid" アドレスを検索するキー(初期値:uid C社員番号) ldapVals = "cn.mail" LDAP検索時の属性名。ひとつならアドレス。2つなら名前とアドレス host = S M T Pサーバー名 subject = タイトル fileURL = 添付ファイルのセーブディレクトリ filename = 添付ファイル名(ローカルにセーブされたファイル名)(複数登録可能) debug = true/false:	
複数登録可能な個所は、カンマ区切りで、複数登録できます。 アドレス(from,to,cc,bcc,replyTo)は、『"名称"<メールアドレス>』形式が指定できます。< と > で囲まれた範囲がメールアドレスになります。 名称には、漢字が使用できます。	
host は、システムパラメータ の COMMON_MAIL_SERVER 属性が初期値で設定されます。 charset は、システムパラメータ の MAIL_DEFAULT_CHARSET 属性が初期値で設定されます。	
1.subject	【TAG】タイトルを指定します。 タイトルを指定します。
2.from	【TAG】送信元(F R O M)アドレスを指定します。 送信元(F R O M)アドレスを指定します。
3.to	【TAG】送信先(T O)アドレスをCSV形式で指定します。 複数アドレスをカンマ区切りでセットできます。
4.cc	【TAG】送信先(C C)アドレスをCSV形式で指定します。 複数アドレスをカンマ区切りでセットできます。
5.bcc	【TAG】送信先(B C C)アドレスをCSV形式で指定します。 複数アドレスをカンマ区切りでセットできます。
6.replyTo	【TAG】返信先(replyTo)アドレスをCSV形式で指定します。 複数アドレスをカンマ区切りでセットできます。
7.host	【TAG】メールサーバーを指定します (初期値:COMMON_MAIL_SERVER [=])。 メールサーバーを指定します。 (初期値:システム定数のCOMMON_MAIL_SERVER [= @og.value org.opengion.hayabusa.common.SystemData#COMMON_MAIL_SERVER])。
8.charset	【TAG】文字エンコーディングを指定します (初期値:MAIL_DEFAULT_CHARSET [=ISO-2022-JP])。 文字エンコーディングは、メール本文に対するエンコーディング指定です。 これは、Windows-31J、MS932、Shift_JIS のどれかが指定された場合、 『指定の文字コード + 8bit 送信』します。 それ以外の指定は、『ISO-2022-JP に独自変換 + 7bit 送信』になります。 (初期値:システム定数のMAIL_DEFAULT_CHARSET [= @og.value org.opengion.hayabusa.common.SystemData#MAIL_DEFAULT_CHARSET])。
9.fileURL	【TAG】添付ファイルのセーブディレクトリを指定します (初期値:FILE_URL [=filetemp/])。 この属性で指定されるディレクトリに、添付ファイルが存在すると仮定します。 指定方法は、通常の fileURL 属性と同様に、先頭が、'/' (UNIX) または、2 文字目が、 ":\" (Windows) の場合は、指定のURLそのままのディレクトリに、そうでない場合は、 fileURL = "[@USER.ID]" と指定すると、FILE_URL 属性で指定のフォルダの下に、 さらに、各個人ID別のフォルダを作成して、そこを使用します。 (初期値:システム定数のFILE_URL [= @og.value org.opengion.hayabusa.common.SystemData#FILE_URL])。
10.filename	【TAG】添付ファイル名をCSV形式で指定します。 複数ファイルをセットできます。 設定方法は、カンマで区切って並べ複数指定できます。
11.useLDAP	【TAG】メールアドレスをLDAPを利用して取得するかどうか[true/false]を指定します(初期値:false)。 LDAP のメールアドレスを取得する場合は、true をセットします。false の場合は、 from,to,cc,bcc,replyTo の設定値がそのまま使用されます。 上記アドレスに、@ を含む場合は、LDAP を検索せず、メールアドレスとして そのまま使用します。LDAP の検索結果が存在しない場合は、不正なアドレスとして、 エラーとします。 初期値は、false です。
12.ldapKey	【TAG】アドレスを検索するキーを指定します(初期値:uid C社員番号)。 LDAP のメールアドレスを取得する場合の、検索するキーを指定します。 from,to,cc,bcc,replyTo の設定値を、このキーにセットして、検索します。 uid C社員番号、employeeNumber 社員番号 などが考えられます。 初期値は、uid(C社員番号)です。 このキーで複数の検索結果が得られた場合は、エラーとします。
	【TAG】LDAP検索時の属性名を指定します(初期値:cn.mail)。

バージョン タグ名 HTMLTEXT BODY要素	内容
13.ldapVals	LDAPを検索する場合、“担当者名”<メールアドレス>形式のメールアドレスを作成する場合、検索する属性値を指定します。 ひとつならアドレス。2つなら名前とアドレスの順で指定されたと認識します。 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。
14.debug	デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

形式サンプル

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

5.5.4.2
og:mailSender2

【TAG】 メール定型文のIDを指定します。
定型文マスタに定義されている定型文IDを指定します。

Tag

●形式 : <og:mailSender ptnId="..." action="..." from="..." to="..." />
●body : なし

●Tag定義 :

<og:mailSender2
 ptnId
 from
 action
 addrCheck
 to
 cc
 bcc
 tableId
 scope
 fileURL
 filename
 useStop
 debug
>

○【TAG】 メール定型文のIDを指定します。(必須)
○【TAG】 送信元(F R O M)の社員IDを指定します。(必須)
○【TAG】 アクション[CHECK/SEND/NOCHECK]をセットします。(必須)
【TAG】 メールアドレスの構文とメールアカウントのチェックをするかどうか[true/false]を指定します
【TAG】 送信先(T O)の社員ID、グループIDをCSV形式で指定します
【TAG】 送信先(C C)の社員ID、グループIDをCSV形式で指定します
【TAG】 送信先(B C C)の社員ID、グループIDをCSV形式で指定します
【TAG】 (通常は使いません)宛先のDBTableModelを、sessionに登録するときのキーを指定します
【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)
【TAG】 添付ファイルのセーブディレクトリを指定します(初期値:FILE_URL [=filetemp/])
【TAG】 添付ファイル名をCSV形式で指定します
【TAG】 例外発生した場合、後続JSPの評価を中止するかどうか[true:中止/false:継続]を指定します
【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)

●使用例

<og:mailSender2 >
 ptnId = PtnId
 action = Action
 from = From
 to = To
 cc = Cc
 bcc = Bcc
 fileURL = 添付ファイルのセーブディレクトリ
 filename = 添付ファイル名(ローカルにセーブされたファイル名)(コンマ区切りで複数登録可能)
 addrCheck = true/false(メールアカウントの有効チェック)
 useStop = true/false エラー発生時に後続JSPの評価を中止する(true)/中止しない(false)
 scope = request/session 宛先テーブルの格納スコープ(デフォルト: session)
 tableId = TableId 宛先テーブルのID(通常はデフォルトのテーブルモデルID名称を利用します)
 debug = true/false
</og:mailSender >

from には社員IDしかセットできません。
to, cc, bccには社員ID、またはグループIDをコンマ区切りで複数セットできます。
action:CHECK は送信前に、一度送信内容を確認したい場合に利用します。action=CHECKの場合、scopeにはsessionしかセットできません。
action:SEND は確認済のメール文を送信する場合に利用します。
action:NOCHECK は確認なしで送信したい場合に利用します。

1.ptnId

【TAG】 メール定型文のIDを指定します。
定型文マスタに定義されている定型文IDを指定します。

2.from

【TAG】 送信元(F R O M)の社員IDを指定します。
送信元(F R O M)の社員IDを指定します。社員マスタに存在している社員ID(例: "C12345")しかセットできません。

3.action

【TAG】 アクション[CHECK/SEND/NOCHECK]をセットします。
送信前に、一度送信内容を確認する場合、"CHECK" をセットします。
確認済のメール文を送信する場合、"SEND" をセットします。
確認なしで送信する場合、"NOCHECK" をセットします。

4.addrCheck

【TAG】 メールアドレスの構文とメールアカウントのチェックをするかどうか[true/false]を指定します。
メールアドレスの構文とメールアカウントのチェックをする(true)/しない(false)を指定します。
メール文合成の段階では、メールアドレスの構文文法についてチェックします。
メール送信の段階では、メールアカウントが有効かについてチェックします。
"true"と指定する場合、エラーが検出されたら、例外を投げて本タグの処理が中止されます。
"false"と指定する場合、エラーが検出されても、例外を投げません。

5.to

【TAG】 送信先(T O)の社員ID、グループIDをCSV形式で指定します。
複数のID(社員ID、グループID)をカンマ区切りでセットできます。
グループIDはグループマスタ管理画面により定義する必要があります。"GP. XXXXX"の形式でセットします。

6.cc

【TAG】 送信先(C C)の社員ID、グループIDをCSV形式で指定します。
複数のID(社員ID、グループID)をカンマ区切りでセットできます。
グループIDはグループマスタ管理画面により定義する必要があります。"GP. XXXXX"の形式でセットします。

7.bcc

【TAG】 送信先(B C C)の社員ID、グループIDをCSV形式で指定します。
複数のID(社員ID、グループID)をカンマ区切りでセットできます。
グループIDはグループマスタ管理画面により定義する必要があります。"GP. XXXXX"の形式でセットします。

8.tableId

【TAG】 (通常は使いません)宛先のDBTableModelを、sessionに登録するときのキーを指定します。
本タグでは、宛先のリストに基づいて DBTableModelオブジェクトを作成します。
この tableId は sessionに登録するときのキーとして指定します。
初期値は、HybsSystem.TBL_MDL_KEY です。

9.scope

【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)。
"request", "page", "session", "applicaton" が指定できます。
JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、主にテーブルモデルを管理するスコープを指します。

スコープ	変数の有効範囲	
page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会
request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ
session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。 内部的には、同じキーワー
applicaton	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。

10.fileURL

【TAG】 添付ファイルのセーブディレクトリを指定します
(初期値:FILE_URL [=filetemp/])。
この属性で指定されるディレクトリに、添付ファイルが存在すると仮定します。
指定方法は、通常の fileURL 属性と同様に、先頭が、"/" (UNIX) または、2 文字目が、
"." (Windows) の場合は、指定のURLそのままのディレクトリに、そうでない場合は、
fileURL = "[@USER. ID]" と指定すると、FILE_URL 属性で指定のフォルダの下に、
さらに、各個人ID別のフォルダを作成して、そこを使用します。
(初期値:システム定数のFILE_URL [=[@og.value.org.opengion.hayabusa.common.SystemData#FILE_URL]])。

11.filename

【TAG】 添付ファイル名をCSV形式で指定します。
複数ファイルをセットできます。
設定方法は、カンマで区切って並べ複数指定できます。

12.useStop

【TAG】 例外発生した場合、後続JSPの評価を中止するかどうか[true:中止/false:継続]を指定します。
"true"と指定する場合、例外が発生したら、後続JSPが評価されません。
"false"と指定する場合、例外が発生しても、後続JSPが評価されます。後続のJSPでは変数
[@MAIL.ERR.CODE]で本タグの実行状況(エラー発生したか)を取得できます。

【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。

バージョン タグ名 HTMLTEXT BODY要素	内容
13.debug	デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。 <div>形式サンプル</div>

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:mainProcess	【TAG】（通常使いません）処理の実行を指定する command を設定できます（初期値:NEW）。この処理は、command="NEW" の場合のみ実行されます。RENEW時にはなにも行いません。初期値は、NEW です。
JSP	
●形式 : <og:mainProcess useJspLog = "[true/false]" useDisplay = "[true/false]" > <og:process processID = "ZZZ" > <og:param key = "AAA" value = "111" /> </og:process > </og:mainProcess > ●body : あり (process)	
●Tag定義 : <og:mainProcess command useJspLog useDisplay useThread delayTime debug > ... Body ... </og:mainProcess>	
●使用例 <og:mainProcess useJspLog = "true" > <og:process processID = "DBReader" > <og:param key = "dbid" value = "FROM" /> <og:param key = "sql" value = "select * from GEO2" /> </og:process > <og:process processID = "DBWriter" > <og:param key = "dbid" value = "T0" /> <og:param key = "table" value = "GEO2" /> </og:process > </og:mainProcess >	
1.command	【TAG】（通常使いません）処理の実行を指定する command を設定できます（初期値:NEW）。この処理は、command="NEW" の場合のみ実行されます。RENEW時にはなにも行いません。初期値は、NEW です。
2.useJspLog	【TAG】ログ出力先に、JspWriter（つまり、HTML上の返り値）を使用するかどうか[true/false]を指定します（初期値:false）。ログファイルは、processタグで、Logger を指定する場合に、パラメータ logFile にてファイル名/System.out/System.err 形式で指定します。この場合、JSP 特有のWriterである、JspWriter（つまり、HTML上の返り値）は指定できません。ここでは、特別に ログの出力先を、JspWriter に切り替えるかどうかを指示できます。true を指定すると、画面出力 (JspWriter) に切り替わります。初期値は、false です。
3.useDisplay	【TAG】画面表示先に、JspWriter（つまり、HTML上の返り値）を使用するかどうか[true/false]を指定します（初期値:false）。画面表示は、processタグで、Logger を指定する場合に、パラメータ dispFile にてファイル名/System.out/System.err 形式で指定します。この場合、JSP 特有のWriterである、JspWriter（つまり、HTML上の返り値）は指定できません。ここでは、特別に ログの出力先を、JspWriter に切り替えるかどうかを指示できます。true を指定すると、画面出力 (JspWriter) に切り替わります。初期値は、false です。
4.useThread	【TAG】独立した別スレッドで実行するかどうか[true/false]を指定します（初期値:false）。MainProcess 処理を実行する場合、比較的执行時間が長いケースが考えられます。そこで、実行時に、スレッドを生成して処理を行えば、非同期に処理を行う事が可能です。ただし、その場合の出力については、JspWriter 等で返すことは出来ません。起動そのものを、URL 指定の http で呼び出すのであれば、返り値を無視することで、アプリサーバー側のスレッドで処理できます。初期値は、順次処理 (false) です。
5.delayTime	【TAG】要求に対して、処理の実行開始を遅延させる時間を指定します（初期値:0秒）。プロセス起動が、同時に大量に発生した場合に、すべての処理を行うのではなく、ある程度待って、複数の処理を1回だけで済ますことが出来る場合があります。例えば、更新データ毎にトリガが起動されるケースなどです。それらの開始時刻を遅らせる事で、同時発生のトリガを1回のプロセス処理で実行すれば、処理速度が向上します。ここでは、処理が開始されると、タイマーをスタートさせ、指定時間経過後に、処理を開始するようにしますが、その間、受け取ったリクエストは、すべて処理せず破棄されます。ここでは、リクエストのタイミングと処理の開始タイミングは厳密に制御していませんので、処理が重複する可能性があります。よって、アプリケーション側でリクエストが複数処理されても問題ないように、制限をかける必要があります。遅延は、リクエスト引数単位に制御されます。
6.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します（初期値:false）。デバッグ情報を [true:出力する/false:しない]を指定します。出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
5.5.4.2 og:message JSP	<p>【TAG】ラベルリソースのラベルIDを指定します。 ラベルを変更するときに、lbl属性を使います。</p> <p>ラベルID は、所定の language に基づく ResourceManager の getLabelData(id) を呼び出し、その結果のLabelInterfaceを使います。 getMsglbl() で取り出せます。 ラベルIDとメッセージIDは同時には登録できません。 ラベルとメッセージは統一されました。</p> <p>●形式: <og:message msg="..." val0=[...] ... /> ●body: あり</p> <p>●Tag定義:</p> <pre> <og:message lbl language command comment type val0 val1 val2 val3 val4 val5 val6 val7 val8 val9 caseKey caseVal debug > ... Body ... </og:message> </pre> <p>●使用例</p> <pre> <og:message msg="MSG0032" language="ja" /> msg : MessageResource.properties のキー language : ロケール(オプション) </pre> <p>例 1)</p> <pre> <og:message msg="MSG0032">検索条件</og:message> </pre> <p>BODY 部分は、無視されます。コメント等に使用できます。 HTMLファイルには、コメント部分は出力されません。</p>
1.lbl	<p>【TAG】ラベルリソースのラベルIDを指定します。 ラベルを変更するときに、lbl属性を使います。</p> <p>ラベルID は、所定の language に基づく ResourceManager の getLabelData(id) を呼び出し、その結果のLabelInterfaceを使います。 getMsglbl() で取り出せます。 ラベルIDとメッセージIDは同時には登録できません。 ラベルとメッセージは統一されました。</p>
2.language	<p>【TAG】タグ内部で使用する言語コード[ja/en/zh/...]を指定します。 この言語コードに基づいて、表示のラベルをリソースから作成します。</p>
3.command	<p>【TAG】コマンド(INSERT, COPY, MODIFY, DELETE)をセットします。 コマンドは、HTMLから(get/post)指定されますので、CMD_xxx で設定される フィールド定数値のいずれかを、指定できます。 コマンド(INSERT, COPY, MODIFY, DELETE)に対応したメッセージを表示します。</p> <p>INSERT : MSG0044 追加処理を行います。 COPY : MSG0045 複写処理を行います。 MODIFY : MSG0046 変更処理を行います。 DELETE : MSG0047 削除処理を行います。</p> <p>なお、command によるメッセージの自動選択は、msg , lbl とともにセットされていない 場合にのみ有効になります。</p>
4.comment	<p>【TAG】コメントを指定します。 msg 属性でメッセージリソースから読み取りますが、開発途中で リソース登録を済ませていない状態でコメントを入れる場合に、 直接コメントをかけるようにします。</p>
5.type	<p>【TAG】タイプを(Label, Short, Tips, Description)から指定します(初期値:Label)。 タイプは、ラベル/メッセージリソースのどの情報を取得したいかを指定できます。 従来のラベル/メッセージは、Label に相当します。 なお、指定は、それぞれの頭文字('L','S','T','D')のみでも可能です。 初期値は、"Label" です。</p> <ul style="list-style-type: none"> ・Label : ラベル/メッセージ表示 ・Short : 表形式で使用する短いラベル表示 ・Tips : ラベルの上にTips表示 ・Description : 概要説明を表示
6.val0	【TAG】メッセージの引数[0]を指定します。 メッセージの引数を指定します。
7.val1	【TAG】メッセージの引数[1]を指定します。 メッセージの引数を指定します。
8.val2	【TAG】メッセージの引数[2]を指定します。 メッセージの引数を指定します。
9.val3	【TAG】メッセージの引数[3]を指定します。 メッセージの引数を指定します。
10.val4	【TAG】メッセージの引数[4]を指定します。 メッセージの引数を指定します。
11.val5	【TAG】メッセージの引数[5]を指定します。 メッセージの引数を指定します。
12.val6	【TAG】メッセージの引数[6]を指定します。 メッセージの引数を指定します。
13.val7	【TAG】メッセージの引数[7]を指定します。 メッセージの引数を指定します。
14.val8	【TAG】メッセージの引数[8]を指定します。 メッセージの引数を指定します。
15.val9	【TAG】メッセージの引数[9]を指定します。 メッセージの引数[9]を指定します。
16.caseKey	<p>【TAG】このタグ自体を利用するかどうかの条件キーを指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
17.caseVal	何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。 【TAG】このタグ自体を利用するかどうかの条件値を指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。
18.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTML5 BODY要素	内容
形式サンプル	
5.5.4.2 og:option 拡張 Tag ●形式 : <og:option value="..." lbl ="..." selected="..." /> ●body : なし ●Tag定義 : <og:option value selected lbl lbls label caseKey caseVal id lang dir title style disabled clazz language roles debug > ●使用例 フルダウンで選択する 値(value) に対して ラベル(lbl) を付けられます。 <og:select name="ORDER_BY" msg="ORDER_BY"> <og:option value="SYSTEM_ID, CLM, LANG" lbl ="CLM" selected="selected" /> </og:select> ラベルが複数ある場合は、lbls 属性を使用します。 <og:select name="ORDER_BY" msg="ORDER_BY"> <og:option value="SYSTEM_ID, CLM, LANG" lbls="LANG, NAME_JA" /> </og:select> MessageResource プロパティの値を使用したいときはmsg属性を使います。 <og:select name="CDC"> <og:option msg="MSG0001" /> </og:select> LabelResource プロパティの値を使用したいときはlbl属性を使います。 <og:select name="CDC"> <og:option lbl="CDC1" /> </og:select> 選択肢の中から複数選択できるようにするときはmultiple属性を使います。 <og:select name="CDC" multiple="multiple"> <og:option value="AAA" /> </og:select> 選択不可にするときはdisabled属性を使います。 <og:select name="CDC" disabled="disabled"> <og:option value="AAA" /> </og:select> 選択肢をSELECT文の結果から作成したいときはqueryOptionタグと組み合わせて使います。 <og:select name="CDC"> <og:queryOption> select NOSYN, NOSYN, ':' , NMSYN from DB01 ORDER BY 1 </og:queryOption> </og:select>	【TAG】 値を指定します。 ここで指定した値がプルダウンメニュー中に存在する場合、選択状態になります。 【TAG】 オプションを選択済みの状態(selected)にセットします(初期値:未選択)。 selected="selected" または selected="true" 以外の値はセットできないように制限をかけます。 初期値は、未選択 です。 【TAG】 ラベルリソースのラベルIDを指定します。 ラベルを変更するときに、lbl属性を使います。 ラベルID は、所定の language に基づく ResourceManager の getLabelData(id) を呼び出し、その結果のLabelInterfaceを使います。 getMsgLbl() で取り出せます。 ラベルIDとメッセージIDは同時には登録できません。 ラベルとメッセージは統一されました。 【TAG】 ラベルをCSV形式で複数指定します。 シングルラベルの lbl 属性との違いは、ここではラベルを複数 カンマ区切りで渡すことが可能であることです。これにより、“A, B, C” という値に対して、“Aのラベル表示, Bのラベル表示, Cのラベル表示” という具合に文字列を再合成します。 これは、SQL文のOrder By 句で、ソート順を指定する場合などに便利です。 <og:option lbls="MKNMJP, MKCD, MKNMEN" /> 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスる為です。 【TAG】 optionタグのラベルを指定します。 ここでのラベルは、optionタグのラベルです。(lbl属性は、異なります。) これは、optgroup とともに使用される階層化メニュー時に使用されます。 【TAG】 このタグ自体を利用するかどうかの条件キーを指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。 【TAG】 このタグ自体を利用するかどうかの条件値を指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。
1.value	【TAG】 値を指定します。 ここで指定した値がプルダウンメニュー中に存在する場合、選択状態になります。
2.selected	【TAG】 オプションを選択済みの状態(selected)にセットします(初期値:未選択)。 selected="selected" または selected="true" 以外の値はセットできないように制限をかけます。 初期値は、未選択 です。
3.lbl	【TAG】 ラベルリソースのラベルIDを指定します。 ラベルを変更するときに、lbl属性を使います。 ラベルID は、所定の language に基づく ResourceManager の getLabelData(id) を呼び出し、その結果のLabelInterfaceを使います。 getMsgLbl() で取り出せます。 ラベルIDとメッセージIDは同時には登録できません。 ラベルとメッセージは統一されました。
4.lbls	【TAG】 ラベルをCSV形式で複数指定します。 シングルラベルの lbl 属性との違いは、ここではラベルを複数 カンマ区切りで渡すことが可能であることです。これにより、“A, B, C” という値に対して、“Aのラベル表示, Bのラベル表示, Cのラベル表示” という具合に文字列を再合成します。 これは、SQL文のOrder By 句で、ソート順を指定する場合などに便利です。 <og:option lbls="MKNMJP, MKCD, MKNMEN" /> 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスる為です。
5.label	【TAG】 optionタグのラベルを指定します。 ここでのラベルは、optionタグのラベルです。(lbl属性は、異なります。) これは、optgroup とともに使用される階層化メニュー時に使用されます。
6.caseKey	【TAG】 このタグ自体を利用するかどうかの条件キーを指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。
7.caseVal	【TAG】 このタグ自体を利用するかどうかの条件値を指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。
	【HTML】 要素に対して固有の名前(id)をつける場合に設定します。

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
8.id	特別な使用方法として、id="FOCUS" とすることで、フィールド系要素にフォーカスを移動させます。これは、そのページ内で唯一の id 属性として使用ください。 【HTML】要素の内容と他の属性値の言語(lang, xml:lang)を指定します。
9.lang	HTMLの言語属性に使われます。指定する値は、ISO 639で規定されている「言語コード」です。 [ja/en/zh/...]などのほかに、en-US:アメリカ英語、en-cockney:コックニー英語 など、副言語を指定する方法も定められています。 ここでは、lang と xml:lang の両方に同じ値がセットされます。 タグの language 属性とは使用用途が異なります。lang セット
10.dir	【HTML】文字表記の方向(dir)を指定します。 当該要素の書字方向を指定する属性です。 ltr で、左から右に、rtl で、右から左に並べます。
11.title	【HTML】要素に対する補足的情報(title)を設定します。 title セットは、ボタンなどに適用すると、マウスオーバーによりこのメッセージがチップスのように表示されます。これを利用して、説明文を登録することが可能です。 ここに登録した文字列が、メッセージリソースに存在する場合は、そのメッセージを存在しない場合は、そのままの値を設定します。
12.style	【HTML】この要素に対して適用させるスタイルシート(style)を設定します。 タグにstyle属性を設定します。これは、キー:値: のセットを複数記述できます。 通常は、class属性や、id属性で登録しておき、&lg:style type="text/css"> で外部から指定する方がソースは読みやすくなります。
13.disabled	【TAG】その部品に対して、選択や変更が出来ないように(disabled)指定します(サーバーに送信されない)。 BUTTON/INPUT/OPTGROUP/OPTION/SELECT/TEXTAREA 系に対して、指定可能です。 disabled="disabled" , disabled="true" が指定された場合は、有効です。 false も指定値としては、有効です。(大文字小文字の区別も不要) それ以外の指定は、エラーとします。
14.clazz	【HTML】要素に対して class 属性を設定します。 Javaの言語使用上、class で作成できないため、代用として clazz を使用しています。 html で作成される属性は、class で作成されます。
15.language	【TAG】タグ内部で使用する言語コード[ja/en/zh/...]を指定します。 この言語コードに基づいて、表示のラベルをリソースから作成します。
16.roles	【TAG】ロールをセットします。 ここで指定したカラムロールを元に、ユーザー毎のアクセス許可がチェックされます。 アクセス許可されないと、表示されません。 このロールを指定しない場合は、カラムリソースのロールが使用されます。
17.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
5.5.4.2 og:orderBy Tag ●形式: <og:orderBy columns="..." /> ●body: なし ●Tag定義: <og:orderBy columns checked lbls checkedValue caseKey caseVal debug /> ●使用例 <og:orderBy columns="PN, YOBI" checked="checked" /> <og:orderBy columns="YOBI, PN" /> <og:orderBy columns="MKNM, YOBI" />	【TAG】表示順対象のカラムをCSV形式で複数指定します。 表示順対象のカラムを指定します。 【TAG】表示順対象のカラムをCSV形式で複数指定します。(必須) 【TAG】初めからチェックされた状態 (checked) で表示します 【TAG】ラベルをCSV形式で複数指定します 【TAG】初めからチェックされた状態にする値を指定します 【TAG】このタグ自体を利用するかどうかの条件キーを指定します (初期値: null) 【TAG】このタグ自体を利用するかどうかの条件値を指定します (初期値: null) 【TAG】デバッグ情報を出力するかどうか [true/false] を指定します (初期値: false)
1.columns	【TAG】表示順対象のカラムをCSV形式で複数指定します。 表示順対象のカラムを指定します。
2.checked	【TAG】初めからチェックされた状態 (checked) で表示します。 check 値が "checked" の場合のみ、実行されます。
3.lbls	【TAG】ラベルをCSV形式で複数指定します。 ラベル (複数) を変更するときに、lbls属性を使います。 シングルラベルの lbl 属性との違いは、ここではラベルを複数 カンマ区切りで渡すことが可能であることです。これにより、"A, B, C" という値に対して、 "Aのラベル表示, Bのラベル表示, Cのラベル表示" という具合に文字列を再合成します。 これは、SQL文のOrder By 句で、ソート順を指定する場合などに便利です。
4.checkedValue	【TAG】初めからチェックされた状態にする値を指定します。 チェックバリエーションが、自分自身のカラム名と同じであれば、チェックが付きま。通常、checkedValue = "[@ORDER BY]" で、自動的にチェックされます。
5.caseKey	【TAG】このタグ自体を利用するかどうかの条件キーを指定します (初期値: null)。 caseKey.matches (caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。
6.caseVal	【TAG】このタグ自体を利用するかどうかの条件値を指定します (初期値: null)。 caseKey.matches (caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。
7.debug	【TAG】デバッグ情報を出力するかどうか [true/false] を指定します (初期値: false)。 デバッグ情報を [true:出力する/false:しない] を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

5.5.4.2
og:plsqlUpdate

JSP

【TAG】 Query を発行する為のクラスIDを指定します(初期値:JDBCPLSQL)。
引数指定のINSERT/UPDATE文を実行する場合の、queryType 属性を使用します。
このタグでは、execute(String ,String , DBSysArg[] , DBUserArg[])を実行します。
代表的なクラスとして、“JDBCPLSQL” が標準で用意されています。

タグにより使用できる/出来ないがありますが、これは、org.opengion.hayabusa.db
以下の Query_**** クラスの **** を与えます。
これらは、Query インターフェースを継承したサブクラスです。

●形式: <og:plsqlUpdate command="..." names="..." dbType="..." queryType="JDBCPLSQL">[plsql(?.?.?.?.?.)] <og:plsqlUpdate>

●body: あり

●Tag定義:

```
<og:plsqlUpdate
  queryType
  command
  scope
  maxRowCount
  skipRowCount
  notfoundMsg
  names
  dbType
  selectedAll
  tableId
  dbId
  stopError
  debug
>
... Body ...
</og:plsqlUpdate>
```

●使用例

・ 引数/プロシジャーを他のJSPから渡す場合

```
<copy.jsp>
<og:hidden name="names" value="UNIQ,USRID,ECNO,EDBN" />
<og:hidden name="SQL" value="{ call RK0271E.RK0271E( ?.?.?.?.?. ) }" />
</copy.jsp>
<entry.jsp>
<og:plsqlUpdate
  command = "[@command]"
  names    = "[@names]"
  dbType   = "RK0271ARG"
  queryType = "JDBCPLSQL" >
  { @SQL }
</og:plsqlUpdate>
```

・ 引数/プロシジャーを直接書く場合

```
<entry.jsp>
<og:plsqlUpdate
  command = "[@command]"
  names    = "UNIQ,USRID,ECNO,EDBN"
  dbType   = "RK0271ARG"
  queryType = "JDBCPLSQL" >
  { call RK0271E.RK0271E( ?.?.?.?.?. ) }
</og:plsqlUpdate>
```

<<参考>>

・ RK0271E.RK0271E(?.?.?.?.?.)の「?」の意味

```
(RK0271E.sp)
CREATE OR REPLACE PACKAGE RK0271E AS
PROCEDURE RK0271E(
  P_KEKKA OUT NUMBER -- 1個目の「?」⇒結果 0:正常 1:警告 2:異常
  ,P_ERRMSG OUT ERR_MSG_ARRAY -- 2個目の「?」⇒エラーメッセージ配列
  ,P_NAMES IN VARCHAR2 -- 3個目の「?」⇒カラム名チェック用文字列
  ,P_SYSARGS IN SYSARG_ARRAY -- 4個目の「?」⇒登録条件配列(改廃(A:追加/C:変更/D:削除)等がセットされます)
  ,P_RK0271 IN RK0271ARG_ARRAY -- 5個目の「?」⇒登録データ配列
);
/
CREATE OR REPLACE TYPE RK0271ARG_ARRAY AS VARRAY(100) OF RK0271ARG;
```

・ RK0271ARGの定義の仕方

```
(RK0271ARG.sql)
DROP TYPE RK0271ARG_ARRAY;
CREATE OR REPLACE TYPE RK0271ARG AS OBJECT
(
  UNIQ VARCHAR2(11)
  ,USRID VARCHAR2(5)
  ,ECNO VARCHAR(7)
  ,EDBN VARCHAR(2)
);
/
CREATE OR REPLACE TYPE RK0271ARG_ARRAY AS VARRAY(100) OF RK0271ARG;
```

1.queryType

2.command

3.scope

【TAG】 Query を発行する為のクラスIDを指定します(初期値:JDBCPLSQL)。
引数指定のINSERT/UPDATE文を実行する場合の、queryType 属性を使用します。
このタグでは、execute(String ,String , DBSysArg[] , DBUserArg[])を実行します。
代表的なクラスとして、“JDBCPLSQL” が標準で用意されています。

タグにより使用できる/出来ないがありますが、これは、org.opengion.hayabusa.db
以下の Query_**** クラスの **** を与えます。
これらは、Query インターフェースを継承したサブクラスです。

【TAG】 コマンド(NEW,RENEW)をセットします(PlsqlUpdateTag,UpdateTag の場合は、ENTRY)。
コマンドは、HTMLから(get/post)指定されますので、CMD_*** で設定される
フィールド定数値のいずれかを、指定できます。

【TAG】 キャッシュする場合のスコープ[request/page/session/application]を指定します(初期値:session)。
“request”, “page”, “session”, “application” が指定できます。
JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、
主にテーブルモデルを管理するスコープを指します。

スコープ	変数の有効範囲	
page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会
request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ
session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。内部的には、同じキーワー
application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。

【TAG】 (通常は使いません) データの最大読み込み件数を指定します

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
4.maxRowCount	<p>(初期値:DB_MAX_ROW_COUNT[=1000])。</p> <p>データベース自体の検索は、指定されたSQLの全件を検索しますが、DBTableModelのデータとして登録する最大件数をこの値に設定します。サーバーのメモリ資源と応答時間の確保の為です。</p> <p>0 をセットすると、無制限(Integer.MAX_VALUE)になります。</p> <p>(初期値:ユーザー定数のDB_MAX_ROW_COUNT[=@og.value org.opengion.hayabusa.common.SystemData#DB_MAX_ROW_COUNT])。</p>
5.skipRowCount	<p>【TAG】(通常は使いません)データの読み始めの初期値を指定します。</p> <p>データベース自体の検索は、指定されたSQLの全件を検索しますが、DBTableModelのデータとしては、スキップ件数分は登録されません。サーバーのメモリ資源と応答時間の確保の為です。</p>
6.notfoundMsg	<p>【TAG】検索結果がゼロ件の場合に表示するメッセージリソースIDを指定します(初期値:MSG0077[対象データはありませんでした])。</p> <p>ここでは、検索結果がゼロ件の場合のみ、特別なメッセージを表示させます。</p> <p>従来は、displayMsg と兼用で、『0 件検索しました』という表示でしたが、displayMsg の初期表示は、OFF になりましたので、ゼロ件の場合のみ別に表示させます。</p> <p>表示させたくない場合は、notfoundMsg = "" をセットしてください。</p> <p>初期値は、MSG0077[対象データはありませんでした]です。</p>
7.names	<p>【TAG】PL/SQLを利用する場合の引数にセットすべき データの名称をCSV形式で複数指定します。</p> <p>複数ある場合は、カンマ区切り文字で渡します。</p> <p>names 属性は、queryType に応じて設定可否が異なりますので、ご注意ください。</p> <p>names なし: JDBC, JDBCUpdate</p> <p>names あり: JBCCallable, JBCErrMsg, JDBCUpdate</p> <p>(JDBCUpdateは、names 属性のあり/なし両方に対応しています。)</p>
8.dbType	<p>【TAG】Queryオブジェクトに渡す引数のタイプ定義(例:type名_ARRAY)。</p> <p>ここでは、type 定義のPL/SQL名を指定します。</p> <p>行を表す配列は、type名_ARRAY という名称です。</p>
9.selectedAll	<p>【TAG】データを全件選択済みとして処理するかどうか[true/false]を指定します(初期値:false)。</p> <p>全てのデータを選択済みデータとして扱って処理します。</p> <p>全件処理する場合に、(true/false)を指定します。</p> <p>初期値は false です。</p>
10.tableId	<p>【TAG】(通常は使いません)結果のDBTableModelを、sessionに登録するときのキーを指定します。</p> <p>検索結果より、DBTableModelオブジェクトを作成します。これを、以下のview 等のタグに渡す場合に、通常は、session を利用します。その場合の登録キーです。</p> <p>query タグを同時に実行して、結果を求める場合、同一メモリに配置される為、この tableId 属性を利用して、メモリ空間を分けます。</p> <p>初期値は、HybsSystem.TBL_MDL_KEY です。</p>
11.dbid	<p>【TAG】(通常は使いません)Queryオブジェクトを作成する時のDB接続IDを指定します。</p> <p>Queryオブジェクトを作成する時のDB接続IDを指定します。</p> <p>これは、システムリソースで、DEFAULT_DB_URL 等で指定している データベース接続先情報に、XX_DB_URL を定義することで、dbid="XX" とすると、この 接続先を使用してデータベースにアクセスできます。</p>
12.stopError	<p>【TAG】PLSQL/SQL処理エラーの時に処理を中止するかどうか[true/false]を設定します(初期値:true)。</p> <p>false(中止しない)に設定する場合、後続処理では、{@DB_ERR_CODE}の値により、PLSQL/SQLの異常/正常終了によって分岐処理は可能となります。</p> <p>初期値は、true(中止する)です。</p>
13.debug	<p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。</p> <p>デバッグ情報を [true:出力する/false:しない]を指定します。</p> <p>出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTML EXT BODY要素	内容
形式サンプル	
<div>5.5.4.2</div> <div>og:popup</div> <div>拡張</div> <div>Tag</div> <div>●形式 : <og:popup gamenId="..." href="..." keys="..." vals="..." rtnKeys="..." /></div> <div>●body : なし</div> <div>●Tag定義 :</div> <div><div><og:popup</div><div>name</div><div>gamenId</div><div>href</div><div>keys</div><div>vals</div><div>rtnKeys</div><div>column</div><div>onMark</div><div>markList</div><div>height</div><div>width</div><div>src</div><div>charset</div><div>target</div><div>caseKey</div><div>caseVal</div><div>id</div><div>lang</div><div>dir</div><div>title</div><div>style</div><div>tabindex</div><div>accesskey</div><div>clazz</div><div>lbl</div><div>language</div><div>command</div><div>debug</div><div>></div></div> <div>●使用例</div> <div>・普通のポップアップ</div> <div><og:popup gamenId="GE0002" href="index.jsp" target="CONTENTS"</div> <div>keys="AAA, BBB, CCC" vals="[@AAA], [@B], ABC"</div> <div>rtnKeys="DDD, EEE" /></div> <div>・QUERYの入力フィールドに組み込む</div> <div>①keys、vals、rtnKeys がセットされる場合</div> <div>セットされる keys に対して vals の値をポップアップ画面に渡し、ポップアップ画面の戻り値を rtnKeys にセットされます。</div> <div><og:column name="AAA" value="[@AAA]" ></div> <div><og:popup gamenId="GE0002"</div> <div>keys="AAA, BBB, CCC" vals="[@AAA], [@B], ABC"</div> <div>rtnKeys="DDD, EEE" /></div> <div></og:column></div> <div>②keys、vals、rtnKeys がセットされない場合</div> <div>ポップアップ画面にリクエスト変数 XNAME に外側のカラム名、XVALUE にカラムの入力値、rtnKeys にカラム名を自動的にセットされます。</div> <div><og:column name="AAA" value="[@AAA]" ></div> <div><og:popup gamenId="GE0002" /></div> <div></og:column></div> <div>・RESULTの入力フィールドに組み込む(viewMarker 経由)</div> <div>gamenId を指定しなければこの画面へのURLを作成します。</div> <div>このURLは、その画面の index.jsp に直接ジャンプするためのURLです。</div> <div><og:viewMarker command="[@command]" isRenderer="false"></div> <div><og:popup gamenId="GE0002" column="ABC" onMark="[ABC]" markList="1 2 3"</div> <div>keys="AAA, BBB, CCC" vals="[AAA], [BBB].value, [CCC]"</div> <div>rtnKeys="DDD, EEE" /></div> <div></og:viewMarker></div> <div>・ポップアップ側のJSPファイル(view 部分)</div>	<div>【TAG】 この要素をリンクの対象とするための名前を指定します。</div> <div>この要素をリンクの対象とするための名前を指定します。</div> <div>【TAG】 この要素をリンクの対象とするための名前を指定します</div> <div>【TAG】 画面ID属性を登録します</div> <div>【TAG】 ポップアップ先のURLを指定します (初期値:index.jsp)</div> <div>【TAG】 ポップアップ先に渡す入力キー配列をCSV形式で複数指定します</div> <div>【TAG】 ポップアップ先に渡す値配列をCSV形式で複数指定します</div> <div>【TAG】 ポップアップ先より受け取るキー配列をCSV形式で複数指定します</div> <div>【TAG】 リンクをつきたいカラムを指定します</div> <div>【TAG】 リンクを張る(true or 1)/張らない(false or 0)の設定を指定します (初期値:true)</div> <div>【TAG】 処理するマークを、“ ”区切りの文字列で指定します</div> <div>【TAG】 ダイアログの縦幅をピクセル数で指定します</div> <div>【TAG】 ダイアログの横幅をピクセル数で指定します</div> <div>【TAG】 表示させる画像のURLを指定します</div> <div>【TAG】 ポップアップ先の文字コードセットを指定します</div> <div>【TAG】 ポップアップ先の表示ターゲットを指定します (初期値:CONTENTS)</div> <div>【TAG】 このタグ自体を利用するかどうかの条件キーを指定します (初期値:null)</div> <div>【TAG】 このタグ自体を利用するかどうかの条件値を指定します (初期値:null)</div> <div>【HTML】 要素に対して固有の名前(id)をつける場合に設定します</div> <div>【HTML】 要素の内容と他の属性値の言語(lang,xml:lang)を指定します</div> <div>【HTML】 文字表記の方向(dir)を指定します</div> <div>【HTML】 要素に対する補足的情報(title)を設定します</div> <div>【HTML】 この要素に対して適用させるスタイルシート(style)を設定します</div> <div>【HTML】 タブの移動順(tabindex)を指定します (0 ~ 32767)</div> <div>【HTML】 アクセスキー(alt+キーで直接指定)を割り当てます</div> <div>【HTML】 要素に対して class 属性を設定します</div> <div>【TAG】 ラベルリソースのラベルIDを指定します</div> <div>【TAG】 タグ内部で使用する言語コード[ja/en/zh/...]を指定します</div> <div>【TAG】 (通常使いません)処理の実行を指定する command を設定できます (初期値:NEW)</div> <div>【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します (初期値:false)</div> <div>【TAG】 この要素をリンクの対象とするための名前を指定します</div> <div>【TAG】 画面ID属性を登録します</div> <div>【TAG】 ポップアップ先のURLを指定します (初期値:index.jsp)</div> <div>【TAG】 ポップアップ先に渡す入力キー配列をCSV形式で複数指定します</div> <div>【TAG】 ポップアップ先に渡す値配列をCSV形式で複数指定します</div> <div>【TAG】 ポップアップ先より受け取るキー配列をCSV形式で複数指定します</div> <div>【TAG】 リンクをつきたいカラムを指定します</div> <div>【TAG】 リンクを張る(true or 1)/張らない(false or 0)の設定を指定します (初期値:true)</div> <div>【TAG】 処理するマークを、“ ”区切りの文字列で指定します</div> <div>【TAG】 ダイアログの縦幅をピクセル数で指定します</div> <div>【TAG】 ダイアログの横幅をピクセル数で指定します</div> <div>【TAG】 表示させる画像のURLを指定します</div> <div>【TAG】 ポップアップ先の文字コードセットを指定します</div> <div>【TAG】 ポップアップ先の表示ターゲットを指定します (初期値:CONTENTS)</div> <div>【TAG】 このタグ自体を利用するかどうかの条件キーを指定します (初期値:null)</div> <div>【TAG】 このタグ自体を利用するかどうかの条件値を指定します (初期値:null)</div> <div>【HTML】 要素に対して固有の名前(id)をつける場合に設定します</div> <div>【HTML】 要素の内容と他の属性値の言語(lang,xml:lang)を指定します</div> <div>【HTML】 文字表記の方向(dir)を指定します</div> <div>【HTML】 要素に対する補足的情報(title)を設定します</div> <div>【HTML】 この要素に対して適用させるスタイルシート(style)を設定します</div> <div>【HTML】 タブの移動順(tabindex)を指定します (0 ~ 32767)</div> <div>【HTML】 アクセスキー(alt+キーで直接指定)を割り当てます</div> <div>【HTML】 要素に対して class 属性を設定します</div> <div>【TAG】 ラベルリソースのラベルIDを指定します</div> <div>【TAG】 タグ内部で使用する言語コード[ja/en/zh/...]を指定します</div> <div>【TAG】 (通常使いません)処理の実行を指定する command を設定できます (初期値:NEW)</div> <div>【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します (初期値:false)</div> <div>【TAG】 この要素をリンクの対象とするための名前を指定します。</div> <div>この要素をリンクの対象とするための名前を指定します。</div> <div>【TAG】 画面ID属性を登録します。</div> <div>gamenId 属性は、画面IDをキーに、実アドレスを求めるのに使用します。</div> <div>画面IDが指定された場合は、実アドレスに変換する。指定されない場合は、href 属性をそのままリンク情報を作成します。</div> <div>画面IDを指定する場合の href は、JSPファイル名(例えば、index.jsp 等)です。</div> <div>【TAG】 ポップアップ先のURLを指定します (初期値:index.jsp)。</div> <div>通常、gamenId を指定している場合は、それ以降(index.jsp , result.jsp など)を指定します。</div> <div>通常のリンクの場合、GAMENID/href というアドレスに合成されます。(GE0001/result.jsp など)</div> <div>(正確には、画面IDではなく、対応する画面ADDRESSを使用します。)</div> <div>ポップアップ専用のリンクなので、初期値は、index.jsp です。</div> <div>【TAG】 ポップアップ先に渡す入力キー配列をCSV形式で複数指定します。</div> <div>ポップアップ先に渡す入力キー配列を指定します。</div> <div>分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。</div> <div>こうしないとデータ自身にカンマを持っている場合に分解をミスの為です。</div> <div>【TAG】 ポップアップ先に渡す値配列をCSV形式で複数指定します。</div> <div>ポップアップ先に渡す値配列を指定します。</div> <div>分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。</div> <div>こうしないとデータ自身にカンマを持っている場合に分解をミスの為です。</div> <div>【TAG】 ポップアップ先より受け取るキー配列をCSV形式で複数指定します。</div> <div>ポップアップ先より受け取るキー配列を指定します。</div> <div>分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。</div> <div>こうしないとデータ自身にカンマを持っている場合に分解をミスの為です。</div> <div>【TAG】 リンクをつきたいカラムを指定します。</div> <div>このカラム名のTableModelに対して、 ViewLink タグが存在する場合は、リンクを張ります。</div> <div>【TAG】 リンクを張る(true or 1)/張らない(false or 0)の設定を指定します (初期値:true)。</div> <div>リンクを張る場合は、“true”(または “1”) リンクを張らない場合は、“true以外”(または “0”)をセットします。</div> <div>初期値は、“true”(リンクを張る)です。</div> <div>さらに、[カラム名] で、動的にカラムの値で、リンクを張る、張らないを</div>
1.name	【TAG】 この要素をリンクの対象とするための名前を指定します。

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
9.markList	<p>選ぶ事が可能になります。値は、“true”(または“1”) で、マークします。</p> <p>ただし、単独 linkタグ時は使用できません。</p> <p>追記 3.5.2.0 (2003/10/20) : markList属性に、処理対象文字列郡を指定できます。</p> <p>これにより、リンクを処理するかどうかの判断が、true,1 以外にも使用できるようになりました。</p> <p>【TAG】 処理するマークを、“ ”区切りの文字列で指定します。</p> <p>markListで与えられた文字列に、onMark文字列(大文字/小文字の区別あり)が 含まれていれば、処理します。</p> <p>例えば、“A”、“B”、“C” という文字列が、onMark で指定された 場合に処理するようにしたい場合は、“A B C” をセットします。</p> <p>markList には、[カラム名]指定が可能です。(ただし、単独 linkタグ時は除く)</p> <p>初期値は、“true TRUE 1”です。</p>
10.height	<p>【TAG】 ダイアログの縦幅をピクセル数で指定します。</p> <p>縦幅を指定します。単位も含めて入力してください。</p> <p>設定しない場合は、ogPopup JavaScript の内部初期値が採用されます。</p>
11.width	<p>【TAG】 ダイアログの横幅をピクセル数で指定します。</p> <p>横幅を指定します。単位も含めて入力してください。</p> <p>設定しない場合は、ogPopup JavaScript の内部初期値が採用されます。</p>
12.src	<p>【TAG】 表示させる画像のURLを指定します。</p> <p>この属性を設定すると、自動的に イメージリンクでの onClick属性で ポップアップの起動画面を作成します。</p> <p>この属性が設定されていない場合は、button での起動になります。</p>
13.charset	<p>【TAG】 ポップアップ先の文字コードセットを指定します。</p> <p>ポップアップ先の文字コードセットを指定します。</p>
14.target	<p>【TAG】 ポップアップ先の表示ターゲットを指定します(初期値:CONTENTS)。</p> <p>ポップアップ先の表示ターゲットを指定します。</p>
15.caseKey	<p>【TAG】 このタグ自体を利用するかどうかの条件キーを指定します(初期値:null)。</p> <p>caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。</p> <p>false の場合は、このタグは使用されません。</p> <p>何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。</p>
16.caseVal	<p>【TAG】 このタグ自体を利用するかどうかの条件値を指定します(初期値:null)。</p> <p>caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。</p> <p>false の場合は、このタグは使用されません。</p> <p>何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。</p>
17.id	<p>【HTML】 要素に対して固有の名前(id)をつける場合に設定します。</p> <p>特別な使用方法として、id="FOCUS" とすることで、フィールド系要素に フォーカスを移動させます。これは、そのページ内で唯一の id 属性として使用ください。</p>
18.lang	<p>【HTML】 要素の内容と他の属性値の言語(lang,xml:lang)を指定します。</p> <p>HTMLの言語属性に使われます。指定する値は、ISO 639で規定されている「言語コード」です。 [ja/en/zh/...]などのほかに、en-US:アメリカ英語、en-cockney:コックニー英語 など、 副言語を指定する方法も定められています。</p> <p>ここでは、lang と xml:lang の両方に同じ値がセットされます。</p> <p>タグの language 属性とは使用用途が異なります。lang セット</p>
19.dir	<p>【HTML】 文字表記の方向(dir)を指定します。</p> <p>当該要素の書字方向を指定する属性です。</p> <p>ltr で、左から右に、rtl で、右から左に並べます。</p>
20.title	<p>【HTML】 要素に対する補足的情報(title)を設定します。</p> <p>title セットは、ボタンなどに適用すると、マウスオーバーによりこのメッセージが チップスのように表示されます。これを利用して、説明文を登録することが可能です。</p> <p>ここに登録した文字列が、メッセージリソースに存在する場合は、そのメッセージを 存在しない場合は、そのままの値を設定します。</p>
21.style	<p>【HTML】 この要素に対して適用させるスタイルシート(style)を設定します。</p> <p>タグにstyle属性を設定します。これは、キー:値: のセットを複数記述できます。</p> <p>通常は、class属性や、id属性で登録しておき、&lg:style type="text/css"> で 外部から指定する方がソースは読みやすくなります。</p>
22.tabindex	<p>【HTML】 タブの移動順(tabindex)を指定します(0 ~ 32767)。</p> <p>Tabキーを押したときに要素が選択される順番を指定します。</p> <p>値には、選択させたい順番を数値で記述します。</p>
23.accesskey	<p>【HTML】 アクセスキー(alt+キーで直接指定)を割り当てます。</p> <p>アクセスキーは、マウスの使えない環境でも、リンクにジャンプする、ボタンを押す、入力フォームに フォーカスを移すなどの操作を簡単に行うことができるように考慮されたものです。</p> <p>Windows の「ファイル(F)」メニューについている、F と同じような動きをします。</p>
24.clazz	<p>【HTML】 要素に対して class 属性を設定します。</p> <p>Javaの言語使用上、class で作成できないため、代用として clazz を使用しています。</p> <p>html で作成される属性は、class で作成されます。</p>
25.lbl	<p>【TAG】 ラベルリソースのラベルIDを指定します。</p> <p>ラベルを変更するときに、lbl属性を使います。</p> <p>ラベルID は、所定の language に基づく ResourceManager の getLabelData(id) を呼び出し、その結果のLabelInterfaceを使用します。 getMsgLbl() で取り出せます。</p> <p>ラベルIDとメッセージIDは同時には登録できません。</p> <p>ラベルとメッセージは統一されました。</p>
26.language	<p>【TAG】 タグ内部で使用する言語コード[ja/en/zh/...]を指定します。</p> <p>この言語コードに基づいて、表示のラベルをリソースから作成します。</p> <p>【TAG】 (通常使いません)処理の実行を指定する command を設定できます(初期値:NEW)。</p>
27.command	<p>ここで指定されたcommandは、ポップアップ画面の引数として渡されます。</p> <p>初期値は、NEW です。</p>
28.debug	<p>【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。</p> <p>デバッグ情報を [true:出力する/false:しない]を指定します。</p> <p>出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
5.5.4.2 og:process JSP	<div>形式サンプル</div> <p>【TAG】 リクエスト情報 に登録するキーをセットします。 processID は、org.opengion.fukurou.process.HybsProcess インターフェースを実装した Process_**** クラスの **** を与えます。 これらは、HybsProcess インターフェースを継承したサブクラスである必要があります。 標準で、org.opengion.fukurou.process 以下の Process_**** クラスが、Process_**** 宣言 と して、定義されています。 属性クラス定義の {@link org.opengion.fukurou.process.HybsProcess HybsProcess} を参照願います。</p>
	<p>●形式 : <og:process processID="ZZZ" > <og:param key="AAA" value="111" /> </og:process > ●body : あり (param)</p> <p>●Tag定義 : <og:process processID ○【TAG】リクエスト情報 に登録するキーをセットします。(必須) debug 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false) > ... Body ... </og:process></p> <p>●使用例 一般的な変数の渡し方 <og:mainProcess > <og:process processID="DBReader" > <og:param key="dbid" value="FROM" /> <og:param key="sql" value="select * from GE02" /> </og:process > <og:process processID="DBWriter" > <og:param key="dbid" value="T0" /> <og:param key="table" value="GE02" /> </og:process > </og:mainProcess ></p> <p> BODY 部に記述した変数の渡し方 <og:process processID="DBReader" > <og:param key="SQL" > SELECT COUNT(*) FROM GEA03 WHERE SYSTEM_ID=[SYSTEM_ID] AND CLM=[CLM] AND FGJ = '1' </og:param> </og:process ></p>
1.processID	<p>【TAG】 リクエスト情報 に登録するキーをセットします。 processID は、org.opengion.fukurou.process.HybsProcess インターフェースを実装した Process_**** クラスの **** を与えます。 これらは、HybsProcess インターフェースを継承したサブクラスである必要があります。 標準で、org.opengion.fukurou.process 以下の Process_**** クラスが、Process_**** 宣言 と して、定義されています。 属性クラス定義の {@link org.opengion.fukurou.process.HybsProcess HybsProcess} を参照願います。</p>
2.debug	<p>【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容																																														
	形式サンプル																																														
5.5.4.2 og:query JSP	<p>【TAG】Query を発行する為のクラスID(JDBC, JDBCCallable, JDBCErrMsg, JDBCUpdate)を指定します(初期値:JDBC)。</p> <p>検索を実行する手段は、Query インターフェースの実装クラスになります。</p> <p>このタグでは、Query.execute(String[]) メソッドが呼ばれます。</p> <p>例えば、ストアプロシージャ等を実行する場合に、queryType="JDBCErrMsg" を指定することができます。</p> <p>初期値は、“JDBC” です。</p> <p>queryType は、システムリソースの Query_**** 宣言の **** を与えます。</p> <p>これらは、Query インターフェースを継承したサブクラスである必要があります。</p> <p>標準で、org.opengion.hayabusa.db 以下の Query_**** クラスが、Query_**** 宣言として、定義されています。</p> <p>属性クラス定義の {@link org.opengion.hayabusa.db.Query Query} を参照願います。</p>																																														
●形式：	<pre><og:query command="NEW" > SELECT文 </og:query> <og:query command="NEW" names="..." queryType="JDBCErrMsg" > { call PL/SQL(?, ?, ?, ?) } </og:query></pre>																																														
●body：あり																																															
●Tag定義：	<p>og:query</p> <table><tr><td>queryType</td><td>【TAG】Query を発行する為のクラスID(JDBC, JDBCCallable, JDBCErrMsg, JDBCUpdate)を指定します(初期値:JDBC)</td></tr><tr><td>command</td><td>【TAG】 コマンド(NEW, RENEW)をセットします(PlsqlUpdateTag, UpdateTag の場合は、ENTRY)</td></tr><tr><td>scope</td><td>【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)</td></tr><tr><td>maxRowCount</td><td>【TAG】 (通常は使いません)データの最大読み込み件数を指定します(初期値:DB_MAX_ROW_COUNT[=1000])</td></tr><tr><td>skipRowCount</td><td>【TAG】 (通常は使いません)データの読み始めの初期値を指定します</td></tr><tr><td>displayMsg</td><td>【TAG】 検索結果を画面上に表示するメッセージリソースIDを指定します(初期値:VIEW_DISPLAY_MSG[=])</td></tr><tr><td>overflowMsg</td><td>【TAG】 検索データが最大検索数をオーバーした場合に表示するメッセージリソースIDを指定します(初期値:MSG0007[検索結果が、制限行数を超えまし</td></tr><tr><td>notfoundMsg</td><td>【TAG】 検索結果がゼロ件の場合に表示するメッセージリソースIDを指定します(初期値:MSG0077[対象データはありませんでした])</td></tr><tr><td>names</td><td>【TAG】 PL/SQLを利用する場合の引数にセットすべき データの名称をCSV形式で複数指定します</td></tr><tr><td>stopZero</td><td>【TAG】 検索結果が0件のとき処理を続行するかどうか[true/false]を指定します(初期値:false[続行する])</td></tr><tr><td>tableId</td><td>【TAG】 (通常は使いません)結果のDBTableModelを、sessionに登録するときのキーを指定します</td></tr><tr><td>dbId</td><td>【TAG】 (通常は使いません)Queryオブジェクトを作成する時のDB接続IDを指定します</td></tr><tr><td>trace</td><td>【TAG】 引数の SQL 文を EXPLAIN PLAN を[true:行う/それ以外:行わない]を指定します(初期値:false)</td></tr><tr><td>checkNames</td><td>【TAG】 リクエスト変数の正規化を行うカラムをCSV形式で複数指定します</td></tr><tr><td>modifyType</td><td>【TAG】 DB検索時の モディファイタイプを指定します[A:追加/C:更新/D:削除]</td></tr><tr><td>quotCheck</td><td>【TAG】 リクエスト情報の クォーティション(‘’) 存在チェックを実施するかどうか[true/false]を設定します(初期値:USE_SQL_INJECTION_CHECK)</td></tr><tr><td>stopError</td><td>【TAG】 PLSQL/SQL処理エラーの時に処理を中止するかどうか[true/false]を設定します(初期値:true)</td></tr><tr><td>xssCheck</td><td>【TAG】 リクエスト情報の HTMLTag開始/終了文字(<>) 存在チェックを実施するかどうか[true/false]を設定します(初期値:USE_XSS_CHECK[=true])</td></tr><tr><td>mainTrans</td><td>【TAG】 (通常使いません)タグで処理される処理がメインとなるトランザクション処理かどうかを指定します(初期値:true)</td></tr><tr><td>caseKey</td><td>【TAG】 このタグ自体を利用するかどうかの条件キーを指定します(初期値:null)</td></tr><tr><td>caseVal</td><td>【TAG】 このタグ自体を利用するかどうかの条件値を指定します(初期値:null)</td></tr><tr><td>useBeforeHtmlTag</td><td>【TAG】 処理時間(queryTime)などの情報出力[true:有効/false:無効]を指定します(初期値:true)</td></tr><tr><td>debug</td><td>【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)</td></tr></table> <pre>> ... Body ... </og:query></pre>	queryType	【TAG】Query を発行する為のクラスID(JDBC, JDBCCallable, JDBCErrMsg, JDBCUpdate)を指定します(初期値:JDBC)	command	【TAG】 コマンド(NEW, RENEW)をセットします(PlsqlUpdateTag, UpdateTag の場合は、ENTRY)	scope	【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)	maxRowCount	【TAG】 (通常は使いません)データの最大読み込み件数を指定します(初期値:DB_MAX_ROW_COUNT[=1000])	skipRowCount	【TAG】 (通常は使いません)データの読み始めの初期値を指定します	displayMsg	【TAG】 検索結果を画面上に表示するメッセージリソースIDを指定します(初期値:VIEW_DISPLAY_MSG[=])	overflowMsg	【TAG】 検索データが最大検索数をオーバーした場合に表示するメッセージリソースIDを指定します(初期値:MSG0007[検索結果が、制限行数を超えまし	notfoundMsg	【TAG】 検索結果がゼロ件の場合に表示するメッセージリソースIDを指定します(初期値:MSG0077[対象データはありませんでした])	names	【TAG】 PL/SQLを利用する場合の引数にセットすべき データの名称をCSV形式で複数指定します	stopZero	【TAG】 検索結果が0件のとき処理を続行するかどうか[true/false]を指定します(初期値:false[続行する])	tableId	【TAG】 (通常は使いません)結果のDBTableModelを、sessionに登録するときのキーを指定します	dbId	【TAG】 (通常は使いません)Queryオブジェクトを作成する時のDB接続IDを指定します	trace	【TAG】 引数の SQL 文を EXPLAIN PLAN を[true:行う/それ以外:行わない]を指定します(初期値:false)	checkNames	【TAG】 リクエスト変数の正規化を行うカラムをCSV形式で複数指定します	modifyType	【TAG】 DB検索時の モディファイタイプを指定します[A:追加/C:更新/D:削除]	quotCheck	【TAG】 リクエスト情報の クォーティション(‘’) 存在チェックを実施するかどうか[true/false]を設定します(初期値:USE_SQL_INJECTION_CHECK)	stopError	【TAG】 PLSQL/SQL処理エラーの時に処理を中止するかどうか[true/false]を設定します(初期値:true)	xssCheck	【TAG】 リクエスト情報の HTMLTag開始/終了文字(<>) 存在チェックを実施するかどうか[true/false]を設定します(初期値:USE_XSS_CHECK[=true])	mainTrans	【TAG】 (通常使いません)タグで処理される処理がメインとなるトランザクション処理かどうかを指定します(初期値:true)	caseKey	【TAG】 このタグ自体を利用するかどうかの条件キーを指定します(初期値:null)	caseVal	【TAG】 このタグ自体を利用するかどうかの条件値を指定します(初期値:null)	useBeforeHtmlTag	【TAG】 処理時間(queryTime)などの情報出力[true:有効/false:無効]を指定します(初期値:true)	debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)
queryType	【TAG】Query を発行する為のクラスID(JDBC, JDBCCallable, JDBCErrMsg, JDBCUpdate)を指定します(初期値:JDBC)																																														
command	【TAG】 コマンド(NEW, RENEW)をセットします(PlsqlUpdateTag, UpdateTag の場合は、ENTRY)																																														
scope	【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)																																														
maxRowCount	【TAG】 (通常は使いません)データの最大読み込み件数を指定します(初期値:DB_MAX_ROW_COUNT[=1000])																																														
skipRowCount	【TAG】 (通常は使いません)データの読み始めの初期値を指定します																																														
displayMsg	【TAG】 検索結果を画面上に表示するメッセージリソースIDを指定します(初期値:VIEW_DISPLAY_MSG[=])																																														
overflowMsg	【TAG】 検索データが最大検索数をオーバーした場合に表示するメッセージリソースIDを指定します(初期値:MSG0007[検索結果が、制限行数を超えまし																																														
notfoundMsg	【TAG】 検索結果がゼロ件の場合に表示するメッセージリソースIDを指定します(初期値:MSG0077[対象データはありませんでした])																																														
names	【TAG】 PL/SQLを利用する場合の引数にセットすべき データの名称をCSV形式で複数指定します																																														
stopZero	【TAG】 検索結果が0件のとき処理を続行するかどうか[true/false]を指定します(初期値:false[続行する])																																														
tableId	【TAG】 (通常は使いません)結果のDBTableModelを、sessionに登録するときのキーを指定します																																														
dbId	【TAG】 (通常は使いません)Queryオブジェクトを作成する時のDB接続IDを指定します																																														
trace	【TAG】 引数の SQL 文を EXPLAIN PLAN を[true:行う/それ以外:行わない]を指定します(初期値:false)																																														
checkNames	【TAG】 リクエスト変数の正規化を行うカラムをCSV形式で複数指定します																																														
modifyType	【TAG】 DB検索時の モディファイタイプを指定します[A:追加/C:更新/D:削除]																																														
quotCheck	【TAG】 リクエスト情報の クォーティション(‘’) 存在チェックを実施するかどうか[true/false]を設定します(初期値:USE_SQL_INJECTION_CHECK)																																														
stopError	【TAG】 PLSQL/SQL処理エラーの時に処理を中止するかどうか[true/false]を設定します(初期値:true)																																														
xssCheck	【TAG】 リクエスト情報の HTMLTag開始/終了文字(<>) 存在チェックを実施するかどうか[true/false]を設定します(初期値:USE_XSS_CHECK[=true])																																														
mainTrans	【TAG】 (通常使いません)タグで処理される処理がメインとなるトランザクション処理かどうかを指定します(初期値:true)																																														
caseKey	【TAG】 このタグ自体を利用するかどうかの条件キーを指定します(初期値:null)																																														
caseVal	【TAG】 このタグ自体を利用するかどうかの条件値を指定します(初期値:null)																																														
useBeforeHtmlTag	【TAG】 処理時間(queryTime)などの情報出力[true:有効/false:無効]を指定します(初期値:true)																																														
debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)																																														
●使用例	<pre><og:query command="NEW"> select PN, YOBI, NMEN, HINM from XX01 where PN = '{@PN}' order by PN </og:query></pre> <p>・ 検索条件が入力された時([@PN]がNOT NULLのとき) 作成されるSQL文⇒select PN, YOBI, NMEN, HINM from XX01 where PN = 'AAA' order by PN</p> <p>・ 検索条件が入力されなかった時([@PN]がNULLのとき) 作成されるSQL文⇒select PN, YOBI, NMEN, HINM from XX01 where PN = '' order by PN</p> <pre><og:query command="NEW"> select PN, YOBI, NMEN, HINM from XX01 <og:where> <og:and value="PN = '{@PN}%" /> <og:and value="YOBI like '{@YOBI}%" /> </og:where> order by PN </og:query></pre> <p>・ 検索条件が入力された時([@PN]がNOT NULLのとき) 作成されるSQL文⇒select PN, YOBI, NMEN, HINM from XX01 PN = 'AAA%' and YOBI like 'BBB%' order by PN</p> <p>・ 検索条件が入力されなかった時([@PN]がNULLのとき) WHERE句がなくなる。 作成されるSQL文⇒select PN, YOBI, NMEN, HINM from XX01 order by PN</p> <p>注意:WhereTagを使った場合、下のようにはなりません。</p> <pre>select PN, YOBI, NMEN, HINM from XX01 PN = '' and YOBI like '%' order by PN</pre> <pre><og:query command="NEW"> select PN, YOBI, NMEN, HINM from XX01 where PN="11111" <og:where startKey="and"> <og:and value="YOBI like '{@PN}%" /> </og:where> order by PN </og:query></pre> <p>・ 検索条件が入力された時([@YOBI]がNOT NULLのとき) 作成されるSQL文⇒select PN, YOBI, NMEN, HINM from XX01 PN = '11111' and YOBI like 'BBB%' order by PN</p> <p>・ 検索条件が入力されなかった時([@YOBI]がNULLのとき) WHERE句がなくなる。 作成されるSQL文⇒select PN, YOBI, NMEN, HINM from XX01 PN = '11111' order by PN</p> <pre><og:query command = "NEW" names = "SYSTEM_ID, LANG, CLM, NAME_JA, LABEL_NAME, KBSAKU, USER, ID" checkNames = "CLM, NAME_JA" queryType = "JDBCErrMsg" displayMsg = "" [call TYPE3B01, TYPE3B01(?, ?, ?, ?)] </og:query></pre> <p>・ queryType に JDBCErrMsg を指定して、PL/SQL をコールできます。 引数は、names 属性をキーにリクエスト変数から読み込みます。 ・ checkNames にカラム名を指定すると、columns.valueSet による</p>																																														

バージョン タグ名 HTMLTEXT BODY要素	内容															
形式サンプル																
リクエスト変数の正規化を行います。																
1.queryType	<p>【TAG】 Query を発行する為のクラスID(JDBC, JDBCCallable, JDBCErrMsg, JDBCUpdate)を指定します(初期値:JDBC)。</p> <p>検索を実行する手段は、Query インターフェースの実装クラスになります。</p> <p>このタグでは、Query.execute(String[]) メソッドが呼ばれます。</p> <p>例えば、ストアドプロシージャ等を実行する場合に、queryType="JDBCErrMsg"を指定することができます。</p> <p>初期値は、“JDBC” です。</p> <p>queryType は、システムリソースの Query_**** 宣言の **** を与えます。</p> <p>これらは、Query インターフェースを継承したサブクラスである必要があります。</p> <p>標準で、org.opengion.hayabusa.db 以下の Query_**** クラスが、Query_**** 宣言 として、定義されています。</p> <p>属性クラス定義の [@link org.opengion.hayabusa.db.Query Query] を参照願います。</p>															
2.command	<p>【TAG】 コマンド(NEW,RENEW)をセットします(PisqlUpdateIag,UpdateIag の場合は、ENTRY)。</p> <p>コマンドは、HTMLから(get/post)指定されますので、CMD_xxx で設定されるフィールド定数値のいづれかを、指定できます。</p>															
3.scope	<p>【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)。</p> <p>“request”, “page”, “session”, “applicaton” が指定できます。</p> <p>JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、主にテーブルモデルを管理するスコープを指します。</p> <table><tr><th>スコープ</th><th>変数の有効範囲</th><th></th></tr><tr><td>page</td><td>JSPページ内</td><td>そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会</td></tr><tr><td>request</td><td>HTTPリクエスト</td><td>リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ</td></tr><tr><td>session</td><td>HTTPセッション</td><td>初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。 内部的には、同じキーワー</td></tr><tr><td>applicaton</td><td>Webアプリケーション</td><td>ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。</td></tr></table>	スコープ	変数の有効範囲		page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会	request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ	session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。 内部的には、同じキーワー	applicaton	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。
スコープ	変数の有効範囲															
page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会														
request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ														
session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。 内部的には、同じキーワー														
applicaton	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。														
4.maxRowCount	<p>【TAG】 (通常は使いません) データの最大読み込み件数を指定します (初期値:DB_MAX_ROW_COUNT[=1000])。</p> <p>データベース自体の検索は、指定されたSQLの全件を検索しますが、DBTableModelのデータとして登録する最大件数をこの値に設定します。</p> <p>サーバーのメモリ資源と応答時間の確保の為です。</p> <p>0 をセットすると、無制限(Integer.MAX_VALUE)になります。</p> <p>(初期値:ユーザー定数のDB_MAX_ROW_COUNT[=@og.value org.opengion.hayabusa.common.SystemData#DB_MAX_ROW_COUNT]) 。</p>															
5.skipRowCount	<p>【TAG】 (通常は使いません) データの読み始めの初期値を指定します。</p> <p>データベース自体の検索は、指定されたSQLの全件を検索しますが、DBTableModelのデータとしては、スキップ件数分は登録されません。</p> <p>サーバーのメモリ資源と応答時間の確保の為です。</p>															
6.displayMsg	<p>【TAG】 検索結果を画面上に表示するメッセージリソースIDを指定します (初期値:VIEW_DISPLAY_MSG[=])。</p> <p>ここでは、検索結果の件数や登録された件数をまず出力し、その次に、ここで指定したメッセージをリソースから取得して表示します。</p> <p>件数を表示させたい場合は、displayMsg = “MSG0033”[件検索しました] をセットしてください。</p> <p>表示させたくない場合は、displayMsg = “” をセットしてください。</p> <p>(初期値:システム定数のVIEW_DISPLAY_MSG[=@og.value org.opengion.hayabusa.common.SystemData#VIEW_DISPLAY_MSG]) 。</p>															
7.overflowMsg	<p>【TAG】 検索データが最大検索数をオーバーした場合に表示するメッセージリソースIDを指定します (初期値:MSG0007[検索結果が、制限行数を超えましたので、残りはカットされました]) 。</p> <p>検索結果が、maxRowCount で設定された値より多い場合、何らかのデータは検索されず切り捨てられたことになります。</p> <p>ここでは、displayMsg を表示した後、必要に応じて、このメッセージを表示します。</p> <p>表示させたくない場合は、overflowMsg = “” をセットしてください。</p> <p>初期値は、MSG0007[検索結果が、制限行数を超えましたので、残りはカットされました]です。</p>															
8.notfoundMsg	<p>【TAG】 検索結果がゼロ件の場合にのみ、特別なメッセージを表示させます。</p> <p>ここでは、検索結果がゼロ件の場合のみ、特別なメッセージを表示させます。</p> <p>従来は、displayMsg と兼用で、『0 件検索しました』という表示でしたが、displayMsg の初期表示は、OFF になりましたので、ゼロ件の場合のみ別に表示させます。</p> <p>表示させたくない場合は、notfoundMsg = “” をセットしてください。</p> <p>初期値は、MSG0077[対象データはありませんでした]です。</p>															
9.names	<p>【TAG】 PL/SQLを利用する場合の引数にセットすべき データの名称をCSV形式で複数指定します。</p> <p>複数ある場合は、カンマ区切り文字で渡します。</p> <p>names 属性は、queryType に応じて設定可否が異なりますので、ご注意ください。</p> <p>names なし: JDBC, JDBCUpdate</p> <p>names あり: JDBCCallable, JDBCErrMsg, JDBCUpdate</p> <p>(JDBCUpdateは、names 属性のあり/なし両方に対応しています。)</p>															
10.stopZero	<p>【TAG】 検索結果が0件のとき処理を続行するかどうか[true/false]を指定します(初期値:false[続行する])。</p> <p>初期値は、false(続行する)です。</p>															
11.tableId	<p>【TAG】 (通常は使いません) 結果のDBTableModelを、sessionに登録するときのキーを指定します。</p> <p>検索結果より、DBTableModelオブジェクトを作成します。これを、以下のview 等のタグに渡す場合に、通常は、session を利用します。その場合の登録キーです。</p> <p>query タグを同時に実行して、結果を求める場合、同一メモリに配置される為、この tableId 属性を利用して、メモリ空間を分けます。</p> <p>初期値は、HybsSystem.TBL_MD_L KEY です。</p>															
12.dbid	<p>【TAG】 (通常は使いません) Queryオブジェクトを作成する時のDB接続IDを指定します。</p> <p>Queryオブジェクトを作成する時のDB接続IDを指定します。</p> <p>これは、システムリソースで、DEFAULT_DB_URL 等で指定している データベース接続先情報に、XX_DB_URL を定義することで、dbid="XX" とすると、この 接続先を使用してデータベースにアクセスできます。</p>															
13.trace	<p>【TAG】 引数の SQL 文を EXPLAIN PLAN を[true:行う/それ以外:行わない]を指定します(初期値:false)。</p> <p>ここでは、以下の処理を行います。</p> <p>1. 引数の SQL 文を画面に表示します。</p> <p>2. 引数の SQL 文を EXPLAIN PLAN した結果を、画面に表示します。</p> <p>なお、以前は、セッションのトレースを行っていましたが、その機能は、廃止いたします。</p> <p>初期値は、行わない: false です。</p>															
14.checkNames	<p>【TAG】 リクエスト変数の正規化を行うカラムをCSV形式で複数指定します。</p> <p>PL/SQLを利用する場合の引数にセットすべき データを、リクエスト変数の値のままではなく、カラムオブジェクトの valueSet メソッド経由で正規化した値を使用するようにします。</p>															
15.modifyType	<p>【TAG】 DB検索時の モディファイタイプを指定します[A:追加/C:更新/D:削除]。</p> <p>DB検索時に、そのデータをA(追加)、C(更新)、D(削除)のモディファイタイプをつけた状態にします。</p> <p>その状態で、そのまま、update する事が可能になります。</p> <p>【TAG】 リクエスト情報の クォーティション(‘) 存在チェックを実施するかどうか[true/false]を設定します (初期値:USE_SQL_INJECTION_CHECK[=true])。</p>															
16.quoteCheck	<p>S Q LインJECTION対策の一つとして、暫定的ではありますが、SQLのパラメータに渡す文字列にクォーティション(‘)を許さない設定にすれば、ある程度は防止できます。</p> <p>数字タイプの引数には、 or 5=5 などのクォーティションを使用しないコードを埋めても、数字チェックで検出可能です。文字タイプの場合は、必ず (‘) をはずして、 or 'A' like 'A' のような形式になる為、(‘) チェックだけでも有効です。</p> <p>(‘) が含まれていたエラーにする(true)／かノーチェックか(false)を指定します。</p> <p>初期値は、SystemData#USE_SQL_INJECTION_CHECK です。</p>															
	<p>【TAG】 PLSQL/SQL処理エラーの時に処理を中止するかどうか[true/false]を設定します(初期値:true)。</p> <p>false(中止しない)に設定する場合、後続処理では、{@DB.ERR_CODE}の値により、</p>															

バージョン タグ名 HTMLTEXT BODY要素	内容
17.stopError	<p>PLSQL/SQLの異常/正常終了によって分岐処理は可能となります。 初期値は、true(中止する)です。</p> <p>【TAG】リクエスト情報の HTMLTag開始/終了文字(×) 存在チェックを実施するかどうか[true/false]を設定します (初期値:USE_XSS_CHECK[=true])。</p>
18.xssCheck	<p>クロスサイトスクリプティング(XSS)対策の一環としてless/greater than signについてのチェックを行います。 (×) が含まれていたエラーにする(true)／か／チェックか(false)を指定します。 (初期値:システム定数のUSE_XSS_CHECK[=]@og.value org.opengion.hayabusa.common.SystemData#USE_XSS_CHECK。</p>
19.mainTrans	<p>【TAG】(通常使いません)タグで処理される処理がメインとなるトランザクション処理かどうかを指定します(初期値:true)。 この値は、ファイルダウンロード処理に影響します。この値がtrueに指定された時にcommitされたDBTableModelが ファイルダウンロードの対象の表になります。</p> <p>このパラメーターは、通常、各タグにより実装され、ユーザーが指定する必要はありません。 但し、1つのJSP内でDBTableModelが複数生成される場合に、前に処理したDBTableModelについてファイルダウンロードをさせたい 場合は、後ろでDBTableModelを生成するタグで、明示的にこの値をfalseに指定することで、ファイルダウンロード処理の対象から 除外することができます。</p>
20.caseKey	<p>【TAG】このタグ自体を利用するかどうかの条件キーを指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。</p>
21.caseVal	<p>【TAG】このタグ自体を利用するかどうかの条件値を指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。</p>
22.useBeforeHtmlTag	<p>【TAG】 処理時間(queryTime)などの情報出力[true:有効/false:無効]を指定します(初期値:true)。 Query で、検索する場合に、処理時間(queryTime)などの情報を出力していますが、 ViewForm で、CustomData などの 非HTML表示ビューを使用する場合、データとして、 紛れ込んでしまうため、出力を抑制する必要があります。 true(有効)にすると、これらのHTMLが出力されます。false にすると、出力されません。 初期値は、true(有効) です。</p>
23.debug	<p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
<p>5.5.4.2 og:queryButton</p> <p>JSP</p> <p>●形式 : <og:queryButton /></p> <p>●body : あり</p> <p>●Tag定義 :</p> <pre><og:queryButton command pageSize maxRowCount useCreat type scope lbl accesskey usePrevNext usePageSize useRowCount useTimeView clearAccesskey debug > ... Body ... </og:queryButton></pre> <p>●使用例</p> <pre><og:queryButton command = "NEW" type = "ENTRY" pageSize = "100" maxRowCount = "100" useCreat = "true" scope = "request" msg = "MSG0010" accesskey = "F" usePrevNext = "true" usePageSize = "true" useRowCount = "true" useTimeView = "true" /></pre> <p>標準的な検索ボタン</p> <pre><og:queryButton /></pre> <p>標準的な登録ボタン(command="NEW" で、query.jsp で使用するボタン)</p> <pre><og:queryButton type="ENTRY" /></pre> <p>BODY 部に追加要素を表示</p> <pre><og:queryButton > <td><jsp:directive.include file="/jsp/common/timeViewImage.jsp" /></td> </og:queryButton ></pre>	<p>【TAG】 リクエストとして送信するコマンドを指定します (初期値:NEW)。 NEW: 通常の検索 (初期値) / ENTRY: エントリ登録時</p> <p>【TAG】 リクエストとして送信するコマンドを指定します (初期値:NEW)</p> <p>【TAG】 pageSize ブルダウンメニューの初期値を指定します (初期値:HTML_PAGESIZE[=100])</p> <p>【TAG】 maxRowCount ブルダウンメニューの初期値を指定します (初期値:DB_MAX_ROW_COUNT[=1000])</p> <p>【TAG】 JavaScriptによる初期化ボタンを使用するかどうか[true/false]を指定します (初期値:true)</p> <p>【TAG】 典型的なボタンの形状に設定します (初期値:null)</p> <p>【TAG】 リクエストとして送信するスコープを指定します (初期値:null)</p> <p>【TAG】 ラベルリソースのラベルIDを指定します</p> <p>【TAG】 検索ボタンに使用するショートカットキー文字を指定します (初期値:F)</p> <p>【TAG】 進む／戻るのprev_next リンクを表示するかどうかを指定します (初期値:true)</p> <p>【TAG】 pageSize 引数を使用するかどうかを指定します (初期値:true)</p> <p>【TAG】 maxRowCount 引数を使用するかどうかを指定します (初期値:true)</p> <p>【TAG】 処理時間を表示する TimeView を表示するかどうかを指定します (初期値:true)</p> <p>【TAG】 クリアボタンに使用するショートカットキー文字を指定します (初期値:C)</p> <p>【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します (初期値:false)</p> <p>NEW: 通常の検索 (初期値) / ENTRY: エントリ登録時</p> <p>null: 標準 , SELECT: 検索ボタンのみ , ENTRY: 登録ボタンのみ</p> <p>pageSize の初期値設定 (初期値:HTML_PAGESIZE)</p> <p>maxRowCount の初期値設定 (初期値:DB_MAX_ROW_COUNT)</p> <p>true: JavaScriptによる初期化 / false: 通常のresetボタン</p> <p>null (初期値) / session: セッション登録 / request: リクエスト登録</p> <p>検索ボタンに表示するメッセージID (初期値:MSG0010)</p> <p>ショートカットキー文字 (初期値:F)</p> <p>Prev-Nextリンクを使用するかどうか (初期値:true)</p> <p>pageSize 引数を使用するかどうか (初期値:true)</p> <p>maxRowCount 引数を使用するかどうか (初期値:true)</p> <p>timeView表示 (実行時間表示)を使用するかどうか (初期値:true)</p>
1.command	<p>【TAG】 リクエストとして送信するコマンドを指定します (初期値:NEW)。 NEW: 通常の検索 (初期値) / ENTRY: エントリ登録時</p>
2.pageSize	<p>【TAG】 pageSize ブルダウンメニューの初期値を指定します (初期値:HTML_PAGESIZE[=100])。 ブルダウンメニューを表示する場合の初期値を指定します。 (初期値:ユーザー定数のDB_MAX_ROW_COUNT[=(@og.value org.opengion.hayabusa.common.SystemData#HTML_PAGESIZE)])。</p>
3.maxRowCount	<p>【TAG】 maxRowCount ブルダウンメニューの初期値を指定します (初期値:DB_MAX_ROW_COUNT[=1000])。 ブルダウンメニューを表示する場合の初期値を指定します。 (初期値:ユーザー定数のDB_MAX_ROW_COUNT[=(@og.value org.opengion.hayabusa.common.SystemData#DB_MAX_ROW_COUNT)])。</p>
4.useCreat	<p>【TAG】 JavaScriptによる初期化ボタンを使用するかどうか[true/false]を指定します (初期値:true)。 通常の初期化ボタンでは、HTMLが作成された状態に戻すだけです。例えば、 引数付きで設定された場合、その引数が設定されている状態へ戻します。 このJavaScriptによる初期化ボタンの場合(trueの場合)は、JavaScriptにより INIT状態でもう一度QUERYフレームを呼び出しますので、指定された引数も クリアされます。 初期値は、true: JavaScriptによる初期化です。</p>
5.type	<p>【TAG】 典型的なボタンの形状に設定します (初期値:null)。 クエリーボタンでは、検索ボタン以外に、pageSize、maxRowCount、 prevNext、timeView などの項目も表示可能です。 これらを、個別に設定可能ですが、通常使用するタイプとして、3タイプを 用意します。これは、設定状態の初期値といえます。 null : 通常の検索ボタンで、フル装備の指定です。 SELECT: 1 件のみ検索するようなケースで使います。 検索ボタンとクリアボタンのみを表示します。 ENTRY : いきなり登録するようなケースで使います。 登録ボタンとクリアボタンのみを表示します。(command="NEW" です。)</p>
6.scope	<p>【TAG】 リクエストとして送信するスコープを指定します (初期値:null)。 リクエスト時に、付与する引数情報のうちの、スコープを指定します。 これは、検索専用画面の場合は、scope="request" にする事で、メモリの 無駄な確保を行わずに、検索結果を表示できます。 ただし、その結果に対して、NEXT-PREV や、ヘッダーソートリンクなどの 機能が使えなくなります。 通常は、何も指定しない状態(null)で、RESULT画面の設定に任せるのが良いでしょう。 初期値は、null = session(セッション登録) です。</p>
7.lbl	<p>【TAG】 ラベルリソースのラベルIDを指定します。 ラベルを変更するときに、lbl属性を使います。 ラベルID は、所定の language に基づく ResourceManager の getLabelData(id) を呼び出し、その結果のLabelInterfaceを使います。 getMsgLbl() で取り出せます。 ラベルIDとメッセージIDは同時には登録できません。 ラベルとメッセージは統一されました。</p>
8.accesskey	<p>【TAG】 検索ボタンに使用するショートカットキー文字を指定します (初期値:F)。 ボタンを、Alt-XX で実行できるようにするショートカットキーを設定します。 標準では、検索ボタンは、"F" に設定していますので、Alt-F となります。 初期値は、Find の "F" です。</p>
	<p>【TAG】 進む／戻るのprev_next リンクを表示するかどうかを指定します (初期値:true)。 進む／戻るリンクは、RESULT画面にコマンド(FIRST, PREV, NEXT, LAST) という</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
9.usePrevNext	コマンドを送る事で実現している画像リンクです。 PREV_NEXT が存在しない画面では、消去しておく必要があります。 初期値は、true(表示する)です。
10.usePageSize	【TAG】 pageSize 引数を使用するかどうかを指定します(初期値:true)。 pageSize 引数は、RESULT画面のviewタグで使用する。一回の表示に 何件のデータを表示させるかを指定する引数です。 この件数が多いと、多くのデータを見ることが出来ますが、その分表示が 遅くなります。逆に、少ないと、PREV_NEXT を多用することになり、 目的の行を探し出しにくくなります。 pageSize の値は、pageSizeパラメータで画面単位に指定するか、システム パラメータの HTML_PAGESIZE でシステム単位に指定できます。 初期値は、true(使用する)です。
11.useRowCount	【TAG】 maxRowCount 引数を使用するかどうかを指定します(初期値:true)。 maxRowCount 引数は、RESULT画面のqueryタグで使用する。一回の検索時に 何件のデータを取得させるかを指定する引数です。 この件数が多いと、多くのデータを検索することが出来ますが、その分表示が 遅くなります。逆に、少ないと、目的の行まで探し出せないかもしれません。 maxRowCount の値は、rowCountパラメータで画面単位に指定するか、システム パラメータの DB_MAX_ROW_COUNT でシステム単位に指定できます。 初期値は、true(使用する)です。
12.useTimeView	【TAG】 処理時間を表示する TimeView を表示するかどうかを指定します(初期値:true)。 true に設定すると、処理時間を表示するバーイメーが表示されます。 これは、DB検索、APサーバー処理、画面表示の各処理時間をバーイメーで 表示させる機能です。処理時間の目安になります。 初期値は、true です。
13.clearAccesskey	【TAG】 クリアボタンに使用するショートカットキー文字を指定します(初期値:C)。 ボタンを、Alt-XX で実行できるようにするショートカットキーを設定します。 標準では、検索ボタンは、“C” に設定していますので、Alt-C となります。 初期値は、Clear の “C” です。
14.debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:queryOption	【TAG】Optionの初期値で選ばれる値を指定します。 Optionの初期値で選ばれる値を指定します。
JSP ●形式： <div> <div><og:queryOption ></div> <div>SELECT文</div> <div></og:queryOption ></div> </div> ●body：あり ●Tag定義： <div> <div><og:queryOption</div> <div>value</div> <div>separator</div> <div>defaultVal</div> <div>language</div> <div>quotCheck</div> <div>dbid</div> <div>addKey</div> <div>classUseNo</div> <div>groupUseNo</div> <div>titleUseNo</div> <div>xssCheck</div> <div>caseKey</div> <div>caseVal</div> <div>debug</div> <div>></div> <div>... Body ...</div> <div></og:queryOption></div> </div> ●使用例 <div> <div><og:select name="CDC" ></div> <div><og:queryOption></div> <div>select NOSYN,NOSYN,NMSYN from DB01 ORDER BY 1</div> <div></og:queryOption></div> <div></og:select></div> </div> <div> <div><og:select name="CDC" ></div> <div><og:option msg="" /></div> <div><og:queryOption value="61200" separator=":" ></div> <div>select CDBK,CDBK,NMBK from DB02 ORDER BY 1</div> <div></og:queryOption></div> <div></og:select></div> </div>	【TAG】Optionの初期値で選ばれる値を指定します 【TAG】複数のラベルを合成するときに使用する項目区切り文字をセットします(初期値:スペース) 【TAG】value値がNULLの場合に使用される初期値を設定します 【TAG】タグ内部で使用する言語コード[ja/en/zh/...]を指定します 【TAG】リクエスト情報の クォーティション(') 存在チェックを実施するかどうか[true/false]を設定します (初期値:USE_SQL_INJECTION_CHECK[=true]) 【TAG】(通常は使いません)Queryオブジェクトを作成する時のDB接続IDを指定します 【TAG】項目が一つだけの場合の` `リソースに、キー情報を追加するかどうかを指定します(初期値:false) 【TAG】オプションに追加する class 属性の カラム番号を指定します 【TAG】オプションのグループ化を行うカラム番号を指定します 【TAG】オプションに追加する title 属性の カラム番号を指定します 【TAG】リクエスト情報の HTMLTag開始/終了文字(<>) 存在チェックを実施するかどうか[true/false]を設定します (初期値:USE_XSS_CHECK[=true]) 【TAG】このタグ自体を利用するかどうかの条件キーを指定します(初期値:null) 【TAG】このタグ自体を利用するかどうかの条件値を指定します(初期値:null) 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)
1.value	【TAG】Optionの初期値で選ばれる値を指定します。 Optionの初期値で選ばれる値を指定します。
2.separator	【TAG】複数のラベルを合成するときに使用する項目区切り文字をセットします(初期値:スペース)。 初期値は、スペースです。
3.defaultVal	【TAG】value値がNULLの場合に使用される初期値を設定します。 value値がNULLの場合に、この初期値をセットします。
4.language	【TAG】タグ内部で使用する言語コード[ja/en/zh/...]を指定します。 この言語コードに基づいて、表示のラベルをリソースから作成します。
5.quotCheck	【TAG】リクエスト情報の クォーティション(') 存在チェックを実施するかどうか[true/false]を設定します (初期値:USE_SQL_INJECTION_CHECK[=true])。 SQLインジェクション対策の一つとして、暫定的ではありますが、SQLのパラメータに渡す文字列にクォーティション(') を許さない設定にすれば、ある程度は防止できます。 数字タイプの引数には、 or 5=5 などのクォーティションを使用しないコードを埋めても、数字チェックで検出可能です。文字タイプの場合は、必ず (')をはずして、 ' or 'A' like 'A' のような形式になる為、(')チェックだけでも有効です。 (') が含まれていたエラーにする(true)／かノーチェックか(false)を指定します。 (初期値:システム定数のUSE_SQL_INJECTION_CHECK[=@og.value org.opengion.hayabusa.common.SystemData#USE_SQL_INJECTION_CHECK])。
6.dbid	【TAG】(通常は使いません)Queryオブジェクトを作成する時のDB接続IDを指定します。 Queryオブジェクトを作成する時のDB接続IDを指定します。 これは、システムリソースで、DEFAULT_DB_URL 等で指定している データベース接続先情報に、XX_DB_URL を定義することで、 dbid="XX" とすると、この 接続先を使用してデータベースにアクセスできます。
7.addKey	【TAG】項目が一つだけの場合の` `リソースに、キー情報を追加するかどうかを指定します(初期値:false)。 Queryオブジェクトの項目が一つの場合、` `部には、` `リソースを使用します。 この時、` `無しの場合は、キーが表示されますが、` `ありの場合は、キーは表示されず` `のみ表示されます。 都合によっては、キーも表示したい場合がありますので、その様なケースでは、addKey = "true" を設定する事で、キー:` `のセットを` `として扱います。 初期値はfalse(キーは付加しない)です。
8.classUseNo	【TAG】オプションに追加する class 属性の カラム番号を指定します。 オプションは、データベースを検索して作成されますが、そのSQL文のカラム情報を 使用して オプションに class 属性を追加します。 各オプションに色をつける場合は、この class 属性に対応する CSS ファイルを用意します。 ここでは、class 属性に使用する SQL文の カラム番号(先頭が 0) を指定します。 通常、カラム番号=0 は キー情報、=1 はラベル情報 です。2 か 3 を指定します。 初期値は、使用しない(-1)です。
9.groupUseNo	【TAG】オプションのグループ化を行うカラム番号を指定します。 オプションは、データベースを検索して作成されますが、そのSQL文のカラム情報を 使用して オプションをグループ化します。グループ化は optgroup要素をブレイク時に出力する事で対応します。 ここでは、グループ化に使用する SQL文の カラム番号(先頭が 0) を指定します。 通常、カラム番号=0 は キー情報、=1 はラベル情報 です。2 か 3 を指定します。 初期値は、使用しない(-1)です。
10.titleUseNo	【TAG】オプションに追加する title 属性の カラム番号を指定します。 オプションは、データベースを検索して作成されますが、そのSQL文のカラム情報を 使用して オプションに title 属性を追加します。 title属性はマウスオーバー時にツールチップとして表示されるため、プルダウンの横幅を短くしたい場合に有効です。 通常、カラム番号=0 は キー情報、=1 はラベル情報 です。2 か 3 を指定します。 初期値は、使用しない(-1)です。
11.xssCheck	【TAG】リクエスト情報の HTMLTag開始/終了文字(<>) 存在チェックを実施するかどうか[true/false]を設定します (初期値:USE_XSS_CHECK[=true])。 クロスサイトスクリプティング(XSS)対策の一環としてless/greater than signについてのチェックを行います。 (<>) が含まれていたエラーにする(true)／かノーチェックか(false)を指定します。 (初期値:システム定数のUSE_XSS_CHECK[=@og.value org.opengion.hayabusa.common.SystemData#USE_XSS_CHECK])。
12.caseKey	【TAG】このタグ自体を利用するかどうかの条件キーを指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
13.caseVal	【TAG】 このタグ自体を利用するかどうかの条件値を指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。
14.debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:readTable Tag	<p>【TAG】実際に読み出すクラス名の略称(TableReader_**** の ****)をセットします(初期値:Default)。 実際に読み出すクラス名(の略称)をセットします。 これは、org.opengion.hayabusa.io 以下の TableReader_**** クラスの **** を与えます。これらは、TableReader インターフェースを継承したサブクラスです。 属性クラス定義の [link org.opengion.hayabusa.io.TableReader TableReader] を参照願います。</p>
●形式： <og:readTable command fileURL filename encode maxRowCount /> ●body：なし ●Tag定義： <og:readTable readerClass fileURL filename encode maxRowCount separator tableId command modifyType displayMsg notFoundMsg sheetName columns useNumber adjustColumns checkColumns nullCheck language stopZero scope mainTrans skipRowCount useRenderer debug /> ●使用例 <og:readTable command readerClass modifyType fileURL filename encode maxRowCount columns useNumber adjustColumns checkColumns nullCheck stopZero skipRowCount />	<p>【TAG】実際に読み出すクラス名の略称(TableReader_**** の ****)をセットします(初期値:Default)。 【TAG】読み取り元ディレクトリ名を指定します(初期値:FILE_URL) 【TAG】ファイルを作成するときのファイル名をセットします(初期値:FILE_FILENAME[=file.xls]) 【TAG】ファイルを作成するときのファイルエンコーディング名をセットします(初期値:FILE_ENCODE) 【TAG】読取時の最大取り込み件数をセットします(初期値:DB_MAX_ROW_COUNT[=1000]) 【TAG】可変長ファイルを作成するときの項目区切り文字をセットします 【TAG】(通常使いません)sessionから所得する DBTableModelオブジェクトの ID 【TAG】コマンド(NEW, RENEW)をセットします(初期値:NEW) 【TAG】ファイル取り込み時の モディファイタイプ(A(追加),C(更新),D(削除))を指定します 【TAG】query の結果を画面上に表示するメッセージIDを指定します(初期値:MSG0033[件検索しました]) 【TAG】検索結果がゼロ件の場合に表示するメッセージリソースIDを指定します(初期値:MSG0077[対象データはありませんでした]) 【TAG】EXCELファイルを読み込むときのシート名を設定します(初期値:第1シート) 【TAG】読み取り元ファイルのカラム列を、外部(タグ)より指定します 【TAG】行番号情報を、使用している/していない[true/false]を指定します(初期値:true) 【TAG】読み取り元ファイルのデータ変換を行うカラム列をカンマ指定します 【TAG】読み取り元ファイルの整合性チェックを行うカラム列をカンマ指定します 【TAG】NULL チェックすべきカラム列をカンマ区切り(CVS形式)で指定します 【TAG】タグ内部で使用する言語コード[ja/en/zh/...]を指定します 【TAG】読込件数が0件のとき処理を続行するかどうか[true/false]を指定します(初期値:false[続行する]) 【TAG】キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session) 【TAG】(通常使いません)タグで処理される処理がメインとなるトランザクション処理かどうかを指定します(初期値:false) 【TAG】(通常は使いません)データの読み飛ばし件数を設定します 【TAG】読取処理でラベルをコードリソースに逆変換を行うかどうかを指定します(初期値:USE_TABLE_READER_RENDERER[=false]) 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)</p> <p> = "NEW" = "[@USER_ID]" 読み取り元ディレクトリ名 = "[@filename]" 読み取り元ファイル名 = "UnicodeLittle" 読み取り元ファイルエンコード名 = "10000" 読取最大件数 = "Fixed" 固定長データの読み取り = "[@modifyType]" 読取時のモディファイタイプ(A,C等) = "[@USER_ID]" 読み取り元ディレクトリ名 = "[@filename]" 読み取り元ファイル名 = "Shift_JIS" 読み取り元ファイルエンコード名 = "10000" 読取最大件数 = "OYA,KO,HJO,SU,DYSTR,DYEND" #NAME に対応するカラム列 = "false" 行番号の存在しないデータを読み取ります。 = "OYA,KO,HJO,SU" データ変換するカラム列("*"で全カラム) = "OYA,KO,HJO,SU" 整合性チェックするカラム列("*"で全カラム) = "OYA,KO,SU" NULLチェックを実行します("*"で全カラム) = "true" 取得0件の場合に以降の処理を停止します = "4" データの読み飛ばし件数(読み込み開始は、この数字+1行目から)</p>
1.readerClass	<p>【TAG】実際に読み出すクラス名の略称(TableReader_**** の ****)をセットします(初期値:Default)。 実際に読み出すクラス名(の略称)をセットします。 これは、org.opengion.hayabusa.io 以下の TableReader_**** クラスの **** を与えます。これらは、TableReader インターフェースを継承したサブクラスです。 属性クラス定義の [link org.opengion.hayabusa.io.TableReader TableReader] を参照願います。</p>
2.fileURL	<p>【TAG】読み取り元ディレクトリ名を指定します (初期値:FILE_URL[=filetemp/])。 この属性で指定されるディレクトリより、ファイルを読み取ります。 指定方法は、通常の fileURL 属性と同様に、先頭が、"/" (UNIX) または、2文字目が、 "." (Windows) の場合は、指定のURLそのままのディレクトリに、そうでない場合は、 fileURL = "[@USER_ID]" と指定すると、FILE_URL 属性で指定のフォルダの下に、 さらに、各個人ID別のフォルダを作成して、そこを操作します。 (初期値:システム定数のFILE_URL[=[@og.value org.opengion.hayabusa.common.SystemData#FILE_URL]])。 【TAG】ファイルを作成するときのファイル名をセットします (初期値:FILE_FILENAME[=file.xls])。</p>
3.filename	<p>ファイルを作成するときのファイル名をセットします。 (初期値:システム定数のFILE_FILENAME[=[@og.value org.opengion.hayabusa.common.SystemData#FILE_FILENAME]])。 【TAG】ファイルを作成するときのファイルエンコーディング名をセットします (初期値:FILE_ENCODE[=UnicodeLittle])。</p>
4.encode	<p>Shift_JIS, MS932, Windows-31J, UTF-8, ISO-8859-1, UnicodeLittle (初期値:システム定数のFILE_ENCODE[=[@og.value org.opengion.hayabusa.common.SystemData#FILE_ENCODE]])。 【TAG】読取時の最大取り込み件数をセットします (初期値:DB_MAX_ROW_COUNT[=1000])。</p>
5.maxRowCount	<p>DBTableModelのデータとして登録する最大件数をこの値に設定します。 サーバーのメモリ資源と応答時間の確保の為です。 0 をセットすると、無制限(Integer.MAX_VALUE)になります。 (初期値:ユーザー定数のDB_MAX_ROW_COUNT[=[@og.value org.opengion.hayabusa.common.SystemData#DB_MAX_ROW_COUNT]])。 【TAG】可変長ファイルを作成するときの項目区切り文字をセットします。 可変長ファイルを作成するときの項目区切り文字をセットします。</p>
6.separator	<p>【TAG】(通常使いません)sessionから所得する DBTableModelオブジェクトの ID。 表示処理後に、(内部ポインタを書き換えた)DBTableModelオブジェクトを 同じキーで、sessionに登録します。 初期値は、HybsSystem.TBL_MDL_KEY です。</p>
8.command	<p>【TAG】コマンド(NEW, RENEW)をセットします(初期値:NEW)。 コマンドは、HTMLから(get/post)指定されますので、CMD_xxx で設定される フィールド定数値のいずれかを、指定できます。 何も設定されない、または、null の場合は、"NEW" が初期値にセットされます。 【TAG】ファイル取り込み時の モディファイタイプ(A(追加),C(更新),D(削除))を指定します。</p>
9.modifyType	<p>ファイル読み込み時に、そのデータをA(追加)、C(更新)、D(削除)の モディファイタイプをつけた状態にします。 その状態で、そのまま、update する事が可能になります。</p>

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

10.displayMsg

【TAG】 query の結果を画面上に表示するメッセージIDを指定します(初期値:MSG0033[件検索しました])。
ここでは、検索結果の件数や登録された件数をまず出力し、
その次に、ここで指定したメッセージをリソースから取得して
表示します。
表示させたくない場合は、displayMsg = "" をセットしてください。
初期値は、検索件数を表示します。
【TAG】 検索結果がゼロ件の場合に表示するメッセージリソースIDを指定します(初期値:MSG0077[対象データはありませんでした])。
ここでは、検索結果がゼロ件の場合のみ、特別なメッセージを表示させます。
従来は、displayMsg と兼用で、『0 件検索しました』という表示でしたが、
displayMsg の初期表示は、OFF になりましたので、ゼロ件の場合のみ別に表示させます。
表示させたくない場合は、notfoundMsg = "" をセットしてください。
初期値は、MSG0077[対象データはありませんでした]です。
【TAG】 EXCELファイルを読み込むときのシート名を設定します(初期値:第1シート)。
EXCELファイルを読み込む時に、シート名を指定します。これにより、複数の形式の
異なるデータを順次読み込むことや、シートを指定して読み取ることが可能になります。
初期値は、第一シートです。
【TAG】 読み取り元ファイルのカラム列を、外部(タグ)より指定します。
読み取り元ファイルのカラム列を、外部(タグ)より指定します。
ファイルに記述された #NAME より優先して使用されます。
これは、元ファイルのカラムを順番に指定のカラム名に割り当てる機能で
ファイルの特定のカラム列を抜き出して取り込む機能ではありません。
【TAG】 行番号情報を、使用している/していない[true/false]を指定します(初期値:true)。
通常のフォーマットでは、各行の先頭に行番号が出力されています。
読み取り時に、#NAME 属性を使用する場合は、この行番号を無視しています。
#NAME 属性を使用せず、columns 属性でカラム名を指定する場合(他システムの
出力ファイルを読み取るケース等)では、行番号も存在しないケースがあり、
その様な場合に、useNumber="false" を指定すれば、データの最初から読み取り始めます。
この場合、出力データのカラムの並び順が変更された場合、columns 属性も
指定しなおす必要がありますので、できるだけ、#NAME 属性を使用するように
してください。
なお、EXCEL 入力には、この設定は適用されません。(暫定対応)
初期値は、true(使用する) です。
【TAG】 読み取り元ファイルのデータ変換を行うカラム列をカンマ指定します。
カラムオブジェクトのDBType属性に対応したデータ変換を行います。
指定のカラム名をカンマ区切り(CSV)で複数指定できます。
分解方法は、通常のパラメータ取得後に、CSV分解します。
【TAG】 読み取り元ファイルの整合性チェックを行うカラム列をカンマ指定します。
カラムオブジェクトのDBType属性に対応したチェックを行います。
指定のカラム名をカンマ区切り(CSV)で複数指定できます。
全てのカラムのチェックを行う場合は、allColumnCheck = "true" を
指定して下さい。
分解方法は、通常のパラメータ取得後に、CSV分解します。
【TAG】 NULL チェックすべきカラム列をカンマ区切り(CVS形式)で指定します。
nullCheck="AAA, BBB, CCC, DDD"
~~先に配列に分解してからリクエスト変数の値を取得するようにします。~~
~~こうする事で、リクエストにカンマ区切りの値を設定できるようになります。~~
分解方法は、通常のパラメータ取得後に、CSV分解します。
【TAG】 タグ内部で使用する言語コード[ja/en/zh/...]を指定します。
この言語コードに基づいて、表示のラベルをリソースから作成します。
【TAG】 読込件数が0件のとき処理を続行するかどうか[true/false]を指定します(初期値:false[続行する])。
初期値は、false(続行する)です。
【TAG】 キャッシュする場合のスコープ[request/page/session/application]を指定します(初期値:session)。
"request","page","session","application" が指定できます。
JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、
主にテーブルモデルを管理するスコープを指します。

スコープ	変数の有効範囲	
page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会
request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ
session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。 内部的には、同じキーワー
application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。

【TAG】 (通常使いません) タグで処理される処理がメインとなるトランザクション処理かどうかを指定します(初期値:false)。
この値は、ファイルダウンロード処理に影響します。この値がtrueに指定された時にcommitされたDBObjectModelが
ファイルダウンロードの対象の表になります。

このパラメータは、通常、各タグにより実装され、ユーザーが指定する必要はありません。
但し、1つのJSP内でDBObjectModelが複数生成される場合に、前に処理したDBObjectModelについてファイルダウンロードをさせたい
場合は、後ろでDBObjectModelを生成するタグで、明示的にこの値をfalseに指定することで、ファイルダウンロード処理の対象から
除外することが出来ます。
【TAG】 (通常は使いません) データの読み飛ばし件数を設定します。
TAB区切りテキストやEXCEL等のデータの読み始めの初期値を指定します。
ファイルの先頭行が、0行としてカウントしますので、設定値は、読み飛ばす
件数になります。(1と指定すると、1件読み飛ばし、2行目から読み込みます。)
読み飛ばしは、コメント行などは、無視しますので、実際の行数分読み飛ばします。
#NAME属性や、columns 属性は、有効です。
【TAG】 読取処理でラベルをコードリソースに逆変換を行うかどうかを指定します
(初期値:USE_TABLE_READER_RENDERER[=false])。
TableWriter_Renderer 系のクラスで出力した場合は、コードリソースがラベルで出力されます。
そのファイルを読み取ると、当然、エラーになります。
ここでは、コードリソースのカラムに対して、ラベルからコードを求める逆変換を行うことで、
Renderer 系で出力したファイルを取り込むことができるようになります。

ここでは、TableWriter 系と同様に、TableReader_Renderer 系のクラスを作るのではなく、
属性値のフラグで、制御します。
将来的には、TableWriter 系も廃止して、同様のフラグで制御するように変更する予定です。
(初期値:システム定数のUSE_TABLE_READER_RENDERER[=@og.value.org.opengion.hayabusa.common.SystemData#USE_TABLE_READER_RENDERER])。
【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。
デバッグ情報を [true:出力する/false:しない] を指定します。
出力形式自体は、個々のタグによって異なります。

21.mainTrans

22.skipRowCount

23.useRenderer

24.debug

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

5.5.4.2

og:regQuery

Tag

●形式: <og:regQuery baseKey="..." ... />

●body: なし

●Tag定義:

<og:regQuery
baseKey
key
value
maxRowCount
orderBy
maxLevel
like
tableId
command
displayMsg
notfoundMsg
overflowMsg
scope
mainTrans
debug
>

●使用例

<og:regQuery debug="false"
command = "NEW"
baseKey = "/microsoft/windows nt"
key = "driver"
value = "Ne"
maxRowCount = "0"
maxLevel = "0"
like = "true"
>

like は、key / value を設定したときのみ有効です。また、key / value を両方同時に設定した場合は、like 属性は両方に同時に適用されます。

1.baseKey

2.key

3.value

4.maxRowCount

5.orderBy

6.maxLevel

7.like

8.tableId

9.command

10.displayMsg

11.notfoundMsg

12.overflowMsg

13.scope

14.mainTrans

15.debug

【TAG】 検索ベースキーを設定します (HKEY_CURRENT_USER\Software 以下をベースとします)。
キーは、HKEY_CURRENT_USER\Software 以下をベースとします。

○【TAG】 検索ベースキーを設定します (HKEY_CURRENT_USER\Software 以下をベースとします)。(必須)
【TAG】 検索キーを設定します
【TAG】 検索バリューを設定します
【TAG】 レジストリの最大検索件数をセットします (初期値:0[無制限])
【TAG】 検索した結果を表示する表示順をファイル属性名で指定します
【TAG】 検索時の最大展開レベル (0は無制限) を指定します (初期値:1)
【TAG】 キーおよびバリューについて、like 検索を行うかどうか[true/false]を指定します (初期値:false)
【TAG】 (通常は使いません) 結果をDBTableModelに書き込んで、sessionに登録するときのキーを指定します
【TAG】 コマンド (NEW, RENEW) をセットします (初期値:NEW)
【TAG】 検索結果を画面に表示するメッセージIDを指定します (初期値:MSG0033[件検索しました])
【TAG】 検索結果がゼロ件の場合に表示するメッセージリソースIDを指定します (初期値:MSG0077[対象データはありませんでした])
【TAG】 オーバーフロー時に画面に表示するメッセージIDを指定します (初期値:MSG0007)
【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します (初期値:session)
【TAG】 (通常使いません) タグで処理される処理がメインとなるトランザクション処理かどうかを指定します (初期値:false)
【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します (初期値:false)

NEW, RENEW が使用できます。
検索時のベースとなるレジストリキー名
検索したいレジストリキー名 (初期値は全件)
検索したいレジストリ値 (初期値は全件)
最大検索件数 (0で無制限)
最大検索階層レベル (0で無制限)
true で曖昧検索/false は一致検索

【TAG】 検索ベースキーを設定します (HKEY_CURRENT_USER\Software 以下をベースとします)。
キーは、HKEY_CURRENT_USER\Software 以下をベースとします。

【TAG】 検索キーを設定します。
検索キーを設定します。

【TAG】 検索バリューを設定します。
検索バリューを設定します。

【TAG】 レジストリの最大検索件数をセットします (初期値:0[無制限])。
DBTableModelのデータとして登録する最大件数をこの値に設定します。
サーバーのメモリ資源と応答時間の確保の為です。
初期値は、0 は、無制限です。

【TAG】 検索した結果を表示する表示順をファイル属性名で指定します。
現状様では、複数のキーを指定することは出来ません。

【TAG】 検索時の最大展開レベル (0は無制限) を指定します (初期値:1)。
0 を指定すると、無制限に階層を展開します。
初期値は、1 レベルです。

【TAG】 キーおよびバリューについて、like 検索を行うかどうか[true/false]を指定します (初期値:false)。
like検索とは、キーの一部の文字を含む場合にマッチしたとして、値を取り出します。
ここでは設定は、キーもバリューも同時に適用されます。また、大文字小文字の区別も行いません。

【TAG】 (通常は使いません) 結果をDBTableModelに書き込んで、sessionに登録するときのキーを指定します。
初期値は、HybsSystem.TBL_MD_L_KEY です。

【TAG】 コマンド (NEW, RENEW) をセットします (初期値:NEW)。
コマンドは、HTMLから (get/post) 指定されますので、CMD_xxx で設定される
フィールド定数値のいつれかを、指定できます。
何も設定されない、または、null の場合は、“NEW” が初期値にセットされます。

【TAG】 検索結果を画面に表示するメッセージIDを指定します (初期値:MSG0033[件検索しました])。
ここでは、検索結果の件数や登録された件数をまず出力し、
その次に、ここで指定したメッセージをリソースから取得して
表示します。
表示させたくない場合は、displayMsg = "" をセットしてください。
初期値は、検索件数を表示します。

【TAG】 検索結果がゼロ件の場合に表示するメッセージリソースIDを指定します (初期値:MSG0077[対象データはありませんでした])。
ここでは、検索結果がゼロ件の場合のみ、特別なメッセージを表示させます。
従来は、displayMsg と兼用で、『0 件検索しました』という表示でしたが、
displayMsg の初期表示は、OFF になりましたので、ゼロ件の場合のみ別に表示させます。
表示させたくない場合は、notfoundMsg = "" をセットしてください。
初期値は、MSG0077[対象データはありませんでした]です。

【TAG】 オーバーフロー時に画面に表示するメッセージIDを指定します (初期値:MSG0007)。
表示させたくない場合は、overflowMsg = "" をセットしてください。

【TAG】 キャッシュする場合のスコープ[request/page/session/application]を指定します (初期値:session)。
“request”, “page”, “session”, “application” が指定できます。
JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、
主にテーブルモデルを管理するスコープを指します。

スコープ	変数の有効範囲	
page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会
request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ
session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。内部的には、同じキーワ
application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。

【TAG】 (通常使いません) タグで処理される処理がメインとなるトランザクション処理かどうかを指定します (初期値:false)。
この値は、ファイルダウンロード処理に影響します。この値がtrueに指定された時にcommitされたDBTableModelが
ファイルダウンロードの対象の表になります。

このパラメーターは、通常、各タグにより実装され、ユーザーが指定する必要はありません。
但し、1つのJSP内でDBTableModelが複数生成される場合に、前に処理したDBTableModelについてファイルダウンロードをさせたい
場合は、後ろでDBTableModelを生成するタグで、明示的にこの値をfalseに指定することで、ファイルダウンロード処理の対象から
除外することができます。

【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します (初期値:false)。
デバッグ情報を [true:出力する/false:しない] を指定します。
出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容	
形式サンプル		
5.5.4.2 og:report	【TAG】 帳票IDを指定します。 雛形ファイルは、帳票ID.html となります。また、ファーストページ対応の場合は、 帳票ID_FIRST.html になります。 なお、filename 属性が指定された場合は、そちらが優先されます。	
Tag	●形式 : <og:report fileURL=["..."] listId=["..."] ... /> ●body : なし	
●Tag定義 :	<og:report listId fileURL programFile outFileURL outFilename headerKeys headerVals footerKeys footerVals pageEndCut reportClass language scope tableId debug >	
●使用例	○【TAG】 帳票IDを指定します。(必須) 【TAG】 雛型のHTMLファイルの保存してある ディレクトリを指定します 【TAG】 HTMLファイルをEXCEL変換する場合に使用するBATファイルを指定します 【TAG】 出力HTMLファイルの保存してあるディレクトリを指定します ○【TAG】 ファイルを作成するときの出力ファイル名をセットします。(必須) 【TAG】 固定部の{@KEY} の KEY 部分をCSV形式で複数指定します 【TAG】 固定部のKEY に対応する値をCSV形式で複数指定します 【TAG】 繰り返し部の終了後に表示する key 部分をCSV形式で複数指定します 【TAG】 固定部のKEY に対応する値をCSV形式で複数指定します 【TAG】 ボディー部(繰り返し部)がなくなったときに、それ以降を表示するかどうか[true/false]を指定します (初期値:true) 【TAG】 実際に書き出すクラス名の略称(DBTableReport.**** の ****)をセットします (初期値:HTML) 【TAG】 タグ内部で使用する言語コード[ja/en/zh/...]を指定します 【TAG】 キャッシュする場合のスコープ[request/page/session/application]を指定します (初期値:session) 【TAG】 (通常使いません)sessionから所得する DBTableModelオブジェクトの ID 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します (初期値:false)	
1.listId	【TAG】 帳票IDを指定します。 雛形ファイルは、帳票ID.html となります。また、ファーストページ対応の場合は、 帳票ID_FIRST.html になります。 なお、filename 属性が指定された場合は、そちらが優先されます。	
2.fileURL	【TAG】 雛型のHTMLファイルの保存してある ディレクトリを指定します。 この属性で指定されるディレクトリのファイルを読み取ります。 指定方法は、通常の fileURL 属性と同様に、先頭が、'/' (UNIX) または、2文字目が、 "." (Windows) の場合は、指定のURLそのままのディレクトリに、そうでない場合は、 システムパラメータ の FILE_URL 属性で指定のフォルダの下に、作成されます。 fileURL = "[@USER.ID]" と指定すると、FILE_URL 属性で指定のフォルダの下に、 さらに、各個人ID別のフォルダを作成して、そこを操作します。	
3.programFile	【TAG】 HTMLファイルをEXCEL変換する場合に使用するBATファイルを指定します。 ファイルは、フルパスで指定してください。 指定がない場合は、変換処理は行いません。 通常は、dbdef2/def/Script/runExcelPrint.bat を呼び出してください。 初期値は、null(変換処理しない)です。	
4.outFileURL	【TAG】 出力HTMLファイルの保存してあるディレクトリを指定します。 この属性で指定されるディレクトリにファイルを出力します。 指定方法は、通常の fileURL 属性と同様に、先頭が、'/' (UNIX) または、2文字目が、 "." (Windows) の場合は、指定のURLそのままのディレクトリに、そうでない場合は、 システムパラメータ の FILE_URL 属性で指定のフォルダの下に、作成されます。 fileURL = "[@USER.ID]" と指定すると、FILE_URL 属性で指定のフォルダの下に、 さらに、各個人ID別のフォルダを作成して、そこに出力します。	
5.outFilename	【TAG】 ファイルを作成するときの出力ファイル名をセットします。 ファイルを作成するときの出力ファイル名をセットします。	
6.headerKeys	【TAG】 固定部の{@KEY} の KEY 部分をCSV形式で複数指定します。 カンマで複数指定できます。	
7.headerVals	分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスの為です。 【TAG】 固定部のKEY に対応する値をCSV形式で複数指定します。 カンマで複数指定で、リクエスト情報でも設定できます。	
8.footerKeys	分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスの為です。 【TAG】 繰り返し部の終了後に表示する key 部分をCSV形式で複数指定します。 カンマで複数指定できます。	
9.footerVals	分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスの為です。	
10.pageEndCut	【TAG】 ボディー部(繰り返し部)がなくなったときに、それ以降を表示するかどうか[true/false]を指定します (初期値:true)。 true では、それ以降を出力しません。 初期値は "true" (なくなった時点で、出力しない。)です。	
11.reportClass	【TAG】 実際に書き出すクラス名の略称(DBTableReport.**** の ****)をセットします (初期値:HTML)。 これは、org.opengion.hayabusa.report 以下の DBTableReport.**** クラスの **** を 与えます。これらは、DBTableReport インターフェースを継承したサブクラスです。 初期値は、"HTML" です。 属性クラス定義の {@link org.opengion.hayabusa.report.DBTableReport DBTableReport} を参照願います。	
12.language	【TAG】 タグ内部で使用する言語コード[ja/en/zh/...]を指定します。 この言語コードに基づいて、表示のラベルをリソースから作成します。	
13.scope	【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します (初期値:session)。 "request", "page", "session", "applicaton" が指定できます。 JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、 主にテーブルモデルを管理するスコープを指します。	
14.tableId	【TAG】 (通常使いません)sessionから所得する DBTableModelオブジェクトの ID。 表示処理後に、(内部ポインタを書き換えた)DBTableModelオブジェクトを 同じキーで、sessionに登録します。 初期値は、HybsSystem.TBL_MD_KEY です。	
15.debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します (初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。	

バージョン タグ名 HTMLTEXT BODY要素	内容															
形式サンプル																
5.5.4.2 og:report2 Tag	【TAG】 雛型のHTMLファイルの保存してある ディレクトリを指定します。 この属性で指定されるディレクトリのファイルを読み取ります。 指定方法は、通常の fileURL 属性と同様に、先頭が、'/' (UNIX) または、2 文字目が、 "." (Windows) の場合は、指定のURLそのままのディレクトリに、そうでない場合は、 システムパラメータ の FILE_URL 属性で指定のフォルダの下に、作成されます。 fileURL = "[@USER_ID]" と指定すると、FILE_URL 属性で指定のフォルダの下に、 さらに、各個人ID別のフォルダを作成して、そこを操作します。															
●形式 : <og:report fileURL="[...]" listId="[...]" ... /> ●body : なし ●Tag定義 : <og:report2 fileURL listId outFileURL outFilename headerKeys headerVals footerKeys footerVals pageEndCut useLocalResource fgrun printerName language scope tableId debug /> ●使用例	【TAG】 雛型のHTMLファイルの保存してある ディレクトリを指定します ○【TAG】 帳票IDを指定します。(必須) ○【TAG】 出力HTMLファイルの保存してあるディレクトリを指定します ○【TAG】 ファイルを作成するときの出力ファイル名をセットします。(必須) 【TAG】 固定部の{@KEY} の KEY 部分をCSV形式で複数指定します 【TAG】 固定部のKEY に対応する値をCSV形式で複数指定します 【TAG】 繰り返し部の終了後に表示する key 部分をCSV形式で複数指定します 【TAG】 固定部のKEY に対応する値をCSV形式で複数指定します 【TAG】 ボディー部(繰り返し部)がなくなったときに、それ以降を表示するかどうか[true/false]を指定します (初期値:true) 【TAG】 各システムのリソース(ローカルリソース)を使用するか[true/false]を指定します (初期値:true) 【TAG】 出力方法を指定します (初期値:P(PDF出力)) 【TAG】 プリンター名を指定します 【TAG】 タグ内部で使用する言語コード[ja/en/zh/...]を指定します 【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します (初期値:session) 【TAG】 (通常使いません)sessionから所得する DBTableModelオブジェクトの ID 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します (初期値:false)															
1.fileURL	【TAG】 雛型のHTMLファイルの保存してある ディレクトリを指定します。 この属性で指定されるディレクトリのファイルを読み取ります。 指定方法は、通常の fileURL 属性と同様に、先頭が、'/' (UNIX) または、2 文字目が、 "." (Windows) の場合は、指定のURLそのままのディレクトリに、そうでない場合は、 システムパラメータ の FILE_URL 属性で指定のフォルダの下に、作成されます。 fileURL = "[@USER_ID]" と指定すると、FILE_URL 属性で指定のフォルダの下に、 さらに、各個人ID別のフォルダを作成して、そこを操作します。															
2.listId	【TAG】 帳票IDを指定します。															
3.outFileURL	【TAG】 出力HTMLファイルの保存してあるディレクトリを指定します。 この属性で指定されるディレクトリにファイルを出力します。 指定方法は、通常の fileURL 属性と同様に、先頭が、'/' (UNIX) または、2 文字目が、 "." (Windows) の場合は、指定のURLそのままのディレクトリに、そうでない場合は、 システムパラメータ の FILE_URL 属性で指定のフォルダの下に、作成されます。 fileURL = "[@USER_ID]" と指定すると、FILE_URL 属性で指定のフォルダの下に、 さらに、各個人ID別のフォルダを作成して、そこに出力します。															
4.outFilename	【TAG】 ファイルを作成するときの出力ファイル名をセットします。 ファイルを作成するときの出力ファイル名をセットします。 紙に印字する場合などファイルに出力しない場合は不要です。															
5.headerKeys	【TAG】 固定部の{@KEY} の KEY 部分をCSV形式で複数指定します。 カンマで複数指定できます。 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスの為です。															
6.headerVals	【TAG】 固定部のKEY に対応する値をCSV形式で複数指定します。 カンマで複数指定で、リクエスト情報でも設定できます。 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスの為です。															
7.footerKeys	【TAG】 繰り返し部の終了後に表示する key 部分をCSV形式で複数指定します。 カンマで複数指定できます。 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスの為です。															
8.footerVals	【TAG】 固定部のKEY に対応する値をCSV形式で複数指定します。 カンマで複数指定で、リクエスト情報でも設定できます。 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスの為です。															
9.pageEndCut	【TAG】 ボディー部(繰り返し部)がなくなったときに、それ以降を表示するかどうか[true/false]を指定します (初期値:true)。 true では、それ以降を出力しません。 初期値は "true" (なくなった時点で、出力しない。)です。															
10.useLocalResource	【TAG】 各システムのリソース(ローカルリソース)を使用するか[true/false]を指定します (初期値:true)。 true の場合、各システムに登録されたリソース情報を使用して帳票データが変換されます。 false の場合は、帳票デーモンが起動しているシステム(通常は'GE')のリソースが適用されます。 初期値は "true" (ローカルリソースを使用する)です。															
11.fgrun	【TAG】 出力方法を指定します (初期値:P(PDF出力))。 出力方法のコードは、FGRUNのコードリソースと同じものが指定できます。 初期値は "P" (PDF出力)です。															
12.printerName	【TAG】 プリンター名を指定します。 プリンター名を指定します。このプリンター名は帳票サーバー上でのプリンタ名です。 ファイル出力等、紙に印刷しない場合は不要です。															
13.language	【TAG】 タグ内部で使用する言語コード[ja/en/zh/...]を指定します。 この言語コードに基づいて、表示のラベルをリソースから作成します。															
14.scope	【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します (初期値:session)。 "request", "page", "session", "applicaton" が指定できます。 JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、 主にテーブルモデルを管理するスコープを指します。 <table><tr><th>スコープ</th><th>変数の有効範囲</th><th></th></tr><tr><td>page</td><td>JSPページ内</td><td>そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会</td></tr><tr><td>request</td><td>HTTPリクエスト</td><td>リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ</td></tr><tr><td>session</td><td>HTTPセッション</td><td>初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。 内部的には、同じキーワー</td></tr><tr><td>application</td><td>Webアプリケーション</td><td>ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。</td></tr></table>	スコープ	変数の有効範囲		page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会	request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ	session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。 内部的には、同じキーワー	application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。
スコープ	変数の有効範囲															
page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会														
request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ														
session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。 内部的には、同じキーワー														
application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。														
15.tableId	【TAG】 (通常使いません)sessionから所得する DBTableModelオブジェクトの ID。 表示処理後に、(内部ポインタを書き換えた)DBTableModelオブジェクトを 同じキーで、sessionに登録します。 初期値は、HybsSystem.TBL_MODL_KEY です。															
16.debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します (初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。															

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

5.5.4.2
og:requestCache

【TAG】 キャッシュするサブキーを指定します(初期値:"")。
キャッシュすべき値を、キー毎に指定できるようにします。
例えば、これに、画面IDを追加しておけば、画面ごとに、自分のリクエスト情報を
キャッシュしておき、自分の画面が呼ばれたら、再度使用することができるようになります。
NEW でキャッシュ登録を行い、RENEW で、通常のキャッシュキーに値を取り出します。

●形式 : <og:requestCache cacheKey= ["..."] />
●body : なし

●Tag定義 :
<og:requestCache
 cacheKey 【TAG】 キャッシュするサブキーを指定します(初期値:"")
 action 【TAG】 アクション(SET, DELETE)をセットします
 keys 【TAG】 リンク先に渡すキーを指定します
 vals 【TAG】 keys属性に対応する値をCSV形式で複数指定します
 debug 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)
>

●使用例
<og:requestCache
 cacheKey="@[GUI.KEY]" キャッシュするサブキーを指定します。
>

1.cacheKey

【TAG】 キャッシュするサブキーを指定します(初期値:"")。
キャッシュすべき値を、キー毎に指定できるようにします。
例えば、これに、画面IDを追加しておけば、画面ごとに、自分のリクエスト情報を
キャッシュしておき、自分の画面が呼ばれたら、再度使用することができるようになります。
NEW でキャッシュ登録を行い、RENEW で、通常のキャッシュキーに値を取り出します。

2.action

【TAG】 アクション(SET, DELETE)をセットします。
アクションは、HTMLから(get/post)指定されますので、ACT_xxx で設定される
フィールド定数値のいずれかを、指定できます。
無指定の場合は、なにもしません。

	action	名称	機能
SET	登録	指定の keys のキーに vals のキャッシュをセットします。	
DELETE	削除	指定の keys のキャッシュを削除します。	

3.keys

【TAG】 リンク先に渡すキーを指定します。
戻る時に、検索時のキャッシュに指定した引数以外に指定したり、別の値に置き換えたり
する場合のキーを設定できます。カンマ区切りで複数指定できます。
vals 属性には、キーに対応する値を、設定してください。
分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。
こうしないとデータ自身にカンマを持っている場合に分解をミスる為です。

4.vals

【TAG】 keys属性に対応する値をCSV形式で複数指定します。
キーに設定した値を、カンマ区切り文字で複数して出来ます。
指定順序は、キーと同じにしておいて下さい。
分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。
こうしないとデータ自身にカンマを持っている場合に分解をミスる為です。

5.debug

【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。
デバッグ情報を [true:出力する/false:しない]を指定します。
出力形式自体は、個々のタグによって異なります。

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

5.5.4.2
og:requestValue
Tag

【TAG】{@SYS. XXXX} と同等のシステムパラメータ の値を取得します。
本来は、システムパラメータ の値を取得できますが、システム的な共有値も
取得できるように機能追加しています。
また、ユーザー個別にシステムパラメータを変更できます。この取得は、
システムパラメータとして取得します。(値はユーザー個別値を返します。)

SERVER_NAME	このTomcatが実行しているサーバー名	localhost 等
SERVER_URL	Portも含むURLアドレス	http://localhost:8823/
CONTEXT_URL	実行しているコンテキストのURLアドレス	http://localhost:8823/dbdef2/
REAL_PATH	/ ルートに対応する物理ディレクトリ	d:/webapps/dbdef2/ 等
CONTEXT_NAME	コンテキスト名(webapps 直下の仮想フォルダ名)	dbdef 等
DATE	YMDH とほぼ同じですが、'yyyy/MM/dd HH:mm:ss' の形式で取得できます。	
任意	ユーザーパラメータ (GE16) の値/システムパラメータ (GE12) の値を取得	

{@SYS. COOKIE. カラム名} で、Cookie の読み込みを行います。
上記以外に、予約語や、システムパラメータ の値の読取が可能です。

●形式：
<og:requestValue
sys = "XXXX"
user = "ID"
gui = "KEY"
mem = "XXXX"
lbl = "XXXX"
msg = "XXXX"
db = "COUNT"
session = "XXXX"
date = "XXXX"
nvar = "XXXX"
>
●body : なし
●Tag定義：
<og:requestValue
sys
user
gui
mem
lbl
db
session
date
nvar
debug
>/>

1.sys

【TAG】{@SYS. XXXX} と同等のシステムパラメータ の値を取得します。
本来は、システムパラメータ の値を取得できますが、システム的な共有値も
取得できるように機能追加しています。
また、ユーザー個別にシステムパラメータを変更できます。この取得は、
システムパラメータとして取得します。(値はユーザー個別値を返します。)

SERVER_NAME	このTomcatが実行しているサーバー名	localhost 等
SERVER_URL	Portも含むURLアドレス	http://localhost:8823/
CONTEXT_URL	実行しているコンテキストのURLアドレス	http://localhost:8823/dbdef2/
REAL_PATH	/ ルートに対応する物理ディレクトリ	d:/webapps/dbdef2/ 等
CONTEXT_NAME	コンテキスト名(webapps 直下の仮想フォルダ名)	dbdef 等
DATE	YMDH とほぼ同じですが、'yyyy/MM/dd HH:mm:ss' の形式で取得できます。	
任意	ユーザーパラメータ (GE16) の値/システムパラメータ (GE12) の値を取得	

{@SYS. COOKIE. カラム名} で、Cookie の読み込みを行います。
上記以外に、予約語や、システムパラメータ の値の読取が可能です。

2.user

【TAG】{@USER. XXXX} と同等のUserInfoオブジェクトの値を取得します。
ユーザーパラメータとは異なります。

以下の値は UserInfo オブジェクトの項目から取得します。

- ・JNAME ユーザー日本語名称
- ・ID ユーザー I D
- ・IDNO USER ID が5Byte以上の時のみ先頭1文字を除いたユーザー I D
- ・INFO ユーザー情報(ユーザーID: 日本語名称)
- ・LANG 言語
- ・ROLES ロール
- ・IPADDRESS IPアドレス
- ・LOGINTIME ログイン時刻

以下の値はあらかじめ、動的に作成されます。

- ・YMD 8byte の今日のシステム日付
- ・YMDH 1 4byte の今日のシステム日時

それ以外は、外部より設定された値です。

3.gui

【TAG】{@GUI. XXXX} と同等のGUIInfoオブジェクトの値を取得します。
GUIInfo オブジェクトで定義されている属性情報を取り出すことが出来ます。
引数は、"XXXX ID" という状態で受け取ります。(GUI. ははずした形)
XXXXには、画面情報(GUIInfo)の属性キーを指定します。IDが指定されない場合は、
実行中の自分自身の画面が指定されたことになります。
これは、session 登録項目 の HybsSystem#GUIINFO_KEY キー の値で取得します。
この値は、画面が呼び出される毎に毎回設定されており、リクエスト毎に
所得し直す必要があります。

ID に、画面IDを指定した場合は、@ 指定によるリクエスト引数の値を適用できます。
{@GUI. @XXXX ID} や、{@GUI. XXXX @ID} です。(同時指定も可能)

- ・KEY 画面ID
- ・ADDRESS 実行アドレス
- ・REALADDRESS 実行実アドレス
- ・SEQUENCE 表示順
- ・GROUPS メニューグループ
- ・CLASSIFY メニュー分類
- ・LEVEL メニュー階層番号
- ・LABEL 画面名称
- ・NAME 画面名称(=SNAME)

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
	<ul style="list-style-type: none">・SNAME 画面名称(short)・LNAME 画面名称(long)・ROLES ロールズ・MODE アクセスモード列(mr,mw,-r,-wの羅列)・TARGET ターゲット・PARAM 設定値(パラメータ)・KBLINK リンク区分・DESCRIPTION 概要説明
4.mem	【TAG】{@MEM. XXXX} と同等のREQUEST_CACHEの値を取得します。 検索時の request 情報をキャッシュしているメモリ上から値を取り出します。 通常は、query_init.jsp で定義されている GAMENID 付のリクエストキャッシュ情報を取り出します。
5.lbl	【TAG】{@LBL. XXXX} と同等のラベルリソースの値を取得します。 ラベルリソースのロケールは、ログインユーザーのロケールになります。 これは、{@LBL. XXXX Y} 引数処理をおこなうための、サポートメソッドです。 引数は、“XXXX Y” という状態で受け取ります。(LBL. ははずした形) ラベルには、通常のラベル(Label) 以外に、Short, Tips, Comment の情報を持っています。 {@LBL. XXXX Y} の Y に、先のLabel, Short, Tips, Comment の頭文字(‘L’, ‘S’, ‘T’, ‘C’) を指定することで、それぞれの状態を取得することが可能になります。 Y を指定しない場合({@LBL. XXXX}) は、‘L’ が指定された場合と同じ効果があります。 Y は、先頭文字 1 文字で判断していますので、{@LBL. XXXX Short} と記述できます。 Y 引数には、@で始まるリクエスト引数が指定できます。例えば、{@LBL. XXXX @ZZ} とすると、ZZ のリクエスト引数の値が Y に適用されます。 ラベルキーそのものをパラメータ化することが出来ます。 これは、{@LBL. @XXXX} という形式になります。引数は、先の説明と同じです。 この場合は、XXXX をキーにリクエスト引数の値が、ラベルリソースのキーになります。
6.db	【TAG】データベース実行結果を取得します。 検索結果の件数を、“DB. COUNT” キーでリクエストにセットしています。 検索結果を、“DB. ERR_CODE” キーでリクエストにセットしています。 これらは、リクエストのアトリビュートへセットしている為、query タグや、directWriteTable タグの処理以降のリクエストスコープ中での取得になります。
7.session	【TAG】session Attribute の値を取得します。 session スコープに登録されているアトリビュートオブジェクトの値を取り出します。
8.date	【TAG】日付関係の値を取得します。 これは、{@DATE. XXXX AA BB} 引数処理をおこなうための、サポートメソッドです。 引数は、“XXXX AA BB” という状態で受け取ります。(DATE. ははずした形) “XXXX” は、日付処理を行うキー文字列で予約語になっています。 <ul style="list-style-type: none">・YMD : 8 文字の日付データ (200X年XX月XX日) を扱います。・Y2MD : 6 文字の日付データ (0X年XX月XX日) を扱います。・YM : 6 文字の日付データ (200X年XX月) を扱います。・HMS : 6 文字の時刻データ (XX時XX分XX秒) を扱います。・YMDHMS : 1 4 文字の日付データ (200X年XX月XX日XX時XX分XX秒) を扱います。・EEE : 曜日をデフォルトロケールで表示します。 F付きは、フォーマットされた日付を返します。 <ul style="list-style-type: none">・YMDF : 8 文字の日付データ (200X年XX月XX日) を扱います。・Y2MDF : 6 文字の日付データ (0X年XX月XX日) を扱います。・YMF : 6 文字の日付データ (200X年XX月) を扱います。・HMSF : 6 文字の時刻データ (XX時XX分XX秒) を扱います。・YMDHMSF : 1 4 文字の日付データ (200X年XX月XX日XX時XX分XX秒) を扱います。 AA 引数は、基準となる日付を、YYYYMMDD形式で指定します。無指定の場合は、処理時刻を基準にした、同一タグ内での同一日付(年月日時分秒)を使用して値を取得します。 指定できる日付は、必ず、YYYYMMDD形式とし、時分秒は、ゼロにリセットされた状態になります。 AA には、数字で始まる (20050701 など) 実日付と@で始まるパラメータが使用できます。 @ZZ と記述することで、{@ZZ} で指定する場合と同様のリクエストパラメータが使用できます。 このパラメータの値は、YYYYMMDD形式の数字か、null (またはゼロストリング) です。null の場合は、なにも指定されていないと判断して、処理時刻を使用します。 第2引数は、(数字か、@) 以外の場合は、省略されたと判断されます。 BB 引数は、日付についての加減算処理を行います。 省略すると、なにも加減算処理を行いません。 <ul style="list-style-type: none">・SD : 当月の最初の日付にセットします。(当月1日)・ED : 当月の最後の日付にセットします。(当月月末)・SM : 本年の最初の月にセットします。(本年1月)・EM : 本年の最後の月にセットします。(本年12月)・SW : 日付処理の週初め(月曜日)にセットします。日付は当日より前に移動します。・EW : 日付処理の週末(日曜日)にセットします。日付は当日より後ろに移動します。・D1 ~ DXXX : 日を指定の分だけ進めます。D1なら翌日、D200 なら200日後・M1 ~ MXXX : 月を指定の分だけ進めます。M1なら翌月、M6 なら半年後
9.nvar	【TAG】Unicode文字列の値を HTML のエスケープ記号(&#xZZZZ;)に変換します。 HTMLでのエスケープ文字を使用して、Unicodeを文字列化して登録する為の変換を行います。 ここでは、入力文字を、キャラクタ(char)型に分解し、(&#xZZZZ;)に変換していきます。 よって、通常に1文字(Shift-JISで2Byte, UTF-8で3Byte)が、8 Byteになります。 この変換された文字列を、HTML上でそのまま取り出すと、元のUnicode文字に戻る為、通常のShift-JISでは、扱えない文字(中国語など)でも表示可能になります。
10.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値: false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
5.5.4.2 og:responseHeader	<p>【TAG】レスポンスヘッダ に、Cache-Control の値を設定します(初期値:"max-age=0")。このヘッダは、クライアントに対してドキュメントをキャッシュする場合の条件を伝えます。初期値は、max-age=0 に設定しています。指定する値は、以下のどれかです。</p> <p>public : ドキュメントをキャッシュして良い private : ドキュメントが共有されないプライベートの中なら、キャッシュして良い。 no-cache : ドキュメントをキャッシュしてはいけない。 no-store : ドキュメントのキャッシュや、ディスク上の一時ファイルも禁止する。 must-revalidate : クライアントは、ドキュメントをプロキシではなく、本来のサーバーに確認する必要がある。 proxy-revalidate : must-revalidate と同じであるが、共有キャッシュに対してのみ適用される。 max-age=xxx : ドキュメントが、xxx秒後に陳腐化する。Expires より優先される。 s-max-age=xxx : 共有キャッシュは、ドキュメントが、xxx秒後に陳腐化する。</p>
●形式 : <og:responseHeader ●body : なし	cacheKey="[...]" />
●Tag定義 : <og:responseHeader cacheControl contentType refresh refreshURL redirect status location debug >	<p>【TAG】レスポンスヘッダ に、Cache-Control の値を設定します(初期値:"max-age=0")</p> <p>【TAG】レスポンスヘッダ に、content-Type の値を設定します</p> <p>【TAG】レスポンスヘッダ に、refresh の値を設定します</p> <p>【TAG】レスポンスヘッダ に、refresh の値を設定するときに、指定のURLをロードします</p> <p>【TAG】指定されたURLへ一時的なリダイレクトレスポンスを送信します</p> <p>【TAG】ステータスコードを設定します</p> <p>【TAG】レスポンスヘッダ に、location の値を設定します</p> <p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)</p>
●使用例	
1.cacheControl	<p>【TAG】レスポンスヘッダ に、Cache-Control の値を設定します(初期値:"max-age=0")。このヘッダは、クライアントに対してドキュメントをキャッシュする場合の条件を伝えます。初期値は、max-age=0 に設定しています。指定する値は、以下のどれかです。</p> <p>public : ドキュメントをキャッシュして良い private : ドキュメントが共有されないプライベートの中なら、キャッシュして良い。 no-cache : ドキュメントをキャッシュしてはいけない。 no-store : ドキュメントのキャッシュや、ディスク上の一時ファイルも禁止する。 must-revalidate : クライアントは、ドキュメントをプロキシではなく、本来のサーバーに確認する必要がある。 proxy-revalidate : must-revalidate と同じであるが、共有キャッシュに対してのみ適用される。 max-age=xxx : ドキュメントが、xxx秒後に陳腐化する。Expires より優先される。 s-max-age=xxx : 共有キャッシュは、ドキュメントが、xxx秒後に陳腐化する。</p>
2.contentType	<p>【TAG】レスポンスヘッダ に、content-Type の値を設定します。このヘッダは、これから返すドキュメントのMIMEタイプを与えます。MIMEタイプの詳しい規格は、RFC1521 と、RFC1522 です。</p> <p>最新リスト 未登録タイプ(x-タイプ)</p>
3.refresh	<p>【TAG】レスポンスヘッダ に、refresh の値を設定します。このヘッダは、更新されたページをブラウザが今から何秒後にリクエストすればよいかということを伝えます。</p>
4.refreshURL	<p>【TAG】レスポンスヘッダ に、refresh の値を設定するときに、指定のURLをロードします。このヘッダは、refresh と共に使用され、リクエストする場合のURLを指定します。</p>
5.redirect	<p>【TAG】指定されたURLへ一時的なリダイレクトレスポンスを送信します。指定されたリダイレクト先のURLを用いて、クライアントに一時的なリダイレクトレスポンスを送信します。URLとしては相対URLを指定することができます。</p>
6.status	<p>【TAG】ステータスコードを設定します。</p> <p>ステータスコードを設定します。</p> <p>100 ~ 199 1 0 0番台はおしらせ的な情報です。 200 ~ 299 2 0 0番台はリクエストが成功したことを表します。 300 ~ 399 3 0 0番台はファイルが移動したことを表します。 400 ~ 499 4 0 0番台はクライアント側のエラーを表します。 500 ~ 599 5 0 0番台はサーバー側のエラーを表します。</p>
7.location	<p>【TAG】レスポンスヘッダ に、location の値を設定します。このヘッダは、ドキュメントのアドレスを通知します。3 0 0番台のステータスコードには、このヘッダが必ず付随する必要があります。</p>
8.debug	<p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。デバッグ情報を [true:出力する/false:しない]を指定します。出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:schedule JSP	【TAG】 コマンド(SET, VIEW, REMOVE, CANCEL)をセットします(初期値:SET)。 コマンドは、HTMLから(get/post) 指定されますので、CMD_xxx で設定される フィールド定数値のいずれかを、指定できます。 何も設定されない、または、null の場合は、“SET” が初期値にセットされます。
●形式: <og:schedule timerTask="..." name="..." command="[SET VIEW REMOVE CANCEL]" ... /> ●body: あり	
●Tag定義: <og:schedule command timerTask name comment singleton delay period fixedDelay keys vals startTime stopTime scope tableId debug > ... Body ... </og:schedule>	【TAG】 コマンド(SET, VIEW, REMOVE, CANCEL)をセットします(初期値:SET) ○【TAG】 処理を実行するクラス名(HybsTimerTask のサブクラス)を指定します。(必須) ○【TAG】 名称(ユニークキー)を設定します。(必須) 【TAG】 タイマータスクの説明を設定します 【TAG】 Name属性が同一のタスクを2重登録出来ないよう[true/false]にします(初期値:true[出来ない]) 【TAG】 ミリ秒単位の遅延時間を設定します(初期値:0) 【TAG】 ミリ秒単位の繰り返し間隔を設定します(初期値:60000) 【TAG】 固定遅延実行を指定するかどうか[true:固定遅延実行/false:固定頻度実行]を設定します(初期値:true) 【TAG】 HybsTimerTask に渡す為のキー情報、CSV形式で複数指定します 【TAG】 HybsTimerTask に渡す為の値を、CSV形式で複数指定します 【TAG】 24時間制(YMMDD)の開始時刻を設定します(初期値:000000) 【TAG】 24時間制(YMMDD)の終了時刻を設定します(初期値:000000) 【TAG】 キャッシュする場合のスコープ(local, request, page, session, applicaton)を指定します(初期値:local) 【TAG】 (通常は使いません)結果をDBTableModelに書き込んで、sessionに登録するときのキーを指定します 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)
●使用例 帳票デモンスケジュール登録 period は、ms 単位で指定する必要があるため、1000倍しています。 <og:schedule command scope timerTask name comment singleton delay period fixedDelay keys vals > </>	= "{@CMD}" = "request" = "org.opengion.hayabusa.report.ReportDaemon" = "{@NAME}" = "Start-up By {@USER, INFO}" = "true" = "0" = "{@PERIOD}000" = "true" = "SYSTEM_ID, DMN_GRP" = "{@SYSTEM_ID}, {@DMN_GRP}"
1.command	【TAG】 コマンド(SET, VIEW, REMOVE, CANCEL)をセットします(初期値:SET)。 コマンドは、HTMLから(get/post) 指定されますので、CMD_xxx で設定される フィールド定数値のいずれかを、指定できます。 何も設定されない、または、null の場合は、“SET” が初期値にセットされます。
2.timerTask	【TAG】 処理を実行するクラス名(HybsTimerTask のサブクラス)を指定します。 処理を実行する HybsTimerTask のクラス名を指定します。
3.name	【TAG】 名称(ユニークキー)を設定します。 タイマータスクの名称(ユニークキー)を設定します。
4.comment	【TAG】 タイマータスクの説明を設定します。 タイマータスクの説明を設定します。
5.singleton	【TAG】 Name属性が同一のタスクを2重登録出来ないよう[true/false]にします(初期値:true[出来ない])。 同一種類のタスクを複数同時立上げたくない場合に、利用します。 一般に、タイマータスクそのものが、繰り返し実行されますので、 初期値は、2重登録禁止にしています。
6.delay	【TAG】 ミリ秒単位の遅延時間を設定します(初期値:0)。 タスクが実行される前のミリ秒単位の遅延を設定します(初期値:0)
7.period	【TAG】 ミリ秒単位の繰り返し間隔を設定します(初期値:60000)。 連続するタスク実行のミリ秒単位の時間を設定します(初期値:60000 ms)
8.fixedDelay	【TAG】 固定遅延実行を指定するかどうか[true:固定遅延実行/false:固定頻度実行]を設定します(初期値:true)。 固定遅延実行では、前の実行の実際の実行時間を基準にしてそれぞれの実行がスケジュールされます。 何らかの理由で実行が遅延した場合 そのあとの実行も遅延されます。 最終的に、実行の頻度は通常、指定した期間の対応する頻度よりも若干遅くなります 固定頻度実行では、最初の実行のスケジュールされた実行時間を基準にしてそれぞれの実行が スケジュールされます。何らかの理由で実行が遅延した場合、「遅れを取り戻す」ために 2 以上の実行が連続して行われます。最終的に実行の頻度は、指定した期間の対応する頻度と 同じになります 初期値は、true:固定遅延実行 です。
9.keys	【TAG】 HybsTimerTask に渡す為のキー情報、CSV形式で複数指定します。 HybsTimerTask に渡す為の、キー情報をカンマ区切りで複数指定できます。 vals 属性には、キーに対応する値を、設定してください。 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスる為です。
10.vals	【TAG】 HybsTimerTask に渡す為の値を、CSV形式で複数指定します。 キーに設定した値を、カンマ区切り文字で複数して出来ます。 指定順序は、キーと同じにしておいて下さい。 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスる為です。
11.startTime	【TAG】 24時間制(YMMDD)の開始時刻を設定します(初期値:000000)。 指定時刻範囲内での実行のみ許可するように開始時刻を設定します。 これは、タイマーで指定した間隔ごとにチェックを入れるので、チェック時間が 長い場合は、正確に開始時刻から始まるというものではありません。 初期値は、“000000” です。
12.stopTime	【TAG】 24時間制(YMMDD)の終了時刻を設定します(初期値:000000)。 指定時刻範囲内での実行のみ許可するように終了時刻を設定します。 これは、タイマーで指定した間隔ごとにチェックを入れるので、チェック時間が 長い場合は、正確に終了時刻で終了するというものではありません。 (終了時刻を超えてからの新規実行はありません。) 初期値は、“000000” です。
13.scope	【TAG】 キャッシュする場合のスコープ(local, request, page, session, applicaton)を指定します(初期値:local)。 “local”, “request”, “page”, “session”, “applicaton” が指定できます。 “local” は、過去のバージョンとの互換性を保つ為の特殊な値です。 初期値は、“local” です。
14.tableId	【TAG】 (通常は使いません)結果をDBTableModelに書き込んで、sessionに登録するときのキーを指定します。 初期値は、HybsSystem.TBL_MD_KEY です。
15.debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:select 拡張 JSP	【TAG】 メニューの名称を指定します。 メニューの名称を指定します。
●形式 : <og:select name="..." /> ●body : あり	
●Tag定義 : <og:select name value defaultVal lbl must mustAny addNoValue td colspan optionAttributes size multiple id caseKey caseVal lang dir title style disabled tabindex clazz language onClick onChange onBlur onFocus ondblClick onMouseDown onMouseUp onMouseMove onMouseOut onMouseOver onSelect onKeydown onKeyPress onKeyUp roles eventColumn eventURL eventCallback debug > ... Body ... </og:select>	
●使用例 <og:select name="CDC"> <og:option code="CDC" /> </og:select> <og:select name="CDC"> <og:option msg="MSG0001" /> <og:option code="CDC" /> </og:select> <og:select name="CDC"> <og:option lbl="CDC1" /> <og:option code="CDC" /> </og:select> <og:select name="CDC" multiple="multiple"> <og:option code="CDC" /> </og:select> <og:select name="CDC" disabled="disabled"> <og:option code="CDC" /> </og:select> <og:select name="CDC"> <og:queryOption> select NOSYN,NOSYN,':',NMSYN from DB01 ORDER BY 1 </og:queryOption> </og:select> <og:select name="CDC" eventColumn="OYA" > <og:queryOption> select NOSYN,NOSYN,':',NMSYN from DB01 where GLM = "[@OYA]" ORDER BY 1 </og:queryOption> </og:select>	
1.name	【TAG】 メニューの名称を指定します。 メニューの名称を指定します。
2.value	【TAG】 値を指定します。 ここで指定された値が、optionタグの値と同じであれば、選択状態になります。 <og:select name="CDC" value="[@ABC]" defaultVal="XYZ" > <og:option value="ABC" lbl="CDC1" /> <og:option value="XYZ" lbl="CDC2" /> </og:select>
	【TAG】 value属性に値がセットされていないとき使用する、初期値を指定します。 value属性に値がセットされていないときに、初期値をセットします。

バージョン タグ名 HTML5 BODY要素	内容
	形式サンプル
3.defaultVal	なお、commandがRENEWの場合は、このdefaultValは適用されません。 これは、defaultValの値が埋め込まれた項目が存在する画面に戻った際に、defaultValの項目の値がnullとなっていた場合に、defaultValの値が復元されると、検索結果との不一致が発生するためです。 【TAG】ラベルリソースのラベルIDを指定します。 ラベルを変更するときに、lbl属性を使います。
4.lbl	ラベルID は、所定の language に基づく ResourceManager の getLabelData(id) を呼び出し、その結果のLabelInterfaceを使います。 getMsglbl() で取り出せます。 ラベルIDとメッセージIDは同時には登録できません。 ラベルとメッセージは統一されました。
5.must	【TAG】必須入力を表す色に変えるかどうか[true/false]を指定します(初期値:false)。 初期値は、必須でない("false") です。
6.mustAny	<code><og:input name="PN" must="true" /></code> 【TAG】選択必須入力(どれかひとつ必須)を表す色[true/mustAny/その他]を指定します(初期値:無指定)。 複数のカラムのうち、どれかひとつを必須とする選択必須入力を示す色を指定します。 true または、mustAny を設定すると、class属性に、mustAny がセットされます。 mustAny は、CSSファイルに初期設定されています。true または、mustAny 以外の値をセットすると、その値がそのまま、class属性にセットされますので、選択必須のグループ化が可能です。 なお、実際の選択必須入力チェックは、ここではなく、columnCheck タグで指定が必要です。 自動処理は、mustAny="true" 指定の場合のみ有効です。 初期値は、無指定です。
7.addNoValue	【TAG】値なしのOptionを含めるかどうか[true/false]を指定します(初期値:false)。 カラムがプルダウンメニュー形式の場合、選択肢には通常データベース登録できる候補が並びます。しかし、検索時には、候補を絞りたくない(指定しない)場合があり、その様な場合には、値なしのOptionを含める(true)ことで、対応可能です。 初期値:false(値なしのOptionを含めない)は、過去の互換性を重視しているため、column タグと異なります。
8.td	【TAG】テーブル形式の <td> タグを使用するかどうか[yes/no/false]を指定します(初期値:yes)。 初期値は、使用する("yes") です。
9.colspan	【TAG】フィールド部分の colspan を指定します。 通常は、3 を指定して、フィールド、ラベル、フィールドの 組み合わせ部分に、長いフィールドが適用されるように使用します。
10.optionAttributes	<code><og:select name="CDK" colspan="3" /></code> 【TAG】JavaScript などの HTML基本タグ以外の属性を、そのままタグとして使用します。 JavaScript などの HTML基本タグ以外の属性を、そのまま タグとして使用します。 そのときに、()は、()に置き換えます。
11.size	【TAG】リストボックスとして表示する場合の表示行数を指定します。 表示行数を指定します。
12.multiple	【TAG】複数選択を可能(multiple)にします(初期値:未設定)。 multiple 値が "multiple" または "true" の場合、複数選択状態になります。 また、"ALL" を設定する場合は、OptionTag上で、すべてのOption が選択状態に 初期設定されます。
13.id	大文字、小文字の種別はなくしますが、指定以外の文字が設定された場合は、エラーになります。 【HTML】要素に対して固有の名前(id)をつける場合に設定します。 特別な使用方法として、id="FOCUS" とすることで、フィールド系要素に フォーカスを移動させます。これは、そのページ内で唯一の id 属性として使用ください。
14.caseKey	【TAG】このタグ自体を利用するかどうかの条件キーを指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。
15.caseVal	何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。 【TAG】このタグ自体を利用するかどうかの条件値を指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。
16.lang	何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。 【HTML】要素の内容と他の属性値の言語(lang.xml:lang)を指定します。 HTMLの言語属性に使われます。指定する値は、ISO 639で規定されている「言語コード」です。 [ja/en/zh/...]などのほかに、en-US:アメリカ英語、en-cockney:コックニー英語 など、 副言語を指定する方法も定められています。 ここでは、lang と xml:lang の両方に同じ値がセットされます。 タグの language 属性とは使用用途が異なります。lang セット
17.dir	【HTML】文字表記の方向(dir)を指定します。 当該要素の書字方向を指定する属性です。 ltr で、左から右に、rtl で、右から左に並べます。
18.title	【HTML】要素に対する補足的情報(title)を設定します。 title セットは、ボタンなどに適用すると、マウスオーバーによりこのメッセージが チップスのように表示されます。これを利用して、説明文を登録することが可能です。 ここに登録した文字列が、メッセージリソースに存在する場合は、そのメッセージを 存在しない場合は、そのままの値を設定します。
19.style	【HTML】この要素に対して適用させるスタイルシート(style)を設定します。 タグにstyle属性を設定します。これは、キー:値: のセットを複数記述できます。 通常は、class属性や、id属性で登録しておく、&lg:style type="text/css"> で 外部から指定する方がソースは読みやすくなります。
20.disabled	【TAG】その部品に対して、選択や変更が出来ないように(disabled)指定します(サーバーに送信されない)。 BUTTON/INPUT/OPTGROUP/OPTION/SELECT/TEXTAREA 系に対して、指定可能です。 disabled="disabled" , disabled="true" が指定された場合は、有効です。 false も指定値としては、有効です。(大文字小文字の区別も不要) それ以外の指定は、エラーとします。
21.tabindex	【HTML】タブの移動順(tabindex)を指定します(0 ~ 32767)。 Tabキーを押したときに要素が選択される順番を指定します。 値には、選択させたい順番を数値で記述します。
22.clazz	【HTML】要素に対して class 属性を設定します。 Javaの言語使用上、class で作成できないため、代用として clazz を使用しています。
23.language	html で作成される属性は、class で作成されます。 【TAG】タグ内部で使用する言語コード[ja/en/zh/...]を指定します。 この言語コードに基づいて、表示のラベルをリソースから作成します。
24.onClick	【HTML】JavaScriptのイベント onClick を設定します(例:onClick="renew(' query. jsp', ' QUERY'):")。 onClick をセットします。 例えば、<og:column name="KBSAKU" onClick="renew(' query. jsp', ' QUERY'):" /> のように指定することで、プルダウンメニューの絞込み検索が可能になります。
25.onChange	【HTML】JavaScriptのイベント onChange を設定します(例:onChange="renew(' query. jsp', ' QUERY'):")。 onChange をセットします。 例えば、<og:column name="KBSAKU" onChange="renew(' query. jsp', ' QUERY'):" /> のように指定することで、プルダウンメニューの絞込み検索が可能になります。
26.onBlur	【HTML】JavaScriptのイベント onBlur を設定します(例:onBlur="this.value=value.toUpperCase();")。 onBlur は、フォーカスが離れたときに発生するイベントです。
27.onFocus	【HTML】JavaScriptのイベント onFocus を設定します。 onFocus は、フォーカスされたときに発生するイベントです。
	【HTML】JavaScriptのイベント ondblClick を設定します。

バージョン タグ名 HTML&EXT BODY要素	内容
	形式サンプル
28.ondblClick	ondblClick は、マウスでダブルクリックされたときに発生するイベントです。
29.onMouseDown	【HTML】JavaScriptのイベント onMouseDown を設定します。 onMouseDown は、マウスダウンされたときに発生するイベントです。
30.onMouseUp	【HTML】JavaScriptのイベント onMouseUp を設定します。 onMouseUp は、マウスアップされたときに発生するイベントです。
31.onMouseMove	【HTML】JavaScriptのイベント onMouseMove を設定します。 onMouseMove は、マウスが移動されたときに発生するイベントです。
32.onMouseOut	【HTML】JavaScriptのイベント onMouseOut を設定します。 onMouseOut は、マウスが離れたときに発生するイベントです。
33.onMouseOver	【HTML】JavaScriptのイベント onMouseOver を設定します。 onMouseOver は、マウスが重なったときに発生するイベントです。
34.onSelect	【HTML】JavaScriptのイベント onSelect を設定します。 onSelect は、テキストフィールド/テキストエリアのテキストが 選択されたときに発生するイベントです。
35.onKeyDown	【HTML】JavaScriptのイベント onKeyDown を設定します。 onKeyDown は、キーが押されたときに発生するイベントです。
36.onKeyPress	【HTML】JavaScriptのイベント onKeyPress を設定します。 onKeyPress は、キーが押され続けていたときに発生するイベントです。
37.onKeyUp	【HTML】JavaScriptのイベント onKeyUp を設定します。 onKeyUp は、キーが押された状態から離されたときに発生するイベントです。
38.roles	【TAG】ロールを設定します。 ここで指定したカラムロールを元に、ユーザー毎のアクセス許可がチェックされます。 アクセス許可されないと、表示されません。 このロールを指定しない場合は、カラムリソースのロールが使用されます。
39.eventColumn	【TAG】イベントカラム(親カラム)を指定します。 親子関係のカラムを作成する場合に利用します。 queryOptionタグとセットで利用して下さい。 ※詳細はcolumnTagのeventColumn属性の説明をご覧ください。
40.eventURL	【TAG】イベントカラム指定時に呼び出すURL。 イベントカラム指定時に部品を作成するJSPを指定します。 初期値はシステムリソースのEVENT_COLUMN_URLです。 (例: eventURL="makeColumn_custom.jsp") ※詳細はcolumnTagのeventURL属性の説明をご覧ください。
41.eventCallback	【TAG】eventColumn実行後のcallback関数指定。 eventColumnの動作終了後に実行するcallback関数の指定を行います。 関数名のみを指定して下さい。 ex) sampleCallback() を実行する場合はeventCallback="sampleCallback" 実行する関数の第一引数には自動的に次の要素が入った配列が渡されます [0] 変更された子カラムのID属性値 [1] eventColumn実行後の子カラムの値 [2] eventColumn実行前の子カラムの値
42.debug	この属性は、optionAttributesへの設定と同様の動作を行います。 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値: false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
<p>5.5.4.2 og:shell</p> <p>Tag</p> <p>●形式 : <og:shell program="..." /></p> <p>●body : なし</p> <p>●Tag定義 :</p> <pre><og:shell program useBatch wait timeout stdout stderr info action workDir envKeys envVals display debug /></pre> <p>●使用例</p> <pre><og:shell program = "プログラム" dir /S や、abc.bat など action = "アクション" [RUN/SET/GET/REMOVE] useBatch = "バッチ使用" バッチ : true/false (初期値:false) wait = "終了待ち" 処理終了まで待つ:true/待たない:false (初期値:true) stdout = "標準出力使用" 標準出力使用:true/未使用:false (初期値:false) stderr = "エラー出力使用" エラー出力使用:true/未使用:false (初期値:false) info = "状態表示" シェルの状態表示:true/未使用:false (初期値:false) /></pre> <p><og:value command="GET" key="SHELL_RTN_CODE" /></p>	<p>【TAG】 シェルファイルを実行する時に引き渡すコマンド。 シェルファイルを実行する時に引き渡すコマンド</p> <p>○ 【TAG】 シェルファイルを実行する時に引き渡すコマンド。(必須)</p> <p>【TAG】 BATCHプロセスを実行するかどうか[true/false]を指定します(初期値:false)</p> <p>【TAG】 プロセスの終了を待つかどうか[true/false]を指定します(初期値:true[待つ])</p> <p>【TAG】 プロセスの実行処理のタイムアウトを設定します (初期値:SHELL_TIMEOUT[-3600])</p> <p>【TAG】 実行結果を標準出力に出力するかどうか[true/false]を指定します(初期値:false)</p> <p>【TAG】 実行結果をエラー出力に出力するかどうか[true/false]を指定します(初期値:false)</p> <p>【TAG】 インフォメーション情報を出力するかどうか[true/false]を指定します(初期値:false)</p> <p>【TAG】 アクション(RUN, SET, GET, REMOVE)をセットします(初期値:RUN)</p> <p>【TAG】 作業ディレクトリを指定します(初期値:Java仮想マシンの作業ディレクトリ)</p> <p>【TAG】 環境変数のキーをC S V形式で指定します(初期値:現状のJava仮想マシンの環境)</p> <p>【TAG】 環境変数の値をC S V形式で指定します(初期値:現状のJava仮想マシンの環境)</p> <p>【TAG】 処理結果を画面に表示するかどうか[true/false]指定します(初期値:true)</p> <p>【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)</p>
1.program	【TAG】 シェルファイルを実行する時に引き渡すコマンド。 シェルファイルを実行する時に引き渡すコマンド
2.useBatch	【TAG】 BATCHプロセスを実行するかどうか[true/false]を指定します(初期値:false)。 BATCHファイルの実行と、EXEファイルの実行では、cmd.exe 経由するか どうか、判断が必要です。true(BATCHプロセス)では、cmd.exe 経由で実行、 false(EXEファイル)では、直接実行します。 初期値は、false(EXEファイル)です。
3.wait	【TAG】 プロセスの終了を待つかどうか[true/false]を指定します(初期値:true[待つ])。 プロセスの終了を待つ(true)/待たない(false) (初期値:true)
4.timeout	【TAG】 プロセスの実行処理のタイムアウトを設定します (初期値:SHELL_TIMEOUT[-3600])。 ゼロ(0) の場合は、割り込みが入るまで待ちつづけます。 その場合、確実に終了させる手段を用意しておかないと、いつまで待っても以下の処理が 実行されない(デッドロック)状態になってしまいます。 (初期値:システム定数のSHELL_TIMEOUT [= @og.value org.opengion.hayabusa.common.SystemData#SHELL_TIMEOUT])。
5.stdout	【TAG】 実行結果を標準出力に出力するかどうか[true/false]を指定します(初期値:false)。 実行結果の標準出力を出力するかどうか(初期値:false)
6.stderr	【TAG】 実行結果をエラー出力に出力するかどうか[true/false]を指定します(初期値:false)。 実行結果のエラー出力を出力するかどうか(初期値:false)
7.info	【TAG】 インフォメーション情報を出力するかどうか[true/false]を指定します(初期値:false)。 インフォメーション情報を出力するかどうか(初期値:false)
8.action	【TAG】 アクション(RUN, SET, GET, REMOVE)をセットします(初期値:RUN)。 アクションは、HTMLから(get/post) 指定されますので、ACT_XXX で設定される フィールド定数値のいずれかを、指定できます。 RUN Shellオブジェクトを生成/実行します。(セッションに登録しません。) SET Shellオブジェクトを生成/実行し、セッションに登録します。 GET セッションに登録されている Shellオブジェクトを呼び出します。 REMOVE セッションから、Shellオブジェクトを削除します。 Shellが実行中の場合は強制終了します。
9.workDir	【TAG】 作業ディレクトリを指定します(初期値:Java仮想マシンの作業ディレクトリ)。 シェルを実行する、作業ディレクトリを指定します。 絶対パスで指定することも、相対パスで指定することも出来ます。 相対パスの場合は、ドキュメントベースの物理パスが基準になります。 (そのJSPファイル自身のディレクトリが基準ではありません。) 指定しない場合は、このJava仮想マシンの作業ディレクトリで実行されます。
10.envKeys	【TAG】 環境変数のキーをC S V形式で指定します(初期値:現状のJava仮想マシンの環境)。 シェル実行時の環境変数のキーを、C S V形式で指定します。 指定がない場合は、現状のJava仮想マシンの環境が引き継がれます。 envVals と対応していなければなりません。 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスる為です。
11.envVals	【TAG】 環境変数の値をC S V形式で指定します(初期値:現状のJava仮想マシンの環境)。 シェル実行時の環境変数の値を、C S V形式で指定します。 指定がない場合は、現状のJava仮想マシンの環境が引き継がれます。 envKeys と対応していなければなりません。 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスる為です。
12.display	【TAG】 処理結果を画面に表示するかどうか[true/false]指定します(初期値:true)。 処理結果が、異常でない場合のみ、この機能が有効になります。
13.debug	各種処理を連続で行う場合、最後にエラーが発生したにもかかわらず、 先に正常終了している場合に、不要なメッセージを出すと、紛らわしい為です。 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない] を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:space	【TAG】 全体の大きさ (width 属性) を指定します。 通常の space タグは、td 2 個分のタグを出力しますが、width 属性を指定した場合は、colspan="2" をセット下上で、width 属性に設定されます。
Tag	
●形式 : <og:space />	
●body : なし	
●Tag定義 :	
<og:space	【TAG】 全体の大きさ (width 属性) を指定します
width	【TAG】 全体の大きさ (width 属性) を指定します
colspan	【TAG】 ラベル部分 (前半部分) の大きさ (width 属性) を指定します
labelWidth	【TAG】 値部分 (後半部分) の大きさ (width 属性) を指定します
valueWidth	【TAG】 デバッグ情報を出力するかどうか [true/false] を指定します (初期値: false)
debug	
/>	
●使用例	
通常の、ラベル部、フィールド部が空の td タグを生成します。	
<og:space />	
td タグの大きさを指定することができます。	
<og:space	
labelWidth = "ラベル部分 (前半部分) の大きさ (width 属性) を指定します。"	
valueWidth = "値部分 (後半部分) の大きさ (width 属性) を指定します。"	
/>	
1.width	【TAG】 全体の大きさ (width 属性) を指定します。 通常の space タグは、td 2 個分のタグを出力しますが、width 属性を指定した場合は、colspan="2" をセット下上で、width 属性に設定されます。
2.colspan	【TAG】 全体の大きさ (width 属性) を指定します。 属性は td タグの colspan 属性に設定されます。
3.labelWidth	【TAG】 ラベル部分 (前半部分) の大きさ (width 属性) を指定します。 属性は td タグの width 属性に設定されます。
4.valueWidth	【TAG】 値部分 (後半部分) の大きさ (width 属性) を指定します。 属性は td タグの width 属性に設定されます。
5.debug	【TAG】 デバッグ情報を出力するかどうか [true/false] を指定します (初期値: false)。 デバッグ情報を [true:出力する/false:しない] を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
5.5.4.2 og:splitView JSP	<p>【TAG】 固定するカラム名を、CSV形式(カンマ区切り文字列)で設定します。 Viewタグの左側(固定部)には、このタグで指定された fixDisplay のカラムを columnDisplay に設定します。 Viewタグの右側には、fixDisplay のカラムを noDisplay のカラムに設定します。 既存の設定値(noDisplay)があれば、それに追記されます。</p> <p>●形式 : <og:splitView fixDisplay="CLM, NAME_JA"><og:view . . . /></og:splitView> ●body : あり</p> <p>●Tag定義 : <og:splitView fixDisplay 【TAG】 固定するカラム名を、CSV形式(カンマ区切り文字列)で設定します debug 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false) > . . . Body . . . </og:splitView></p> <p>●使用例 <og:splitView fixDisplay="CLM, NAME_JA"> <og:view viewFormType = "HTMLTable" command = "{@command}" checked = "{@checked}" startNo = "{@startNo}" pageSize = "{@pageSize}" noWritable = "{@noWritable}" columnWritable = "{@columnWritable}" /> </og:splitView></p>
1.fixDisplay	<p>【TAG】 固定するカラム名を、CSV形式(カンマ区切り文字列)で設定します。 Viewタグの左側(固定部)には、このタグで指定された fixDisplay のカラムを columnDisplay に設定します。 Viewタグの右側には、fixDisplay のカラムを noDisplay のカラムに設定します。 既存の設定値(noDisplay)があれば、それに追記されます。</p>
2.debug	<p>【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。</p>

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

5.5.4.2
og:submit
拡張
Tag

【TAG】アクション (INSERT, COPY, MODIFY, DELETE, ENTRY, RESET) を指定します。
Ver5 になって、update.jsp ですべての処理を行う方向で開発するに当たり、command と lbl を記述する必要ができました。
そこで、現在最新版では、action 属性を用意し、command を記述するだけで良くなりました。

[action属性]	[初期設定されるパラメータ郡]			
INSERT	value="update"	command="INSERT"	lbl="INSERT"	accesskey="I"
COPY	value="update"	command="COPY"	lbl="COPY"	accesskey="C"
MODIFY	value="update"	command="MODIFY"	lbl="MODIFY"	accesskey="M"
DELETE	value="update"	command="DELETE"	lbl="DELETE"	accesskey="Z"
ENTRY	value="entry"	command="ENTRY"	lbl="ENTRY"	accesskey="E"
RESET	value="reset"	command="RESET"	lbl="ENTRY"	accesskey="R"

●形式 : <og:submit value="..." lbl="..." />
●body : なし

●Tag定義 :
<og:submit
 action
 value
 command
 gamenId
 lbl
 target
 keys
 vals
 roles
 dbkeys
 optionAttributes
 columnWritable
 noWritable
 caseKey
 caseVal
 id
 lang
 dir
 title
 style
 readonly
 disabled
 tabindex
 accesskey
 clazz
 language
 onClick
 onBlur
 onFocus
 ondblClick
 onMouseDown
 onMouseUp
 onMouseMove
 onMouseOut
 onMouseOver
 img
 debug
/>

●使用例
Ver5 推奨ケース : 指定の action 属性に設定された値に応じた、value , command , lbl , accesskey を初期化します。
(5.2.2.0 (2010/11/01)以降)

<og:writeCheck>
 <og:submit action="COPY" noWritable="FGJ,UNIQ" />
 <og:submit action="MODIFY" noWritable="CLM,FGJ,UNIQ" />
 <og:submit action="DELETE" columnWritable="null" />
 <og:submit gamenId="GF9110" value="index" dbkeys="SYSTEM_ID,LANG,CLM" command="NEW" lbl="GF9100" target="CONTENTS" />

</og:writeCheck>

Ver4 一般的なケース : value は共通になったため、command と lbl の設定が必要(下位互換性のための設定)

<og:writeCheck>
 <og:submit value="update" command="COPY" lbl="COPY" accesskey="C" />
 <og:submit value="update" command="MODIFY" lbl="MODIFY" accesskey="M" />
 <og:submit value="update" command="DELETE" lbl="DELETE" accesskey="Z" />
 <og:submit gamenId="GF9110" value="index" dbkeys="SYSTEM_ID,LANG,CLM" command="NEW" lbl="GF9100" target="CONTENTS" />

</og:writeCheck>

従来のケース : value に設定された値JSPに対してサブミットされます。(下位互換性のための設定)

<og:writeCheck>
 <og:submit value="copy" lbl="MSG0035" accesskey="C" />
 <og:submit value="modify" lbl="MSG0036" accesskey="M" />
 <og:submit value="delete" lbl="MSG0037" accesskey="Z" />
 <og:submit gamenId="GF9110" value="index" dbkeys="SYSTEM_ID,LANG,CLM" command="NEW" lbl="GF9100" target="CONTENTS" />

</og:writeCheck>

1.action

【TAG】アクション (INSERT, COPY, MODIFY, DELETE, ENTRY, RESET) を指定します。
Ver5 になって、update.jsp ですべての処理を行う方向で開発するに当たり、command と lbl を記述する必要ができました。
そこで、現在最新版では、action 属性を用意し、command を記述するだけで良くなりました。

[action属性]	[初期設定されるパラメータ郡]			
INSERT	value="update"	command="INSERT"	lbl="INSERT"	accesskey="I"
COPY	value="update"	command="COPY"	lbl="COPY"	accesskey="C"
MODIFY	value="update"	command="MODIFY"	lbl="MODIFY"	accesskey="M"
DELETE	value="update"	command="DELETE"	lbl="DELETE"	accesskey="Z"
ENTRY	value="entry"	command="ENTRY"	lbl="ENTRY"	accesskey="E"
RESET	value="reset"	command="RESET"	lbl="ENTRY"	accesskey="R"

2.value

【TAG】forward したい J S P ファイル名を記述します(例:insert,copy,modify,delete など)。
J S P ファイル名は、標準で、insert,copy,modify,delete などと指定します。
実際には、各JSP画面(insert.jsp,copy.jsp,modify.jsp,delete.jsp)に
リクエストが転送されます。
この J S P ファイル名は、同一画面 I D 内のフォルダに属している必要があります。

バージョン タグ名 HTML&EXT BODY要素	内容
	形式サンプル
3.command	この ID は、JSP 画面そのものですので、大文字小文字は区別されます。 [TAG] 処理コマンドを登録します(初期値:大文字の value 属性値[INSERT,COPY,MODIFY,DELETE など])。 command 属性を指定しない場合は、このvalue 属性値が、コマンドになります。 value 属性に、insert,copy,modify,delete などと指定されていた場合は、それぞれ、INSERT,COPY,MODIFY,DELETE というコマンドになります。 コマンドは、大文字です。
4.gamenId	[TAG] gamenId 属性を登録します。 gamenId 属性は、別の画面にforward する場合に使用します。 実際は、forward ではなく、sendRedirect されます。
5.lbl	[TAG] ラベルリソースのラベルIDを指定します。 ラベルを変更するときに、lbl属性を使います。 ラベルID は、所定の language に基づく ResourceManager の getLabelData(id) を呼び出し、その結果のLabelInterfaceを使用します。 getMsglbl() で取り出せます。 ラベルIDとメッセージIDは同時に登録できません。 ラベルとメッセージは統一されました。
6.target	[TAG] サブミット先の文書を表示させるフレーム、またはウィンドウの名前を指定します。 サブミット先のフレーム名(ターゲット属性)を設定します。
7.keys	[TAG] ボタン専用のリクエストキーをCSV形式で複数指定します。 このサブミットボタンが押された場合のみ、転送されるリクエスト情報の キーを設定できます。カンマ区切りで複数指定できます。 vals 属性には、キーに対応する値を、設定してください。 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスの為です。
8.vals	[TAG] ボタン専用のリクエスト値をCSV形式で複数指定します。 キーに対応した値を、カンマ区切り文字で複数指定出来ます。 指定順序は、キーと同じにしておいて下さい。 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスの為です。
9.roles	[TAG] ロールをセットします。 ここで指定したカラムロールを元に、ユーザー毎のアクセス許可がチェックされます。 アクセス許可されないと、表示されません。 このロールを指定しない場合は、カラムリソースのロールが使用されます。
10.dbkeys	[TAG] commonForward の dbkeys にカラム指定を行います。 カラム指定は、CSV形式(カンマ区切り)で指定してください。 なお、引数は、自動的に受けるのではなく、commonForward タグに、 dbkeys="[@dbkeys]" の記述は必要です。
11.optionAttributes	[TAG] JavaScript などの HTML 基本タグ以外の属性を、そのままタグとして使用します。 JavaScript などの HTML 基本タグ以外の属性を、そのまま タグとして使用します。そのときに、() は、() に置き換えます。
12.columnWritable	[TAG] 書き込み可能カラム名を、カンマ区切りで与えます。 これは、書き込み不可カラム名の指定(noWritable)と同時にセットする ことは出来ません。 なお、カラム名の代わりに、“null”を指定すると、なにも指定しないこと になります。つまり、noWritable にすべてのカラムを指定すること 同じになります。(デフォルトなので、あまり意味はありません。) “*”を指定すると、すべてのカラムを(columnWritable)指定したことになります。
13.noWritable	[TAG] 書き込み不可カラム名を、カンマ区切りで与えます。 これは、書き込み可能カラム名の指定(columnWritable)と同時にセットする ことは出来ません。 なお、カラム名の代わりに、“null”を指定すると、なにも指定しないこと になります。つまり、columnWritable にすべてのカラムを指定すること 同じになります。 “*”を指定すると、すべてのカラムを(noWritable)指定したことになります。
14.caseKey	[TAG] このタグ自体を利用するかどうかの条件キーを指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。
15.caseVal	[TAG] このタグ自体を利用するかどうかの条件値を指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。
16.id	[HTML] 要素に対して固有の名前(id)をつける場合に設定します。 特別な使用方法として、id="FOCUS" とすることで、フィールド系要素に フォーカスを移動させます。これは、そのページ内で唯一の id 属性として使用ください。
17.lang	[HTML] 要素の内容と他の属性値の言語(lang.xml::lang)を指定します。 HTMLの言語属性に使われます。指定する値は、ISO 639で規定されている「言語コード」です。 [ja/en/zh/...]などのほかに、en-US:アメリカ英語、en-cockney:コックニー英語 など、 副言語を指定する方法も定められています。 ここでは、lang と xml::lang の両方に同じ値がセットされます。 タグの language 属性とは使用用途が異なります。lang セット
18.dir	[HTML] 文字表記の方向(dir)を指定します。 当該要素の書字方向を指定する属性です。 ltr で、左から右に、rtl で、右から左に並べます。
19.title	[HTML] 要素に対する補足的情報(title)を設定します。 title セットは、ボタンなどに適用すると、マウスオーバーによりこのメッセージが チップスのように表示されます。これを利用して、説明文を登録することが可能です。 ここに登録した文字列が、メッセージリソースに存在する場合は、そのメッセージを 存在しない場合は、そのままの値を設定します。
20.style	[HTML] この要素に対して適用させるスタイルシート(style)を設定します。 タグにstyle属性を設定します。これは、キー:値: のセットを複数記述できます。 通常は、class属性や、id属性で登録しておき、&lg:style type="text/css"> で 外部から指定する方がソースは読みやすくなります。
21.readonly	[TAG] その部品に対して変更が出来ないように(readonly)指定します(サーバーに送信される)。 INPUT/TEXTAREA 系に対して、指定可能です。 readonly="readonly", readonly="true" が指定された場合は、有効です。 false も指定値としては、有効です。(大文字小文字の区別も不要) それ以外の指定は、エラーとします。
22.disabled	[TAG] その部品に対して、選択や変更が出来ないように(disabled)指定します(サーバーに送信されない)。 BUTTON/INPUT/OPTGROUP/OPTION/SELECT/TEXTAREA 系に対して、指定可能です。 disabled="disabled", disabled="true" が指定された場合は、有効です。 false も指定値としては、有効です。(大文字小文字の区別も不要) それ以外の指定は、エラーとします。
23.tabindex	[HTML] タブの移動順(tabindex)を指定します(0 ~ 32767)。 Tabキーを押したときに要素が選択される順番を指定します。 値には、選択させたい順番を数値で記述します。
24.accesskey	[HTML] アクセスキー(alt+キーで直接指定)を割り当てます。 アクセスキーは、マウスの使えない環境でも、リンクにジャンプする、ボタンを押す、入力フォームに フォーカスを移すなどの操作を簡単に行うことができるように考慮されたものです。 Windows の「ファイル(F)」メニューについての、F と同じような動きをします。 [HTML] 要素に対して class 属性を設定します。 Javaの言語使用上、class で作成できないため、代用として

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
25.clazz	clazz を使用しています。 html で作成される属性は、 class で作成されます。
26.language	【TAG】 タグ内部で使用する言語コード[ja/en/zh/...]を指定します。 この言語コードに基づいて、表示のラベルをリソースから作成します。
27.onClick	【HTML】 JavaScriptのイベント onClick を設定します(例:onClick="renew(' query. jsp', ' QUERY');")。 onClick をセットします。 例えば、<og:column name="KBSAKU" onClick="renew(' query. jsp', ' QUERY');" /> のように指定することで、プルダウンメニューの絞り込み検索が可能になります。
28.onBlur	【HTML】 JavaScriptのイベント onBlur を設定します(例:onBlur="this.value=value.toUpperCase();")。 onBlur は、フォーカスが離れたときに発生するイベントです。
29.onFocus	【HTML】 JavaScriptのイベント onFocus を設定します。 onFocus は、フォーカスされたときに発生するイベントです。
30.ondblClick	【HTML】 JavaScriptのイベント ondblClick を設定します。 ondblClick は、マウスでダブルクリックされたときに発生するイベントです。
31.onMouseDown	【HTML】 JavaScriptのイベント onMouseDown を設定します。 onMouseDown は、マウスダウンされたときに発生するイベントです。
32.onMouseUp	【HTML】 JavaScriptのイベント onMouseUp を設定します。 onMouseUp は、マウスアップされたときに発生するイベントです。
33.onMouseMove	【HTML】 JavaScriptのイベント onMouseMove を設定します。 onMouseMove は、マウスが移動されたときに発生するイベントです。
34.onMouseOut	【HTML】 JavaScriptのイベント onMouseOut を設定します。 onMouseOut は、マウスが離れたときに発生するイベントです。
35.onMouseOver	【HTML】 JavaScriptのイベント onMouseOver を設定します。 onMouseOver は、マウスが重なったときに発生するイベントです。
36.img	【TAG】 画像ボタンを作る場合の、画像ファイルを指定します。 画像ボタン作成支援の属性です。 inputタグでtype=imageにした場合、IEではname, valueのセットが 次の画面に渡されない仕様になっているためエンジンのsubmitでは 利用できません。(どのボタンが押されたか分からない) そこで、typeはsubmitのままcssの背景画像としてここで指定した 画像を配置します。 内部的にはbackground: url(imgFile) left top no-repeat; をstyleタグに書く事と同じです。 高さ、幅は把握できないため、別途style属性でheight, widthを指定して下さい。 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。
37.debug	デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:switch	【TAG】 switch のマッチ判定用のキーを設定します。 switch のマッチ判定用のキーを設定します。
JSP	
●形式 : <og:switch key="..." /> <og:case match="A" /> ... </og:case> <og:case match="B" /> ... </og:case> <og:case match="C" /> ... </og:case> <og:case isDefault="true" /> ... </og:case> </og:switch> ●body : あり	
●Tag定義 : <og:switch key debug > ... Body ... </og:switch>	○ 【TAG】 switch のマッチ判定用のキーを設定します。(必須) 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)
●使用例	<pre><og:switch key="@PARAM" /> <og:case match="A" /> 処理A </og:case> <og:case match="B" /> 処理B </og:case> <og:case match="C" /> 処理C </og:case> <og:case isDefault="true" /> 処理X </og:case> </og:switch></pre> <p>・ switch の key に対して、case の match に指定された値が、マッチ (switch_key.match(case_match)) した場合に、case の BODY 部分が処理されます。 マッチしなければ、BODY部は、スキップされます。</p> <p>・ isDefault="true" の場合は、どれも マッチしなかった場合に、実行されます。</p> <p>・ Javaの switch-case 文は、最初に処理された case 以降を処理します。通常は、break を入れて 後続処理を実行されないようにしています。 この、switch-case タグは、caseタグの isBreak 属性で制御します。初期値が isBreak="true" に、なっているため、通常は、どれかの case が実行された段階で、switchの処理は、終了されます。 isBreak="false" にすると、switchから抜けずに、継続して case との match を実行します。 この場合、Java等と異なるのは、直後のcase文が実行されるのではなく、あくまで match 作業が 継続されるということです。つまり、複数の case で処理を行いたい場合は、isBreak="false" にすると同時に、match 条件もそれぞれで、マッチするように設定する必要があります。</p> <pre><og:switch key="@PARAM" /> <og:case match="[1]" isBreak="false" /> 処理A </og:case> <og:case match="[12]" isBreak="false" /> 処理B </og:case> <og:case match="[123]" isBreak="false" /> 処理C </og:case> <og:case isNull="true" /> 処理X </og:case> <og:case isDefault="true" /> 処理Y </og:case> </og:switch></pre> <p>・ 上記指定では、isBreak="false" が指定されているため、マッチした後も継続して判定処理が実施されます。</p> <p>・ 上記例で言うと、PARAM が "1" の場合、上記 3 つともにマッチします。</p> <p>・ isNull="true" は、switch の key が null の場合に成立します。(null とは、ゼロ文字列も含む)</p>
1.key	【TAG】 switch のマッチ判定用のキーを設定します。 switch のマッチ判定用のキーを設定します。
2.debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:sysInstall Tag ●形式: <og:sysInstall fileURL="..." insFile="..." /> ●body: なし ●Tag定義: <og:sysInstall insFile fileURL debug /> ●使用例 ・ Calc(ods) ファイルをPDFに変換 <og:sysInstall insFile="context.zip" />	【TAG】インストールファイルを指定します。 インストールファイルを指定します。 各コンテキストのインストール用アーカイブを指定する必要があります。 ○【TAG】インストールファイルを指定します。(必須) 【TAG】操作するファイルのディレクトリを指定します (初期値:FILE_URL[=filetemp]) 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)
1.insFile	【TAG】インストールファイルを指定します。 インストールファイルを指定します。 各コンテキストのインストール用アーカイブを指定する必要があります。
2.fileURL	【TAG】操作するファイルのディレクトリを指定します (初期値:FILE_URL[=filetemp])。 この属性で指定されるディレクトリのファイル进行操作します。 指定方法は、通常の fileURL 属性と同様に、先頭が、'/' (UNIX) または、2 文字目が、 ":" (Windows) の場合は、指定のURLそのままのディレクトリに、そうでない場合は、 (初期値:システム定数のFILE_URL[=@og.value org.opengion.hayabusa.common.SystemData#FILE_URL])。
3.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:tab JSP	<p>【TAG】ラベルリソースのラベルIDを指定します。 ラベルを変更するときに、lbl属性を使います。</p> <p>ラベルID は、所定の language に基づく ResourceManager の getLabelData(id) を呼び出し、その結果のLabelInterfaceを使います。 getMsgLbl() で取り出せます。 ラベルIDとメッセージIDは同時には登録できません。 ラベルとメッセージは統一されました。</p>
●形式 : <og:tabTable ... >	<pre> <og:tab name="page1" lbl="page 1" > <jsp:directive.include file="page1.jsp" /> </og:tab> <og:tab name="page2" lbl="page 2" > <jsp:directive.include file="page2.jsp" /> </og:tab> </og:tabTable> </pre>
●body : あり	
●Tag定義 :	
<og:tab	【TAG】ラベルリソースのラベルIDを指定します
lbl	【廃止】要素に対して固有の名前(id)をつける場合に設定します
name	【廃止】要素に対して固有の名前(id)をつける場合に設定します
term	【廃止】処理する条件を指定します(初期値:null)
termList	【廃止】処理する条件を含むような文字列を指定します
delTerm	【廃止】処理しないタブを選択する条件を指定します(初期値:null)
delTermList	【廃止】処理しない条件を含むような文字列を指定します
style	【TAG】タブに与える 初期 style 属性を指定します
keys	【廃止】タブが選択状態(OPEN選択タブ時)に使用するキーをCSV形式で複数指定します
vals	【廃止】タブが選択状態(OPEN選択タブ時)に使用する値をCSV形式で複数指定します
debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)
> ... Body ...	
</og:tab>	
●使用例	
<og:tabTable>	
<og:tab lbl="page 1">	
<table summary="layout">	
<tr>	
<og:column name="SYSTEM_ID" must="true" />	
<og:column name="COLUMN_NAME" />	
<og:column name="RENDERER" />	
</tr>	
</table>	
</og:tab>	
<og:tab lbl="page 2">	
<table summary="layout">	
<tr>	
<og:column name="DBTYPE" />	
<og:column name="NAME_JA" />	
<og:column name="EDITOR" />	
</tr>	
</table>	
</og:tab>	
</og:tabTable>	
1.lbl	<p>【TAG】ラベルリソースのラベルIDを指定します。 ラベルを変更するときに、lbl属性を使います。</p> <p>ラベルID は、所定の language に基づく ResourceManager の getLabelData(id) を呼び出し、その結果のLabelInterfaceを使います。 getMsgLbl() で取り出せます。 ラベルIDとメッセージIDは同時には登録できません。 ラベルとメッセージは統一されました。</p>
2.name	<p>【廃止】要素に対して固有の名前(id)をつける場合に設定します。 名前セット</p>
3.term	<p>【廃止】処理する条件を指定します(初期値:null)。 termListで与えられた文字列に、term文字列(大文字/小文字の区別あり)が 含まれていれば、OPEN選択タブとして処理します。 OPEN選択タブでないタブは、初期値OPENにならないだけで、タブそのものは表示されます。 OPEN選択とDELETE選択が競合した場合は、DELETE選択が優先されます。 初期値は、 null です。</p>
4.termList	<p>【廃止】処理する条件を含むような文字列を指定します。 termListで与えられた文字列に、term文字列(大文字/小文字の区別あり)が 含まれていれば、OPEN選択タブとして処理します。 例えば、“A”、“B”、“C” という文字列が、term で指定された 場合に処理するようにしたい場合は、“A B C” をセットします。 初期値は、 null です。 判定は、文字列を含むかどうかなので、1文字のみの場合は、“ ”区切りにする 必要はありません。 OPEN選択とDELETE選択が競合した場合は、DELETE選択が優先されます。</p>
5.delTerm	<p>【廃止】処理しないタブを選択する条件を指定します(初期値:null)。 delTermListで与えられた文字列に、delTerm文字列(大文字/小文字の区別あり)が 含まれていれば、DELETE選択タブとして処理します。 DELETE選択タブは、タブそのものが表示されません。 ただし、タブのselectIndex は、DELETEされたタブも含めて、カウントされますので、 JSPでの設定時の順番がインデックス番号になります。 OPEN選択とDELETE選択が競合した場合は、DELETE選択が優先されます。 初期値は、 null です。</p>
6.delTermList	<p>【廃止】処理しない条件を含むような文字列を指定します。 delTermListで与えられた文字列に、delTerm文字列(大文字/小文字の区別あり)が 含まれていれば、DELETE選択タブとして処理します。 例えば、“A”、“B”、“C” という文字列が、delTerm で指定された 場合に処理しないようにしたい場合は、“A B C” をセットします。 初期値は、 null です。 判定は、文字列を含むかどうかなので、1文字のみの場合は、“ ”区切りにする 必要はありません。 OPEN選択とDELETE選択が競合した場合は、DELETE選択が優先されます。</p>
7.style	<p>【TAG】タブに与える 初期 style 属性を指定します。 ts:tab 本体では、初期選択時のスタイルシートを、defaultStyle と selectedStyle で与える必要があります。これは、id 属性を設定して、 外部でスタイルシートを定義する形式で指定できません。 ここで指定した style 属性 は、tabTableTag で与える style 属性 より優先度は</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
8.keys	<p>高くなります。</p> <p>ここでの style は、タブの OPEN 選択には関係ありません。</p> <p>【廃止】タブが選択状態 (OPEN 選択タブ時) に使用するキーを CSV 形式で複数指定します。</p> <p>タブの BODY 内部でのみ有効な scope="request" 変数を定義できます。</p> <p>条件は、term、termList による OPEN 選択タブ時にのみ指定のタブの BODY 部内のみ有効になります。</p> <p>タブの前に設定されている scope="request" 変数 は退避され、BODY 部が終了した時点で、元の変数に値が戻されます。</p> <p>分解方法は、CSV 変数を先に分解してから、getRequestParameter で値を取得します。</p> <p>こうしないとデータ自身にカンマを持っている場合に分解をミスる為です。</p>
9.vals	<p>【廃止】タブが選択状態 (OPEN 選択タブ時) に使用する値を CSV 形式で複数指定します。</p> <p>タブの BODY 内部でのみ有効な scope="request" 変数を定義できます。</p> <p>条件は、term、termList による OPEN 選択タブ時にのみ指定のタブの BODY 部内のみ有効になります。</p> <p>タブの前に設定されている scope="request" 変数 は退避され、BODY 部が終了した時点で、元の変数に値が戻されます。</p> <p>分解方法は、CSV 変数を先に分解してから、getRequestParameter で値を取得します。</p> <p>こうしないとデータ自身にカンマを持っている場合に分解をミスる為です。</p>
10.debug	<p>【TAG】デバッグ情報を出力するかどうか [true/false] を指定します (初期値: false)。</p> <p>デバッグ情報を [true: 出力する/false: しない] を指定します。</p> <p>出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTML5 BODY要素	内容
形式サンプル	
5.5.4.2 og:tabLink	<p>【TAG】タブの一覧をどこから取得するかを指定します(初期値: AUTO)。 タブの一覧をどこから取得するかを指定します。 現状の実装では、クエリを発行して一覧を生成する「DB」と、子タグであるtabListタグを列挙してタブを定義する「TAG」が実装されています。</p>
JSP	<p>また、「AUTO」と指定した場合は、Body部分の内容に応じて自動的に判定されます。 初期値は、「AUTO」です。</p>
●形式: <og:tabLink href="..." />	
●body: あり	
●Tag定義:	
<og:tabLink	
listType	【TAG】タブの一覧をどこから取得するかを指定します(初期値: AUTO)
href	【TAG】リンク先のJSPを指定します(初期値: result.jsp)
target	【TAG】リンクのターゲットを指定します(初期値: RESULT)
openTab	【TAG】リンク表示にタブリンクを自動で開くかどうか[true/false]を指定します(初期値: true[=開く])
openTabName	【TAG】最初に関くタブリンクの名前を指定します
constKeys	【TAG】次画面に渡す定数パラメーターのキーを指定します
constVals	【TAG】次画面に渡す定数パラメーターの値を指定します
listCount	【TAG】1行辺りに表示するタブの数を指定します(初期値: 10)
selClass	【TAG】選択タブのクラスを指定します(初期値: selTab)
unselClass	【TAG】非選択タブのクラスを指定します(初期値: unselTab)
orientation	【TAG】タブの方向、横型(Horizontal)か縦型(Vertical)を指定します(初期値: 横型)
width	【TAG】タブリンクの幅を %、px、または "auto" で指定します
height	【TAG】タブの高さを %、px、または "auto" で指定します
debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値: false)
> ... Body ...	
</og:tabLink>	
●使用例	
①DBからタブリストを取得する場合	<p>Body部分に記述されたクエリよりタブ一覧を生成します。</p>
	<pre> <og:tabLink listType = "DB" タブの一覧をどこから取得するか href = "result.jsp" リンク先のJSP target = "RESULT" リンクターゲット openTab = "[true/false]" タブ表示後にタブを自動で開く openTabName = "[@PN]" 自動で開くタブの名前 constKeys = "KEY1" 次画面に固定で渡すキー一覧 constVals = "[@VAL1]" 次画面に固定で渡す値一覧 listCount = "10" 1行辺りに表示するタブの数 selClass = "selTab" 選択タブのクラス unselClass = "unselTab" 非選択タブのクラス width = "100px" タブリンクの幅 height = "50px" タブリンクの高さ > select PN, HINM, 'tabClass', 'query.jsp', 'ADMIN', 'false' from XX01 where PN = '[@PN]' order by PN </og:tabLink> </pre>
②tabListタグからタブリストを生成する場合	<p>tabListタグよりタブ一覧を生成します。</p>
	<pre> <og:tabLink listType = "DB" タブの一覧をどこから取得するか href = "result.jsp" リンク先のJSP target = "RESULT" リンクターゲット openTab = "[true/false]" タブ表示後にタブを自動で開く openTabName = "[@PN]" 自動で開くタブの名前 constKeys = "KEY1" 次画面に固定で渡すキー一覧 constVals = "[@VAL1]" 次画面に固定で渡す値一覧 listCount = "10" 1行辺りに表示するタブの数 selClass = "selTab" 選択タブのクラス unselClass = "unselTab" 非選択タブのクラス width = "100px" タブリンクの幅 height = "50px" タブリンクの高さ > <og:tabList name="TAB1" href="result1.jsp" keys="PN, CDK" vals="ABC, V" /> <og:tabList name="TAB2" href="result2.jsp" keys="PN, CDK" vals="BCD, W" /> <og:tabList name="TAB3" href="result3.jsp" keys="PN, CDK" vals="CDE, X" /> </og:tabLink> </pre>
1.listType	<p>【TAG】タブの一覧をどこから取得するかを指定します(初期値: AUTO)。 タブの一覧をどこから取得するかを指定します。 現状の実装では、クエリを発行して一覧を生成する「DB」と、子タグであるtabListタグを列挙してタブを定義する「TAG」が実装されています。</p> <p>また、「AUTO」と指定した場合は、Body部分の内容に応じて自動的に判定されます。 初期値は、「AUTO」です。</p>
2.href	<p>【TAG】リンク先のJSPを指定します(初期値: result.jsp)。 リンク先のJSPを指定します。 このタブリンクは、あくまで「タブの形をしたリンク」なので、target属性と合わせてセットする必要があります。 初期値は、「result.jsp」です。</p>
3.target	<p>【TAG】リンクのターゲットを指定します(初期値: RESULT)。 リンクのターゲットを指定します。 このタブリンクは、あくまで「タブの形をしたリンク」なので、target属性を設定し、別のフレームに実画面を表示するようにします。 初期値は、「RESULT」です。</p>
4.openTab	<p>【TAG】リンク表示にタブリンクを自動で開くかどうか[true/false]を指定します(初期値: true[=開く])。 リンク表示にタブリンクを自動で開くかを指定します。 openTabName属性が指定されていない場合、自動で開くタブは「1番目に表示されたタブリンク」です。 指定されている場合は、その名前を持つ「1番目」のタブが自動で開かれます。 タブが選択不可能な状態の場合は、「1番目」の条件から除外されます。 初期値は、「true(開く)」です。</p>
5.openTabName	<p>【TAG】最初に関くタブリンクの名前を指定します。 最初に関くタブリンクのキーを指定します。</p>
6.constKeys	<p>【TAG】次画面に渡す定数パラメーターのキーを指定します。 次画面に渡す定数パラメーターのキーを指定します。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
7.constVals	キーはカンマ区切りで複数指定が可能です。 パラメーターの値は、constVals属性の数と一致している必要があります。 【TAG】次画面に渡す定数パラメーターの値を指定します。 次画面に渡す定数パラメーターの値を指定します。 値はカンマ区切りで複数指定が可能です。 パラメーターの値は、constKeys属性の数と一致している必要があります。 【TAG】1行辺りに表示するタブの数を指定します(初期値:10)。 1行辺りに表示するタブの数を指定します。 1行辺りのタブの数がこの設定を超えると、自動的に折り返します。 また、折り返し毎に、左に10pxのマージンを設けます。 初期値は、10です。 この属性は、orientationがHorizontal(水平方向)の場合のみ有効です。
9.selClass	【TAG】選択タブのクラスを指定します(初期値:selTab)。 タブが選択されている状態にある場合の、タブ部分のクラス名を指定します。 このクラス名を変更する場合は、そのクラスをcustom/custom.css等で再定義して下さい。 初期値は、selTabです。
10.unselClass	【TAG】非選択タブのクラスを指定します(初期値:unselTab)。 タブが選択されていない状態にある場合の、タブ部分のクラス名を指定します。 このクラス名を変更する場合は、そのクラスをcustom/custom.css等で再定義して下さい。 初期値は、unselTabです。
11.orientation	【TAG】タブの方向、横型(Horizontal)か縦型(Vertical)を指定します(初期値:横型)。 タブは、上にタブが並ぶ横型と左にタブが並ぶ縦型があります。 この属性では、横型は、Horizontal、縦型は、Vertical を指定します。 指定は、文字列の最初の一文字を見ているだけです。HかVでも構いません。 縦型(Vertical)にした場合、各タブ要素は、フレームサイズの幅に合わせて 最大で表示されます。幅を固定する場合は、width属性を指定して下さい。 初期値は、横型(Horizontal) です。
12.width	【TAG】タブリンクの幅を %、px、または“auto”で指定します。 初期値は、“auto”(自動設定) です。 autoの場合、横型表示では、文字の幅に合わせて自動的に調整され、 縦型表示の場合は、フレームサイズに合わせて拡大して表示されます。
13.height	【TAG】タブの高さを、%、px、または“auto”で指定します。 タブの高さを、%、px、または“auto”で指定します 初期値は、“auto”(自動設定) です。
14.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:tabList Tag ●形式 : <og:tabLink href="..." > <og:tabList name="TAB1" ... /> ... </og:tabLink > ●body : なし ●Tag定義 : <og:tabList name lbl href term termList delTerm delTermList unselClass keys vals roles debug /> ●使用例 tabListタグからタブリストを生成する場合 <og:tabLink listType href target openTab openTabName constKeys constVals listCount selClass unselClass width height > <og:tabList name="TAB1" href="result1.jsp" keys="PN, CDK" vals="ABC, V" /> <og:tabList name="TAB2" href="result2.jsp" keys="PN, CDK" vals="BCD, W" /> <og:tabList name="TAB3" href="result3.jsp" keys="PN, CDK" vals="CDE, X" /> </og:tabLink>	【TAG】要素に対して固有の名前をつけます。 要素に対して固有の名前をつけます。 ここで設定された名称は、自動でタブを表示するためのopenTabName属性判定にも使用されます。 ○【TAG】要素に対して固有の名前をつけます。(必須) 【TAG】ラベルリソースのラベルIDを指定します 【TAG】リンク先のJSPを指定します(初期値:result.jsp) 【TAG】処理する条件を指定します(初期値:null) 【TAG】処理する条件を含むような文字列を指定します 【TAG】処理しないタブを選択する条件を指定します(初期値:null) 【TAG】処理しない条件を含むような文字列を指定します 【TAG】非選択タブのクラスを指定します(初期値:unselTab) 【TAG】リンク先のJSPに引数として渡すキーをCSV形式で指定します 【TAG】リンク先のJSPに引数として渡す値をCSV形式で指定します 【TAG】ロールをセットします 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false) タブの一覧をどこから取得するか リンク先のJSP リンクターゲット タブ表示後にタブを自動で開く 自動で開くタブの名前 次画面に固定で渡すキー一覧 次画面に固定で渡す値一覧 1行辺りに表示するタブの数 選択タブのクラス 非選択タブのクラス タブリンクの幅 タブリンクの高さ
1.name	【TAG】要素に対して固有の名前をつけます。 要素に対して固有の名前をつけます。 ここで設定された名称は、自動でタブを表示するためのopenTabName属性判定にも使用されます。
2.lbl	【TAG】ラベルリソースのラベルIDを指定します。 ラベルを変更するときに、lbl属性を使います。 ラベルID は、所定の language に基づく ResourceManager の getLabelData(id) を呼び出し、その結果のLabelInterfaceを使用します。 getMsgLbl() で取り出せます。 ラベルIDとメッセージIDは同時には登録できません。 ラベルとメッセージは統一されました。
3.href	【TAG】リンク先のJSPを指定します(初期値:result.jsp)。 リンク先のJSPを指定します。 ここで指定しない場合は、tagLinkタグの値が適用されます。
4.term	【TAG】処理する条件を指定します(初期値:null)。 termListで与えられた文字列に、term文字列(大文字/小文字の区別あり)が 含まれていれば、OPEN選択タブとして処理します。 OPEN選択タブでないタブは、初期値OPENにならないだけで、タブそのものは表示されます。 OPEN選択とDELETE選択が競合した場合は、DELETE選択が優先されます。 初期値は、 null です。
5.termList	【TAG】処理する条件を含むような文字列を指定します。 termListで与えられた文字列に、term文字列(大文字/小文字の区別あり)が 含まれていれば、OPEN選択タブとして処理します。 例えば、“A”、“B”、“C” という文字列が、term で指定された 場合に処理するようにしたい場合は、“A B C” をセットします。 初期値は、 null です。 判定は、文字列を含むかどうかなので、1文字のみの場合は、“ ”区切りにする 必要はありません。 OPEN選択とDELETE選択が競合した場合は、DELETE選択が優先されます。
6.delTerm	【TAG】処理しないタブを選択する条件を指定します(初期値:null)。 delTermListで与えられた文字列に、delTerm文字列(大文字/小文字の区別あり)が 含まれていれば、DELETE選択タブとして処理します。 DELETE選択タブは、タブそのものが表示されません。 ただし、タブのselectIndex は、DELETEされたタブも含めて、カウントされますので、 JSPでの設定時の順番がインデックス番号になります。 OPEN選択とDELETE選択が競合した場合は、DELETE選択が優先されます。 初期値は、 null です。
7.delTermList	【TAG】処理しない条件を含むような文字列を指定します。 delTermListで与えられた文字列に、delTerm文字列(大文字/小文字の区別あり)が 含まれていれば、DELETE選択タブとして処理します。 例えば、“A”、“B”、“C” という文字列が、delTerm で指定された 場合に処理しないようにしたい場合は、“A B C” をセットします。 初期値は、 null です。 判定は、文字列を含むかどうかなので、1文字のみの場合は、“ ”区切りにする 必要はありません。 OPEN選択とDELETE選択が競合した場合は、DELETE選択が優先されます。
8.unselClass	【TAG】非選択タブのクラスを指定します(初期値:unselTab)。 タブが選択されていない状態にある場合の、タブ部分のクラス名を指定します。 このクラス名を変更する場合は、そのクラスをcustom/custom.css等で再定義して下さい。 tabListタグで指定された値は、tabLinkタグで指定されたものより優先されます。 初期値は、unselTabです。
9.keys	【TAG】リンク先のJSPに引数として渡すキーをCSV形式で指定します。 リンク先のJSPに引数として渡すキーをCSV形式で指定します。
10.vals	【TAG】リンク先のJSPに引数として渡す値をCSV形式で指定します。 リンク先のJSPに引数として渡す値をCSV形式で指定します。
11.roles	【TAG】ロールをセットします。 ここで指定したカラムロールを元に、ユーザー毎のアクセス許可がチェックされます。 アクセス許可されないと、表示されません。 このロールを指定しない場合は、カラムリソースのロールが使用されます。

バージョン タグ名 HTMLTEXT BODY要素	内容
12.debug	<div>形式サンプル</div> <div>【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。</div>

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:tabTable JSP	<p>【廃止】タブの方向、横型(horizontal)か縦型(vertical)を指定します(初期値:横型)。タブは、上にタブが並ぶ横型と左にタブが並ぶ縦型があります。この属性では、横型は、horizontal、縦型は、vertical を指定します。指定は、文字列の最初の一文字を見ているだけです、HかVでも構いません。初期値は、横型(horizontal) です。</p>
●形式: <og:tabTable	<pre> orientation = "[horizontal/vertical]" height = "[100px / 100%]" width = "[100px / 100%]" selectedIndex = "[0..]" </pre> <p>タブの方向(横/縦) テーブルの高さ(px OR %) テーブルの幅(px OR %) 初期表示するタブ番号</p> <pre> > <og:tab lbl="..." > ... </og:tab > タブそのもの <og:tab lbl="..." > ... </og:tab > タブそのもの </og:tabTable > </pre> <p>●body: あり(tab)</p>
●Tag定義:	<pre> <og:tabTable orientation height width selectedIndex style debug > ... Body ... </og:tabTable> </pre> <p>【廃止】タブの方向、横型(horizontal)か縦型(vertical)を指定します(初期値:横型) 【廃止】タブの高さを、%、px、または“auto”で指定します 【廃止】タブの幅を %、px、または“auto”で指定します 【廃止】初期表示するページ番号を指定します(初期値:0) 【廃止】初期表示時のタブに与える style 属性を指定します 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)</p>
●使用例	<p>横型(horizontal)の場合。横型が初期値のため、無指定で使用できます。</p> <pre> <og:tabTable> <og:tab lbl="page 1"><jsp:directive.include file="page1.jsp" /></og:tab> <og:tab lbl="page 2"><jsp:directive.include file="page2.jsp" /></og:tab> <og:tab lbl="page 3"><jsp:directive.include file="page3.jsp" /></og:tab> </og:tabTable> </pre> <p>縦型(vertical)の場合。テーブルの高さを指定する必要があります。(px OR %)</p> <pre> <og:tabTable orientation="vertical" height="100px"> <og:tab lbl="page 1"><jsp:directive.include file="page1.jsp" /></og:tab> <og:tab lbl="page 2"><jsp:directive.include file="page2.jsp" /></og:tab> <og:tab lbl="page 3"><jsp:directive.include file="page3.jsp" /></og:tab> </og:tabTable> </pre>
1.orientation	<p>【廃止】タブの方向、横型(horizontal)か縦型(vertical)を指定します(初期値:横型)。タブは、上にタブが並ぶ横型と左にタブが並ぶ縦型があります。この属性では、横型は、horizontal、縦型は、vertical を指定します。指定は、文字列の最初の一文字を見ているだけです、HかVでも構いません。初期値は、横型(horizontal) です。</p>
2.height	<p>【廃止】タブの高さを、%、px、または“auto”で指定します。縦型(orientation="vertical")の初期値は、“auto”です。横型(orientation="horizontal")の初期値は、“200px”です。横型の場合は、“auto”に設定すると、高さが“0”になってしまいます。必ず、なにかの値(px)で指定する必要があります。縦型で“auto”に設定すると、各タブ毎に中の記述情報によって、タブの大きさが替わります。タブを切り替えた時に、違和感がない様にするには、高さを固定(px 指定)するとよいです。</p>
3.width	<p>【廃止】タブの幅を %、px、または“auto”で指定します。縦型(orientation="vertical")の初期値は、“auto”です。横型(orientation="horizontal")の初期値は、“100%”です。※ 縦型の場合、幅に px で数字を設定しても、有効に作用しません。</p>
4.selectedIndex	<p>【廃止】初期表示するページ番号を指定します(初期値:0)。タブテーブルには、複数のタブを含みます。初期表示時にどのタブを表示するかを指定します。ページ番号は、0から始まる数字です。初期値は、0です。</p>
5.style	<p>【廃止】初期表示時のタブに与える style 属性を指定します。ts:tab 本体では、初期選択時のスタイルシートを、defaultStyle と selectedStyle で与える必要があります。これは、id 属性を設定して、外部でスタイルシートを定義する形式で指定できません。ここで指定した style 属性 は、個別の tabTag に与える style 属性 より優先度は低くなります。</p>
6.debug	<p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。デバッグ情報を [true:出力する/false:しない]を指定します。出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容																
形式サンプル																	
5.5.4.2 og:tableExist	【廃止】 コマンド(ENTRY) をセットします。 コマンドは、HTMLから (get/post) 指定されますので、CMD_xxx で設定される フィールド定数値のいつれかを、指定できます。																
Tag																	
●形式：																	
・<og:tableExist																	
command	=	"{@command}"															
names	=	"[...]"															
from	=	"..." 必須															
where	=	"..." 必須															
exist	=	"[auto true false one notuse]" 必須															
errRemove	=	"[true false]"															
>																	
●body：なし																	
●Tag定義：																	
<og:tableExist																	
command	【廃止】 コマンド(ENTRY) をセットします																
scope	【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します (初期値:session)																
names	【廃止】 引数にセットすべき データの名称(カラム名)をCSV形式で複数指定します																
from	○ 【廃止】 チェックするデータベース名(from 句)を指定します。(必須)																
where	○ 【廃止】 チェックする検索条件(where句)を指定します。(必須)																
exist	【廃止】 データベースのチェック方法(auto/true/false/one/notuse)を指定します (初期値:auto:自動)																
tableId	【廃止】 (通常は使いません)結果をDBTableModelに書き込んで、sessionに登録するときのキーを指定します																
dbid	【廃止】 (通常は使いません)Queryオブジェクトを作成する時のDB接続IDを指定します																
errRemove	【廃止】 エラー時の選択行を取り除いて継続処理を行うかどうか[true/false]を指定します (初期値:false)																
debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します (初期値:false)																
>																	
●使用例																	
・<og:tableExist																	
command	=	"{@command}"															
names	=	"USERID, SYSTEM_ID"															
from	=	"GE10"															
where	=	"USERID=? AND SYSTEM_ID=?"															
exist	=	"true" >															
・where 条件の ? 文字に、names で指定したカラム名の値が、DBTableModelより取得されます。 値の取得は、先に選択された行のみについて、実行されます。																	
・exist 属性の値に応じて、チェック方法が異なります。 auto, true, false, one, notuse が指定できます。																	
・テーブルは、1 つのみ指定できます。複数指定や、UNION で結合する場合は、ビュー等を作成して対応してください。																	
・<og:tableExist																	
command	=	"{@command}"															
from	=	"GE10"															
where	=	"USERID=[USERID] AND SYSTEM_ID=[SYSTEM_ID]" >															
・where 条件の [カラム名] 文字に、DBTableModelより値がセットされます。																	
・exist は、初期値(auto)になります。内部のA, C, Dに応じて自動判別します。																	
1.command	【廃止】 コマンド(ENTRY) をセットします。 コマンドは、HTMLから (get/post) 指定されますので、CMD_xxx で設定される フィールド定数値のいつれかを、指定できます。																
	【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します (初期値:session)。 "request", "page", "session", "applicaton" が指定できます。 JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、主にテーブルモデルを管理するスコープを指します。																
2.scope	<table><tr><th>スコープ</th><th>変数の有効範囲</th><th></th></tr><tr><td>page</td><td>JSPページ内</td><td>そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会</td></tr><tr><td>request</td><td>HTTPリクエスト</td><td>リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やボ</td></tr><tr><td>session</td><td>HTTPセッション</td><td>初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。 内部的には、同じキーワー</td></tr><tr><td>application</td><td>Webアプリケーション</td><td>ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。</td></tr></table>		スコープ	変数の有効範囲		page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会	request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やボ	session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。 内部的には、同じキーワー	application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。
スコープ	変数の有効範囲																
page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会															
request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やボ															
session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。 内部的には、同じキーワー															
application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。															
3.names	【廃止】 引数にセットすべき データの名称(カラム名)をCSV形式で複数指定します。 複数ある場合は、カンマ区切り文字で渡します。 引数をnames ではなく、[カラム名]形式で直接指定するほうが、SQL文が判りやすくなります。																
4.from	【廃止】 チェックするデータベース名(from 句)を指定します。 from 句 に指定するデータベース名です。																
5.where	【廃止】 チェックする検索条件(where句)を指定します。 where 区 に指定する検索条件です。? の部分に、names 属性で指定した カラムのデータが、DBTableModelより取り出されて適用されます。 または、[カラム名]形式で、直接指定することもできます。その場合は、 name 属性は指定する必要がありません。 [カラム名]の前後に、() シングルコーテーションは、不要です。																
6.exist	【廃止】 データベースのチェック方法(auto/true/false/one/notuse)を指定します (初期値:auto:自動)。 exist 属性に指定された、 「true:存在する」、「false:存在しない」、「one:ひとつ以下」、 の値は、いずれの場合も、成立時は、正常とみなします。 「auto:自動」は、DBTableModelのmodifyType(A, C, D)に応じて、チェックします。 A, C, D は、entryタグにコマンドを渡してデータを作成したときに、内部で作成されます。 notuse は、 チェックを行いません。これは、 このタグを共有使用する場合に、 外部で チェックを行うかどうかを指定できるようにするために使用します。 (「true:存在する」には、データが存在した場合に、OKで、 なければエラーです。) 初期値は、「auto:自動」です。																
7.tableId	【廃止】 (通常は使いません)結果をDBTableModelに書き込んで、sessionに登録するときのキーを指定します。 初期値は、HybsSystem.TBL_MDL_KEY です。																
8.dbid	【廃止】 (通常は使いません)Queryオブジェクトを作成する時のDB接続IDを指定します。 Queryオブジェクトを作成する時のDB接続IDを指定します。																
9.errRemove	【廃止】 エラー時の選択行を取り除いて継続処理を行うかどうか[true/false]を指定します (初期値:false)。 exist 属性に指定された、 「true:存在する」、「false:存在しない」、「one:ひとつ以下」、 に対して、エラーが発生した選択行番号を、取り除いて以下の処理を継続するかどうかを 指定します。 true に設定した場合は、エラーデータを削除し、継続処理を行うことができます。 false の場合は、エラーデータを表示して、継続処理を停止します。 初期値は、「false:エラー時停止」です。																
10.debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します (初期値:false)。 デバッグ情報を 「true:出力する/false:しない」を指定します。																

バージョン タグ名 HTMLTEXT BODY要素	内容
	出力形式自体は、個々のタグによって異なります。 <div>形式サンプル</div>

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

5.5.4.2
og:tableFilter

【TAG】 データベース処理を実行するクラスパスを指定します。
ここで指定するクラスIDは、システムリソース にて TableFilter のサブクラス(インターフェース継承)として指定する必要があります。

JSP

クラス自身は、org.opengion.hayabusa.db.TableFilter インターフェースを継承している必要があります。

●形式 : <og:tableFilter classId="..." />
●body : あり

●Tag定義 :
<og:tableFilter
classId
tableId
modifyType
keys
vals
selectedAll
scope
dbid
debug
>
... Body ...
</og:tableFilter>

○ 【TAG】 データベース処理を実行するクラスパスを指定します。(必須)
【TAG】 (通常は使いません)DBTableModel sessionに登録されているキーを指定します
【TAG】 データ処理の方法(A:追加 C:更新 D:削除)を指定します
【TAG】 リンク先に渡すキーを指定します
【TAG】 names属性に対応する値をCSV形式で複数指定します
【TAG】 データを全件選択済みとして処理するかどうか[true/false]を指定します(初期値:false)
【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)
【TAG】 (通常は使いません)Queryオブジェクトを作成する時のDB接続IDを指定します
【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)

●使用例
・ 引数/プロシジャーを直接書く場合
【entry.jsp】
<og:tableFilter
classId
tableId
keys
vals
selectedAll
modifyType
>
</>

= "WL_LOGICSET" :TableFilter のサブクラス(実行クラス)
= "WL0000" :登録元のDBTableModelのsession/request変数内の取得キー
= "AA, BB, CC" :実行クラスへの引数のキー
= "[&64;AA], [&64;BB], [&64;CC]" :実行クラスへの引数の値
= "false/true" :処理対象の行を全行選択するかどうか(初期値:false)
= "A/C/D" :処理の方法(A:追加 C:更新 D:削除)を指定します。初期値は自動です。

1.classId

【TAG】 データベース処理を実行するクラスパスを指定します。
ここで指定するクラスIDは、システムリソース にて TableFilter のサブクラス(インターフェース継承)として指定する必要があります。

クラス自身は、org.opengion.hayabusa.db.TableFilter インターフェースを継承している必要があります。

2.tableId

【TAG】 (通常は使いません)DBTableModel sessionに登録されているキーを指定します。
初期値は、HybsSystem.TBL_MD_L_KEY です。

3.modifyType

【TAG】 データ処理の方法(A:追加 C:更新 D:削除)を指定します。
通常は、DBTableModel に自動設定されている modifyType を元に、データ処理方法を
選別します。(A:追加 C:更新 D:削除)
この場合、行単位で modifyType の値を取得して判別する必要がありますが、一般には
処理対象は、全件おなじ modifyType である可能性が高いです。
また、selectedAll などで強制的に全件処理対象とする場合は、modifyType に値が
設定されていません。その様な場合に外部より modifyType を指定します。
初期値は、自動判定 です。

4.keys

【TAG】 リンク先に渡すキーを指定します。
戻る時に、検索時のキャッシュに指定した引数以外に指定したり、別の値に置き換えたり
する場合のキーを設定できます。カンマ区切りで複数指定できます。
vals 属性には、キーに対応する値を、設定してください。
分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。
こうしないとデータ自身にカンマを持っている場合に分解をミスの為です。

5.vals

【TAG】 names属性に対応する値をCSV形式で複数指定します。
キーに設定した値を、カンマ区切り文字で複数して出来ます。
指定順序は、キーと同じにしておいて下さい。
分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。
こうしないとデータ自身にカンマを持っている場合に分解をミスの為です。

6.selectedAll

【TAG】 データを全件選択済みとして処理するかどうか[true/false]を指定します(初期値:false)。
全てのデータを選択済みデータとして扱って処理します。
全件処理する場合に、指定します。(true/false)
デフォルト false です。

7.scope

【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)。
"request", "page", "session", "applicaton" が指定できます。
JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、
主にテーブルモデルを管理するスコープを指します。

スコープ	変数の有効範囲	
page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使いう機会
request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ
session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。内部的には、同じキーワー
application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。

8.dbid

【TAG】 (通常は使いません)Queryオブジェクトを作成する時のDB接続IDを指定します。
Queryオブジェクトを作成する時のDB接続IDを指定します。
これは、システムリソースで、DEFAULT_DB_URL 等で指定している データベース接続先
情報に、XX_DB_URL を定義することで、dbid="XX" とすると、この 接続先を使用して
データベースにアクセスできます。

9.debug

【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。
デバッグ情報を [true:出力する/false:しない]を指定します。
出力形式自体は、個々のタグによって異なります。

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

5.5.4.2
og:tableMerge

【TAG】 コマンド(NEW, RENEW)をセットします(初期値:NEW)。
コマンドは、HTMLから(get/post)指定されますので、CMD_xxx で設定される
フィールド定数値のいずれかを、指定できます。
初期値は NEW です。

Tag

●形式：
・<og:tableMerge
 action
 tableId
 scope
 masterTableId
 masterScope
 slaveTableId
 slaveScope
 masterKeys
 slaveKeys
 diffKeys
 unionCfms
 modifyCfms
 noSideEffect
 useDiffData
 useCheckOnly
 groupAddCfms
 display
/>
●body：なし
●Tag定義：
 <og:tableMerge
 command
 action
 tableId
 scope
 masterTableId
 masterScope
 slaveTableId
 slaveScope
 masterKeys
 slaveKeys
 diffKeys
 unionCfms
 modifyCfms
 groupAddCfms
 noSideEffect
 useDiffData
 useCheckOnly
 display
 mainTrans
 caseKey
 caseVal
 separator
 debug
 />
●使用例
 例1)デフォルト以外に必要な属性のみ指定するサンプル
 <og:tableMerge action="UNION"
 slaveScope = "request" masterKeys = "AA, BB, CC"
 />
 ・出力先、マスターともに初期値は、tableId="DEFAULT" scope="session" です。
 スレーブは、tableId か scope をける必要がある為、scope="request" を指定しています。
 比較するカラム名は、マスタ、スレーブ同じであれば、マスタのみ指定でかまいません。
 例2)マスタ、スレーブともメモリに残らないように request で作成します。
 <og:tableMerge action="INTERSECT"
 masterScope = "request"
 slaveScope = "request"
 slaveTableId = "SLAVE"
 masterKeys = "AA, BB, CC"
 />
 ・マスタ、スレーブともメモリに残らないように request で作成した場合は、
 どちらかの TableId を変える必要があります。
 出力先は初期値の、tableId="DEFAULT" scope="session" です。

1.command

【TAG】 コマンド(NEW, RENEW)をセットします(初期値:NEW)。
コマンドは、HTMLから(get/post)指定されますので、CMD_xxx で設定される
フィールド定数値のいずれかを、指定できます。
初期値は NEW です。

2.action

【TAG】 アクションを指定します(UNION_ALL|UNION|INTERSECT|MINUS|DIFFERENCE|UNION_CLM|ADD_CLM|GROUP|UNION_SELROW)。
指定できるアクションは、全体集合(UNION_ALL)、全体集合(挿入位置指定)(UNION_SELROW)、和集合(UNION)、
積集合(INTERSECT)、差集合(MINUS)、差分集合(DIFFERENCE)、列合成(UNION_CLM)、列追加(ADD_CLM)、 グループ(GROUP)です。
列合成とグループ以外の処理では、カラム順とカラム数は同数でなければなりません。
primaryKeys や unionCfms などの指定のキー名は、マスタテーブルに存在する必要があります。
マスタテーブルと同じカラム番号でスレーブテーブルよりデータを読み出します。
(カラム名や属性は、異なってもかまいませんが、マスタテーブルに準拠します。)
また、単独(マスタテーブルのみ)で、和集合と同等の、グループ(GROUP)を使用すると、指定の
カラムでのユニーク化を行うことが可能になります。グループ処理では、先行優先とし、
2回目に現れた情報を削除することになります。グループ が指定された場合は、
スレーブテーブルは無視されます。いずれの処理においても、集合処理を行う主キーで
一旦グループ化されます。全体集合(UNION_ALL)で処理する場合でも、主キーがユニークで
ない場合は、マスター、スレーブの各テーブルで一旦グループ化された後で、マージされます。
(マージ後は、同一主キーを持つ行は存在します。)
全体集合(UNION_ALL)の場合のみ、mergeKeys を指定する必要はありません。その場合は、
キーなしのため、マスターとスレーブのテーブルを単に合成するだけになります。

action	名称	処理概要	1	2	3	4
UNION_ALL	全体集合	マスタとスレーブを合成				
UNION_SELROW	全体集合	マスタとスレーブを合成(マスタ表のチェック行を起点に追加)	<input type="radio"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
UNION	和集合	マスタとスレーブのユニーク部のみ合成	<input type="radio"/>	<input type="radio"/>	<input type="checkbox"/>	<input type="checkbox"/>

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

	INTERSECT	積集合	マスタとスレーブのユニーク部が一致するマスタのみ選択	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	MINUS	差集合	マスタからスレーブに存在するユニーク部を削除した残り	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	DIFFERENCE	差分集合	ユニーク部が一致し、差分カラムが異なるマスタのみ選択	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	UNION_CLM	列合成	マスタとキー致するスレーブのカラム情報を追加	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	ADD_CLM	列追加	UNION_CLMとの違いは、カラムのみ追加することです。	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	GROUP	グループ	マスタのユニーク部化	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	※：マスタテーブルオブジェクトは、常に必須 1：スレーブテーブルオブジェクト必須 2：masterKeys 属性必須 3：unionClms 属性必須(スレーブテーブルのカラム名または追加するカラム名) 4：diffKeys 属性必須(DIFFERENCE時の差分カラム名)、modifyClms 属性使用可能						
3.tableId	【TAG】出力先のtableIdを指定します (初期値:HybsSystem#TBL_MD_L_KEY[=h_tblmdl])。 集合処理結果の DBTableModel をメモリにセットする場合のキー(tableId)を指定します。 (初期値:HybsSystem#TBL_MD_L_KEY[=@og_value HybsSystem#TBL_MD_L_KEY])。						
4.scope	【TAG】出力先のscopeを指定します(初期値:session)。 集合処理結果の DBTableModel をメモリにセットする場合のスコープを指定します。 ここでは、マスタやスレーブのスコープ設定が必要な為、superクラスのメソッドを オーバーライドしてこのオブジェクト内でキープしています。 初期値は、session です。						
5.masterTableId	【TAG】マスタテーブルのtableIdを指定します (初期値:HybsSystem#TBL_MD_L_KEY[=h_tblmdl])。 集合処理のマスタとなる DBTableModel をメモリから取り出す場合のキー(tableId)を指定します。 (初期値:HybsSystem#TBL_MD_L_KEY[=@og_value HybsSystem#TBL_MD_L_KEY])。						
6.masterScope	【TAG】マスタテーブルのscopeを指定します(初期値:session)。 集合処理のマスタとなる DBTableModel をメモリから取り出す場合のスコープを指定します。 初期値は、session です。						
7.slaveTableId	【TAG】スレーブテーブルのtableIdを指定します (初期値:HybsSystem#TBL_MD_L_KEY[=h_tblmdl])。 集合処理のスレーブとなる DBTableModel をメモリから取り出す場合のキー(tableId)を指定します。 なお、アクションがグループ(GROUP)の場合は、スレーブテーブルは使用されません。 (初期値:HybsSystem#TBL_MD_L_KEY[=@og_value HybsSystem#TBL_MD_L_KEY])。						
8.slaveScope	【TAG】スレーブテーブルのscopeを指定します(初期値:session)。 集合処理のスレーブとなる DBTableModel をメモリから取り出す場合のスコープを指定します。 なお、アクションがグループ(GROUP)の場合は、スレーブテーブルは使用されません。 初期値は、session です。						
9.masterKeys	【TAG】マスタテーブルの集合処理を行う主キーを指定します。 集合処理を行う場合の、カラム名を、カンマ区切り文字(CSV形式)で指定します。 このキーの組み合わせを元に、集合処理の突合せを行います。 なお、アクションがグループ(GROUP)以外の処理では、マスタとスレーブのカラム数と 並び順は、同じでなければなりません。カラム名は、各々別々でもかまいません。 アクションが全体集合(UNION_ALL)以外の場合は、必須属性になります。						
10.slaveKeys	【TAG】スレーブテーブルの集合処理を行う主キーを指定します。 集合処理を行う場合の、カラム名を、カンマ区切り文字(CSV形式)で指定します。 このキーの組み合わせを元に、集合処理の突合せを行います。 なお、アクションがグループ(GROUP)以外の処理では、マスタとスレーブのカラム数と 並び順は、同じでなければなりません。カラム名は、各々別々でもかまいません。 null の場合は、masterKeys と同じとします。						
11.diffKeys	【TAG】マスタテーブルのDIFFERENCE時の差分カラム名を(CSV形式)指定します。 アクションが差分処理(DIFFERENCE)の場合に、差分チェックを行うカラム名を、 カンマ区切り文字(CSV形式)で指定します。 差分処理とは、masterKeys で指定されたキーでは、マスタ、スレーブともに存在し かつ、このキー(diffKeys)で指定されたキーの値が異なるマスタレコードを 抜き出します。 つまり、主キーは存在し、属性が異なる情報のピックアップになりますので、 データ更新(UPDATE)対象を見つける場合に使用できます。 アクションが差分処理(DIFFERENCE)の場合は、必須属性になります。						
12.unionClms	【TAG】スレーブからマスタへ追加するカラム名をCSV形式で指定します。 アクションが列合成(UNION_CLM)または列追加(ADD_CLM)の場合に使用されます。 列合成(UNION_CLM)は、マスタとスレーブの主キーに対して、ここで指定のスレーブの カラム列名を、マスタの列に追加します。主キーがマッチしない行に関しては、 カラムの初期値が適用されたデータを作成します。 列追加(ADD_CLM)は、マスタテーブルに指定のカラムを追加するだけです、スレーブテーブルは 参照しません。よって、主キーも指定不要です。						
13.modifyClms	【TAG】スレーブからマスタへ値を更新するカラム名をCSV形式で指定します。 アクションが差分処理(DIFFERENCE)の場合に、結果にマスタテーブルが抜き出されますが、 更新する場合に、スレーブ特有のユニークキー(例:UNIQ)を用いて更新する場合、 指定のカラム値は、スレーブの値にセットしておきたい場合があります。 ここでは、指定のカラムについて、値だけスレーブからマスタへセットします。 なお、値の更新については、マスタとスレーブが同一キーという制約があります。						
14.groupAddClms	【TAG】集合処理するとき、相違データをCSV連結して残すカラム名をCSV形式で指定します。 masterKeysで集合処理するとき、通常、最初に見つかった行データののみ残りますが、 ここに指定したカラムについては、発生都度、自分自身の情報に、CSV形式で連結して いきます。 この操作により、本来削除された情報が、1行のCSV形式で取得できる効果が得られます。 これは、value タグの action="APPEND" を、DBTableModel に対して実施するような感じです。						
15.noSideEffect	【TAG】テーブルモデルに対する副作用の有無[true:ない/false:ある]を指定します(初期値:false:ある)。 すべての処理で、DBTableModel に対して、ユニーク化やグループ化などの集合処理を 行う過程で、マスタテーブルに対して直接処理を行うと、副作用が発生します。 同様に、スレーブテーブルにおいても、一旦キー列でグループ化されるため、副作用が 発生します。これは、無駄なメモリ領域の確保と、テーブル(マスタ、スレーブとも)の コピー処理時間の節約になります。初期値の設定も副作用がある状態になっています。 ところが、都合によっては、色々な action を連続して行いたい場合など、毎回、 データベースを検索するよりもメモリ上でコピーしたほうが都合がよいケースでは、 副作用が出ないように、noSideEffect="true" に設定します。 ただし、マスタ、スレーブともテーブルをコピーを行い、結果のテーブルも派生する為、 通常、2つの領域(マスタと結果は同じテーブルに書かれる)で良い所を、5つの領域が 作成されます。 初期値は、副作用がある(noSideEffect="false")です。						
16.useDiffData	【TAG】差分のスレーブデータを結果テーブルに追加するかどうかを指定します(初期値:true)。 アクションが差分処理(DIFFERENCE)の場合に、結果にマスタテーブルが抜き出されますが、 差分対象のスレーブデータと比較したい場合があります。 このフラグを true にセットすると、書き込み禁止属性が付いた状態で、スレーブデータが 結果テーブルに追加されます。 なお、この処理では、通常と異なり、マスタテーブルにはグループ化の副作用は発生しますが、 結果テーブルは新規に作成され、先頭行に必ず WRITABLE カラムが付加されます。 初期値は、true:追加する です。						
	【TAG】マスタテーブルの選択行のデータのみを対象に処理を行うかどうかを指定します(初期値:false)。 処理対象のマスタテーブルについて、選択行が指定された場合に、選択行のみを処理対象に						

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
17.useCheckOnly	するか、全件を対象にするかを指定します。 積集合や差分集合など通常は、全件を対象にすることになりますが、列合成や列追加など、マスタテーブルに対してのみ作用を及ぼす処理では、選択行のみを対象に処理を行う事が考えられます。その場合、初期グループ化と同じで、対象とする行が選択行のみになります。 初期値は、false：全件対象 です。
18.display	【TAG】マージの結果を表示するかどうかを指定します(初期値:true)。 true で、マージ結果を表示します。 false では、何も表示しません(初期値:true) マスタテーブルの件数は、通常、キーでグループ化されるため、入力件数と異なります。 同様に、スレーブ件数も異なります。結果の件数は、処理結果が現実的かどうかの判断に使用されます。 初期値は、true：表示する です。
19.mainTrans	【TAG】(通常使いません)タグで処理される処理がメインとなるトランザクション処理かどうかを指定します(初期値:false)。 この値は、ファイルダウンロード処理に影響します。この値がtrueに指定された時にcommitされたDBTableModelがファイルダウンロードの対象の表になります。
20.caseKey	このパラメータは、通常、各タグにより実装され、ユーザーが指定する必要はありません。 但し、1つのJSP内でDBTableModelが複数生成される場合に、前に処理したDBTableModelについてファイルダウンロードをさせたい場合は、後ろでDBTableModelを生成するタグで、明示的にこの値をfalseに指定することで、ファイルダウンロード処理の対象から除外することができます。
21.caseVal	【TAG】このタグ自体を利用するかどうかの条件キーを指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。
22.separator	【TAG】このタグ自体を利用するかどうかの条件値を指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。
23.debug	【TAG】groupAddCllmsで文字列を連結する項目区切り文字をセットします(初期値:",")。 groupAddCllmsで文字列を連結する項目区切り文字をセットします(初期値:",")。 初期値は、"," に設定されています。
	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:tableUpdate JSP	<p>【TAG】Query を発行する為のクラスIDを指定します(初期値:JDBCTableUpdate)。 引数指定のINSERT/UPDATE文を実行する場合の、queryType 属性を使用します。 このタグでは、execute(int[] ,DBTableModel)を実行します。 代表的なクラスとして、“JDBCTableUpdate” が標準で用意されています。</p> <p>タグにより使用できる／出来ないがありますが、これは、org.opengion.hayabusa.db 以下の Query_**** クラスの **** を与えます。 これらは、Query インターフェースを継承したサブクラスです。</p>
<p>●形式: <og:tableUpdate command=“...” names=“...” queryType=“JDBCTableUpdate” > {@SQL} </og:update> ●body : あり ●Tag定義: <og:tableUpdate queryType 【TAG】Query を発行する為のクラスIDを指定します(初期値:JDBCTableUpdate) sqlType 【TAG】BODY部に書かれている Param の SQLタイプを指定します command 【TAG】コマンド(NEW, RENEW)をセットします(PlsqlUpdateTag, UpdateTag の場合は、ENTRY) scope 【TAG】キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session) displayMsg 【TAG】検索結果を画面上に表示するメッセージリソースIDを指定します (初期値:VIEW_DISPLAY_MSG[=]) resourceType 【特殊】クリアするリソースの種類[GEO03/GEO04/GEO08]を指定します conditionKey 【TAG】条件判定するカラム I D を指定します(初期値:null) conditionList 【TAG】条件判定する値のリストを、“ ”で区切って登録します(初期値:無条件) tableId 【TAG】(通常は使いません)結果のDBTableModelを、sessionに登録するときのキーを指定します dbId 【TAG】(通常は使いません)Queryオブジェクトを作成する時のDB接続IDを指定します selectedAll 【TAG】データを全件選択済みとして処理するかどうか[true/false]を指定します(初期値:false) commitTableModel 【特殊】SQL実行後に結果をDBTableModelに反映させるかどうか[true/false]を指定します(初期値:true) followCdkh 【TAG】DBTableModelの改廃Cに従って処理を行うかを指定します quotCheck 【TAG】リクエスト情報の クォーティション(') 存在チェックを実施するかどうか[true/false]を設定します(初期値:false) debug 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false) > ... Body ... </og:tableUpdate></p> <p>●使用例 ・QUERYを他のJSPから渡す場合 【copy.jsp】 <og:hidden name=“SQL” > INSERT INTO GE41 (CLM, NAME_JA, LABEL_NAME, KBSAKU, SYSTEM_ID, LANG, FGJ, DYSET, DYUPD, USRSET, USRUPD, PGUPD) VALUES (C[CLM], [NAME_JA], [LABEL_NAME], [KBSAKU], [SYSTEM_ID], [LANG], '1', '[@USER.YMDH]', '[@USER.YMDH]', '[@USER.ID]', '[@USER.ID]', '[@GUI.KEY]') </og:value></p> <p> 【entry.jsp】 <og:tableUpdate command = “[@command]” queryType = “JDBCTableUpdate” {@SQL} </og:tableUpdate></p> <p>・tableUpdateParamを使用する場合 【entry.jsp】 <og:tableUpdate command = “[@command]” queryType = “JDBCTableUpdate” sqlType = “[@sqlType]” // tableUpdateParam の sqlType と一致 > <og:tableUpdateParam sqlType = “[@sqlType]” // INSERT, COPY, UPDATE, MODIFY, DELETE table = “[@TABLE_NAME]” // 処理対象のテーブル名 names = “[@names]” // 処理対象のカラム名 omitNames = “[@omitNames]” // 処理対象外のカラム名 where = “[@where]” // 処理対象を特定するキー constKeys = “[@constKeys]” // 処理カラム名の中の固定情報カラム名 constVals = “[@constVals]” // 処理カラム名の中の固定情報設定値 /> </og:tableUpdate></p> <p>・処理の可否を指定する場合 【entry.jsp】 <og:tableUpdate command = “[@command]” queryType = “JDBCTableUpdate” conditionKey = “...” : 条件判定するカラム I D を指定(初期値は columnName) conditionList = “...” : 条件判定する値のリストを、“ ”で区切って登録(初期値は、無条件) {@SQL} </og:tableUpdate></p>	
1.queryType	<p>【TAG】Query を発行する為のクラスIDを指定します(初期値:JDBCTableUpdate)。 引数指定のINSERT/UPDATE文を実行する場合の、queryType 属性を使用します。 このタグでは、execute(int[] ,DBTableModel)を実行します。 代表的なクラスとして、“JDBCTableUpdate” が標準で用意されています。</p> <p>タグにより使用できる／出来ないがありますが、これは、org.opengion.hayabusa.db 以下の Query_**** クラスの **** を与えます。 これらは、Query インターフェースを継承したサブクラスです。</p>
2.sqlType	<p>【TAG】BODY部に書かれている Param の SQLタイプを指定します。 TableUpdateParamTag は、上位の TableUpdateTag の sqlType 属性 と同じ sqlType 属性の場合のみ、SQL文を合成・出力します。 つまり、TableUpdateTag側のsqlType 属性をパラメータに、TableUpdateParamTag の sqlType 属性を固定値にすることで、どのパラメータを使用するかを 選択できる機能を実現する事が可能です。</p>
3.command	<p>【TAG】コマンド(NEW, RENEW)をセットします(PlsqlUpdateTag, UpdateTag の場合は、ENTRY)。 コマンドは、HTMLから(get/post)指定されますので、CMD_xxx で設定される フィールド定数値のいづれかを、指定できます。</p>
	<p>【TAG】キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)。 “request”, “page”, “session”, “applicaton” が指定できます。 JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、 主にテーブルモデルを管理するスコープを指します。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容		
	形式サンプル		
4.scope	スコープ	変数の有効範囲	
	page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会
	request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やボ
	session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。内部的には、同じキーワ
	application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。
5.displayMsg	【TAG】 検索結果を画面上に表示するメッセージリソースIDを指定します (初期値:VIEW_DISPLAY_MSG[-])。 ここでは、検索結果の件数や登録された件数をまず出力し、 その次に、ここで指定したメッセージをリソースから取得して表示します。 件数を表示させたい場合は、displayMsg = "MSG0033"[件検索しました] をセットしてください。 表示させたくない場合は、displayMsg = "" をセットしてください。 (初期値:システム定数のVIEW_DISPLAY_MSG[=@og.value.org.opengion.hayabusa.common.SystemData#VIEW_DISPLAY_MSG])。		
	【特殊】 クリアするリソースの種類[GEO3/GEO4/GEO8]を指定します。 注意: この属性は、リソース関連DBのメンテナンス時にのみ、内部リソースキャッシュを クリアする目的で使います。一般の属性としては、使用することはないため、 ご注意ください。 リソース関連のテーブルを更新した場合、リソースキャッシュをクリアして 置かないと、データベースの値が反映されません。 昔は、リソースの更新ごとに、全件クリアしていましたが、部分クリアが できるようになったため、部分クリアを行います。 ここでは、(GEO3、GEO4、GEO8) のどれかを指定してください。		
6.resourceType	【TAG】 条件判定するカラムIDを指定します(初期値:null)。 指定のカラムIDの値と、conditionList の値を比較して、 存在する場合は、Query 処理を実行します。 例えば、conditionKey="CDKH" として、conditionList="A" とすれば、 改廃コードが"A"のデータで、かつ選択されたデータのみを処理します。 設定しない場合は、通常の処理と同様に、選択行のみ処理されます。		
	【TAG】 条件判定する値のリストを、"/"で区切って登録します(初期値:無条件)。 conditionKey とペアで指定します。ここには、カラムの設定値のリストを 指定することで、複数条件(OR結合)での比較を行い、リストにカラム値が 存在する場合のみ、Query 処理を実行します。 値が設定されている場合は、その値とマッチする必要があります。なにもセット されない場合、または、null の場合は、null データとマッチする場合のみ処理 されますので、ご注意ください。		
9.tableId	【TAG】 (通常は使いません) 結果のDBTableModelを、sessionに登録するときのキーを指定します。 検索結果より、DBTableModelオブジェクトを作成します。これを、以下のview 等のタグに 渡す場合に、通常は、session を利用します。その場合の登録キーです。 query タグを同時に実行して、結果を求める場合、同一メモリに配置される為、 この tableId 属性を利用して、メモリ空間を分けます。 初期値は、HybsSystem.TBL_MD_L KEY です。		
	【TAG】 (通常は使いません) Queryオブジェクトを作成する時のDB接続IDを指定します。 Queryオブジェクトを作成する時のDB接続IDを指定します。 これは、システムリソースで、DEFAULT_DB_URL 等で指定している データベース接続先 情報に、XX_DB_URL を定義することで、dbid="XX" とすると、この 接続先を使用して データベースにアクセスできます。		
11.selectedAll	【TAG】 データを全件選択済みとして処理するかどうか[true/false]を指定します(初期値:false)。 全てのデータを選択済みデータとして扱って処理します。 全件処理する場合に、(true/false)を指定します。 初期値は false です。		
	【特殊】 SQL実行後に結果をDBTableModelに反映させるかどうか[true/false]を指定します(初期値:true)。 注意: この属性は、リソース関連DBのメンテナンス時に、複数DBへの登録を行うための、 暫定対応として定義しています。 falseにした場合は、実データとDBTableModelの整合性が取れなくなるため、使用には十分注して下さい。 初期値は true です。		
13.followCdkh	【TAG】 DBTableModelの改廃Cに従って処理を行うかを指定します。 この属性は、sqlTypeが指定されている場合のみ有効です。 sqlTypeが指定されている場合、そのsqlTypeに対応した、改廃Cが設定されている 行のみを処理します。 対応関係は、以下の通りです。 sqlType = "INSERT" or "COPY" -> 改廃C= 'A' のみ処理 sqlType = "UPDATE" or "CHANGE" -> 改廃C= 'G' のみ処理 sqlType = "DELETE" -> 改廃C= 'D' のみ処理		
14.quotCheck	【TAG】 リクエスト情報のクォーティション(') 存在チェックを実施するかどうか[true/false]を設定します(初期値:false)。 SQLインジェクション対策の一つとして、暫定的ではありますが、SQLのパラメータに 渡す文字列にクォーティション(') を許さない設定にすれば、ある程度は防止できます。 数字タイプの引数には、 or 5=5 などのクォーティションを使用しないコードを埋めても、 数字チェックで検出可能です。文字タイプの場合は、必ず (') をはずして、 ' or 'A' like 'A' のような形式になる為、 (') チェックだけでも有効です。 (') が含まれていたエラーにする(true)／かノーチェックか(false)を指定します。 ※(他のタグは、システムリソースのUSE_SQL_INJECTION_CHECK[=@og.value.org.opengion.hayabusa.common.SystemData#USE_SQL_INJECTION_CHECK]) ですが、JSPの互換性を考慮し、初期値を固定でfalseにしています)		
15.debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。		

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:tableUpdateParam Tag	<p>【TAG】BODY部に書かれている SQLタイプを指定します。 SQLタイプは、INSERT、COPY、UPDATE、MODIFY、DELETE の中から指定する 必要があります。これらは、内部に書かれるSQLの形式を指定するのに使用します。 内部処理は、DBTableModelの改廃コード(A、C、D)に対して使用される SQL を選択する場合の情報に使用されます。 なお、COPY と MODIFY は、command で指定できる簡易機能として用意しています。 上位の TableUpdateTag の sqlType 属性 と同じsqlType 属性の場合のみ、SQL文を 合成・出力します。(上位のsqlTypeがnullの場合は、無条件実行します。) 指定のタイプが、異なる場合は、なにも処理を行いません。</p>
<p>●形式：<og:tableUpdate command="[@command]" queryType="JDBCTableUpdate"> <og:tableUpdateParam sqlType = "[@sqlType]" // INSERT、COPY、UPDATE、MODIFY、DELETE table = "[@TABLE_NAME]" // 処理対象のテーブル名 names = "[@names]" // 処理対象のカラム名 omitNames = "[@omitNames]" // 処理対象外のカラム名 where = "[@where]" // 処理対象を特定するキー constKeys = "[@constKeys]" // 処理カラム名の中の固定情報カラム名 constVals = "[@constVals]" // 処理カラム名の中の固定情報設定値 asNames = "[@asNames]" // 別名を付けたカラム名(select A as B from TBL の B を指定) orgNames = "[@orgNames]" // table実際のカラム名(select A as B from TBL の A を指定) funcKeys = "[@funcKeys]" // 関数等を設定するカラム名 funcVals = "[@funcVals]" // 関数等の設定値 logicalDelete = "[@logicalDelete]" // sqlTypeがDELETEの場合にもUPDATE文を発行 /> </og:tableUpdate></p> <p>●body：なし</p> <p>●Tag定義： <og:tableUpdateParam sqlType ○ 【TAG】 BODY部に書かれている SQLタイプを指定します。(必須) table ○ 【TAG】 処理対象のテーブル名を指定します。(必須) names 【TAG】 処理対象のカラム名をCSV形式で複数指定します omitNames 【TAG】 処理対象外のカラム名をCSV形式で複数指定します where 【TAG】 処理対象を特定するキー条件(where句)を指定します constKeys 【TAG】 設定値を固定値と置き換える対象となるカラム名をCSV形式で複数指定します constVals 【TAG】 設定値を固定値と置き換える対象となる設定値をCSV形式で複数指定します funcKeys 【TAG】 関数等を設定するカラム名をCSV形式で複数指定します funcVals 【TAG】 関数等の設定値をCSV形式で複数指定します asNames 【TAG】 別名を付けたカラム名(select A as B from TBL の B を指定)をCSV形式で複数指定します orgNames 【TAG】 table実際のカラム名(select A as B from TBL の A を指定)をCSV形式で複数指定します quotCheck 【TAG】 リクエスト情報の クォーティション(') 存在チェックを実施するかどうか[true/false]を設定します (初期値:USE_SQL_INJECTION_CHECK[=true]) constObjKey 【TAG】 固定情報カラムの処理オブジェクトを特定するキーを設定します(初期値:SYSTEM_ID) logicalDelete 【TAG】 sqlType="DELETE"の場合に論理削除(UPDATE)を行うかどうかを指定します(初期値:false) debug 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false) /></p> <p>●使用例 ・【entry.jsp】 <og:tableUpdate command="[@command]" queryType="JDBCTableUpdate"> <og:tableUpdateParam sqlType = "[@sqlType]" table = "[@MEM. TABLE_NAME]" where = "ROWID = [ROWID]" /> </og:tableUpdate></p>	
1.sqlType	<p>【TAG】 BODY部に書かれている SQLタイプを指定します。 SQLタイプは、INSERT、COPY、UPDATE、MODIFY、DELETE の中から指定する 必要があります。これらは、内部に書かれるSQLの形式を指定するのに使用します。 内部処理は、DBTableModelの改廃コード(A、C、D)に対して使用される SQL を選択する場合の情報に使用されます。 なお、COPY と MODIFY は、command で指定できる簡易機能として用意しています。 上位の TableUpdateTag の sqlType 属性 と同じsqlType 属性の場合のみ、SQL文を 合成・出力します。(上位のsqlTypeがnullの場合は、無条件実行します。) 指定のタイプが、異なる場合は、なにも処理を行いません。</p>
2.table	<p>【TAG】 処理対象のテーブル名を指定します。 テーブル名を指定することで、sqlTypeに応じた QUERYを生成することが出来ます。 生成する場合のカラムを特定する場合は、names 属性で指定できます。 また、WHERE条件は、where属性で指定します。</p>
3.names	<p>【TAG】 処理対象のカラム名をCSV形式で複数指定します。 生成するQUERYのカラム名をカンマ区切り文字(CSV)で複数指定します。 指定がない場合は、DBTableModel の全カラム(※)を使用して、QUERYを構築します。 一般に、テーブル結合してDBTableModelを構築した場合は、登録すべきカラムを 指定する必要があります。 (※)正確には、DBTableModel の全カラムのうち、ROWID、ROWNUM、WRITABLE カラムは 無視します。 分解方法は、通常のパラメータ取得後に、CSV分解します。</p>
4.omitNames	<p>【TAG】 処理対象外のカラム名をCSV形式で複数指定します。 生成するQUERYのカラム名に指定しないカラム名をカンマ区切り文字(CSV)で複数指定します。 指定がない場合は、DBTableModel の全カラム(※)を使用して、QUERYを構築します。 テーブル結合などで、処理したくないカラム数の方が少ない場合に、names ですべてを 指定するより少ない記述ですみます。 (※)正確には、DBTableModel の全カラムのうち、ROWID、ROWNUM、WRITABLE カラムは 無視します。</p>
5.where	<p>【TAG】 処理対象を特定するキー条件(where句)を指定します。 生成するQUERYのwhere 句を指定します。通常の WHERE 句の書き方と同じで、 DBTableModelの値を割り当てたい箇所に[カラム名] を記述します。 文字列の場合、設定値をセットするときに、シングルクォーテーションを 使用しますが、[カラム名]で指定する場合は、その前後に、(')シング ルクォーテーションは、不要です。 [@XXX] 変数を使用する場合は、パース時に固定文字に置き換えられる為、 文字列指定時の(')シングルクォーテーションが必要になります。 例：FGJ='1' and CLM=[CLM] and SYSTEM_ID in ([SYSID], '**') and KBSAKU=[@KBSAKU]</p>
6.constKeys	<p>【TAG】 設定値を固定値と置き換える対象となるカラム名をCSV形式で複数指定します。 names 属性のカラムや table 属性より、QUERYを作成して、DBTableModelの値を 割り当てする場合、DBTableModelの値ではなく、外部から指定した固定値を 割り当てたい場合に、そのカラム名をカンマ区切り文字(CSV)で複数指定します。 ここで指定するカラム名は、names 属性に含まれるか、DBTableModelのカラムとして 存在する必要があります。なお、names 属性に含まれる場合は、DBTableModelのカラムに 含まれる必要はありません。 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスの為です。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
7.constVals	<p>【TAG】設定値を固定値と置き換える対象となる設定値をCSV形式で複数指定します。 names 属性のカラムや table 属性より、QUERYを作成して、DBTableModelの値を割り当てる場合、DBTableModelの値ではなく、外部から指定した固定値を割り当てたい場合に、そのカラム名に対応する設定値をカンマ区切り文字 (CSV) で複数指定します。ここで指定する設定値は、constKeys 属性と対応させます。 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスる為です。</p>
8.funcKeys	<p>【TAG】関数等を設定するカラム名をCSV形式で複数指定します。 constVals 属性で設定する値は、必ずシングルクオートが付与されます。 その場合、関数などを設定したい場合でも、文字列として設定しようとしています。 ここで指定するカラム名(funcKeys)自身は、constKeys と同じ書式です。 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスる為です。</p>
9.funcVals	<p>【TAG】関数等の設定値をCSV形式で複数指定します。 funcKeys 属性に対応する 関数などの設定値を割り当てます。 constVals 属性との違いは、funcVals の設定値は、そのままの形で、SQL文の構築に使われます。 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスる為です。</p>
10.asNames	<p>【TAG】別名を付けたカラム名(select A as B from TBL の B を指定)をCSV形式で複数指定します。 SELECT 文を記述したとき、別名を付けていたり、SELECTしたテーブルと別のテーブルにDBTableModelの値を書き込む場合、DBTableModel の持っているカラム名と、実際に書き込むカラム名が異なります。そのようなケースに、元の別名カラムを指定します。 orgNames属性の並び順と、asNames属性の並び順を合わせておく必要があります。 このカラム名は、DBTableModel には持っているが、テーブル側には持っていない値なので、内部的に omitNames 属性に値を設定します。利用者は、omitNames に書き込む必要はありません。 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスる為です。</p>
11.orgNames	<p>【TAG】tableの実際のカラム名(select A as B from TBL の A を指定)をCSV形式で複数指定します。 SELECT 文を記述したとき、別名を付けていたり、SELECTしたテーブルと別のテーブルにDBTableModelの値を書き込む場合、DBTableModel の持っているカラム名と、実際に書き込むカラム名が異なります。そのようなケースに、テーブルの実カラムを指定します。 orgNames属性の並び順と、asNames属性の並び順を合わせておく必要があります。 このカラム名は、DBTableModel には持っていませんが、テーブル側には持っている値なので、このカラム名で、SQL文を構築します。 UPDATE TBL SET A=[B] WHERE ... となります。 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスる為です。</p>
12.quotCheck	<p>【TAG】リクエスト情報の クォーティション(') 存在チェックを実施するかどうか[true/false]を設定します (初期値:USE_SQL_INJECTION_CHECK[=true])。 SQLインJECTION対策の一つとして、暫定的ではありますが、SQLのパラメータに渡す文字列にクォーティション(') を許さない設定にすれば、ある程度は防止できます。 数字タイプの引数には、 or 5=5 などのクォーティションを使用しないコードを埋めても、数字チェックで検出可能です。文字タイプの場合は、必ず (') をはずして、 ' or 'A' like 'A のような形式になる為、(') チェックだけでも有効です。 (') が含まれていたエラーにする(true)／かノーチェックか(false)を指定します。 (初期値:システム定数のUSE_SQL_INJECTION_CHECK[=@og_value org.opengion.hayabusa.common.SystemData#USE_SQL_INJECTION_CHECK])。</p>
13.constObjKey	<p>【TAG】固定情報カラムの処理オブジェクトを特定するキーを設定します(初期値:SYSTEM_ID)。 固定情報カラム をシステム単位にJavaクラスで管理できます。 そのクラスオブジェクトは、org.opengion.hayabusa.db.DBConstValue インターフェースを継承した、plugin クラスになります。 そのクラスを特定するキーワードを指定します。 初期値は、SYSTEM_ID でシステム単位にクラスを作成します。 もし、他のシステムと共通の場合は、継承だけさせることも可能です。 対応したDBConstValueクラスがプラグインとして存在しない場合は、システムリソースのDEFAULT_CONST_CLASSで指定されたクラスが利用されます。 初期値は、SYSTEM_ID です。</p>
14.logicalDelete	<p>【TAG】sqlType="DELETE" の場合に論理削除(UPDATE)を行うかどうかを指定します(初期値:false)。 sqlType="DELETE" の場合に論理削除(UPDATE)を行うかどうかを指定します。 trueが指定された場合は、DELETE文ではなく、UPDATE文が発行されます。 falseが指定された場合は、DELETE文が発行されます。 さらに論理削除を行う場合、org.opengion.hayabusa.db.DBConstValue インターフェースに定義されている、getLogicalDeleteKeys() 及びgetLogicalDeleteValsを実装することで、論理削除する際のフラグの更新方法を統一的に管理することが可能になります。 初期値は、false(物理削除する)です</p>
15.debug	<p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:tbody JSP ●形式： <pre> <og:tbody rowspan = "表示データを作成する場合のフォーマットの行数(rowspan)をセットします(初期値:2)。(必須)" noClass = "カラムのクラス名 (VARCHAR2, NUMBER など) を使用するかどうか[true/false]を指定します(初期値:false)。” usableKey = "フォーマットの使用可否を判断するキーとなるカラム名を指定します。” usableList = "フォーマットの使用可否を判断する文字列リストを指定します(初期値:“1”)。” > ... Body ... </og:tbody> </pre> ●body：あり ●Tag定義： <pre> <og:tbody rowspan ○【TAG】表示データを作成する場合のフォーマットの行数(rowspan)をセットします(初期値:2)。(必須) noClass 【TAG】カラムのクラス名 (VARCHAR2, NUMBER など) を使用するかどうか[true/false]を指定します(初期値:false) usableKey 【TAG】フォーマットの使用可否を判断するキーとなるカラム名を指定します usableList 【TAG】フォーマットの使用可否を判断する文字列リストを指定します(初期値:“1”) caseKey 【TAG】このタグ自体を利用するかどうかの条件キーを指定します(初期値:null) caseVal 【TAG】このタグ自体を利用するかどうかの条件値を指定します(初期値:null) useTrCut 【TAG】先頭trタグを削除するかどうか[true/false]を指定します(初期値:true) debug 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false) > ... Body ... </og:tbody> </pre> ●使用例 <pre> <og:view viewFormType = "HTMLCustomTable" command = "{@command}" writable = "false" startNo = "{@startNo}" pageSize = "{@pageSize}" rowspan = "1" numberType = "delete" > <og:thead rowspan="1" > <tr><td>&nbsp;</td><td>[VALUENAME]</td><td>[DESCRIPTION]</td></tr> </og:thead> <og:tbody usableKey="SEQ" usableList="1" rowspan="1"> <tr><td colspan="2">&lt;td>[TAGNAME2]</td><td></td></tr> </og:tbody> <og:tbody usableKey="DATAKEY" usableList="0" rowspan="1"> <tr><td>&nbsp;<td>[VALUENAME]</td> <td>[DESCRIPTION]"</td></tr> </og:tbody> <og:tbody usableKey="DATAKEY" usableList="1" rowspan="2"> <tr><td>&gt;</td><td>... Body ...</td><td></td></tr> <tr><td colspan="3">&lt;td>[TAGNAME2]&gt;</td><td></td></tr> </og:tbody> <og:tbody usableKey="DATAKEY" usableList="2" rowspan="1"> <tr><td>/&gt;</td><td></td><td></td></tr> </og:tbody> </og:view> </pre>	<p>【TAG】表示データを作成する場合のフォーマットの行数(rowspan)をセットします(初期値:2)。 表示データを作成する場合のフォーマットの行数をセットします。 上位の viewFormタグより、こちらが優先されます。</p> <p>【TAG】カラムのクラス名 (VARCHAR2, NUMBER など) を使用するかどうか[true/false]を指定します(初期値:false)。 “true” で、クラス属性を設定しません。これは、CSS ファイルに書かれている属性を 使用しないことを意味します。 初期値は、“false” です。</p> <p>【TAG】フォーマットの使用可否を判断するキーとなるカラム名を指定します。 キーが、usableList に含まれる場合は、このフォームを使用できます。 キー(カラム名)が指定されない場合は、常に使用されます。 ※ 現時点では、BODYタイプのみ使用しています。</p> <p>【TAG】フォーマットの使用可否を判断する文字列リストを指定します(初期値:“1”)。 キーが、この文字列リスト中に存在する場合は、このフォームを使用できます。 この文字列リストは、固定な文字列です。{&64:XXXX} は使用できますが、[XXXX] は 使用できません。 初期値は、“1” です。 ※ 現時点では、BODYタイプのみ使用しています。</p> <p>【TAG】このタグ自体を利用するかどうかの条件キーを指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。</p> <p>【TAG】このタグ自体を利用するかどうかの条件値を指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。</p> <p>【TAG】先頭trタグを削除するかどうか[true/false]を指定します(初期値:true)。 フォーマットの先頭がtrタグの場合は削除する処理がありますが、 CustomDataのような場合では削除したくありません。 falseを指定すると削除処理を行わないようになります。</p> <p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。</p>
1.rowspan	<p>【TAG】表示データを作成する場合のフォーマットの行数(rowspan)をセットします(初期値:2)。 表示データを作成する場合のフォーマットの行数をセットします。 上位の viewFormタグより、こちらが優先されます。</p>
2.noClass	<p>【TAG】カラムのクラス名 (VARCHAR2, NUMBER など) を使用するかどうか[true/false]を指定します(初期値:false)。 “true” で、クラス属性を設定しません。これは、CSS ファイルに書かれている属性を 使用しないことを意味します。 初期値は、“false” です。</p>
3.usableKey	<p>【TAG】フォーマットの使用可否を判断するキーとなるカラム名を指定します。 キーが、usableList に含まれる場合は、このフォームを使用できます。 キー(カラム名)が指定されない場合は、常に使用されます。 ※ 現時点では、BODYタイプのみ使用しています。</p>
4.usableList	<p>【TAG】フォーマットの使用可否を判断する文字列リストを指定します(初期値:“1”)。 キーが、この文字列リスト中に存在する場合は、このフォームを使用できます。 この文字列リストは、固定な文字列です。{&64:XXXX} は使用できますが、[XXXX] は 使用できません。 初期値は、“1” です。 ※ 現時点では、BODYタイプのみ使用しています。</p>
5.caseKey	<p>【TAG】このタグ自体を利用するかどうかの条件キーを指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。</p>
6.caseVal	<p>【TAG】このタグ自体を利用するかどうかの条件値を指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。</p>
7.useTrCut	<p>【TAG】先頭trタグを削除するかどうか[true/false]を指定します(初期値:true)。 フォーマットの先頭がtrタグの場合は削除する処理がありますが、 CustomDataのような場合では削除したくありません。 falseを指定すると削除処理を行わないようになります。</p>
8.debug	<p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
5.5.4.2 og:text JSP	<p>【TAG】 value 値に直接書かれた漢字コードをShift_JIS に変換します。 ここで、value に設定した場合は、BODY 部は無視されます。 なお、このタグでは、エラー発生時でも継続して処理を続けられるようにします。 error.jsp などのエラー処理画面で、このタグを使用するケースがある為です。</p> <pre><og:text value="あいうえお" /></pre> <p>●形式 : <og:text >...</og:text> ●body : あり</p> <p>●Tag定義 :</p> <pre><og:text value include debug > ... Body ... </og:text></pre> <p>【TAG】 value 値に直接書かれた漢字コードをShift_JIS に変換します 【TAG】 動的にファイルを include します 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)</p> <p>●使用例</p> <pre><og:text > <p>あいおえお : <input name="PN" value="{@PN}" /> </p> </og:text> <og:text value="あいうえお" /></pre> <p>動的にファイルを include することが出来ます。 <pre><og:text include="{@query}.txt" /></pre></p>
1.value	<p>【TAG】 value 値に直接書かれた漢字コードをShift_JIS に変換します。 ここで、value に設定した場合は、BODY 部は無視されます。 なお、このタグでは、エラー発生時でも継続して処理を続けられるようにします。 error.jsp などのエラー処理画面で、このタグを使用するケースがある為です。</p> <pre><og:text value="あいうえお" /></pre>
2.include	<p>【TAG】 動的にファイルを include します。 指定のファイル名は、自身のディレクトリからの相対パスで表されます。 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。</p>
3.debug	<p>デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTML5 BODY要素	内容
形式サンプル	
5.5.4.2 og:textarea 拡張 JSP	【HTML】 名前を指定します。 名前を指定します。
●形式 : <og:queryOption > SELECT文 </og:queryOption > ●body : あり	
●Tag定義 : <og:textarea name lbl rows cols must mustAny id lang dir title style readonly disabled tabindex accesskey clazz language onClick onBlur onFocus ondblClick onMouseDown onMouseUp onMouseMove onMouseOut onMouseOver onSelect onKeyDown onKeyPress onKeyUp roles debug > ... Body ... </og:textarea>	○【HTML】 名前を指定します。(必須) 【TAG】 ラベルリソースのラベルIDを指定します 【HTML】 高さ(行数)を指定します 【HTML】 横幅(カラム数)を指定します 【TAG】 必須入力を表す色に変えるかどうか[true/false]を指定します(初期値:false) 【TAG】 選択必須入力(どれかひとつ必須)を表す色[true/mustAny/その他]を指定します(初期値:無指定) 【HTML】 要素に対して固有の名前(id)をつける場合に設定します 【HTML】 要素の内容と他の属性値の言語(lang, xml:lang)を指定します 【HTML】 文字表記の方向(dir)を指定します 【HTML】 要素に対する補足的情報(title)を設定します 【HTML】 この要素に対して適用させるスタイルシート(style)を設定します 【TAG】 その部品に対して変更が出来ないように(readonly) 指定します(サーバーに送信される) 【TAG】 その部品に対して、選択や変更が出来ないように(disabled) 指定します(サーバーに送信されない) 【HTML】 タブの移動順(tabindex)を指定します(0 ~ 32767) 【HTML】 アクセスキー(alt+キーで直接指定)を割り当てます 【HTML】 要素に対して class 属性を設定します 【TAG】 タグ内部で使用する言語コード[ja/en/zh/...]を指定します 【HTML】 JavaScriptのイベント onClick を設定します(例:onClick="renew('query.jsp','QUERY');") 【HTML】 JavaScriptのイベント onBlur を設定します(例:onBlur="this.value=value.toUpperCase();") 【HTML】 JavaScriptのイベント onFocus を設定します 【HTML】 JavaScriptのイベント ondblClick を設定します 【HTML】 JavaScriptのイベント onMouseDown を設定します 【HTML】 JavaScriptのイベント onMouseUp を設定します 【HTML】 JavaScriptのイベント onMouseMove を設定します 【HTML】 JavaScriptのイベント onMouseOut を設定します 【HTML】 JavaScriptのイベント onMouseOver を設定します 【HTML】 JavaScriptのイベント onSelect を設定します 【HTML】 JavaScriptのイベント onKeyDown を設定します 【HTML】 JavaScriptのイベント onKeyPress を設定します 【HTML】 JavaScriptのイベント onKeyUp を設定します 【TAG】 ロールを設定します 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)
●使用例 <og:textarea name="BUN" /> <og:textarea name="BUN"> test </og:textarea name="BUN">	初期値に値をセットしたいとき
1.name	【HTML】 名前を指定します。 名前を指定します。 【TAG】 ラベルリソースのラベルIDを指定します。 ラベルを変更するときに、lbl属性を使います。
2.lbl	ラベルID は、所定の language に基づく ResourceManager の getLabelData(id) を呼び出し、その結果のLabelInterfaceを使用します。 getMsgLbl() で取り出せます。 ラベルIDとメッセージIDは同時に登録できません。 ラベルとメッセージは統一されました。
3.rows	【HTML】 高さ(行数)を指定します。 高さ(行数)を指定します。
4.cols	【HTML】 横幅(カラム数)を指定します。 横幅を指定します。
5.must	【TAG】 必須入力を表す色に変えるかどうか[true/false]を指定します(初期値:false)。 初期値は、必須でない("false") です。
6.mustAny	<og:input name="PN" must="true" /> 【TAG】 選択必須入力(どれかひとつ必須)を表す色[true/mustAny/その他]を指定します(初期値:無指定)。 複数のカラムのうち、どれかひとつを必須とする選択必須入力を示す色を指定します。 true または、mustAny を設定すると、class属性に、mustAny がセットされます。 mustAny は、CSSファイルに初期設定されています。true または、mustAny 以外の値をセット すると、その値がそのまま、class属性にセットされますので、選択必須のグループ化が 可能です。 なお、実際の選択必須入力チェックは、ここではなく、columnCheck タグで指定が必要です。 自動処理は、mustAny="true" 指定の場合のみ有効です。 初期値は、無指定です。
7.id	【HTML】 要素に対して固有の名前(id)をつける場合に設定します。 特別な使用方法として、id="FOCUS" とすることで、フィールド系要素に フォーカスを移動させます。これは、そのページ内で唯一の id 属性として使用ください。 【HTML】 要素の内容と他の属性値の言語(lang, xml:lang)を指定します。
8.lang	HTMLの言語属性に使われます。指定する値は、ISO 639で規定されている「言語コード」です。 [ja/en/zh/...]などのほかに、en-US:アメリカ英語、en-cockney:コックニー英語 など、 副言語を指定する方法も定められています。 ここでは、lang と xml:lang の両方に同じ値がセットされます。 タグの language 属性とは使用用途が異なります。lang セット
9.dir	【HTML】 文字表記の方向(dir)を指定します。 当該要素の書字方向を指定する属性です。 ltr で、左から右に、rtl で、右から左に並べます。
10.title	【HTML】 要素に対する補足的情報(title)を設定します。 title セットは、ボタンなどに適用すると、マウスオーバーによりこのメッセージが チップスのように表示されます。これを利用して、説明文を登録することが可能です。 ここに登録した文字列が、メッセージリソースに存在する場合は、そのメッセージを 存在しない場合は、そのままの値を設定します。
11.style	【HTML】 この要素に対して適用させるスタイルシート(style)を設定します。 タグにstyle属性を設定します。これは、キー:値: のセットを複数記述できます。 通常は、class属性や、id属性で登録しておき、&lg:style type="text/css"> で 外部から指定する方がソースは読みやすくなります。
	【TAG】 その部品に対して変更が出来ないように(readonly) 指定します(サーバーに送信される)。 INPUT/TEXTAREA 系に対して、指定可能です。

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
12.readonly	readonly="readonly"、readonly="true" が指定された場合は、有効です。 false も指定値としては、有効です。(大文字小文字の区別も不要) それ以外の指定は、エラーとします。
13.disabled	【TAG】その部品に対して、選択や変更が出来ないように(disabled)指定します(サーバーに送信されない)。 BUTTON/INPUT/OPTGROUP/OPTION/SELECT/TEXTAREA 系に対して、指定可能です。 disabled="disabled"、disabled="true" が指定された場合は、有効です。 false も指定値としては、有効です。(大文字小文字の区別も不要) それ以外の指定は、エラーとします。
14.tabindex	【HTML】タブの移動順(tabindex)を指定します(0 ~ 32767)。 Tabキーを押したときに要素が選択される順番を指定します。 値には、選択させたい順番を数値で記述します。
15.accesskey	【HTML】アクセスキー(alt+キーで直接指定)を割り当てます。 アクセスキーは、マウスの使えない環境でも、リンクにジャンプする、ボタンを押す、入力フォームにフォーカスを移すなどの操作を簡単に行うことができるように考慮されたものです。 Windows の「ファイル(F)」メニューについている、F と同じような働きをします。
16.clazz	【HTML】要素に対して class 属性を設定します。 Javaの言語使用上、class で作成できないため、代用として clazz を使用しています。 html で作成される属性は、class で作成されます。
17.language	【TAG】タグ内部で使用する言語コード[ja/en/zh/...]を指定します。 この言語コードに基づいて、表示のラベルをリソースから作成します。
18.onClick	【HTML】JavaScriptのイベント onClick を設定します(例:onClick="renew('query.jsp','QUERY');")。 onClick をセットします。 例えば、<og:column name="KBSAKU" onClick="renew('query.jsp','QUERY');"> /> のように指定することで、プルダウンメニューの絞り込み検索が可能になります。
19.onBlur	【HTML】JavaScriptのイベント onBlur を設定します(例:onBlur="this.value=value.toUpperCase();")。 onBlur は、フォーカスが離れたときに発生するイベントです。
20.onFocus	【HTML】JavaScriptのイベント onFocus を設定します。 onFocus は、フォーカスされたときに発生するイベントです。
21.ondblClick	【HTML】JavaScriptのイベント ondblClick を設定します。 ondblClick は、マウスでダブルクリックされたときに発生するイベントです。
22.onMouseDown	【HTML】JavaScriptのイベント onMouseDown を設定します。 onMouseDown は、マウスダウンされたときに発生するイベントです。
23.onMouseUp	【HTML】JavaScriptのイベント onMouseUp を設定します。 onMouseUp は、マウスアップされたときに発生するイベントです。
24.onMouseMove	【HTML】JavaScriptのイベント onMouseMove を設定します。 onMouseMove は、マウスが移動されたときに発生するイベントです。
25.onMouseOut	【HTML】JavaScriptのイベント onMouseOut を設定します。 onMouseOut は、マウスが離れたときに発生するイベントです。
26.onMouseOver	【HTML】JavaScriptのイベント onMouseOver を設定します。 onMouseOver は、マウスが重なったときに発生するイベントです。
27.onSelect	【HTML】JavaScriptのイベント onSelect を設定します。 onSelect は、テキストフィールド/テキストエリアのテキストが 選択されたときに発生するイベントです。
28.onKeydown	【HTML】JavaScriptのイベント onKeydown を設定します。 onKeydown は、キーが押されたときに発生するイベントです。
29.onKeyPress	【HTML】JavaScriptのイベント onKeyPress を設定します。 onKeyPress は、キーが押され続けていたときに発生するイベントです。
30.onKeyUp	【HTML】JavaScriptのイベント onKeyUp を設定します。 onKeyUp は、キーが押された状態から離されたときに発生するイベントです。
31.roles	【TAG】ロールをセットします。 ここで指定したカラムロールを元に、ユーザー毎のアクセス許可がチェックされます。 アクセス許可されないと、表示されません。 このロールを指定しない場合は、カラムリソースのロールが使用されます。
32.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:tfoot JSP ●形式 : <og:tfoot rowspan="..." > ... Body ... </og:tfoot> ●body : あり ●Tag定義 : <pre> <og:tfoot rowspan caseKey caseVal useTrCut debug > ... Body ... </og:tfoot> </pre> ●使用例 <pre> <og:view viewFormType = "HTMLCustomTable" command = "@command" writable = "false" startNo = "@startNo" pageSize = "@pageSize" rowspan = "1" numberType = "delete" > ... <og:tfoot rowspan="1" > <tr><td>&nbsp;</td><td>[VALUENAME]</td><td>[DESCRIPTION]</td></tr> </og:tfoot> </og:view> </pre>	【TAG】表示データを作成する場合のフォーマットの行数(rowspan)をセットします(初期値:2)。 表示データを作成する場合のフォーマットの行数をセットします。 上位の viewFormタグより、こちらが優先されます。 【TAG】表示データを作成する場合のフォーマットの行数(rowspan)をセットします(初期値:2)。(必須) 【TAG】このタグ自体を利用するかどうかの条件キーを指定します(初期値:null) 【TAG】このタグ自体を利用するかどうかの条件値を指定します(初期値:null) 【TAG】先頭trタグを削除するかどうか[true/false]を指定します(初期値:true) 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)
1.rowspan	【TAG】表示データを作成する場合のフォーマットの行数(rowspan)をセットします(初期値:2)。 表示データを作成する場合のフォーマットの行数をセットします。 上位の viewFormタグより、こちらが優先されます。
2.caseKey	【TAG】このタグ自体を利用するかどうかの条件キーを指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。
3.caseVal	何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。 【TAG】このタグ自体を利用するかどうかの条件値を指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。
4.useTrCut	何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。 【TAG】先頭trタグを削除するかどうか[true/false]を指定します(初期値:true)。 フォーマットの先頭がtrタグの場合は削除する処理がありますが、 CustomDataのような場合では削除したくありません。 falseを指定すると削除処理を行わないようになります。
5.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:thead JSP	【TAG】 表示データを作成する場合のフォーマットの行数(rowspan)をセットします(初期値:2)。 表示データを作成する場合のフォーマットの行数をセットします。 上位の viewFormタグより、こちらが優先されます。
●形式: <og:thead rowspan="..."> ... Body ... </og:thead> ●body: あり	
●Tag定義: <og:thead rowspan caseKey caseVal useTrCut debug > ... Body ... </og:thead>	○【TAG】 表示データを作成する場合のフォーマットの行数(rowspan)をセットします(初期値:2)。(必須) 【TAG】 このタグ自体を利用するかどうかの条件キーを指定します(初期値:null) 【TAG】 このタグ自体を利用するかどうかの条件値を指定します(初期値:null) 【TAG】 先頭trタグを削除するかどうか[true/false]を指定します(初期値:true) 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)
●使用例 <og:view viewFormType = "HTMLCustomTable" command = "[@command]" writable = "false" startNo = "[@startNo]" pageSize = "[@pageSize]" rowspan = "1" numberType = "delete" > <og:thead rowspan="1"> <tr><td> </td><td>[VALUENAME]</td><td>[DESCRIPTION]</td></tr> </og:thead> ... </og:view>	
1.rowspan	【TAG】 表示データを作成する場合のフォーマットの行数(rowspan)をセットします(初期値:2)。 表示データを作成する場合のフォーマットの行数をセットします。 上位の viewFormタグより、こちらが優先されます。
2.caseKey	【TAG】 このタグ自体を利用するかどうかの条件キーを指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。
3.caseVal	何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。 【TAG】 このタグ自体を利用するかどうかの条件値を指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。
4.useTrCut	何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。 【TAG】 先頭trタグを削除するかどうか[true/false]を指定します(初期値:true)。 フォーマットの先頭がtrタグの場合は削除する処理がありますが、 CustomDataのような場合では削除したくありません。 falseを指定すると削除処理を行わないようになります。
5.debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:timeTableParam Tag	<p>【TAG】タイムテーブルの開始時刻(含む)をセットします(初期値:0800)。時間軸の書き始めの時刻(自分自身を含む時分4桁)を指定します。</p> <p>この時刻は、8:00 なら、“0800” となり、14:30 なら、“1430” となります。初期値は、“0800” です。</p>
●形式: <og:timeTableParam minStartTime="..." ... /> ●body: なし	
●Tag定義: <og:timeTableParam minStartTime maxEndTime timeInterval nullLinkColumn useDyBreak tdClassColumn useBookingMerge />	<p>【TAG】タイムテーブルの開始時刻(含む)をセットします(初期値:0800)</p> <p>【TAG】タイムテーブルの終了時刻(含まない)をセットします(初期値:2100)</p> <p>【TAG】タイムテーブルのインターバル時間をセットします(初期値:30)</p> <p>【TAG】タイムテーブルが空きの場合のリンクを指定しているカラム名をセットします</p> <p>【TAG】日付でブレイク処理を行うかどうかを指定します(初期値:true)</p> <p>【TAG】タイムテーブルにデータを入れるTDタグにclass属性を付与する場合のカラム名をセットします</p> <p>【TAG】同一日付でブッキング時にマージ処理を行うかどうかを指定します(初期値:false)</p>
●使用例 ViewFormTag の viewFormType が、HTMLTimeTable の場合に使用します。 useParam 属性を設定しておかないと、使用されません。 <og:view viewFormType = "HTMLTimeTable" command = "[@command]" startNo = "0" pageSize = "20" useParam = "true" > <og:timeTableParam minStartTime = "0800" maxEndTime = "2100" timeInterval = "30" nullLinkColumn = "DYUSE" tdClassColumn = "FGCD ACTION"	<p>【TAG】タイムテーブルの開始時刻(含む)をセットします(初期値:0800)。時間軸の書き始めの時刻(自分自身を含む時分4桁)を指定します。</p> <p>この時刻は、8:00 なら、“0800” となり、14:30 なら、“1430” となります。初期値は、“0800” です。</p> <p>【TAG】タイムテーブルの終了時刻(含まない)をセットします(初期値:2100)。時間軸の最後の時刻(自分自身を含まない時分4桁)を指定します。</p> <p>この時刻は、9:00 なら、“0900” となり、14:30 なら、“1430” となります。初期値は、“2100” です。</p> <p>【TAG】タイムテーブルのインターバル時間をセットします(初期値:30)。タイムテーブルのインターバル時間とは、時刻の最終単位の事です。</p> <p>この時刻は、“30” なら、30分となります。初期値は、“30” です。</p>
1.minStartTime	<p>【TAG】タイムテーブルの開始時刻(含む)をセットします(初期値:0800)。時間軸の書き始めの時刻(自分自身を含む時分4桁)を指定します。</p> <p>この時刻は、8:00 なら、“0800” となり、14:30 なら、“1430” となります。初期値は、“0800” です。</p>
2.maxEndTime	<p>【TAG】タイムテーブルの終了時刻(含まない)をセットします(初期値:2100)。時間軸の最後の時刻(自分自身を含まない時分4桁)を指定します。</p> <p>この時刻は、9:00 なら、“0900” となり、14:30 なら、“1430” となります。初期値は、“2100” です。</p>
3.timeInterval	<p>【TAG】タイムテーブルのインターバル時間をセットします(初期値:30)。タイムテーブルのインターバル時間とは、時刻の最終単位の事です。</p> <p>この時刻は、“30” なら、30分となります。初期値は、“30” です。</p>
4.nullLinkColumn	<p>【TAG】タイムテーブルが空きの場合のリンクを指定しているカラム名をセットします。これは、タイムテーブルが空きの場合のリンクを作成するにあたり、ベースとなるリンクが適用されているカラムを指定します。</p> <p>このリンクは、特殊で、引数に、パラメータを追加できますが、(\$1) 等の記号で指定します。この(\$1)、(\$2)には、開始時刻、終了時刻がセットされますが、SELECT文の固定カラムと同じ並び順ですが、DBTableModelの値を設定しているわけではありません。空きの場合は、データ自体が存在しない場合がありますが、その場合は、開始時刻、終了時刻はありません。</p> <p>その場合は、それぞれ、最小開始時刻と最大終了時刻がセットされます。</p> <p>&TMSTART=(\$1)&TMEND=(\$2) という文字列の (\$*) 部分を解析して割当ます。</p> <p>TMSTARTやTMENDは、リンク作成側で自由に指定できます。</p> <p>同様の機能は、BODY部にリンクを指定することも可能です。この(\$1)～(\$4)には、開始時刻、終了時刻、日付、キーがセットされます。</p> <p>&TMSTART=(\$1)&TMEND=(\$2)&DYUSE=(\$3)&UNITID=(\$4) という文字列の (\$*) 部分を解析して割当ます。</p> <p>BODY と nullLinkColumn が両方とも指定された場合は、nullLinkColumn の設定が優先されます。</p>
5.useDyBreak	<p>【TAG】日付でブレイク処理を行うかどうかを指定します(初期値:true)。</p> <p>日付でブレイク処理を行う場合、日付単位にテーブルが分かれます。</p> <p>日付は、テーブルの先頭に、ブレイクした時点で表示されます。</p> <p>日付でブレイクするを指定した場合は、自動的に、noDisplay 属性に日付がセットされます。</p> <p>初期値は、true(日付ブレイクする)です。</p>
6.tdClassColumn	<p>【TAG】タイムテーブルにデータを入れるTDタグにclass属性を付与する場合のカラム名をセットします。これは、タイムテーブルのリンクや説明を入れるTDに、class属性を付与する場合のカラム名を指定します。これにより、TD に色を付けたり、表示の条件を外から指定できます。もっとも一般的な想定用途は、タイムテーブルのデータの種別に応じた色分けです。</p>
7.useBookingMerge	<p>【TAG】同一日付でブッキング時にマージ処理を行うかどうかを指定します(初期値:false)。</p> <p>日付、キー(人や施設)で予定時刻が重複している場合の処理方法を指定します。通常(初期値:false)では、ブッキングデータはレコードを分けて表示させます。例えば、人の予定であれば、仮予約や会議招集などのケースで、重複を表示しておき利用者本人に決めさせるというケースが考えられます。これを、true に設定すると、予定時刻が重複している場合は、マージして、一つの予定として表現します。初期値は、false(ブッキング時にマージ処理を行わない)です。</p>

バージョン
タグ名
HTML
EXT
BODY要素

内容

形式サンプル

5.5.4.2
og:topMenu

JSP

●形式 : <og:topMenu />
●body : なし

●Tag定義 :
<og:topMenu
menuType
expand
groups
classify
href
target
imageOnly
sideCount
minCellCount
maxCellCount
cache
match
unmatch
useButton
useButtonScript
buttonRequest
inlineStyle
useDivOneLevel
debug
> ... Body ...
</og:topMenu>

●使用例
<og:topMenu />

<og:topMenu
menuType = "NORMAL" expand = "true" groups = "AA, BB, CC" classify = "ABC" href = "menu.jsp" target = "MENU" imageOnly = "false" sideCount = "6" minCellCount = "8" maxCellCount = "8" cache = "true" match = "正規表現" unmatch = "正規表現" useButton = "false" useButtonScript = "false" useDivOneLevel = "false"

</og:topMenu />

【TAG】作成するメニューの種類 (NORMAL, GROUP, ONELEVEL, NEXTGUI, MATRIX, MATRIX2) を指定します (初期値: NORMAL)。
作成するメニューには、複数の種類があります。

種類	説明
NORMAL	通常の階層メニュー
GROUP	GROUPのみを取り出してリンクを作成します。(topMenuに利用)
ONELEVEL	指定のclassify のメニューのみを取り出してリンクを作成します。(lineMenuに利用)
NEXTGUI	既存のページの次にアクセスされる画面郡のリンクを作成します。
MATRIX	一覧表形式のメニューを作成します。(大分類付きマルチメニュー)
MATRIX2	一覧表形式のメニューを作成します。(大分類なしボタンメニュー)

1.menuType

2.expand

3.groups

4.classify

5.href

【TAG】作成するメニューの種類 (NORMAL, GROUP, ONELEVEL, NEXTGUI, MATRIX, MATRIX2) を指定します (初期値: NORMAL)。
作成するメニューには、複数の種類があります。

種類	説明
NORMAL	通常の階層メニュー
GROUP	GROUPのみを取り出してリンクを作成します。(topMenuに利用)
ONELEVEL	指定のclassify のメニューのみを取り出してリンクを作成します。(lineMenuに利用)
NEXTGUI	既存のページの次にアクセスされる画面郡のリンクを作成します。
MATRIX	一覧表形式のメニューを作成します。(大分類付きマルチメニュー)
MATRIX2	一覧表形式のメニューを作成します。(大分類なしボタンメニュー)

【TAG】折り返しメニューを構築するかどうかを指定します (初期値: true)。
trueを設定すると、JavaScriptによる折り返しメニューを構築します。
falseの場合は、通常のHTMLのみで、階層メニューを構築します。
初期値は、true (折り返しメニュー) です。

【TAG】表示対象となるグループをカンマ区切り文字列で指定します。
メニューの表示対象グループをカンマ区切り文字列で複数指定できます。
指定のグループのメニューだけが、表示対象になります。
メニューにも、複数のグループを指定できるため、1グループの指定で、
複数のくくりを表示することも可能です。
グループを指定しない場合は、全グループが対象になります。
また、メニュー側にグループ指定がない場合は、グループ指定に
関係なく、対象になります。
初期値は、未指定 (全メニューが対象) です。
分解方法は、通常のパラメータ取得後に、CSV分解します。

【TAG】表示対象となる分類 (classify) を指定します。
メニューの表示対象となる分類 (classify) を指定することで、一まとまりの
メニューを作成します。これは、3段階メニューの最終メニューを求める場合に
指定します。
最終メニューは、画面上部に設ける予定のメニューで、上下フレーム分割での
運用時に使用します。
分類の指定がない場合は、すべてが表示対象になります。
初期値は、未指定 (全メニューが対象) です。

【TAG】グループメニューの表示対象となるソース名 (href) を指定します (初期値: menu.jsp)。
GROUPメニューの表示対象となるソース名 (href) を指定することで、
サブメニューを自分自身のフレームに対して割り当てるのか、フレームを分けて
管理するのかを自由に設定できます。
初期値は、menu.jspです。

【TAG】グループメニューの表示対象となるフレーム名 (target) を指定します (初期値: MENU)。
GROUPメニューの表示対象となるフレーム名 (target) を指定することで、
サブメニューを自分自身のフレームに対して割り当てるのか、フレームを分けて

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
6.target	<p>管理するのかを自由に設定できます。</p> <p>フレーム分割を行うと、変更箇所は、サブメニューのみになる為、動きに無駄がなくなります。グループメニューの大きさが固定されてしまいます。</p> <p>自分自身にすると、グループメニューとサブメニューを一つのフレームに入れることで、更新時の画面のちらつきは発生しますが、無駄なスペースは省くことが可能になります。</p> <p>初期値は、MENU(通常のメニューフレーム)です。</p>
7.imageOnly	<p>【TAG】グループメニューの表示に、画像のみかどうか[true:画像のみ/false:画像+ラベル]を指定します(初期値:false)。</p> <p>GROUPメニューの表示対象として、jsp/menuImage 以下に グループ名と同一の画像ファイルが存在する場合は、画像を使用します。</p> <p>このフラグを、true に設定すると、画像のみを使用します。</p> <p>false の場合は、画像+グループ名のラベルを使用します。</p> <p>画像のみの場合でも、title 属性にグループ名のラベルをセットしますので、マウスカーソルをオーバーすれば、名称がTips表示されます。</p> <p>画像が存在しない場合に、true(画像のみ)に設定した場合は、ラベルの最初の1文字のみを出力します。</p> <p>初期値は、false(画像+ラベル)です。</p>
8.sideCount	<p>【TAG】グループメニューの表示対象となるメニューを横並びさせる数を指定します。</p> <p>GROUPメニューなどの表示を行う場合に、横方向に何個のメニューを表示させるかを指定します。例えば、画像のみのリンクと組み合わせれば、より、多くのグループを横方向に並べることで、小領域に多くの情報を結めることが可能になります。</p> <p>0 を設定すると、横方向にのみ並べる(折り返さない)メニューを作ることが可能になります。</p> <p>初期値は、無制限です。</p>
9.minCellCount	<p>【TAG】表形式メニュー(MATRIX)の一つのセルに含まれる最小行数を指定します(初期値:8)。</p> <p>表形式メニュー(MATRIX)では、一つのセルの高さを同一にする為、
タグを挿入します。</p> <p>このタグの挿入する個数を指定します。</p> <p>この個数だけ、メニューの数があろうとなかろうと行を確保します。</p> <p>指定の値が、実際の行数より少ない場合は、実際の行数分だけ拡張されます。</p> <p>初期値は、8 です。</p>
10.maxCellCount	<p>【TAG】表形式メニュー(MATRIX)の一つのセルに含まれる最大行数を指定します(初期値:8)。</p> <p>表形式メニュー(MATRIX)では、一つのセルの高さを同一にする為、指定の行数で新たな セルを作成して、セルを横方向に連結します。</p> <p>初期値は、8 です。</p>
11.cache	<p>【TAG】グループメニューのキャッシュを使用するかどうか[true/false]を指定します(初期値:true)。</p> <p>GROUPメニューの表示は、一旦作成すると、ほとんど書き換えることがありません。</p> <p>作成は、使用できる全メニューをスキャンして、その中からグループ属性をピックアップするという処理を行っている為、明らかに無駄な処理です。</p> <p>そこで、jsp/index.jsp が実行された場合のみキャッシュをクリアして、内部ではキャッシュがなければ作成し、あればキャッシュを使うロジックになっています。</p> <p>ここでは、キャッシュを使用するの、毎回作成しなおすのかを指定します。</p> <p>対象として、jsp/menuImage 以下に グループ名と同一の画像ファイルが存在する場合は、画像を使用します。</p> <p>このフラグを、true に設定すると、画像のみを使用します。</p> <p>false の場合は、画像+グループ名のラベルを使用します。</p> <p>画像のみの場合でも、title 属性にグループ名のラベルをセットしますので、マウスカーソルをオーバーすれば、名称がTips表示されます。</p> <p>画像が存在しない場合は、たとえ、true(画像のみ)に設定しても、ラベルを出力します。</p> <p>初期値は、true(キャッシュする)です。</p>
12.match	<p>【TAG】正判定(マッチする場合に、メニューに出す)条件を設定します。</p> <p>メニューを表示する/しないの判定を、画面IDのマッチングで判断します。</p> <p>これは、正規表現で表される引数と、画面IDがマッチする場合に、メニューを表示させます。</p> <p>マッチしない場合は、表示されません。</p> <p>何も指定しない場合は、ロールの判定みの行われます。</p>
13.unmatch	<p>【TAG】逆判定(マッチする場合に、メニューに出さない)条件を設定します。</p> <p>メニューを表示する/しないの判定を、画面IDのマッチングで判断します。</p> <p>これは、正規表現で表される引数と、画面IDがマッチする場合に、メニューを表示させません。</p> <p>マッチしない場合は、表示されます。</p> <p>何も指定しない場合は、ロールの判定みの行われます。</p>
14.useButton	<p>【TAG】画面リンクにボタンを使用するかどうか[true/false]を指定します(初期値:false)。</p> <p>画面リンクをボタンリンク形式で表示するかを指定します。</p> <p>falseの場合は、通常のリンクになります。</p> <p>初期値は、false(通常リンク)です。</p>
15.useButtonScript	<p>【TAG】(廃止)ボタンリンクを使用した場合に、JavaScriptのイベントを発生させるかどうか[true/false]を指定します(初期値:false)。</p> <p>(この属性は、廃止されています)</p> <p>ボタンリンクを使用した場合に、JavaScriptのイベントを発生させるかを指定します。</p> <p>発生するイベント及び呼ばれる関数は以下の3つです。</p> <p>①クリック時 -> buttonClick(this)</p> <p>②マウスオーバー時 -> buttonOver(this)</p> <p>③カーソルアウト時 -> buttonOut(this)</p> <p>初期値は、false(Javascriptイベントを発生しない)です。</p>
16.buttonRequest	<p>【TAG】マトリクスからの遷移先でボタンメニュー表示するかどうか[true/false]を指定します(初期値:false)。</p> <p>マトリクスメニューからの遷移先でボタンメニューを表示させるためにアドレスに付加するリクエスト変数を指定します。</p> <p>trueにするとbuttonRequest=trueのリクエスト変数を付けます。</p> <p>falseの場合はリクエスト変数を付けません。</p> <p>初期値は、false(プルダウン形式で表示)です。</p>
17.inlineStyle	<p>【TAG】標準画面を初期状態で表示するかどうか[true/false]を指定します(初期値:false)。</p> <p>通常expand=trueの状態では、メニューが折りたたまれています。</p> <p>このinlineStyle属性をtrueにすると標準画面のスタイルにdisplay:inlineを付加する事で、初期状態でメニューが開いた状態になります。</p> <p>expand=falseとの違いは、隠しメニュー及び分類の折りたたみ機能が利用できる事です。</p> <p>初期値は、false(折りたたまれた状態)です。</p>
18.useDivOneLevel	<p>【TAG】ONELEVEL、NEXTGUI で、DIVタグ+design-onelevel 処理をするかどうか[true/false]を指定します(初期値:false)。</p> <p>ONELEVEL、NEXTGUI は、QUERY 画面の上部に表示される簡易メニューです。</p> <p>この表記は、[画面リンク]形式のテキスト表示されていますが、画面名称を固定長にするなどの処理を入れるため、DIVでフォーマットします。</p> <p>作りは、出力される HTML を確認いただきたいと思います。</p> <p>全体を、<div id="design-onelevel"> で、囲い、画面名称は、 で、囲います。</p> <p>これを、標準CSSで、固定幅と背景色、リンクの文字色など書き換えて、体裁を整えます。</p> <p>初期値は、false(従来と同じ)です。</p>
19.debug	<p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。</p> <p>デバッグ情報を [true:出力する/false:しない]を指定します。</p> <p>出力形式自体は、個々のタグによって異なります。</p>

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

5.5.4.2
og:update

JSP

【廃止】 Query を発行する為のクラスIDを指定します(初期値:JDBCPrepared)。
登録を実行する手段は、Query インターフェースの実装クラスになります。
このタグでは、Query.execute(String[]) メソッドが複数回呼ばれます。
これは、DBTableModelの選択された行に対して、繰り返し呼ばれます。

通常、このタグを使用する場合は、queryType="JDBCPrepared" を指定します。

タグにより使用できる／出来ないがありますが、これは、org.opengion.hayabusa.db
以下の Query **** クラスの **** を与えます。
これらは、Query インターフェースを継承したサブクラスです。

●形式 : <og:update command="..." names="..." queryType="JDBCPrepared" >
 {@SQL}
 </og:update>
●body : あり

●Tag定義 :
 <og:update
 queryType 【廃止】 Query を発行する為のクラスIDを指定します(初期値:JDBCPrepared)
 command 【TAG】 コマンド(NEW, RENEW)をセットします(PisqlUpdateTag, UpdateTag の場合は、ENTRY)
 scope 【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)
 displayMsg 【TAG】 検索結果を画面上に表示するメッセージリソースIDを指定します (初期値:VIEW_DISPLAY_MSG[=])
 names 【TAG】 PL/SQLを利用する場合の引数にセットすべき データの名称をCSV形式で複数指定します
 quotCheck 【TAG】 リクエスト情報の クォーティション(') 存在チェックを実施するかどうか[true/false]を設定します(初期値:USE_SQL_INJECTION_CHECK)
 tableId 【TAG】 (通常は使いません)結果のDBTableModelを、sessionに登録するときのキーを指定します
 dbId 【TAG】 (通常は使いません)Queryオブジェクトを作成する時のDB接続IDを指定します
 xssCheck 【TAG】 リクエスト情報の HTMLTag開始/終了文字(<>) 存在チェックを実施するかどうか[true/false]を設定します (初期値:USE_XSS_CHECK[=true])
 debug 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)
 > ... Body ...
 </og:update>

●使用例
・引数/プロシジャーを他のJSPから渡す場合
 【copy.jsp】
 <og:value scope="session" key="names" value="CLM, NAME_JA, LABEL_NAME, KBSAKU, SYSTEM_ID, LANG" />
 <og:value scope="session" key="SQL" >
 INSERT INTO GE41
 (CLM, NAME_JA, LABEL_NAME, KBSAKU, SYSTEM_ID, LANG,
 FGJ, DYSET, DYUPD, USRSET, USRUPD, PGUPD)
 VALUES
 (?, ?, ?, ?, ?, ?,
 '1', '[@USER.YMDH]', ' '[@USER.YMDH]', ' '[@USER.ID]', ' '[@USER.ID]', ' '[@GUI.KEY]')
 </og:value>

 【entry.jsp】
 <og:update
 command = "[@command]"
 queryType = "JDBCPrepared"
 names = "[@names]" >
 {@SQL}
 </og:update>

 <!-- 前面で指定のSQL文を削除します。(scope="session"なので削除が必要。) -->
 <og:value scope="session" key="names" command="REMOVE" />
 <og:value scope="session" key="SQL" command="REMOVE" />

・引数/プロシジャーを直接書く場合
 【entry.jsp】
 <og:update
 command = "[@command]"
 queryType = "JDBCPrepared"
 names = "CLM, NAME_JA, LABEL_NAME, KBSAKU, SYSTEM_ID, LANG"
 >
 INSERT INTO GE41
 (CLM, NAME_JA, LABEL_NAME, KBSAKU, SYSTEM_ID, LANG,
 FGJ, DYSET, DYUPD, USRSET, USRUPD, PGUPD)
 VALUES
 (?, ?, ?, ?, ?, ?,
 '1', '[@USER.YMDH]', ' '[@USER.YMDH]', ' '[@USER.ID]', ' '[@USER.ID]', ' '[@GUI.KEY]')
 </og:update>

1.queryType

【廃止】 Query を発行する為のクラスIDを指定します(初期値:JDBCPrepared)。
登録を実行する手段は、Query インターフェースの実装クラスになります。
このタグでは、Query.execute(String[]) メソッドが複数回呼ばれます。
これは、DBTableModelの選択された行に対して、繰り返し呼ばれます。

通常、このタグを使用する場合は、queryType="JDBCPrepared" を指定します。

タグにより使用できる／出来ないがありますが、これは、org.opengion.hayabusa.db
以下の Query **** クラスの **** を与えます。
これらは、Query インターフェースを継承したサブクラスです。

2.command

【TAG】 コマンド(NEW, RENEW)をセットします(PisqlUpdateTag, UpdateTag の場合は、ENTRY)。
コマンドは、HTMLから(get/post)指定されますので、CMD_xxx で設定される
フィールド定数値のいつれかを、指定できます。

3.scope

【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)。
"request", "page", "session", "applicaton" が指定できます。
JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、
主にテーブルモデルを管理するスコープを指します。

スコープ	変数の有効範囲	
page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会
request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ
session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。内部的には、同じキーワー
application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。

4.displayMsg

【TAG】 検索結果を画面上に表示するメッセージリソースIDを指定します
(初期値:VIEW_DISPLAY_MSG[=])。
ここでは、検索結果の件数や登録された件数をまず出力し、
その次に、ここで指定したメッセージをリソースから取得して表示します。
件数を表示させたい場合は、displayMsg = "MSG0033"[件検索しました] をセットしてください。
表示させたくない場合は、displayMsg = "" をセットしてください。
(初期値:システム定数のVIEW_DISPLAY_MSG[=@og.value org.opengion.hayabusa.common.SystemData#VIEW_DISPLAY_MSG])。

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
5.names	【TAG】PL/SQLを利用する場合の引数にセットすべき データの名称をCSV形式で複数指定します。 複数ある場合は、カンマ区切り文字で渡します。 names 属性は、queryType に応じて設定可否が異なりますので、ご注意ください。 names なし：JDBC, JDBCUpdate names あり：JDBCCallable, JDBCErrMsg, JDBCUpdate (JDBCUpdateは、names 属性のあり/なし両方に対応しています。) 【TAG】リクエスト情報の クォーティション(') 存在チェックを実施するかどうか[true/false]を設定します (初期値:USE_SQL_INJECTION_CHECK[=true])。
6.quotCheck	SQLインジェクション対策の一つとして、暫定的ではありますが、SQLのパラメータに 渡す文字列にクォーティション(') を許さない設定にすれば、ある程度は防止できます。 数字タイプの引数には、 or 5=5 などのクォーティションを使用しないコードを埋めても、 数字チェックで検出可能です。文字タイプの場合は、必ず (')をはずして、 ' or 'A' like 'A' のような形式になる為、(')チェックだけでも有効です。 (') が含まれていたエラーにする(true)／かノーチェックか(false)を指定します。 初期値は、SystemData#USE_SQL_INJECTION_CHECK です。
7.tableId	【TAG】(通常は使いません)結果のDBTableModelを、sessionに登録するときのキーを指定します。 検索結果より、DBTableModelオブジェクトを作成します。これを、以下のview 等のタグに 渡す場合に、通常は、session を利用します。その場合の登録キーです。 query タグを同時に実行して、結果を求める場合、同一メモリに配置される為、 この tableId 属性を利用して、メモリ空間を分けます。 初期値は、HybsSystem.TBL_MDL_KEY です。
8.dbid	【TAG】(通常は使いません)Queryオブジェクトを作成する時のDB接続IDを指定します。 Queryオブジェクトを作成する時のDB接続IDを指定します。 これは、システムリソースで、DEFAULT_DB_URL 等で指定している データベース接続先 情報に、XX_DB_URL を定義することで、 dbid="XX" とすると、この 接続先を使用して データベースにアクセスできます。
9.xssCheck	【TAG】リクエスト情報の HTMLTag開始/終了文字(><) 存在チェックを実施するかどうか[true/false]を設定します (初期値:USE_XSS_CHECK[=true])。 クロスサイトスクリプティング(XSS) 対策の一環としてless/greater than signについてのチェックを行います。 >< が含まれていたエラーにする(true)／かノーチェックか(false)を指定します。 (初期値:システム定数のUSE_XSS_CHECK[=@og.value.org.opengion.hayabusa.common.SystemData#USE_XSS_CHECK。]
10.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容													
	形式サンプル													
5.5.4.2 og:upload Tag	<p>【TAG】ファイルをアップロードするディレクトリを指定します (初期値:FILE_URL [=filetemp/])。</p> <p>この属性で指定されるディレクトリに、アップロードされたファイルをセーブします。 指定方法は、通常の fileURL 属性と同様に、先頭が、'/' (UNIX) または、2文字目が、 "." (Windows) の場合は、指定のURLそのままのディレクトリに、そうでない場合は、 fileURL = "[@USER.ID]" と指定すると、FILE_URL 属性で指定のフォルダの下に、 さらに、各個人ID別のフォルダを作成して、そこにセーブします。 (初期値:システム定数のFILE_URL [=[@og.value org.opengion.hayabusa.common.SystemData#FILE_URL]])。</p> <p>●形式: <og:upload fileURL="..." maxPostSize="..." /> ●body: なし</p> <p>●Tag定義:</p> <pre><og:upload fileURL filename maxPostSize scope tableId language debug /></pre> <p>●使用例:</p> <p>【query.jsp】</p> <pre><form method="POST" action="result.jsp" enctype="multipart/form-data" target="RESULT"> <table summary="layout" > <tr><og:input type="text" name="submitter" value="[@USER.JNAME]" size="20" msg="MSG0014" /></tr> <tr> <og:input type="file" name="file1" size="30" msg="MSG0015" /> <og:input name="file1_NEW" size="10" lbl="FILENAME" /> </tr><tr> <og:input type="file" name="file2" size="30" msg="MSG0015" /> <og:input name="file2_NEW" size="10" lbl="FILENAME" /> </tr><tr> <og:input type="file" name="file3" size="30" msg="MSG0015" /> <og:input name="file3_NEW" size="10" lbl="FILENAME" /> </tr><tr> <og:column name="writable" value="false" /> </tr> </table></pre> <p>【result.jsp】</p> <pre><og:upload fileURL = "[@USER.ID]" />
 <og:message msg="MSG0003" comment="ファイルの登録が完了しました。" /> <og:view command = "NEW" viewFormType = "HTMLTable" writable = "[@writable]" /> <table> <tr><og:input name="submitter" value="[@submitter]" /></tr> <tr><og:input name="writable" value="[@writable]" /></tr> <tr><og:input name="directory" value="[@directory]" /></tr> <tr><og:input name="file1" value="[@file1]" /></tr> <tr><og:input name="file1_NEW" value="[@file1_NEW]" /></tr> <tr><og:input name="file1_ORG" value="[@file1_ORG]" /></tr> <tr><og:input name="file2" value="[@file2]" /></tr> <tr><og:input name="file2_NEW" value="[@file2_NEW]" /></tr> <tr><og:input name="file2_ORG" value="[@file2_ORG]" /></tr> <tr><og:input name="file3" value="[@file3]" /></tr> <tr><og:input name="file3_NEW" value="[@file3_NEW]" /></tr> <tr><og:input name="file3_ORG" value="[@file3_ORG]" /></tr> </table></pre>													
1.fileURL	<p>【TAG】 ファイルをアップロードするディレクトリを指定します (初期値:FILE_URL [=filetemp/])。</p> <p>この属性で指定されるディレクトリに、アップロードされたファイルをセーブします。 指定方法は、通常の fileURL 属性と同様に、先頭が、'/' (UNIX) または、2文字目が、 "." (Windows) の場合は、指定のURLそのままのディレクトリに、そうでない場合は、 fileURL = "[@USER.ID]" と指定すると、FILE_URL 属性で指定のフォルダの下に、 さらに、各個人ID別のフォルダを作成して、そこにセーブします。 (初期値:システム定数のFILE_URL [=[@og.value org.opengion.hayabusa.common.SystemData#FILE_URL]])。</p>													
2.filename	<p>【TAG】 (通常使いません) ファイルを作成するときのファイル名をセットします。 ファイルを作成するときのファイル名をセットします。 これは、複数同時にアップロードファイル名を変更する時に使用できません。 通常、アップロードされたファイル名を指定する場合、アップロードするinput タグの name 属性に指定する名称 + "_NEW" というリクエスト値を同時に送信すれば、 内部的に関連付けて、ファイル名を更新します。 その場合、クライアントより指定したファイル名は、name属性+"_ORG" という リクエスト値として取得することが可能になります。 name属性 には、最終的に設定されたファイル名がセットされています。 いずれの値も、{@name属性+"_ORG"} や、{@name属性+"_NEW"} として、 アップロードのオリジナルと変更後のファイル名を取得することが出来ます。</p>													
3.maxPostSize	<p>【TAG】 最大転送サイズ(Byte)を指定します(初期値:10485760)。 最大転送サイズを指定します。初期値は、10*1024*1024 = 10MB です。 指定は、Byte 単位で指定します。</p>													
4.scope	<p>【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)。 "request","page","session","applicaton" が指定できます。 JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、 主にテーブルモデルを管理するスコープを指します。</p> <table><tr><th>スコープ</th><th>変数の有効範囲</th><th></th></tr><tr><td>page</td><td>JSPページ内</td><td>そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会</td></tr><tr><td>request</td><td>HTTPリクエスト</td><td>リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ</td></tr><tr><td>session</td><td>HTTPセッション</td><td>初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。内部的には、同じキーワ</td></tr></table>		スコープ	変数の有効範囲		page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会	request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ	session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。内部的には、同じキーワ
スコープ	変数の有効範囲													
page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会												
request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ												
session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。内部的には、同じキーワ												

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
	applicationWebアプリケーション ユーザー間で共有する場合の範囲になります。JSP画面の開発では、まず使うことはありません。
5.tableId	【TAG】（通常使いません）sessionから取得する DBTableModelオブジェクトの ID。 初期値は、HybsSystem.TBL_MDL_KEY です。
6.language	【TAG】 タグ内部で使用する言語コード[ja/en/zh/...]を指定します。 この言語コードに基づいて、表示のラベルをリソースから作成します。
7.debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値: false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

5.5.4.2
og:urlConnect

【TAG】 アクセスする URL を指定します(必須)。
接続するURLを指定します。(例: http://)
?以降のパラメータが含まれていても構いません。
このURL に、keys, vals で指定されたパラメータも追加されます。

JSP

●形式：

<og:urlConnect
url
proxyHost
proxyPort
keys
vals
useSystemUser
authUserPass
display
xslFile
saveFile
soapNameSpace
soapMethodName
tableId
rowKey
colKeys
rtnKeys
encode
>

url
proxyHost
proxyPort
keys, vals

postKey
postFile

authUserPass

useSystemUser

xslFile
display

saveFile

soapNameSpace

soapMethodName

tableId
rowKey
colKeys

rtnKeys
encode

= "http://... " 必須
= "proxy.opengion.org"
= "8080"
= "command, SYSTEM_ID"
= "NEW, GE"
= "true/false" 初期値:true
= "admin:*****" 初期値:admin:*****
= "false/true" 初期値:false
= "filter.xsl"
= "outdata.xml"
= "MyWebService"
= "test"
= "DEFAULT"
= "item"
= "person_id, person_name"
= "version, summary"
= "UTF-8"

</>

url : 接続するURLを指定します。必須属性です。
proxyHost : proxy が存在する場合は、そのホスト名(例: proxy.opengion.org)
proxyPort : proxy が存在する場合は、そのポート番号(例: 8080)
keys, vals : URLの指定時に、パラメータ(引数)を追加します。URLに含めても構いません。
 : SOAPIによる呼び出しの場合の詳細については、keysの属性定義を参照して下さい。
postKey : POST を使って、postFile属性のファイル内容を送信する時のキーです。
postFile : POST を使って、postFile属性のファイル内容を送信します。
 : postFile を指定せず、postKey のみ指定して、BODY部に何か書き込めば、
 : そのBODY部の文字列を POSTの内容として送信します。
authUserPass : Basic認証を使用する場合の接続ユーザー: パスワードを指定します。
 : 接続時のユーザーとパスワードを、USER:PASSWD 形式 で指定します。
 : useSystemUser="false" で何も指定しない場合は、Basic認証を使用しません。
useSystemUser : Basic認証の接続ユーザー: パスワードに、システムユーザーを使用
 : するかどうかを指定します(初期値:true)。
 : true の場合は、SYSTEM:***** を使用します。
xslFile : 接続先データを取得し、そのデータを XSLT変換する場合のXSL ファイルを指定します。
display : 接続した結果のレスポンスを画面に表示するかどうかを指定します(初期値:false)。
 : エンジンの場合、コマンドを投げるだけであれば、結果を取得する必要は
 : ありません。イメージ的には、取得データが、このタグの位置に置き換わります。
 : xslFile が指定されている場合、XSLT変換してセーブします。
saveFile : 接続先データを取得した結果を、ファイル出力します。
 : display="true" と、saveFile を併用することはできません。
 : xslFile が指定されている場合、XSLT変換してセーブします。
soapNameSpace : SOAPIによるWebサービスの呼び出しで、メソッド名及びパラメーターの名前空間を指定します。
 : この名前空間は、通常WSDLファイルのdescriptionsタグのtargetNamespace属性の値により
 : 定義されます。
soapMethodName : SOAPIによるWebサービスの呼び出しで、メソッド名を指定します。
 : WSDLファイルで定義されるoperationタグのname属性の値に相当します。
tableId : 結果のXML ファイルをDBTableModelに変換した際に、登録するTableIdを指定します。
rowKey : XMLをDBTableModelに変換する際の、行を表すタグキーを指定します。
colKeys : XMLをDBTableModelに変換する際の、項目を表すタグキーの一覧を指定します。
 : キーにPARENT_TAG、PARENT_FULL_TAGを指定することで、rowKeyで指定されたタグの
 : 直近の親タグ、及びフルの親タグ名(親タグの階層を">[タグA]>[タグB]>[タグC]>"で表現)を
 : 取得することができます。
rtnKeys : XMLのタグキーを指定して値を取り出します。取り出した値は、{@XX} 形式で処理することが可能です。
encode : データの入出力を行うエンコードを指定します。

●body : POSTデータ

●Tag定義：

<og:urlConnect
url
proxyHost
proxyPort
keys
vals
useSystemUser
authUserPass
display
xslFile
saveFile
postKey
postFile
debug
method
errNeglect
soapNameSpace
soapMethodName
tableId
scope
rowKey
colKeys
rtnKeys
encode
mainTrans
>
... Body ...
</og:urlConnect>

○ 【TAG】 アクセスする URL を指定します(必須)。(必須)
【TAG】 プロキシ経由で接続する場合の、プロキシホスト名を指定します
【TAG】 プロキシ経由で接続する場合の、プロキシポート番号を指定します
【TAG】 アクセスパラメータキーをCSV形式で複数指定します
【TAG】 keys属性に対応する値をCSV形式で複数指定します
【TAG】 Basic認証で接続するユーザーにSYSTEMユーザーを使用するかどうか[true/false]を指定します(初期値:true)
【TAG】 Basic認証を使用して接続する場合のユーザー:パスワードを指定します(初期値:null)
【TAG】 接続の結果を表示するかどうかを指定します(初期値:false)
【TAG】 接続の結果を表示する場合にXSLT変換する場合のファイルを指定します
【TAG】 接続の結果をファイルに保存します
【TAG】 POST を使って、postFile属性のファイル内容を送信する時のキーを指定します
【TAG】 POST を使って、postFile属性のファイル内容を送信します
【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)
【TAG】 送信メソッドを指定します(GET/POST/SOAP)
【TAG】 (通常は使いません) 接続エラーを無視する場合にtrueとします(初期値false)
【TAG】 SOAPIによるWebサービスの呼び出しで、メソッド名及びパラメーターの名前空間を指定します
【TAG】 SOAPIによるWebサービスの呼び出しで、メソッド名を指定します
【TAG】 (通常は使いません) 結果のDBTableModelを、sessionに登録するときのキーを指定します
【TAG】 キャッシュする場合のスコープ[request/page/session/application]を指定します(初期値:session)
【TAG】 結果のXMLをDBTableModelに変換する際に、行のキーとなるタグ名を指定します
【TAG】 結果のXMLをDBTableModelに変換する際に、項目のキーとなるタグ名の一覧を指定します
【TAG】 結果のXMLを変換する際に、パラメータ(Attributes)として取り込むキーの一覧を指定します
【TAG】 データの入出力のエンコードを指定します
【TAG】 (通常使いません) タグで処理される処理がメインとなるトランザクション処理かどうかを指定します(初期値:false)

●例：

アドミン関連

http://localhost:8823/gf/jsp/admin?COMMAND=infomation [状況表示]
http://localhost:8823/gf/jsp/admin?COMMAND=close [フル削除]
http://localhost:8823/gf/jsp/admin?COMMAND=loginUser [ログインユーザー]
http://localhost:8823/gf/jsp/admin?COMMAND=システムパラメータ [システムパラメータ]

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
帳票デモン http://localhost:8823/gf/jsp/REP08/result.jsp?cmd=SET&period=5000&command=NEW&timerTask=org.opengion.hayabusa.report.ReportDaemon&name=ReportDaemon デモン起動 http://localhost:8823/gf/jsp/REP08/result.jsp?cmd=CANCEL&period=5000&command=NEW&timerTask=org.opengion.hayabusa.report.ReportDaemon&name=ReportDaemon デモン停	
Tomcat Manager 画面 http://localhost:8823/manager/reload?path=/ver4 アプリケーションを再ロード	
1.url	【TAG】アクセスする URL を指定します(必須)。 接続するURLを指定します。(例: http://) ?以降のパラメータが含まれていても構いません。 このURL に、keys.vals で指定されたパラメータも追加されます。
2.proxyHost	【TAG】プロキシ経由で接続する場合の、プロキシホスト名を指定します。 接続先が、プロキシ経由の場合、プロキシのホスト名を指定します。 例: proxy.opengion.org
3.proxyPort	【TAG】プロキシ経由で接続する場合の、プロキシポート番号を指定します。 接続先が、プロキシ経由の場合、プロキシのポート番号を指定します。 例: 8080
4.keys	【TAG】アクセスパラメータキーをCSV形式で複数指定します。 アクセスする URL に追加するパラメータのキーを指定します。 カンマ区切りで複数指定できます。 vals 属性には、キーに対応する値を、設定してください。 例: keys="command, SYSTEM_10" vals="NEW, GE" 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスの為です。 [SOAP対応] SOAPによるWebサービスの呼び出しの場合、keys.valsに指定された値より、env:Envelopタグを rootタグとするXMLデータを生成します。 (BODY部分に直接XMLデータを出力することも可能です。) この際、項目名に'>'を含めることで、階層的なXMLデータを表現することができます。 例) [属性定義] keys="param0>AAA, param0>BBB, param1>CCC, DDD" vals="v1, v2, v3, v4" [XMLデータ(※データ部のみ)] <param0> <AAA>v1</AAA> <BBB>v2</BBB> </param0> <param1> <CCC>v3</CCC> <DDD>v4</DDD> 項目の値を"null"とすることで、XMLで言うところの「xsi:nil="true"」のデータを表現することもできます。 また、キー名の先頭を'@'にすることで、項目名に名前空間のPREFIXを付加することができます。 一般的には、JavaやRubyで実装されたWebサービスを呼び出しする場合は、必要ありませんが、 .NETで実装されたWebサービスを呼び出しする場合は、各項目にPREFIXを付与しないと、正しく パラメーターを渡すことができません。 ※現時点では、keysの階層定義は、2階層まで対応しています。 3階層以上のXML構造を定義する場合は、postFile属性によるファイル指定又は、Body部分で直接 XMLデータを記述して下さい。
5.vals	【TAG】keys属性に対応する値をCSV形式で複数指定します。 キーに設定した値を、カンマ区切り文字で複数して出来ます。 指定順序は、キーと同じにしておいて下さい。 例: keys="command, SYSTEM_10" vals="NEW, GE" 分解方法は、CSV変数を先に分解してから、getRequestParameter で値を取得します。 こうしないとデータ自身にカンマを持っている場合に分解をミスの為です。
6.useSystemUser	【TAG】Basic認証で接続するユーザーにSYSTEMユーザーを使用するかどうか[true/false]を指定します(初期値:true)。 useSystemUser="true"(初期値)の場合、URL接続時のコネクションに、Basic認証を 使用しますが、その時のユーザーにシステムユーザー(SYSTEM)を使用します。 useSystemUser="false"の場合は、authUserPass で指定したユーザー:パスワードを 使用します。authUserPass で、何も指定されなかった場合は、Basic認証を使用しません。 初期値は、true(SYSTEMユーザー認証する)です。
7.authUserPass	【TAG】Basic認証を使用して接続する場合のユーザー:パスワードを指定します(初期値:null)。 接続時のユーザーとパスワードを、USER:PASSWD 形式で指定します。 useSystemUser="false"の場合は、ここで指定したユーザーとパスワードを使用します。 その場合に、何も指定しない場合は、Basic認証を使用しません。
8.display	【TAG】接続の結果を表示するかどうかを指定します(初期値:false)。 true で、接続結果を表示します。 false では、何も表示しません(初期値:false) 接続結果を表示する使い方より、admin 画面に接続して、キャッシュクリアするような 使い方が多いと考え、初期値は、false になっています。 xslFile が指定されている場合、XSLT変換して画面表示します。 display="true" と、saveFile を併用することはできません。
9.xslFile	【TAG】接続の結果を表示する場合にXSLT変換する場合のファイルを指定します。 接続先のデータが、XML形式の場合、そのままでは、画面出力できない場合が あります。通常は、HTML形式に変換しますが、その変換に、 XSL ファイルを 指定することが可能です。 display="true" の場合や、saveFile を指定した場合に、適用されます。
10.saveFile	【TAG】接続の結果をファイルに保存します。 接続先のデータを受け取って、ファイルに保存します。その場合、 xslFile が指定されている場合、XSLT変換してセーブします。 display="true" と、saveFile を併用することはできません。
11.postKey	【TAG】POST を使って、postFile属性のファイル内容を送信する時のキーを指定します。 接続先にパラメータ(引数)を投げる場合に、POST を使用できます。 そのときの キーをここで指定します。 POSTするデータは、postFileで指定されたファイルか、BODY部に記述された文字列です。
12.postFile	【TAG】POST を使って、postFile属性のファイル内容を送信します。 接続先にパラメータ(引数)を投げる場合に、POST を使用できます。 そのときの 送信データのファイルをここで指定します。 postKey のみ指定されて、postFile が指定されない場合は、BODY部を送信します。 SOAPによる呼び出しの場合は、ここ(BODY部での定義を含む)で、送信するXMLデータを 定義することができます。
13.debug	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。
14.method	【TAG】送信メソッドを指定します(GET/POST/SOAP)。 URLConnectTagのメソッドの初期設定はGETです。 ここで"POST"(大文字)を指定するとkeys.valsの値セットをPOSTで送信します。 (postKeyが設定されている場合はこの値に関係なくPOSTです)
15.errNeglect	【TAG】(通常は使いません) 接続エラーを無視する場合にtrueとします(初期値false)。 trueにするとConnectで発生したエラーを投げずに処理を続行します。 (標準エラー出力にエラー内容は出力されます)

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

16.soapNameSpace

接続エラーが発生しても処理を中断したくない場合に設定します。
【TAG】 SOAPによるWebサービスの呼び出しで、メソッド名及びパラメーターの名前空間を指定します。
SOAPによるWebサービスの呼び出しで、メソッド名及びパラメーターの名前空間を指定します。
この名前空間は、通常WSDLファイルのdescriptionsタグのtargetNamespace属性の値により定義されます。

17.soapMethodName

【TAG】 SOAPによるWebサービスの呼び出しで、メソッド名を指定します。
SOAPによるWebサービスの呼び出しで、メソッド名を指定します。
WSDLファイルで定義されるoperationタグのname属性の値に相当します。

18.tableId

【TAG】 (通常は使いません) 結果のDBTableModelを、sessionに登録するときのキーを指定します。
検索結果より、DBTableModelオブジェクトを作成します。これを、以下のview 等のタグに渡す場合に、通常は、session を利用します。その場合の登録キーです。
query タグを同時に実行して、結果を求める場合、同一メモリに配置される為、この tableId 属性を利用して、メモリ空間を分けます。
初期値は、HybsSystem.TBL_MD_L KEY です。

19.scope

【TAG】 キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)。
"request", "page", "session", "applicaton" が指定できます。
JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、主にテーブルモデルを管理するスコープを指します。

スコープ	変数の有効範囲	
page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会
request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ
session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。 内部的には、同じキーワー
application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。

20.rowKey

【TAG】 結果のXMLをDBTableModelに変換する際に、行のキーとなるタグ名を指定します。
結果のXMLを表形式に変換する際に、行のキーとなるタグ名を指定します。
このキーのタグを基点として、colKeysで指定されたタグの値を各項目の値として取り込みます。
(詳細は、colKeysのタグリブ属性マニュアルを参照して下さい。)

21.colKeys

【TAG】 結果のXMLをDBTableModelに変換する際に、項目のキーとなるタグ名の一覧を指定します。
結果のXMLをDBTableModelに変換する際に、項目のキーとなるタグ名の一覧を指定します。
rowKeyで行を、colKeysで項目を表現し、rowKeyのタグで囲われた固まりを1つの行とみなします。
このため、colKeysに指定されたキーのタグでも、rowKeyの外にある場合は、取り込み対象となりません。

22.rtnKeys

また、キーにPARENT_TAG、PARENT_FULL_TAGを指定することで、rowKeyで指定されたタグの直近の親タグ、及びフルの親タグ名(親タグの階層を">[タグA]>[タグB]>[タグC]>"で表現)を取得することができます。

23.encode

【TAG】 結果のXMLを変換する際に、パラメータ(Attributes)として取り込むキーの一覧を指定します。
結果のXMLを変換する際に、パラメータ(Attributes)として取り込むキーの一覧を指定します。
ここで指定されたキーのタグの値を取り出し、{@XX}形式(Attributes)として処理できるようにします。

24.mainTrans

【TAG】 データの入出力のエンコードを指定します。
データの入出力のエンコードを指定します。
初期値は、"UTF-8"です。
【TAG】 (通常使いません) タグで処理される処理がメインとなるトランザクション処理かどうかを指定します(初期値:false)。
(通常使いません) タグで処理される処理が、メインとなるトランザクション処理かどうかを指定します。(初期値:false)
この値は、ファイルダウンロード処理に影響します。この値がtrueに指定された時にcommitされたDBTableModelが
ファイルダウンロードの対象の表になります。

このパラメーターは、通常、各タグにより実装され、ユーザーが指定する必要はありません。
但し、1つのJSP内でDBTableModelが複数生成される場合に、前に処理したDBTableModelについてファイルダウンロードをさせたい場合は、後ろでDBTableModelを生成するタグで、明示的にこの値をfalseに指定することで、ファイルダウンロード処理の対象から除外することができます。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:userInfo JSP	<p>【TAG】 コマンド(SET, GET, NVLEGT, REMOVE, SQL)をセットします(初期値:SET)。 コマンドは、HTMLから(get/post)指定されますので、CMD_xxx で設定される フィールド定数値のいずれかを、指定できます。 何も設定されない、または、null の場合は、“SET” が初期値にセットされます。 SQL の場合、検索結果の戻り値が複数存在する場合は、最初の 1 件目のみ使用します。</p>
<p>●形式 : <og:userInfo command=[...] key=[...] value=[...] /> ●body : あり</p> <p>●Tag定義 : <og:userInfo command key value dbid save caseKey caseVal debug > ... Body ... </og:userInfo></p> <p>●使用例 <og:userInfo command key value dbid ></p>	
1.command	<p>【TAG】 コマンド(SET, GET, NVLEGT, REMOVE, SQL)をセットします(初期値:SET)。 コマンドは、HTMLから(get/post)指定されますので、CMD_xxx で設定される フィールド定数値のいずれかを、指定できます。 何も設定されない、または、null の場合は、“SET” が初期値にセットされます。 SQL の場合、検索結果の戻り値が複数存在する場合は、最初の 1 件目のみ使用します。</p>
2.key	<p>【TAG】 UserInfo に登録するキーをセットします(予約語:JNAME, ID, PASSWD, IDNO, INFO, LANG, ROLE, IPADDRESS, LOGINTIME)。</p>
3.value	<p>UserInfo に登録するキーをセットします。 【TAG】 UserInfo に登録する値をセットします。</p>
4.dbid	<p>UserInfo に登録する値をセットします。 【TAG】 (通常は使いません)Queryオブジェクトを作成する時のDB接続IDを指定します。</p>
5.save	<p>Queryオブジェクトを作成する時のDB接続IDを指定します。 【TAG】 UserInfo に登録した値を永続化するかを指定します。 UserInfo に登録した値を永続化するかを指定します。 trueが指定された場合、UserInfoに設定された値は、GE20(ユーザー定数)に保存され、 UserInfoが再作成されるタイミングで自動的に復元されます。 ここで、登録された値は、そのユーザーの全てのロールに対して有効となります。 初期値は、false(永続化しない)です。</p>
6.caseKey	<p>【TAG】 このタグ自体を利用するかどうかの条件キーを指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。</p>
7.caseVal	<p>【TAG】 このタグ自体を利用するかどうかの条件値を指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。</p>
8.debug	<p>【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。</p>

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

5.5.4.2
og:value
JSP

【TAG】 コマンド(SET, GET, REMOVE, SQL, SETTBL, GETTBL, KEYTBL, CLEAR, SETMEM) をセットします (初期値:SET)。
コマンドは、HTMLから (get/post) 指定されますので、CMD_xxx で設定される
フィールド定数値のいつれかを、指定できます。
何も設定されない、または、null の場合は、“SET” が初期値にセットされます。

CLEAR 以外のすべての処理は、指定のスコープの内部キャッシュ情報に対して行われます。

	command	名称	機能
SET	セット	指定のキーに、value値を登録します。	
GET	ゲット	指定のキーの値を画面に出力します。	
REMOVE	リムーブ	指定のキーの値を削除します。	
CLEAR	クリア	セッション/アプリケーションスコープのキャッシュ情報をクリアします。	
SQL	S Q L	指定のSQL文の実行結果を、カラム名をキーとして設定します。	
SQLGET	S Q L ゼット	指定のSQL文の実行結果を、画面に出力します。(2項目以降は無視されます)。	
SETTBL	セットテーブル	指定のキーに、DBTableModel の選択されたカラム(横持ち)の値を登録します。	
GETTBL	ゲットテーブル	指定のキーに、DBTableModel の選択されたカラム(横持ち)の値を画面に出力します。	
KEYTBL	キーテーブル	指定のキーに、DBTableModel の選択されたロウ(縦持ち)の値を登録します。	
SETMEM	セットメモリ	指定のキーに、value値を内部キャッシュに登録します。	

●形式: <og:value command="SET" key="ABC" value="123" />
●body: あり

●Tag定義:
<og:value
command
action
key
value
defaultVal
nullSet
separator
parameter
useMultiRows
scope
tableId
tblScope
dbId
xssCheck
caseKey
caseVal
toVal
fromVal
debug
> ... Body ...
</og:value>

●使用例

- <og:value command="SET" key="ABC" value="123" />
- <og:value command="SQL">SELECT 1 TEST FROM DUAL</og:value>
- <og:value command="SET" key="DEF" value="{@NOCOMMAND}" defaultValue="0000" />
- <og:value command="GET" key="NOKEY" defaultValue="NODATA" />
- <og:value command="SET" key="{@ABC}4" value="TEST1234" />
- <og:value command="GET" key="ABC" />
- <og:value command="SET" action="LOWER" key="LOWERTEST" value="ABCDEF" />
- <og:value command="GET" key="LOWERTEST" />
- <og:value command="GET" action="UPPER" key="LOWERTEST" />
- <og:value command="REMOVE" key="ABC" />

	UPPER	LOWER	MESSAGE	APPEND	ALL_APPEND	MAP	ALL_MAP	LIST	ALL_LIST	DAY_WEEK	MERGE	FIRST	ROW_APPEND	REPLACE	SUBSTR	SPLIT
SET	○	○	○	○	×	×	×	×	×	○	○	×	×	×	○	○
GET	○	○	○	×	×	×	×	×	×	○	○	×	×	×	○	○
REMOVE	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
CLEAR	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
SQL	×	×	×	×	○	○	○	○	×	×	×	○	×	×	×	×
SETTBL	○	○	○	○	○	○	○	○	○	×	×	×	×	×	○	○
GETTBL	○	○	○	○	○	×	×	×	×	○	×	×	×	×	○	○
KEYTBL	○	○	○	×	×	×	×	×	×	○	×	×	×	×	○	○
SETMEM	○	○	○	×	×	×	×	×	×	×	×	×	×	×	○	○
SQLGET	×	×	×	×	×	×	×	×	×	×	×	○	×	×	×	×

1.command

【TAG】 コマンド(SET, GET, REMOVE, SQL, SETTBL, GETTBL, KEYTBL, CLEAR, SETMEM) をセットします (初期値:SET)。
コマンドは、HTMLから (get/post) 指定されますので、CMD_xxx で設定される
フィールド定数値のいつれかを、指定できます。
何も設定されない、または、null の場合は、“SET” が初期値にセットされます。

CLEAR 以外のすべての処理は、指定のスコープの内部キャッシュ情報に対して行われます。

	command	名称	機能
SET	セット	指定のキーに、value値を登録します。	
GET	ゲット	指定のキーの値を画面に出力します。	
REMOVE	リムーブ	指定のキーの値を削除します。	
CLEAR	クリア	セッション/アプリケーションスコープのキャッシュ情報をクリアします。	
SQL	S Q L	指定のSQL文の実行結果を、カラム名をキーとして設定します。	
SQLGET	S Q L ゼット	指定のSQL文の実行結果を、画面に出力します。(2項目以降は無視されます)。	
SETTBL	セットテーブル	指定のキーに、DBTableModel の選択されたカラム(横持ち)の値を登録します。	
GETTBL	ゲットテーブル	指定のキーに、DBTableModel の選択されたカラム(横持ち)の値を画面に出力します。	
KEYTBL	キーテーブル	指定のキーに、DBTableModel の選択されたロウ(縦持ち)の値を登録します。	
SETMEM	セットメモリ	指定のキーに、value値を内部キャッシュに登録します。	

【TAG】 アクション(UPPER, LOWER, MESSAGE, APPEND, ALL_APPEND, LIST, ALL_LIST, DAY_WEEK, MERGE, FIRST, ROW_APPEND) をセットします。
アクションは、HTMLから (get/post) 指定されますので、ACT_xxx で設定される
フィールド定数値のいつれかを、指定できます。

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

無指定の場合は、なにもしません。

action	名称	機能
UPPER	アッパー(大文字化)	value値を大文字に変換します。
LOWER	ローワー(小文字化)	value値を小文字に変換します。
MESSAGE	メッセージ変換	引数をメッセージリソースのキーとして、メッセージ変換します。
APPEND	データアペンド	複数リクエストや複数選択時に値を連結します。
ALL_APPEND	オールアペンド	SETTBL, GETTBL 時に、チェック行以外の全行を対象に値の連結を行います。
MAP	マップ	検索結果やTableModelの値を元に値を一括設定します。
LIST	リスト	複数リクエストや複数選択時に値をArrayListにセットします。
ALL_LIST	オールリスト	チェック行以外の全行を対象に値をArrayListにセットします。
DAY_WEEK	日付前方まるめ	日付型文字列(YYYYMMDD) の値を、月曜日に変換します。 指定日が日曜日の場合は、次の日(月曜日)に進めます。その他は、週始めに戻します。
MERGE	データのマージ	重複を除く、ユニークな値に、マージします。(カンマで分解、separatorで合成)
FIRST	1件目取得	最初の1件目を強制的に選択状態にして、処理を行います。
ROW_APPEND	検索結果の連結	検索結果の行方向のデータを連結します。
REPLACE	文字列置換	value の値から、指定された正規表現(from)の部分文字列を、部分文字列(to)で置換します。
SUBSTR	部分文字列	value の値から、指定された(from)から(to)の部分文字列を作成します。
SPLIT	文字列分割	value の値から、指定されたseparatorで分割した文字列を作成します。(key+0～連番)

3.key

4.value

5.defaultVal

6.nullSet

7.separator

8.parameter

9.useMultiRows

10.scope

11.tableId

12.tblScope

13.dbid

14.xssCheck

15.caseKey

16.caseVal

17.toVal

18.fromVal

19.debug

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

5.5.4.2
og:view

JSP

[TAG] ViewForm オブジェクトを作成するクラス(ViewForm_**** クラスの ****)を指定します。
これは、org.opengion.hayabusa.html 以下の ViewForm_**** クラスの **** を
与えます。これらは、ViewForm インターフェースを継承したサブクラスです。
属性クラス定義の {@link org.opengion.hayabusa.html.ViewForm ViewForm} を参照願います。

●形式 : <og:view />

<og:view
 viewFormType = "HTMLTable"
 command
 = "[{@command}]"
 startNo
 = "0"
 pageSize
 = "20"
 language
 = "ja" />

viewFormType : ViewFormオブジェクトを作成する時のキー(オプション) ViewFormFactory参照
command : 処理コマンド (PREV, NEXT, FIRST, LAST, VIEW) (オプション)
startNo : データの書き始めの初期値を指定(オプション)
pageSize : データのページ数(画面表示件数)を指定(オプション)
language : ロケール(オプション)

●body : あり

●Tag定義 :
<og:view
 viewFormType
 command
 scope
 startNo
 pageSize
 pagePlus
 writable
 checked
 rowspan
 skip
 columnWritable
 noWritable
 columnDisplay
 clmSeqDisplay
 noDisplay
 columnGroup
 noGroup
 groupClass
 tableSorterKeys
 tableId
 viewFormId
 viewLinkId
 viewMarkerId
 editMarkerId
 selectedType
 numberType
 optionTypeAttributes
 noMessage
 backLinkCount
 headerSkipCount
 useCheckControl
 useTableSorter
 useParam
 useConsistency
 bgColorCycle
 bgColorClmCln
 useScrollBar
 loadMarker
 useSelectedRow
 useSelRowColor
 useHighlightRow
 useAfterHtmlTag
 writableControl
 popupReturnKeys
 clazz
 language
 height
 width
 fileDwnClms
 skipNoEdit
 noTransition
 useHighlightCheckOnly
 columnBulkSet
 caseKey
 caseVal
 debug
 > ... Body ...
</og:view>

●使用例
 <og:view viewFormId="HTMLTable" command="NEW" startNo="0" pageSize="5" />

1.viewFormType

[TAG] ViewForm オブジェクトを作成するクラス(ViewForm_**** クラスの ****)を指定します。
これは、org.opengion.hayabusa.html 以下の ViewForm_**** クラスの **** を
与えます。これらは、ViewForm インターフェースを継承したサブクラスです。
属性クラス定義の {@link org.opengion.hayabusa.html.ViewForm ViewForm} を参照願います。

2.command

[TAG] コマンド(PREV, NEXT, FIRST, LAST, NEW, RENEW, VIEW, REVIEW, RESET)をセットします。
コマンドは、HTMLから(get/post)指定されますので、CMD_xxx で設定される
フィールド定数値のいつれかを、指定できます。

3.scope

[TAG] キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)。
"request", "page", "session", "applicaton" が指定できます。
JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、
主にテーブルモデルを管理するスコープを指します。

スコープ	変数の有効範囲	
page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会
request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やボ
session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。内部的には、同じキーワ

バージョン タグ名 HTML EXT BODY要素	内容
	<p>形式サンプル</p> <pre>applicationWebアプリケーション ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。</pre>
4.startNo	<p>【TAG】表示データを作成する場合の表示の開始行番号をセットします。 セットされていない場合は、変更しません。</p>
5.pageSize	<p>【TAG】表示データを作成する場合の1ページの行数をセットします。 セットされていない場合は、変更しません。</p>
6.pagePlus	<p>【TAG】1ページの行数の増加分をセットします(初期値:0)。 セットされていない場合は、0(増加なし)です。</p>
7.writable	<p>【TAG】行が書き込み可能かどうか[true/false]を設定します (初期値:DBTableModel#DEFAULT_WRITABLE[=true])。 書き込み許可する場合は、true、許可しない場合は、true 以外を設定します。 ここで、false を設定すると全行書き込み禁止になります。true を設定した場合、デフォルトで書き込み許可が与えられ、先頭カラムが writable であれば、その値を取り込んで、行ごとの書き込み許可/不許可が決定されます。 writable, checked, useCheckControl については、writable が最優先で次に、useCheckControl="2"(または、checked)、次に、checked を判定します。 checked="true"の場合は、useCheckControl が、"1"(true)以上の場合、"2"(checked)に設定されます。 (初期値:DBTableModel#DEFAULT_WRITABLE[=@og. value DBTableModel#DEFAULT_WRITABLE])。</p>
8.checked	<p>【TAG】行のチェックボックスの初期値を設定するかどうか[true/false]を指定します(初期値:null)。 書き込み可能な行(rowWritable == true)のチェックボックスに対して初期値を選択済みにするか、非選択済みにするかを指定します。 writable, checked, useCheckControl については、writable が最優先で次に、useCheckControl="2"(または、checked)、次に、checked を判定します。 checked="true"の場合は、useCheckControl が、"1"(true)以上の場合、"2"(checked)に設定されます。</p>
9.rowspan	<p>【TAG】表示データを作成する場合のフォーマットの行数をセットします(初期値:2)。 表示データを作成する場合のフォーマットの行数をセットします。</p>
10.skip	<p>【TAG】NEXT等で、選択行以外をスキップするかどうか[true/false]を指定します (初期値:ViewForm#DEFAULT_SKIP[=false])。 明細情報の登録/編集時に、必要な箇所のみを順番に処理する場合に使用します。 (初期値:ViewForm#DEFAULT_SKIP[=@og. value ViewForm#DEFAULT_SKIP])。</p>
11.columnWritable	<p>【TAG】書き込み可能カラム名を、カンマ区切りで与えます。 これは、書き込み不可カラム名の指定(noWritable)と同時にセットすることは出来ません。 なお、カラム名の代わりに、“null”を指定すると、なにも指定しないことになります。つまり、noWritable にすべてのカラムを指定することと同じになります。(デフォルトなので、あまり意味はありません)。 “*”を指定すると、すべてのカラムを(columnWritable)指定したことになります。</p>
12.noWritable	<p>【TAG】書き込み不可カラム名を、カンマ区切りで与えます。 これは、書き込み可能カラム名の指定(columnWritable)と同時にセットすることは出来ません。 なお、カラム名の代わりに、“null”を指定すると、なにも指定しないことになります。つまり、columnWritable にすべてのカラムを指定することと同じになります。 “*”を指定すると、すべてのカラムを(noWritable)指定したことになります。</p>
13.columnDisplay	<p>【TAG】表示可能カラム名を、カンマ区切りで与えます。 これは、表示不可カラム名の指定(noDisplay)と同時にセットすることは出来ません。 “*”を指定すると、すべてのカラムを(columnDisplay)指定したことになります。</p>
14.clmSeqDisplay	<p>【TAG】表示可能カラム名を、カラム順に、カンマ区切りで与えます。 これは、ViewForm_HTMLSeqClnTable 用に特別に用意された属性です。 機能的には、columnDisplay 属性で、カラムの表示順まで指定できる機能を用意すべきですが、過去のタグとの互換性の関係で、別に用意しています。(この属性を使用すると、通常の表示より、処理速度が低下します。) 例: “OYA, KO, HJO, SU, DYSET, DYUPD” この順番で表示され、指定以外のカラムは表示されません。</p>
15.noDisplay	<p>【TAG】表示不可カラム名を、カンマ区切りで与えます。 これは、表示可能カラム名の指定(columnDisplay)と同時にセットすることは出来ません。 “*”を指定すると、すべてのカラムを(noDisplay)指定したことになります。</p>
16.columnGroup	<p>【TAG】同一表示データをまとめる(表示しない)カラム名を、カンマ区切りで与えます。 これは、カラムのグループ化指定を行います。 同じカラム番号で一つ上の行番号と同じ内容の場合に、特殊な表示方法で表示します。 対応は、表示(Renderer)時のみとします。 特殊な表示方法の指定は、groupClass 属性で指定します。 (例: groupClass にアルファベット以外の文字・・そのまま表示、アルファベットはクラス属性を付与します。)カラム単位なので、新しいゼブラによる色分けは行いません。(任意のカラムに適用できる為) また、ファイル出力などのデータ自身は変更されていませんので、そのままデータが落ちます。</p> <p>これは、同一表示データをまとめないカラム名の指定(noGroup)と同時にセットすることは出来ません。 “*”を指定すると、すべてのカラムを(columnGroup)指定したことになります。</p>
17.noGroup	<p>【TAG】同一表示データをまとめないカラム名を、カンマ区切りで与えます。 これは、カラムのグループ化指定で、同一でもまとめないカラムを指定するほうが簡単な場合に使用します。例えば、キー情報以外は、グループ化指定したい場合などに便利です。 動作は、columnGroup の動きと同じです。(まとめないカラムを指定するだけです。)</p> <p>これは、同一表示データをまとめるカラム名の指定(columnGroup)と同時にセットすることは出来ません。 “*”を指定すると、すべてのカラムを(noGroup)指定したことになります。</p>
18.groupClass	<p>【TAG】同一表示データをまとめる場合の、表示方法を指定します。 これは、カラムのグループ化指定(columnGroup)を行う場合の、まとめ表示の方法を指定します。 なにも指定しない場合は、まとめカラムは、表示しない(ゼロ文字列)になります。 その場合、先の行と同じ場合も、変更されたカラムが、NULL(ゼロ文字列)であった場合も、同じ用に表示されるため、区別が付きません。 そこで、前の行と同じデータの場合に、特殊な処理を行うことで、区別できるようにします。 指定方法が特殊なので、注意が必要です。 記号の場合: 記号(←, →, ↓, * など)が指定された場合は、そのままの文字に置き換えられます。 初期値は、空文字列(ゼロ文字列)です。 アルファベットの場合: アルファベット(a-Z, A-Z)の場合は、元の文字 例えば、バックカラーに色を付ける、文字を薄い灰色にする、などできます。 ただし、データ量が圧倒的に増えるため、大量のデータ等で使用するのを避けたほうが良いと思います。</p>
19.tableSorterKeys	<p>【TAG】カラム名リンクソートを表示するカラム名を、カンマ区切りで与えます。 ヘッダーにソート用リンクを作成する useTableSorter 属性 に対して、カラム個別に作成する場合のカラム名をカンマ区切り文字で指定します。 この sortKeys 属性は、useTableSorter 属性 と無関係に、指定したカラムのみ、リンクを表示します。 “*”を指定すると、すべてのカラムを(tableSorterKeys)指定したことになります。 例: “OYA, KO, HJO, SU, DYSET, DYUPD”</p>
	<p>【TAG】(通常使いません)sessionから所得する DBTableModelオブジェクトの ID。 表示処理後に、(内部ポインタを書き換えた)DBTableModelオブジェクトを</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
20.tableId	同じキーで、sessionに登録します。 初期値は、HybsSystem.TBL_MD_L KEY です。 [TAG] (通常使いません)sessionから取得する ViewForm オブジェクトの ID。 表示処理後に、(内部ポインタを書き換えた)ViewForm オブジェクトを 同じキーで、sessionに登録します。 初期値は、HybsSystem.VIEWFORM_KEY です。
21.viewFormId	[TAG] (通常つかいません)requestから取得する ViewLink に対応する Attributes オブジェクトの ID。 ViewForm オブジェクトに、ViewLink オブジェクトをこのキーより取得することにより、 カラムに対して、リンク情報を付加して表示させる。 初期値は、HybsSystem.VIEWLINK_KEY です。
22.viewLinkId	[TAG] (通常使いません)requestから取得する ViewMarker に対応する Attributes オブジェクトの ID。 ViewForm オブジェクトに、ViewMarker オブジェクトをこのキーより取得することにより、 カラムに対して、マーカ情報を付加して表示させる。 初期値は、HybsSystem.VIEW_MARK_KEY です。
23.viewMarkerId	[TAG] (通常使いません)requestから取得する EditMarker に対応する Attributes オブジェクトの ID。 ViewForm オブジェクトに、EditMarker オブジェクト(実際は、ViewMarker オブジェクト)を このキーより取得することにより、カラムに対して、マーカ情報を付加して表示させる。 初期値は、HybsSystem.EDIT_MARK_KEY です。
24.editMarkerId	[TAG] 表示時の選択用オブジェクトのタイプ[checkbox/radio/hidden]を指定します(初期値:checkbox)。 複数選択可能時は“checkbox”を指定します。 一つだけ選べる場合は、“radio”を指定します。 隠しフィールドで全件を選ぶ場合は、“hidden”を指定します。 初期値は、“checkbox”です。
25.selectedType	[TAG] 表示時の行番号の表示方法[sequence/none/delete/skip=カラム名/view=カラム名]を指定します(初期値:sequence)。 ・sequenceは、1から始まる連番です。 ・none を指定すると、番号は表示されません。 ・delete を指定すると、行番号表示そのもののタグを取り除きます。(3.5.5.0 追加) ・skip=カラム名を指定すると、そのカラムの値が NULL の場合は、番号をスキップします。 ・view=カラム名を指定すると、そのカラムの値が、使用されます。(ラベルは付きません) skip=XX と、view=XX は、-の前後にスペースを入れないで下さい。 初期値は、“sequence”です。
26.numberType	[TAG] テーブル等のチェックボックスに属性を付加します。 JavaScript などの HTML基本タグ以外の属性を、そのまま チェックボックス/ラジオボタン等に使用します。 そのときに、(')は、(')に置き換えます。
27.optionTypeAttributes	[TAG] 検索結果メッセージを表示しないかどうか[true/false]を設定します(初期値:false[表示する])。 初期値は、表示する(true以外)です。
28.noMessage	[TAG] ページの先頭へのリンクの間隔をセットします (初期値:VIEW_BACK_LINK_COUNT[=0])。 0 にセットすると、リンクは作成されません。 テーブルのヘッダー固定ができないケースでは、データ件数が多いと、登録ボタンが見えなくなります。 その場合、スクロールして上に上がるより、行番号に作成されリンクで、一気に上に戻るほうが早く処理できます。 その場合、段組み等していると、間隔を少なくしておく必要が出てきます。 ヘッダー固定を利用する場合は、不要です。 (初期値:システム定数のVIEW_BACK_LINK_COUNT[=@og.value org.opengion.hayabusa.common.SystemData#VIEW_BACK_LINK_COUNT])。
29.backLinkCount	[TAG] ヘッダーを出力する間隔をセットします (初期値:VIEW_HEADER_SKIP_COUNT[=0])。 0を指定すると、繰り返ししません(つまり、ヘッダーを最初に表示するだけです。) 数字を指定すると、その回数毎に、ヘッダーをデータの行に挿入します。 (初期値:システム定数のVIEW_HEADER_SKIP_COUNT[=@og.value org.opengion.hayabusa.common.SystemData#VIEW_HEADER_SKIP_COUNT])。
30.headerSkipCount	[TAG] チェックボックスの全チェックを選択するJavaScript の機能を使用するかどうかを指定します (初期値:VIEW_USE_CHECK_CONTROL[=true])。 “true” で、コントロール用のチェックボックスが現れて、この機能を使用することができるようになります。 false は、従来どおりです。 使用するにあたり、jsp/compn/default.js にJavaScriptを設定しておきます。(設定済み) false は、0でも代用可能です。true は、1でも代用可能です。 checked は、2でも代用可能です。checked は、初期値を、チェック済みにします。 注意：0,1,2,true,false,checked 以外の文字をセットすると、エラーが発生します。 writable,checked,useCheckControl については、writable が最優先で 次に、useCheckControl=“2”(または、checked)、次に、checked を判定します。 checked=“true”の場合は、useCheckControl が、“1”(true)以上の場合、 “2”(checked)に設定されます。 (初期値:ユーザー定数のVIEW_USE_CHECK_CONTROL[=@og.value org.opengion.hayabusa.common.SystemData#VIEW_USE_CHECK_CONTROL])。
31.useCheckControl	[TAG] ヘッダーにソート用リンクを作成するかどうか[true/false]を指定します (初期値:VIEW_USE_TABLE_SORTER[=true])。 “true” で、ヘッダーにソート用リンクを作成します。 false は、従来どおりです。 検索済みのDBTableModelを、メモリ上でソートし、再表示させます。 (初期値:ユーザー定数のVIEW_USE_TABLE_SORTER[=@og.value org.opengion.hayabusa.common.SystemData#VIEW_USE_TABLE_SORTER])
32.useTableSorter	[TAG] パラメータファイルの使用をかどうか[true/false]を指定します(初期値:false)。 “true” で、パラメータファイルを読み込むため、BODY部を処理します。 false は、パラメータファイルを使用しません。 パラメータファイルは、各ViewFormのサブクラス用に用意します。 初期値は、false です。(false)
33.useParam	[TAG] Consistency キー による整合性チェックを行うかどうか[true/false]を指定します(初期値:true)。 検索結果を DBTableModel にセットする時に、整合性キーの Consistency キーを 作成します。これを、Viewタグでhidden出力しておき、Entryタグでデータ書き換え時に 整合性チェックを行います。これは、IEの戻るボタンで戻った場合に、画面の キーと検索結果の DBTableModel の内容が一致しない場合のエラーチェックに なります。 この属性は、何らかのケースで、このエラーチェックを行いたくない場合に、 false に設定することで、整合性チェックを行いません。 初期値は、true(整合性チェックを行う)です。
34.useConsistency	[TAG] テーブルのバックグラウンドカラーの入れ替えのサイクルをセットします(初期値は、1:ゼブラ)。 0(ゼブラなし)、-1(ワーニング)、-2以下(エラー)、1(ゼブラ)、2以上(行数まとめ) 初期値は、1(ゼブラ)です。
35.bgColorCycle	[TAG] テーブルのバックグラウンドカラーのクラスに指定するカラム名を指定します。 通常のゼブラ模様は、tr 属性のクラス指定として、row_0、row_1 が指定されます。 その代わりに、ここで指定されたカラムの値がクラス属性として設定されます。 (指定するのはカラム名です。行単位にクラス属性の値を変えることが可能です。) 選択行(row_sel)は、優先して使用されます。 出力されるクラス名は、“row_” + 属性値 になります。
36.bgColorClsCIm	[TAG] スクロールバー用のDIV要素を出力するかどうか[true/false]を指定します (初期値:VIEW_USE_SCROLLBAR[=true])。 スクロールバー対応する為、テーブルの先頭に、DIV要素を出力します。 (初期値:ユーザー定数のVIEW_USE_SCROLLBAR[=@og.value org.opengion.hayabusa.common.SystemData#VIEW_USE_SCROLLBAR])。 ※ 互換性の関係より、false になっています。
37.useScrollBar	[TAG] viewLink や viewMarker の情報取り込みを制御する(初期値は、NEW, RENEW, REVIEW, RESET)。 viewLink や viewMarker の情報取り込みは、command が、NEW, RENEW, REVIEW, RESET の場合のみ行われています。 この属性で、“true”を指定すると、コマンドに無関係に取り込みます。 “false”の場合は、form にキャッシュしているリンク、マーカを削除します。

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
38.loadMarker	取り込み時に、viewLink や viewMarker オブジェクトが存在しない場合は、何もしません。これは、viewMarker オブジェクト自身がすでに設定済みの場合 (VIEWなど) NEXTやPREVでは、viewMarkerオブジェクトは、リクエスト変数に設定されない為です。この場合は、登録しませんが、既存の viewMarkerオブジェクトを持っているはずなので、なにも設定しなくても viewMarker は適用されます。 初期値は、NEW, RENEW, REVIEW, RESET です。
39.useSelectedRow	【TAG】SEL_ROW機能[true:有効/false:無効]を指定します (初期値:VIEW_USE_SELECTED_ROW[=true])。 SEL_ROW機能は、戻るリンクから戻ったときや、登録したときに、選択していた行番号や、先にクリックされた行番号を返す機能で、SEL_ROW リクエストに設定された行がマーカーされます。この機能の有効(true)/無効(false)を指定します。 この属性で、“true” を指定すると、有効になります。 “false” の場合は、無効です。 (初期値:ユーザー定数のVIEW_USE_SELECTED_ROW[=@og.value org.opengion.hayabusa.common.SystemData#VIEW_USE_SELECTED_ROW])。
40.useSelRowColor	【TAG】SEL_ROW使用時の選択行色変更[true:有効/false:無効]を指定します (初期値:VIEW_USE_SELROW_COLOR[=true])。 SEL_ROW機能は、現在選択されている行まで、スクロールする機能ですが、そのとき、選択行にマーカーするかどうかを、指定します。 この属性で、“true” を指定すると、有効になります。 “false” の場合は、無効です。 (初期値:ユーザー定数のVIEW_USE_SELROW_COLOR[=@og.value org.opengion.hayabusa.common.SystemData#VIEW_USE_SELROW_COLOR])。
41.useHilighRow	【TAG】HIGHLIGHT_ROW機能[true:有効/false:無効]を指定します (初期値:VIEW_USE_HIGHLIGHT_ROW[=true])。 HIGHLIGHT_ROW機能は、チェックボックスやラジオボタンで選択した際に、どの行が選択されたか判るように行のハイライト表示を行う機能です。 さらに、行の任意の箇所をクリックしても、チェックが入り、ハイライト表示されます。この属性で、“true” を指定すると、有効になります。 “false” の場合は、無効です。 (初期値:ユーザー定数のVIEW_USE_HIGHLIGHT_ROW[=@og.value org.opengion.hayabusa.common.SystemData#VIEW_USE_HIGHLIGHT_ROW])。
42.useAfterHtmlTag	【TAG】処理時間やページサイズなどの情報出力[true:有効/false:無効]を指定します(初期値:true)。 ViewForm で、CustomData などの 非HTML表示ビューを使用する場合、最後に、処理時間とページサイズ (NEXT時に使用) などの情報を、hidden で出力していますが、これを制御するフラグをセットします。 true(有効)にすると、これらのHTMLが出力されます。false にすると、出力されません。 初期値は、true(有効) です。
43.writableControl	【TAG】設定値に “ ” が含まれている場合にレンダラーを使用するカラムをCSV形式で指定します。 これは、従来の カラム定義の WRITABLE エディターと同等の働きを行うようにカラム属性を指定します。 WRITABLE エディターは、設定値にアンダーバー “ ” が含まれている場合に、その値を書込み禁止にする機能です。これは、エディター自身が値を判断して書き込み許可か禁止かを判断しています。 この動きを汎用的にするため、指定のカラムをカンマ区切り文字 (CSV) で指定することにより、レンダラーとエディターを設定値によって動的に切り替える機能を実現します。 その場合、表示/編集ともに、先頭のアンダーバーは削除されます。 また、全カラムについて、有効にする場合は、writableControl=”*” を設定します。
44.popupReturnKeys	【TAG】ogPopup で検索結果の値を返すキーを、CSV形式で指定します。 popup の検索結果を返す画面で、結果のラジオボタンにイベントセットします。 この場合、オープンもとのwindow に値を返しますが、そのキーをCSV形式で指定します。 なお、このメソッドは、一覧表示 (HTMLTable) 関係のビューのみでサポートしていますが、チェックメソッドの関係で、それ以外のビューに適用しても素通りするようにします。(エラーにしません)
45.clazz	【TAG】table要素に対して class 属性を設定します(初期値:viewTable)。 Javaの言語使用上、class で作成できないため、代用として clazz を使用している。 html で作成される属性は、class で作成される。 初期値は、“viewTable” です。
46.language	【TAG】タグ内部で使用する言語コード[ja/en/zh/...]を指定します。 この言語コードに基づいて、表示のラベルをリソースから作成します。
47.height	【TAG】ビューの高さを指定します。 id=posDivのDIV要素に高さを指定します。 javascriptからこの高さを見てビューの高さを決定します。 単位も指定して下さい。
48.width	【TAG】ビューの幅を指定します。 id=posDivのDIV要素に幅を指定します。 javascriptからこの値を見てビューの幅を決定します。 単位も指定して下さい。(%)指定推奨)
49.fileDwnClms	【TAG】ファイルダウンロード時の抜出カラム名 (CSV形式) を指定します。 fileUDのファイル抜出時のカラムをカンマ形式で指定します。
50.skipNoEdit	【TAG】編集行のみを表示するかを指定します。 編集行のみを表示するかを指定します。trueがセットされた場合、編集行 (改変Cがついている行) のみが出力されます。
51.noTransition	【TAG】(通常は使いません) 画面遷移を行わない形式の登録方法を使用するかを指定します。 画面遷移なしの登録を行うかどうかを指定します。 trueが指定された場合、HTML出力時に、各行に対して、行番号と改変Cが属性情報として付加されます。 ここで出力された情報を元に、JavaScriptの入れ替え処理を行います。
52.useHilighCheckOnly	この値は、og:headタグで設定値、または前画面からの値を継承するため、通常、この属性ではセットしません。 【TAG】選択行ハイライトの制御をチェックボックスのみで有効にするかを指定します (初期値:VIEW_USE_HIGHLIGHT_CHECK_ONLY[=false])。 選択行ハイライトの制御をチェックボックスのみで有効にするかを指定します。 trueの場合、チェックボックスのみで有効になります。 falseの場合、行全体の各要素で有効になります。 trueにした場合、チェックボックスが存在しない行に対しては、ハイライトすることはできません。 (初期値:ユーザー定数のVIEW_USE_HIGHLIGHT_CHECK_ONLY[=@og.value org.opengion.hayabusa.common.SystemData#VIEW_USE_HIGHLIGHT_CHECK_ONLY])。
53.columnBulkSet	【TAG】一括入力カラムのアシスト機能を利用するカラム名を、カンマ区切りで与えます。 これは、カラムの書き込み時に、一括入力可能なカラムをヘッダーに表示し、一括登録できる機能を提供します。 この機能は、jsp/common/bulkSet.js JavaScript とともに提供されます。 IE については、クリップボードも利用できます。Ctrl-C と Ctrl-V でそれぞれ一連のカラムの値の取り出しと書き込みが可能になります。 (プルダウンの場合はオプションに上記が追加されます) “*” を指定すると、すべてのカラムを (columnBulkSet) 指定したことになります。 システムリソースで、VIEW_USE_DEFAULT_BULKSET を true に設定すると、初期値が “*” になりますので、個別の view に設定しなくてもこの機能が使用できるようになります。 この機能は一括入力の手間削減が目的なので、selectedType が radio の場合は、この機能は無効になります。
54.caseKey	【TAG】このタグ自体を利用するかどうかの条件キーを指定します(初期値:null)。 caseKey.matches (caseVal) の値が、true の場合は、このタグは使用されます。

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
	false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。 【TAG】このタグ自体を利用するかどうかの条件値を指定します(初期値:null)。 caseKey.matches(caseVal) の値が、true の場合は、このタグは使用されます。 false の場合は、このタグは使用されません。 何も指定しない場合、または、引数が null の場合は、true と同じで使用されます。
55.caseVal	【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。
56.debug	

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:viewLink JSP	<p>【TAG】 コマンド(NEW, RENEW, RESET, REVIEW)をセットします。 コマンドは、HTMLから(get/post)指定されますので、CMD_xxx で設定される フィールド定数値のいずれかを、指定できます。</p>
●形式 : <og:viewLink command="..." > <og:link name="..." ... /> </og:viewLink >	
●body : あり(link)	
●Tag定義 :	
<og:viewLink command viewLinkId debug > ... Body ... </og:viewLink>	<p>○【TAG】 コマンド(NEW, RENEW, RESET, REVIEW)をセットします。(必須) 【TAG】 (通常使いません)requestから取得する ViewLink に対応する Attributes オブジェクトの ID 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)</p>
●使用例	
QueryTagとViewTagの間にviewLinkを使います。 <og:viewLink command="{@command}"> <og:link column="NOSYN" gameId="01PATTERN1" href="query.jsp" target="QUERY" keys="NOSYN, NMSYN, CDBK" value="[NOSYN], [NMSYN], [CDBK]" /> <og:link column="NMSYN" gameId="01PATTERN1" href="query.jsp" target="QUERY" keys="NMSYN" value="[NMSYN]" /> <og:link column="CDBK" gameId="01PATTERN1" href="query.jsp" target="QUERY" keys="NOSYN, NMSYN" value="[NOSYN], [NMSYN]" /> </og:viewLink>	
1.command	<p>【TAG】 コマンド(NEW, RENEW, RESET, REVIEW)をセットします。 コマンドは、HTMLから(get/post)指定されますので、CMD_xxx で設定される フィールド定数値のいずれかを、指定できます。</p>
2.viewLinkId	<p>【TAG】 (通常使いません)requestから取得する ViewLink に対応する Attributes オブジェクトの ID。 ViewLink オブジェクトをこのキーで、登録することにより、 ViewForm オブジェクトで、リンク情報を付加して表示させる。 初期値は、HybsSystem.VIEWLINK_KEY です。</p>
3.debug	<p>【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:viewModel JSP ●形式 : <og:viewModel > ... </og:viewModel > ●body : あり (columnMarker, popup) ●Tag定義 : <pre> <og:viewModel command viewModelId debug isRenderer > ... Body ... </og:viewModel> </pre> ●使用例 商品CD 2 (CDSYHN02)は赤字で表示する。 商品CD 3 (CDSYHN03)は-----と表示する。 <pre> <og:viewModel command="@command"> <og:columnMarker column="CDSYHN02" onMark="true" > [VCDSYHN02] </og:columnMarker> <og:columnMarker column="CDSYHN03" onMark="true" > <CENTER>-----</CENTER> </og:columnMarker> </og:viewModel> </pre>	【TAG】 コマンド(NEW, RENEW, RESET, REVIEW)をセットします。 コマンドは、HTMLから (get/post) 指定されますので、CMD_xxx で設定される フィールド定数値のいずれかを、指定できます。 【TAG】 コマンド(NEW, RENEW, RESET, REVIEW)をセットします。(必須) 【TAG】 (通常使いません)requestから取得する ViewMarker に対応する Attributes オブジェクトの ID 【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false) 【TAG】 マーカーのタイプが renderer かどうか[true/false]を指定します(初期値:true)
1.command	【TAG】 コマンド(NEW, RENEW, RESET, REVIEW)をセットします。 コマンドは、HTMLから (get/post) 指定されますので、CMD_xxx で設定される フィールド定数値のいずれかを、指定できます。
2.viewModelId	【TAG】 (通常使いません)requestから取得する ViewMarker に対応する Attributes オブジェクトの ID。 ViewForm オブジェクトに、ViewMarker オブジェクトをこのキーより取得することにより、 カラムに対して、マーカー情報を付加して表示させる。 初期値は、HybsSystem.VIEW_MARK_KEY です。
3.debug	【TAG】 デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。
4.isRenderer	【TAG】 マーカーのタイプが renderer かどうか[true/false]を指定します(初期値:true)。 このMarker オブジェクトが、ViewMarker か、EditMarker かを指定します。 内部的には、ViewMarker オブジェクトを構築しており、viewForm で、どちらにセット するかを決めているだけです。 true にセットすると、従来からある、viewMarker (renderer) を指定します。 false は、editMarker として作用し、オブジェクトが書き込み可能な場合に表示されます。 初期値は、true:renderer です。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
<p>5.5.4.2 og:where</p> <p>JSP</p> <p>●形式 : <og:where startKey="..." > <og:and startKey="..." ... /> </og:where></p> <p>●body : あり</p> <p>●Tag定義 :</p> <pre><og:where startKey debug > ... Body ... </og:where></pre> <p>●使用例</p> <pre><og:query command="NEW"> select PN, YOB1, NMEN, HINM from XX01 where PN = '[@PN]' order by PN </og:query></pre> <p>・ 検索条件が入力された時 ([@PN] がNOT NULL のとき) 作成されるSQL文⇒select PN, YOB1, NMEN, HINM from XX01 where PN = 'AAA' order by PN</p> <p>・ 検索条件が入力されなかった時 ([@PN] がNULL のとき) 作成されるSQL文⇒select PN, YOB1, NMEN, HINM from XX01 where PN = '' order by PN</p> <pre><og:query command="NEW"> select PN, YOB1, NMEN, HINM from XX01 <og:where> <og:and value="PN = '[@PN]%' " /> <og:and value="YOB1 like '[@YOB1]%' " /> </og:where> order by PN </og:query></pre> <p>・ 検索条件が入力された時 ([@PN] がNOT NULL のとき) 作成されるSQL文⇒select PN, YOB1, NMEN, HINM from XX01 PN = 'AAA%' and YOB1 like 'BBB%' order by PN</p> <p>・ 検索条件が入力されなかった時 ([@PN] がNULL のとき) WHERE 句がなくなる。 作成されるSQL文⇒select PN, YOB1, NMEN, HINM from XX01 order by PN</p> <p>注意:WhereTagを使った場合、下のようにはなりません。</p> <pre>select PN, YOB1, NMEN, HINM from XX01 PN = '' and YOB1 like '%' order by PN</pre> <pre><og:query command="NEW"> select PN, YOB1, NMEN, HINM from XX01 where PN="11111" <og:where startKey="and"> <og:and value="YOB1 like '[@PN]%' " /> </og:where> order by PN </og:query></pre> <p>・ 検索条件が入力された時 ([@YOB1] がNOT NULL のとき) 作成されるSQL文⇒select PN, YOB1, NMEN, HINM from XX01 PN = '11111' and YOB1 like 'BBB%' order by PN</p> <p>・ 検索条件が入力されなかった時 ([@YOB1] がNULL のとき) WHERE 句がなくなる。 作成されるSQL文⇒select PN, YOB1, NMEN, HINM from XX01 PN = '11111' order by PN</p>	<p>【TAG】 SQL条件句の最初の演算子を指定します (初期値:where)。 SQL条件句の最初の演算子を指定します。</p> <p>【TAG】 SQL条件句の最初の演算子を指定します (初期値:where)</p> <p>【TAG】 デバッグ情報を出力するかどうか [true/false] を指定します (初期値:false)</p>
1.startKey	【TAG】 SQL条件句の最初の演算子を指定します (初期値:where)。 SQL条件句の最初の演算子を指定します。
2.debug	【TAG】 デバッグ情報を出力するかどうか [true/false] を指定します (初期値:false)。 デバッグ情報を [true:出力する/false:しない] を指定します。 出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:writeCheck JSP	<p>【TAG】GAMENID、BACK_GAMENID 属性を付加するかどうか[true/false]を設定します(初期値:true)。 “true”の場合は、BACK_GAMENID、BACK_JSPID 属性を hidden で出力します。 “false”の場合は、付加しません。 これは、戻るリンクを作成する場合に利用されます。 初期値は、“true”です。 ボタンで遷移先がポップアップ画面の場合に、戻るリンクを出しても戻れない為、 そのようなケースでは false にします。</p>
●形式 : <og:writeCheck /> ●body : あり	
●Tag定義 : <og:writeCheck useGamenId roles writable debug > ... Body ... </og:writeCheck>	<p>【TAG】GAMENID、BACK_GAMENID 属性を付加するかどうか[true/false]を設定します(初期値:true) 【TAG】画面と別にロールズを与えます 【TAG】書き込み可能かどうか[true/false]を設定します(初期値:true) 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)</p>
●使用例 <og:writeCheck> <og:submit value="copy" lbl="COPY" accesskey="C" /> <og:submit value="modify" lbl="MODIFY" accesskey="M" /> <og:submit value="delete" lbl="DELETE" accesskey="D" /> <jsp:directive.include file="@SYS.JSP"/common/Excel_direct.jsp" /> </og:writeCheck>	
1.useGamenId	<p>【TAG】GAMENID、BACK_GAMENID 属性を付加するかどうか[true/false]を設定します(初期値:true)。 “true”の場合は、BACK_GAMENID、BACK_JSPID 属性を hidden で出力します。 “false”の場合は、付加しません。 これは、戻るリンクを作成する場合に利用されます。 初期値は、“true”です。 ボタンで遷移先がポップアップ画面の場合に、戻るリンクを出しても戻れない為、 そのようなケースでは false にします。</p>
2.roles	<p>【TAG】画面と別にロールズを与えます。 このロールズが与えられた場合は、画面のロールズではなく、 個別のロールズを使用して、表示の制御を行います。</p>
3.writable	<p>【TAG】書き込み可能かどうか[true/false]を設定します(初期値:true)。 書き込み許可する場合は、true、許可しない場合は、true 以外を設定します。 デフォルト値は、互換性のため、true です。 ここで、false を設定するとBODY部の内容は表示されません。true を設定した場合、 通常通り BODY 部は表示されます。 これは、roles の簡易版として、外部から ON/OFF を制御するのに使用できます。</p>
4.useEdit	<p>【TAG】EDITNAME利用時の編集を可能にするか設定します (初期値:USE_EDIT_WITH_EDITNAME[=false])。 カラムの並び替えや表示等を設定した場合に編集可能にするかを設定します。 (つまり、ボタンを出すかどうかをコントロールします) (初期値:USE_EDIT_WITH_EDITNAME[= (@og.value org.opengion.hayabusa.common.SystemData#USE_EDIT_WITH_EDITNAME)])。</p>
5.outputEditName	<p>【TAG】editNameをhiddenタグで出力するか設定します (初期値:OUTPUT_EDITNAME[=false])。 選択したeditNameを次の画面に送信するためのhiddenタグを出力します。 editNameを次の画面に送信した場合は送信先画面のviewタグのviewFormTypeが 内部的に自動セットされるため注意が必要です。 useEditがfalseの場合は出力しません。 (初期値:OUTPUT_EDITNAME[= (@og.value org.opengion.hayabusa.common.SystemData#OUTPUT_EDITNAME)])。</p>
6.debug	<p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。</p>

バージョン タグ名 HTMLTEXT BODY要素	内容
形式サンプル	
5.5.4.2 og:writeTable JSP	【TAG】実際に書き出すクラス名の略称(TableWriter_**** クラスの ****)を指定します(初期値:Default)。 実際に書き出すクラス名(の略称)をセットします。 これは、org.opengion.hayabusa.io 以下の TableWriter_**** クラスの **** を与えます。これらは、TableWriter インターフェースを継承したサブクラスです。 属性クラス定義の {@link org.opengion.hayabusa.io.TableWriter TableWriter} を参照願います。
<p>●形式：</p> <pre><og:writeTable fileURL = "~{@USER.ID}" 保存先ディレクトリ名 filename = "~{@filename}" 保存ファイル名 encode = "UnicodeLittle" 保存ファイルエンコード名 /></pre> <p>●body：あり(writeTableParam)</p> <p>●Tag定義：</p> <pre><og:writeTable writerClass 【TAG】実際に書き出すクラス名の略称(TableWriter_**** クラスの ****)を指定します(初期値:Default) separator 【TAG】可変長ファイルを作成するときの項目区切り文字をセットします (初期値:TableWriter#TAB_SEPARATOR[=]) headerSequence 【TAG】DBTableModelの出力順(LNSCD など)をセットします fileURL 【TAG】ファイル保存先ディレクトリ名を指定します filename 【TAG】ファイルを作成するときのファイル名をセットします (初期値:FILE_ENCODE[=UnicodeLittle]) encode 【TAG】ファイルを作成するときのファイルエンコーディング名をセットします (初期値:FILE_ENCODE[=UnicodeLittle]) fileAppend 【TAG】追加モードで書き込むかどうか[true/false]を指定します(初期値:false[新規モード]) direct 【廃止】結果をダイレクトにEXCEL起動するかどうか[true/false]を指定します(初期値:false[ファイル]) zip 【TAG】結果をファイルに出力するときに、ZIPで圧縮するかどうか[true/false]を指定します(初期値:false) tableId 【TAG】(通常使いません)sessionから所得する DBTableModelオブジェクトの ID scope 【TAG】キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session) disposition 【TAG】Content-Disposition (inline:埋め込む/attachment:ダイアログを起動)を指定します(初期値:attachment) sheetName 【TAG】EXCELファイルを書き出すときのシート名を設定します refSheetName 【TAG】EXCEL雛型参考ファイルのシート名を設定します refFileURL 【TAG】EXCEL雛型参照ファイルのディレクトリ名をセットします refFilename 【TAG】EXCEL雛型参考ファイル名をセットします useNumber 【TAG】行番号情報を、出力する/しない[true/false]を指定します(初期値:true) columns 【TAG】書き込み先ファイルのカラム列を、外部(タグ)より指定します useRenderer 【TAG】書込処理でコードリソースのラベル変換を行うかどうかを指定します (初期値:USE_TABLE_WRITER_RENDERER[=false]) fontName 【TAG】EXCEL出力時のデフォルトフォント名を設定します (初期値:TABLE_WRITER_DEFAULT_FONT_NAME[=MS P ゴシック]) fontPoint 【TAG】EXCEL出力時のデフォルトフォントポイント数を設定します language 【TAG】タグ内部で使用する言語コード[ja/en/zh/...]を指定します debug 【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false) /> ... Body ... </og:writeTable></pre> <p>●使用例</p> <pre><og:writeTable > <og:writeTableParam key = "Tablename" value="GE12" /> <og:writeTableParam key = "First" First:最初に登録 > insert into GE12bk select * from GE12 where SYSTEM_ID= '**' </og:writeTableParam> <og:writeTableParam key = "First" First:の2番目に登録 > delete from GE12 where SYSTEM_ID= '**' and KBSAKU=' 0' </og:writeTableParam> <og:writeTableParam key = "Last" Last:最後に登録 > update GE12 set XXXX= YYYY where SYSTEM_ID= '**' and KBSAKU=' 0' </og:writeTableParam> </og:writeTableParam></pre>	
1.writerClass	【TAG】実際に書き出すクラス名の略称(TableWriter_**** クラスの ****)を指定します(初期値:Default)。 実際に書き出すクラス名(の略称)をセットします。 これは、org.opengion.hayabusa.io 以下の TableWriter_**** クラスの **** を与えます。これらは、TableWriter インターフェースを継承したサブクラスです。 属性クラス定義の {@link org.opengion.hayabusa.io.TableWriter TableWriter} を参照願います。
2.separator	【TAG】可変長ファイルを作成するときの項目区切り文字をセットします (初期値:TableWriter#TAB_SEPARATOR[=])。 可変長ファイルを作成するときの項目区切り文字をセットします。 (初期値:TableWriter#TAB_SEPARATOR[= @og.value TableWriter#TAB_SEPARATOR])。
3.headerSequence	【TAG】DBTableModelの出力順(LNSCD など)をセットします。 Label, Name, Size, Class, Data の各フィールドの頭文字のアルファベットで出力順を設定します。
4.fileURL	【TAG】ファイル保存先ディレクトリ名を指定します。 この属性で指定されるディレクトリに、ファイルをセーブします。 指定方法は、通常の fileURL 属性と同様に、先頭が、'/' (UNIX) または、2文字目が、'.' (Windows) の場合は、指定のURLそのままのディレクトリに、そうでない場合は、システムパラメータ の FILE_URL 属性で指定のフォルダの下に、作成されます。 fileURL = "{@USER.ID}" と指定すると、FILE_URL 属性で指定のフォルダの下に、さらに、各個人ID別のフォルダを作成して、そこにセーブします。
5.filename	【TAG】ファイルを作成するときのファイル名をセットします (初期値:FILE_ENCODE[=UnicodeLittle])。 ファイルを作成するときのファイル名をセットします。 (初期値:システム定数のFILE_ENCODE[= @og.value org.opengion.hayabusa.common.SystemData#FILE_ENCODE])。
6.encode	【TAG】ファイルを作成するときのファイルエンコーディング名をセットします (初期値:FILE_ENCODE[=UnicodeLittle])。 "DEFAULT", "JISAutoDetect", "JIS", "EUC_JP", "MS932", "SJIS", "Windows-31J", "Shift_JIS" (初期値:システム定数のFILE_ENCODE[= @og.value org.opengion.hayabusa.common.SystemData#FILE_ENCODE])。
7.fileAppend	【TAG】追加モードで書き込むかどうか[true/false]を指定します(初期値:false[新規モード])。 DBTableModelのファイルを、追加モードで書き込むかどうかをセットします。
8.direct	【廃止】結果をダイレクトにEXCEL起動するかどうか[true/false]を指定します(初期値:false[ファイル])。 結果をダイレクトに EXCEL ファイルとして出力するかどうかをセットします。
9.zip	【TAG】結果をファイルに出力するときに、ZIPで圧縮するかどうか[true/false]を指定します(初期値:false)。 初期値は、圧縮しないです。
10.tableId	【TAG】(通常使いません)sessionから所得する DBTableModelオブジェクトの ID。 表示処理後に、(内部ポインタを書き換えた)DBTableModelオブジェクトを同じキーで、sessionに登録します。

バージョン
タグ名
HTMLTEXT
BODY要素

内容

形式サンプル

11.scope

初期値は、HybsSystem.TBL_MD_L KEY です。
【TAG】キャッシュする場合のスコープ[request/page/session/applicaton]を指定します(初期値:session)。
“request”, “page”, “session”, “applicaton” が指定できます。
JSPのスコープは、「変数の有効範囲」を表すもので、フレームワーク上では、主にテーブルモデルを管理するスコープを指します。

スコープ	変数の有効範囲	
page	JSPページ内	そのJSPページ内のみで有効です。フレームワーク的には、JSPページにまたがる処理が多いため、ほとんど使う機会
request	HTTPリクエスト	リクエストの一連の処理期間中に有効な変数で、メモリに多くの情報を残したくない場合に利用します。検索系やポ
session	HTTPセッション	初期設定されているスコープで、ログインユーザー単位にログアウトまで保持されます。内部的には、同じキーワー
application	Webアプリケーション	ユーザー間で共有する場合のスコープになります。JSP画面の開発では、まず使うことはありません。

12.disposition

【TAG】Content-Disposition (inline:埋め込む/attachment:ダイアログを起動)を指定します(初期値:attachment)。
Content-Disposition を指定することで、ブラウザにアプリケーションデータを埋め込むのか(inline)、セーブ/オープンダイアログを起動するのか(attachment)を指定します。
この属性が有効なのは、direct=“true” で、zip=“false”(初期値)のときのみです。
値は、inline 、 attachment が指定できます。
初期値は、attachment です。
注意: 動作検証しましたが、inline と attachment で動作に差が現れませんでした。

13.sheetName

【TAG】EXCELファイルを書き出すときのシート名を設定します。
EXCELファイルを書き出す時に、シート名を指定します。これにより、複数の形式の異なるデータを順次書き出したり(appendモードを併用)することや、シートを指定して新規にEXCELを作成することが可能になります。
初期値は、Sheet1 です。

14.refSheetName

【TAG】EXCEL雛型参考ファイルのシート名を設定します。
EXCELファイルを書き出す時に、雛型として参照するシート名を指定します。
これにより、複数の形式の異なるデータを順次書き出したり(appendモードを併用)することや、シートを指定して新規にEXCELを作成する場合にフォームを設定する事が可能になります。
初期値は、null(第一シート) です。

15.refFileURL

【TAG】EXCEL雛型参照ファイルのディレクトリ名をセットします。
この属性で指定されるディレクトリに、ファイルをセーブします。
指定方法は、通常の fileURL 属性と同様に、先頭が、'/' (UNIX) または、2文字目が、':/' (Windows)の場合は、指定のURLそのままのディレクトリに、そうでない場合は、システムパラメータ の FILE_URL 属性で指定のフォルダの下に、作成されます。
fileURL = “@USER_ID” と指定すると、FILE_URL 属性で指定のフォルダの下に、さらに、各個人ID別のフォルダを作成して、そこにセーブします。

16.refFilename

【TAG】EXCEL雛型参考ファイル名をセットします。
EXCEL雛型参考ファイル名をセットします。

17.useNumber

【TAG】行番号情報を、出力する/しない[true/false]を指定します(初期値:true)。
通常のフォーマットでは、各行の先頭に行番号を出力します。
これは、#NAME 属性を使用する場合には、必ず出力する必要があります。
(#NAME 属性は、読み取り時には、必須です。)
この、先頭の行番号が不要な場合(つまり、他のシステムへのデータ出力、このシステムでは、#NAME 属性が出力されないため、読み込みできません。)
この行番号を出力しないようにできます。
なお、EXCEL 出力には、この設定は適用されません。(暫定対応)
初期値は、true(出力する) です。

18.columns

【TAG】書き込み先ファイルのカラム列を、外部(タグ)より指定します。
この指定により、書き込むカラムを限定的に使用することが可能です。
カラム名は、検索したDBTableModel上に含まれる名前で指定する必要があります。
別名で書き込みたい場合は、予めその名前で検索しておく必要があります。

【TAG】書込処理でコードリソースのラベル変換を行うかどうかを指定します
(初期値:USE_TABLE_WRITER_RENDERER[=false])。
コードリソースをそのままの値で出力すると、数字や記号になり何が書かれているのか不明になります。
これは、コードリソースをラベルに変換して出力するかどうかを指定します。
当然、コードはユニークですが、ラベルはユニークになるかどうか保障はされていませんので
TableReader 系で読み込む場合には、リスクが発生します。
また、TableReader 系で読み込む場合にも、ラベルからコードを求める逆変換を行うように、setUseRenderer メソッドで指定する必要があります。

19.useRenderer

従来は、TableWriter 系に、TableWriter_Renderer 系のクラスを作って対応していましたが、このメソッドの属性値のフラグで、制御します。
(初期値:システム定数のUSE_TABLE_WRITER_RENDERER[=@org.value.org.opengion.hayabusa.common.SystemData#USE_TABLE_WRITER_RENDERER])。
【TAG】EXCEL出力時のデフォルトフォント名を設定します
(初期値:TABLE_WRITER_DEFAULT_FONT_NAME[=MS Pゴシック])。
これは、EXCEL追加機能として実装されています。

20.fontName

EXCELファイルを書き出す時に、デフォルトフォント名を指定します。
フォント名は、EXCELのフォント名をそのまま使用してください。
内部的に、POI の org.apache.poi.hssf.usermodel.HSSFFont#setFontName(String) に設定されます。
(初期値:システム定数のTABLE_WRITER_DEFAULT_FONT_NAME[=@org.value.org.opengion.hayabusa.common.SystemData#TABLE_WRITER_DEFAULT_FONT_NAME])

【TAG】EXCEL出力時のデフォルトフォントポイント数を設定します。
これは、EXCEL追加機能として実装されています。

21.fontPoint

EXCELファイルを書き出す時に、デフォルトポイント数を指定します。
内部的に、POI の org.apache.poi.hssf.usermodel.HSSFFont#setFontHeightInPoints(short) に設定されます。
初期値は、システムリソース の TABLE_WRITER_DEFAULT_FONT_POINTS です。

22.language

【TAG】タグ内部で使用する言語コード[ja/en/zh/...]を指定します。
この言語コードに基づいて、表示のラベルをリソースから作成します。
【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。

23.debug

デバッグ情報を [true:出力する/false:しない]を指定します。
出力形式自体は、個々のタグによって異なります。

バージョン タグ名 HTMLTEXT BODY要素	内容
	形式サンプル
5.5.4.2 og:writeTableParam JSP	<p>【TAG】パラメータとして渡すキー情報([First Last TableName])を指定しま。 WriteTable に渡すパラメータのキー情報です。 First と Last を指定した場合は、XML 形式で出力する EXEC_SQL 情報になります。 TableName の場合は、XML 形式の ROWSET の table 属性になります。 値は、value 属性で指定するか、なければ BODY 部に記述します。</p>
<p>●形式：</p> <pre><og:writeTableParam key="~[First Last TableName]" ~> delete from GE12 where SYSTEM_ID='**' and KBSAKU='0' </og:writeTableParam></pre> <p>●body：あり</p> <p>●Tag定義：</p> <pre><og:writeTableParam key value debug > ... Body ... </og:writeTableParam></pre> <p>●使用例</p> <pre><og:writeTable> <og:writeTableParam key = "Tablename" value="GE12" /> <og:writeTableParam key = "First" First:最初に登録 > insert into GE12bk select * from GE12 where SYSTEM_ID='**' </og:writeTableParam> <og:writeTableParam key = "First" First:の2番目に登録 > delete from GE12 where SYSTEM_ID='**' and KBSAKU='0' </og:writeTableParam> <og:writeTableParam key = "Last" Last:最後に登録 > update GE12 set XXXX='YYYY' where SYSTEM_ID='**' and KBSAKU='0' </og:writeTableParam> </og:writeTableParam></pre>	
1.key	<p>【TAG】パラメータとして渡すキー情報([First Last TableName])を指定しま。 WriteTable に渡すパラメータのキー情報です。 First と Last を指定した場合は、XML 形式で出力する EXEC_SQL 情報になります。 TableName の場合は、XML 形式の ROWSET の table 属性になります。 値は、value 属性で指定するか、なければ BODY 部に記述します。</p>
2.value	<p>【TAG】パラメータとして渡す設定値を指定します(初期値:null)。 WriteTable に渡すパラメータの設定値です。 First と Last を指定した場合は、XML 形式で出力する EXEC_SQL 情報になります。 TableName の場合は、XML 形式の ROWSET の table 属性になります。 値は、value 属性で指定するか、なければ BODY 部に記述します。 BODY 部に記述された場合は、文字列を trim() します。 設定値は、value 属性が優先です。この値が、null の場合は、 BODY 要素を値として使用します。</p>
3.debug	<p>【TAG】デバッグ情報を出力するかどうか[true/false]を指定します(初期値:false)。 デバッグ情報を [true:出力する/false:しない]を指定します。 出力形式自体は、個々のタグによって異なります。</p>

ATTCLASS (バージョン5.5.4.2)

バージョン 属性名称 属性クラス	内容
ATTCLASS	
形式サンプル	
5.5.4.2 queryType (Interface)Query	ステートメント文を execute() する事により、データベースを検索した結果を DBTableModel に割り当てるインターフェースです。 実際にこのインターフェースを継承したクラスでは、データベース以外にフラットファイルやXMLデータなどを読み込んで DBTableModel を作成させる様な実装クラスを作成することができます。
org.opengion.hayabusa.db.Query	指定のSQL文を実行して、検索する Query クラスです。
5.5.4.2 queryType JDBC	java.sql.Statement を用いて、データベース検索処理を行います。 引数は無しです。(与えられたSQL文を実行します。) 内部変数の受け渡しのデフォルト実装は、AbstractQuery クラスを継承しているため、ここでは、execute() メソッドを実装しています。 このクラスでは、ステートメント文を execute() する事により、データベースを検索した結果を DBTableModel に割り当てます。
org.opengion.plugin.query.Query_JDBC	
<pre> <og:query command="@command" debug="false"> <!-- 先頭のカラム名が、"WRITABLE" の場合、'true' or '1' で、書き込み許可が与えら、'2' でチェック済みになります。--> select KBSAKU AS WRITABLE, CLM, NAME_JA, LABEL_NAME, KBSAKU, SYSTEM_ID, LANG, FGJ from GE41 <og:where> <og:and value = "FGJ" in ('0','1')"/> <og:and value = "SYSTEM_ID" like '@SYSTEM_ID'"/> <og:and value = "LANG" like '@LANG'"/> <og:and value = "CLM" like '@CLM'"/> <og:and value = "KBSAKU" = '@KBSAKU'"/> </og:where> <og:appear startKey = "order by" value = "@ORDER_BY" defaultVal = "SYSTEM_ID, CLM, LANG" /> </og:query> </pre>	
5.5.4.2 queryType JDBCArrayCallable	Callableのエラー配列対応版です。バッチ系標準のPL/SQL をコールする Query クラスです。 java.sql.CallableStatement を用いて、データベース検索処理を行います。 引数は、従来のPL/SQLの実行が可能のように、第一引数はエラーコード、第二引数は、エラーメッセージを返してきます。第三引数以降は、自由に指定できます。 内部変数の受け渡しのデフォルト実装は、AbstractQuery クラスを継承しているため、ここでは、execute() メソッドを実装しています。
org.opengion.plugin.query.Query_JDBCArrayCallable	
<p>例：</p> <p>第一引数、第二引数は、通常のPL/SQLと同じ、結果(STATUS)と内容(ERR_CODE)を返します。 それ以降の引数については、入力(IN)のみですが、自由に設定できます。 引数に変数を使用する場合は、? 記号を当てはめます。 第一引数、第二引数は、予約済みですが、それ以降は、好きな位置に割り当てられます。 names 属性の順番に、値だけがセットされていきます。 下記の例は、変数の引数は、使用していません。</p> <pre> <og:query command="NEW" queryType="JDBCArrayCallable" displayMsg="" > { call GEPO0002.GEPO0002(?, ?, '@GUI.KEY)', '@USER.ID') } </og:query> </pre> <pre> CREATE OR REPLACE PACKAGE GEPO0002 AS PROCEDURE GEPO0002(P_STATUS OUT NUMBER, P_ERR_CODE OUT ERR_MSG_ARRAY, P_MIDDB IN VARCHAR2, P_USRUPD IN VARCHAR2); END; </pre>	
5.5.4.2 queryType JDBCCallable	バッチ系標準のPL/SQL をコールする Query クラスです。 java.sql.CallableStatement を用いて、データベース検索処理を行います。 引数は、従来のPL/SQLの実行が可能のように、第一引数はエラーコード、第二引数は、エラーメッセージを返してきます。第三引数以降は、自由に指定できます。 内部変数の受け渡しのデフォルト実装は、AbstractQuery クラスを継承しているため、ここでは、execute() メソッドを実装しています。
org.opengion.plugin.query.Query_JDBCCallable	
<p>例：</p> <p>第一引数、第二引数は、通常のPL/SQLと同じ、結果(STATUS)と内容(ERR_CODE)を返します。 それ以降の引数については、入力(IN)のみですが、自由に設定できます。 引数に変数を使用する場合は、? 記号を当てはめます。 第一引数、第二引数は、予約済みですが、それ以降は、好きな位置に割り当てられます。 names 属性の順番に、値だけがセットされていきます。 下記の例は、変数の引数は、使用していません。</p> <pre> <og:query command="NEW" queryType="JDBCCallable" displayMsg="" > { call GEPO0002.GEPO0002(?, ?, '@GUI.KEY)', '@USER.ID') } </og:query> </pre> <pre> CREATE OR REPLACE PACKAGE GEPO0002 AS PROCEDURE GEPO0002(P_STATUS OUT NUMBER, P_ERR_CODE OUT VARCHAR2, P_MIDDB IN VARCHAR2, P_USRUPD IN VARCHAR2); END; </pre>	
5.5.4.2	<p>エントリ系 PL/SQL をコールして、結果カーソルから、DBTableModel を作成します。</p> <p>java.sql.CallableStatement を用いて、データベース検索処理を行います。 引数を配列指定で渡すことが出来、エラー時には、DBErrMsg オブジェクトにエラー情報を</p>

バージョン 属性名称 属性クラス	内容
ATTCLASS	
形式サンプル	
queryType JDBCErrMsg	<p>格納して返すことが可能です。</p> <p>内部変数の受け渡しのデフォルト実装は、AbstractQuery クラスを継承しているため、ここでは、execute() メソッドを実装しています。</p> <p>このクラスでは、ステートメント文を execute() する事により、データベースを検索した結果を DBTableModel に割り当てます。</p>
org.opengion.plugin.query.Query_JDBCErrMsg	
例: jsp/TYPE3B/result.jsp (検索系: カーソル返し)	
例: jsp/TYPE3B/entry.jsp (エントリ系)	
names には、ARG_ARRAY 配列に順次セットされます。	
使用する場合は、一旦わかり易い変数に受けて利用してください。	
呼び出す PL/SQL では、検索系PL/SQL です。(下記の例は、エントリ系)	
<pre><og:query command = "NEW" names = "SYSTEM_ID, LANG, CLM, NAME_JA, LABEL_NAME, KBSAKU, USER, ID" queryType = "JDBCErrMsg" displayMsg = "" > { call TYPE3B01. TYPE3B01(?, ?, ?, ?) } </og:query></pre>	
<pre>CREATE OR REPLACE PACKAGE TYPE3B01 AS TYPE CUST_CURSOR IS REF CURSOR; PROCEDURE TYPE3B01 (P_KEKKA OUT NUMBER, P_ERRMSGSGS OUT ERR_MSG_ARRAY, P_RC1 OUT CUST_CURSOR, P_ARGS IN ARG_ARRAY); END;</pre>	
<pre>P_SYSTEM_ID GEA08.SYSTEM_ID%TYPE := P_ARGS(1); --システム I D P_LANG GEA08.LANG%TYPE := P_ARGS(2); --言語 P_CLM GEA08.CLM%TYPE := P_ARGS(3); --項目 P_NAME_JA GEA08.NAME_JA%TYPE := P_ARGS(4); --名称(漢字) P_LABEL_NAME GEA08.LABEL_NAME%TYPE := P_ARGS(5); --表示名称 P_KBSAKU GEA08.KBSAKU%TYPE := P_ARGS(6); --作成区分 P_USRSET GEA08.USRSET%TYPE := P_ARGS(7); --登録者</pre>	
5.5.4.2 queryType JDBCKeyEntry	<p>PL/SQL にエントリ情報を配列渡しする、Query 実行クラスです。</p> <p>java.sql.CallableStatement を用いて、データベース登録処理を行います。</p> <p>引数に、キーと値をセットで配列指定で渡すことが出来、エラー時には、DBErrMsg オブジェクトにエラー情報を格納して返すことが可能です。</p> <p>内部変数の受け渡しのデフォルト実装は、AbstractQuery クラスを継承しているため、ここでは、execute() メソッドを実装しています。</p> <p>このクラスでは、ステートメント文を execute() する事により、データベースを検索した結果を DBTableModel に割り当てます。</p>
org.opengion.plugin.query.Query_JDBCKeyEntry	
例:	
第一引数、第二引数は、結果(KEKKA)とエラーメッセージ配列を返します。	
キーエントリでは、エントリ(リクエスト情報)のキーと値をセットで	
引数の配列に設定します。	
キーを元に、値を利用する場合に使用します。	
下記の例は、動的カラムを実現しているPL/SQLの例です。	
<pre><og:entryQuery command = "NEW" queryType = "JDBCKeyEntry" > { call DYNAMIC_PKG.DYNAMIC_TEST(?, ?, ?, ?) } </og:entryQuery></pre>	
<pre>CREATE OR REPLACE PACKAGE DYNAMIC_PKG AS PROCEDURE DYNAMIC_TEST (P_KEKKA OUT NUMBER, P_ERRMSGSGS OUT ERR_MSG_ARRAY, P_KEYS IN ARG_ARRAY, P_ARGS IN ARG_ARRAY); END;</pre>	
5.5.4.2 queryType JDBCPLSQL	<p>PL/SQL をコールする 登録系 Queryクラスです。</p> <p>java.sql.CallableStatement を用いて、データベース検索処理を行います。</p> <p>引数に、SYSARG_ARRAYと、ユーザーARG_ARRAY を配列指定で渡すことが出来、エラー時には、DBErrMsg オブジェクトにエラー情報を格納して返すことが可能です。</p> <p>内部変数の受け渡しのデフォルト実装は、AbstractQuery クラスを継承しているため、ここでは、execute() メソッドを実装しています。</p> <p>このクラスでは、ステートメント文を execute() する事により、データベースを検索した結果を DBTableModel に割り当てます。</p>
org.opengion.plugin.query.Query_JDBCPLSQL	
例: jsp/TYPE1B/result.jsp	
names には、GEA08ARG で定義したカラムを指定します。	
呼び出す PL/SQL では、登録系PL/SQL です。	
<pre><og:plsqlUpdate command = "[@command]" names = "SYSTEM_ID, LANG, CLM, NAME_JA, LABEL_NAME, KBSAKU, FGJ, USRSET" dbType = "GEA08ARG" queryType = "JDBCPLSQL" > { call TYPE1B01. TYPE1B01(?, ?, ?, ?, ?) } </og:plsqlUpdate></pre>	
<pre>PROCEDURE TYPE1B01 (P_KEKKA OUT NUMBER, -- エラー結果(0:正常 1:警告 2:異常) P_ERRMSGSGS OUT ERR_MSG_ARRAY, -- エラーのあるときのエラーメッセージ配列 P_NAMES IN VARCHAR2, P_SYSARGS IN SYSARG_ARRAY, -- 引数 SYSTEMデータ P_GEO8ARGS IN GEA08ARG_ARRAY -- 引数 USERデータ);</pre>	
5.5.4.2 queryType JDBCPrepared	<p>引数引き当て(PreparedStatement) を利用した登録系Queryです。</p> <p>java.sql.PreparedStatement を用いて、データベース検索処理を行います。</p> <p>引数に、指定した値を配列で渡します。</p> <p>内部変数の受け渡しのデフォルト実装は、AbstractQuery クラスを継承しているため、ここでは、execute() メソッドを実装しています。</p>

バージョン 属性名称 属性クラス	内容
ATTCLASS	
形式サンプル	
	このクラスでは、ステートメント文を <code>execute()</code> する事により、データベースを 検索した結果を <code>DBTableModel</code> に割り当てます。
<code>org.opengion.plugin.query.Query_JDBCPrepared</code>	
<p>例：</p> <p>可変引数付きのSQL文を実行します。 これは、INSERT、UPDATE、DELETE など、どのようなSQL文でも実行できます。 <code>names</code> 属性で指定するのは、<code>DBTableModel</code>の列名で、その値が順番に、 引数(?)記号)の個所に設定されます。 選択されたデータ(行)の数だけ、繰り返し実行されます。</p> <pre> jsp/TYPE1A/copy.jsp <og:value scope="session" key="names" value="CLM_NAME_JA, LABEL_NAME, KBSAKU, SYSTEM_ID, LANG" /> <og:value scope="session" key="SQL" > INSERT INTO GEA08 (CLM_NAME_JA, LABEL_NAME, KBSAKU, SYSTEM_ID, LANG, FGJ, DYSET, DYUPD, USRSET, USRUPD, PGUPD) VALUES (?, ?, ?, ?, ?, ?, '1', '@USER.YMDH)', '@USER.YMDH)', '@USER.ID)', '@USER.ID)', '@GUI.KEY)') </og:value> jsp/TYPE1A/entry.jsp <it:h:update command = "@{command}" queryType = "JDBCPrepared" names = "@{names}" > @{SQL} </og:update> <!-- 前画面で指定のSQL文を削除します。(scope="session"なので削除が必要。) --> <og:value scope="session" key="names" command="REMOVE" /> <og:value scope="session" key="SQL" command="REMOVE" /> </pre>	
5.5.4.2 queryType JDBCTableUpdate	<p>引数引き当て(<code>PreparedStatement</code>)を利用した登録系Queryです。</p> <p><code>java.sql.PreparedStatement</code> を用いて、データベース検索処理を行います。 引数の指定方法は、<code>DBTableModel</code> の列名に対応する名称を、SQL文の[列名]形式で 記述します。これを解析して、実際に実行する <code>PreparedStatement</code> に対応する文字列を 作成します。 たとえば、<code>INSERT INTO GEXX (CLM_NAME_JA, LABEL_NAME) VALUES ([CLM], [NAME_JA], [LABEL_NAME])</code> と 記述すれば、内部で、<code>DBTableModel</code> の列名に対応する値を取り出し、SQL文として、 <code>INSERT INTO GEXX (CLM_NAME_JA, LABEL_NAME) VALUES (?, ?, ?)</code> を実行します。</p>
<code>org.opengion.plugin.query.Query_JDBCTableUpdate</code>	
●使用例	
<p>・QUERYを直接書く場合 【entry.jsp】</p> <pre> <og:tableUpdate command = "@{command}" queryType = "JDBCTableUpdate" > INSERT INTO GE41 (CLM_NAME_JA, LABEL_NAME, KBSAKU, SYSTEM_ID, LANG, FGJ, DYSET, DYUPD, USRSET, USRUPD, PGUPD) VALUES ([CLM], [NAME_JA], [LABEL_NAME], [KBSAKU], [SYSTEM_ID], [LANG], '1', '@USER.YMDH)', '@USER.YMDH)', '@USER.ID)', '@USER.ID)', '@GUI.KEY)') </og:tableUpdate> </pre>	
5.5.4.2 queryType JDBCUpdate	<p>一般的な PL/SQLをコールする、Query クラスです。</p> <p><code>java.sql.CallableStatement</code> を用いて、データベース登録処理を行います。 引数は、そのまま配列に格納して処理を行います。エラー時の処理や、検索結果の 取り出しは出来ません。 内部変数の受け渡しのデフォルト実装は、<code>AbstractQuery</code> クラスを継承している ため、ここでは、<code>execute()</code> メソッドを実装しています。 このクラスでは、ステートメント文を <code>execute()</code> する事により、データベースを 検索した結果を <code>DBTableModel</code> に割り当てます。</p> <p>例：</p> <p>Hybs独自のステータスやエラーメッセージなどの引数を持たない、 一般的なPL/SQLをCALLします。 <code>names</code> 属性で指定するのは、<code>DBTableModel</code>の列名で、その値が順番に、 引数(?)記号)の個所に設定されます。 引数が、? でない個所には、直接値を設定したり、{@列名}で、 リクエスト変数をセットする事も可能です。 選択されたデータ(行)の数だけ、繰り返し実行されます。 下記の例は、テーブルのアナライザを実行しています。</p> <pre> jsp/ORA08/result.jsp <og:query displayMsg = "" command = "@{command}" names = "TABLE_OWNER, TABLE_NAME" queryType = "JDBCUpdate" > { call DBMS_STATS.GATHER_TABLE_STATS(?, ?) } </og:query> </pre>
<code>org.opengion.plugin.query.Query_JDBCUpdate</code>	
5.5.4.2 renderer AUTOAREA	<p><code>AUTOAREA</code> レンダラは、カラムのデータをテキストエリアで表示する場合に 使用するクラスです。 <code>readonly</code>のテキストエリアで<code>class=renderer-textarea</code>として出力し、 <code>name</code>属性は付けません。(データは送信されません) エリアの縦、横サイズはエディタの自動拡張テキストエリア(<code>AUTOAREA</code>)と同様にして算出されます。</p> <p>カラムの表示に必要な属性は、<code>DBColumn</code> オブジェクト より取り出します。 このクラスは、<code>DBColumn</code> オブジェクト毎に1つ作成されます。</p>
<code>org.opengion.plugin.column.Renderer_AUTOAREA</code>	
5.5.4.2 renderer	<p><code>CHBOX</code> レンダーは、カラムのデータをチェックボックス文字情報として 表示する場合に使用するクラスで、“0”と、“1”のみ使用できます。</p> <p>“0” がOFF状態(全角スペース)、“1”がON状態(レ)です。</p>

バージョン 属性名称 属性クラス	内容
ATTGLASS 形式サンプル	
CHBOX	<p>カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。</p>
org.opengion.plugin.column.Renderer_CHBOX	英数字をバーコードで使用する CODE39 のチェックデジット付き文字列に変換するレンダークラスです。
5.5.4.2 renderer CODE39	<p>CODE39 は、『0～9, A～Z, -, ., ., \$, /, +, %』のコードが使用できるバーコードの体系です。 通常 * で始まり * で終了します。また、チェックデジット に、モジュラス43 が使われます。 ここでは、指定の文字列の前後に、* を付与し、チェックデジットも付与します。</p> <p>カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。</p>
org.opengion.plugin.column.Renderer_CODE39	
5.5.4.2 renderer COLOR	<p>LABEL レンダーは、カラムの値を#FFFFFFの色として表示する場合に 使用するクラスです。 #FFFFFFのように#付き7桁のデータで設定して下さい。</p> <p>カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。</p>
org.opengion.plugin.column.Renderer_COLOR	
5.5.4.2 renderer COLUMN	<p>COLUMN レンダーは、データの値をカラム名と認識して、動的カラムを 表示するクラスです。</p> <p>データの値を、動的カラムのそのカラム属性を持ったフォームを表示します。</p> <p>カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。</p>
org.opengion.plugin.column.Renderer_COLUMN	
5.5.4.2 renderer CRYPT	<p>パスワード情報など、重要な情報の暗号化された情報を表示する場合に使用するクラスです。</p> <p>このクラスの暗号化は秘密キーによる可逆変換なので、変換方式と秘密キーが判ると 元に戻すことが可能です。それでも、何もしないよりははるかにましです。 データベース等へ登録した暗号化されたデータを表示する場合に、使用します。</p>
org.opengion.plugin.column.Renderer_CRYPT	
5.5.4.2 renderer DATE	<p>DATE レンダーは、カラムのデータを表示パラメータで指定されたフォーマットで 日付表示する場合に使用するクラスです。 6桁未満のデータの場合はフォーマットをかげずにそのまま出力します。</p> <p>表示パラメータで指定されたフォーマットを、java.text.SimpleDateFormat で 解析してフォーマット変換します。 フォーマットルールは、java.text.SimpleDateFormat を参照願います。 フォーマット初期値はシステムリソースのSYS_TIMEです。</p> <p>カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。</p>
org.opengion.plugin.column.Renderer_DATE	
5.5.4.2 renderer DBLABEL	<p>DBLABEL レンダーは、値をラベルリソースの表示ラベルに変換するクラスです。</p> <p>DBMENU で同様の処理を記述できますが、ラベルリソース に限定した使い方を 想定してます。</p> <p>カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。</p>
org.opengion.plugin.column.Renderer_DBLABEL	
5.5.4.2 renderer DBMENU	<p>DBMENU レンダーは、表示パラメータで指定された SQL文を実行し、 プルダウンメニューで表示する場合に使用するクラスです。</p> <p>カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。</p>
org.opengion.plugin.column.Renderer_DBMENU	
5.5.4.2 renderer DECIMAL	<p>DECIMAL レンダーは、カラムのデータをDecimal (10進数、少数)表示する場合に 使用するクラスです。 負数の場合はspanタグclass="minus"を付けて出力します。</p> <p>表示パラメータに与えられた文字列は、java.text.DecimalFormat を 使用してフォーマットされます。 フォーマット変換前に、カンマなどの数値変換時にエラーになる情報を削除しておきます。</p> <p>フォーマットルールは、java.text.DecimalFormat を参照願います。 標準のフォーマットは"#,##0.#"です。</p> <p>カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。</p>
org.opengion.plugin.column.Renderer_DECIMAL	
5.5.4.2 renderer FILTER	<p>特定のHTMLタグのエスケープ文字を元のタグに戻して表示するクラスです。</p> <p>HTMLレンダーで、HTMLタグ(<'>や'>)が('&lt;'や'&gt;')に変換 されますが、この FILTERでは、特定の変換済みタグを元のHTMLに戻す処理を行います。 現時点では、<u> , </u> ,
 の3種類です。 これは、帳票システムで、データに含まれるHTMLを変換しないと、HTMLのレイアウトが 崩れる為、3.7.0.1 (2005/01/31) で、帳票データのHTMLエスケープ処理を導入しました。 この時点で、セル内の改行を表す
 も、エスケープされる為、3.7.1.1 (2005/05/31)にて
のみ、そのまま元に戻す処理が入っています。 今回は、指定のデータに下線を引く <u> タグと、今後このような変換対象が現れる 可能性を考慮して、既存のレンダーに実装しました。 現状の帳票システムでは、エンジンのレンダー経由で変換され、HTML可されているため、 カラムリソース(システムIDがGE)に逆変換したいカラムをこの FILTER レンダーで 登録すれば、元に戻すことが可能になります。</p> <p>クロスサイトスクリプティング問題に対応するフィールドに対して 定義することにより、エスケープ処理を行います。</p> <p>カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。</p>
org.opengion.plugin.column.Renderer_FILTER	
5.5.4.2 renderer FORM	<p>FORM レンダーは、表示パラメータで指定された FORM を表示するクラスで、 元のValue を、\$1 として、使用可能です。 (コロンで区切られた値の表示のコントロールを行うレンダーです)</p> <p>カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。</p>

バージョン 属性名称 属性クラス	内容
ATTGLASS	
形式サンプル	
org.opengion.plugin.column.Renderer_FORM	
5.5.4.2 renderer HM	HM レンダーは、カラムのデータを時：分に分けて表示する場合に使用するクラスです。 4桁以上6桁以下である必要があります。5, 6桁目は無視されます。 カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_HM	
5.5.4.2 renderer HMS	HMS レンダーは、カラムのデータを時：分：秒に分けて表示する場合に使用するクラスです。 カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_HMS	
5.5.4.2 renderer HTML	HTML レンダーは、HTMLタグを含むデータを表示する場合に使用するクラスです。 クロスサイトスクリプティング問題に対応するフィールドに対して定義することにより、エスケープ処理を行います。 カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_HTML	
5.5.4.2 renderer KANA	KANA レンダーは、カラムのデータに対し、半角カナを全角カナに変換して表示する場合に使用するクラスです。 カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_KANA	
5.5.4.2 renderer LABEL	LABEL レンダーは、カラムのデータをそのまま文字列として表示する場合に使用するクラスです。 カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_LABEL	
5.5.4.2 renderer MD	YMD レンダーは、カラムのデータを日付(月/日)表示する場合に使用するクラスです。 元の文字列の5, 6桁目を月として、7, 8桁目を日として表示します。 カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_MD	
5.5.4.2 renderer MENU	MENU レンダーは、カラムのデータをコードリソースに対応したラベルでプルダウンメニュー表示する場合に使用するクラスです。 カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_MENU	
5.5.4.2 renderer MONEY	MONEY レンダーは、カラムのデータを金額表示する場合に使用するクラスです。 マイナス時の表示は、id="minus" をキーに CSSファイルで指定しています。 通貨は、標準では、¥ ですが、値:記号 という形式で指定すれば、各値ごとに通貨を指定できます。(ただし、通貨変換は、サポートしていません。) 負数の場合はspanタグclass="minus"を付けて出力します。 カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_MONEY	
5.5.4.2 renderer MULTIQUERY	MULTIQUERY レンダーは、表示パラメータで指定された SQL文の実行結果を表示するクラスです。 複数件のデータが取得された場合、各データは、spanタグで囲われて返されます。 値1値2値3 第2カラムを指定した場合、その値は、spanタグのclass属性として付加されます。 また、SQL文の指定には、元のValue を、\$1 として使用可能です。 (\$Cで自身のカラム名を参照することも可能です) カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_MULTIQUERY	
5.5.4.2 renderer NBSP	NBSP レンダーは、内部のスペースを、 という文字列に置き換えます。 これは、文字列にスペースが含まれている場合、 というコードにエスケープすることで、HTML 上で、連続したスペースを表示します。 通常、PRE レンダーで表示するなどの方法もありますが、NBSP でないとスペースにならない場合(たとえば、プルダウンメニューのオプション文字列など)に対応できます。 また、レンダーパラメータに、数字を指定すれば、その文字数で強制的に Fill埋めするため、固定長の表示にも使用できます。 これにより、連続するスペースを、そのまま表示することが出来ます。 カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_NBSP	
5.5.4.2 renderer NUMBER	NUMBER レンダーは、カラムのデータを数字表示する場合に使用するクラスです。 x, yの形式で表示パラメータを指定可能です。 負数の場合はspanタグclass="minus"を付けて出力します。 フォーマットには、java.text.NumberFormat を使用せずに、独自クラスを使用しておりdouble 以上の精度をもつ値でも正確に変換できます。 カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_NUMBER	
5.5.4.2 renderer PASSWD	PASSWD レンダーは、カラムのデータをパスワード情報(*****)として表示する場合に使用するクラスです。 パスワード表示とは、すべての表示を、***** で表示します。 カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_PASSWD	
5.5.4.2 renderer PN	PN レンダーは、カラムのデータを品番情報(1 1桁の文字列を3-5-3表示)として表示する場合に使用するクラスです。 カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。

バージョン 属性名称 属性クラス	内容
ATTGLASS	
形式サンプル	
	このクラスは、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_PN	
5.5.4.2 renderer PN2	PN2 レンダーは、カラムのデータを品番情報(1桁の文字列を3-5-3)に対して、それぞれ、“PN_1”, “PN_2”, “PN_3” というクラスを付加します。 これにより、真ん中の部分だけの色を変えたりすることができます。 カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_PN2	
5.5.4.2 renderer PRE	PRE レンダーは、カラムのデータをそのまま PRE表示する場合に使用するクラスです。 カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_PRE	
5.5.4.2 renderer QUERY	QUERY レンダーは、表示パラメータで指定された SQL文の実行結果を表示するクラスで、元のValue を、\$1 として使用可能です。 又、\$Cで自身のカラム名が使用可能です。 カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_QUERY	
5.5.4.2 renderer RADIO	RADIO レンダーは、カラムのデータをコードリソースに対応したラジオボタンの代替ラベルで表示する場合に使用するクラスです。 カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_RADIO	
5.5.4.2 renderer SLABEL	SLABEL レンダーは、桁数の長いデータをコンパクトに表示させる LABEL レンダーの類似クラスです。 全角2Byte / 半角および半角カタカナを 1Byte で簡易計算し、指定の桁数でカットします。 初期値は、20Byteで、桁数は、表示パラメータ(RENDERER_PARAM)で指定します。 文字をカットした場合は、最後に『...』を追加し、カット前の文字を title 属性に設定することで、マウスをカット後の文字に載せると、カット前の値がチップ表示されます。 カット文字... カットされなかった場合は、元の文字がそのまま表示されます。 カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、表示パラメータになにも指定しない(デフォルト)場合は、すべて同一のオブジェクトを返します。それ以外は、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_SLABEL	
5.5.4.2 renderer TEXTAREA	TEXTAREA レンダーは、カラムのデータをテキストエリアで表示する場合に使用するクラスです。 readonlyのテキストエリアでclass=renderer-textareaとして出力し、name属性は付けません。(データは送信されません) エリアの縦、横サイズはエディタのテキストエリアと同様にして算出されます。 カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_TEXTAREA	
5.5.4.2 renderer TMSTMP	TMSTMP レンダーは、日付ネイティブのカラムのデータから、数字部分だけをピックアップし、日時(年/月/日 時:分:秒)表示する場合に使用するクラスです。 数字以外は、各データベースによって、まちまちなので、'/'、':'、'-'、' ' を削除します。 なお、並び順は、必須に、YYYYMMDDHHMISS の順番とし、January 8 などの形式や、04:05 AM 、1/8/1999 などの形式は受け付けることは出来ません。 あくまで、先頭から 数字を順次取得し、YYYYMMDDHHMISS 形式に当てはめていきます。 カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_TMSTMP	
5.5.4.2 renderer WRITABLE	先頭1文字目が、アンダーバー“_” の場合は書き込み禁止属性を考慮するレンダークラスです。 (つまり先頭の“_”を削った状態で表示します) データは“_”付きのため、テーブルモデルで登録に使用する際はentryタグのWRTCTRLを利用してください。 カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_WRITABLE	
5.5.4.2 renderer XXXX	XXXX レンダーは、パラメータで指定された XXXXフォーマットに対して、値を変換します。たとえば、郵便番号や、電話番号など、ハイフン編集や、日付や時刻などの / や : 編集を行います。 例) 1234567 XXX-XXXX ⇒ 123-4567 例) 1234567890 (XXX)-XXX-XXXX ⇒ (123)-456-7890 例) 20060715 XXXX/XX/XX ⇒ 2006/07/15 XXXX フォーマットよりデータが少ない場合は、以下表示されません。 XXXX-XXXX の場合に、データが4桁の場合は、ハイフンは含まれません。 フォーマットの初期値はXXXXXXXXXXXXXXXXXです。 カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_XXXX	
5.5.4.2 renderer YM	YM レンダーは、カラムのデータを日付(年/月)表示する場合に使用するクラスです。 カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_YM	
5.5.4.2 renderer YMD	YMD レンダーは、カラムのデータを日付(年/月/日)表示する場合に使用するクラスです。 カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_YMD	
5.5.4.2 renderer YMD31	YMD レンダーは、カラムのデータを日付(年/月/日)表示する場合に使用するクラスです。 9999年のデータは"" (0ストリング)で返します。 カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column.Renderer_YMD31	
5.5.4.2	YMDH レンダーは、カラムのデータを日時(年/月/日 時:分:秒)表示する場合に使用するクラスです。

バージョン 属性名称 属性クラス	内容
ATTCLASS 形式サンプル	
renderer YMDH	カラムの表示に必要な属性は、DBCColumn オブジェクト より取り出します。 このクラスは、DBCColumn オブジェクト毎に 1 つ作成されます。
org.opengion.plugin.column.Renderer_YMDH	
5.5.4.2 editor AUTOAREA	AUTOAREA エディターは、カラムのデータをテキストエリアで編集する場合に使用するクラスです。 エリアの大きさは、表示する文字列によって、自動的に変更されます。初期値や、文字数が小さい場合のサイズは、デフォルト値を使用するか、編集パラメータに、x, y 形式で指定された値を使います。 列方向での最大桁数を、指定することが可能です。システムパラメータで HTML_AUTOAREA_MAX_COL_SIZE を指定することで、折り返し列数の調整も同時に行われます。0 が指定された場合は、無制限になります。 HTML_AUTOAREA_MAX_ROW_SIZE を指定することで、行数の最大値を指定することが可能です。0 が指定された場合は、無制限になります。 編集パラメータは、『行、列』指定可能です。例えば、5, 10 とすると、5 行 10 列のテキストエリアを最小範囲として設定できます。 初期値は、HTML_COLUMNS_MAXSIZE で指定の列数と、定義されているデータサイズ割る HTML_COLUMNS_MAXSIZE で、5 を超えない値を、行数としています。 編集パラメータの、『行、列』指定で、同時に、最大行列数の指定も可能です。5-10, 15-20 とすると、最小 5 行-最大 10 行で、最小 15 列-最大 20 列のテキストエリアを指定できます。編集パラメータでの最大値指定は、システムパラメータでの最大値指定より、優先されます。 このエディターでは、カラムの内容に応じて、書込み禁止属性を強制的に付与する事も可能です。 value (検索結果) の先頭 1 文字目が、アンダーバー (_) の場合は、編集モードになりません。(読取専用) データベースに書き込むときには、通常のアンダーバー無しの文字列に変換して登録します。 カラムの表示に必要な属性は、DBCColumn オブジェクト より取り出します。 このクラスは、DBCColumn オブジェクト毎に 1 つ作成されます。
org.opengion.plugin.column.Editor_AUTOAREA	
5.5.4.2 editor CHBOX	カラムのデータをチェックボックスで編集する場合に使用するエディタークラスです。 このエディターを使用する場合は、entryタグの、checkboxName 属性で、チェックボックスとして使用するカラム名をカンマ区切りで記述する必要があります。 また、値としては、“0” と、“1” のみ 有効です。(“0” がOFF状態で、“1” がON状態です。) チェックボックスは、通常チェックが入らないと何も送信されません。“0”から“1”への状態変化は、チェックされるためリクエスト値が送信されますが、“1”から“0”の場合は、送信されない為、データのクリアを取得することが出来ません。 このクラスでは、状態にかかわらず、hidden で、“0”を送信しています。受信側では、常に“0”がくる為、何もチェックされない場合は、“0”が設定されたことになります。 “1”が設定されると、“0”と“1”が両方 (同じキーで) 取得されます。その場合は、“1”を見つけることで、“1”をセットされたことが判ります。 編集パラメータに“SEQ”と記述することで正方向にしか選べないシークチェックボックスを実現できます。 これにより、シーケンスにステータスを順に挙げていくような、チェックボックスを作成することが出来ます。(逆に戻れないメニュー) このエディタはeventColumnに対応していません。 カラムの表示に必要な属性は、DBCColumn オブジェクト より取り出します。 このクラスは、DBCColumn オブジェクト毎に 1 つ作成されます。
org.opengion.plugin.column.Editor_CHBOX	
5.5.4.2 editor CHBOX2	CHBOX2 エディターは、カラムのデータをチェックボックスで編集する場合に使用するクラスです。 このエディターは、CHBOXとは異なりチェックボックス特有の制御を全く行いません。 特性としては、typeがcheckboxであるという1点を除いて、TEXT エディターと同じです カラムの表示に必要な属性は、DBCColumn オブジェクト より取り出します。 このクラスは、DBCColumn オブジェクト毎に 1 つ作成されます。
org.opengion.plugin.column.Editor_CHBOX2	
5.5.4.2 editor COLOR	COLOR エディターは、カラムのデータをカラーピッカーで選択する場合に使用するクラスです。 値は#FFFFFFのように#付き7桁で入ります。 カラムの表示に必要な属性は、DBCColumn オブジェクト より取り出します。 このクラスは、DBCColumn オブジェクト毎に 1 つ作成されます。
org.opengion.plugin.column.Editor_COLOR	
5.5.4.2 editor COLUMN	動的カラムのデータを編集する場合に使用するエディタークラスです。 値の文字列より、DBCColumn を作成して、そのエディター文字列を動的に作成します。 このエディタはeventColumnに対応していません。 カラムの表示に必要な属性は、DBCColumn オブジェクト より取り出します。 このクラスは、DBCColumn オブジェクト毎に 1 つ作成されます。
org.opengion.plugin.column.Editor_COLUMN	
5.5.4.2 editor CRYPT	パスワード情報など、重要な情報の暗号化された情報を編集する場合に使用するクラスです。 このクラスの暗号化は秘密キーによる可逆変換なので、変換方式と秘密キーが判ると元に戻すことが可能です。それでも、何もしないよりははるかにましです。 データベース等へ登録した暗号化されたデータを編集する場合に、使用します。
org.opengion.plugin.column.Editor_CRYPT	
5.5.4.2 editor DBMENU	カラムの編集パラメーターのSQL文の実行結果より、プルダウンメニューを作成して編集する場合に使用するエディタークラスです。 編集パラメータには、プルダウンメニューを作成するための、SQL文を記述します。 このSQL文は、select KEY, LABEL from xx ... という構文で、KEY部分とLABEL部分が選択されます。現時点では、それ以降の検索結果は無視されますが、将来的な拡張があるため、検索は、KEYとLABELの2つにしてください。 第一カラムはキー、第二カラムはラベルでこの2つは必須です。第三カラムは短縮ラベル、第四カラムはグループ(optgroup)、第五カラムは色付け等に使うクラスです。 短縮ラベルが設定されている場合、一覧でこのエディタが適用されると短縮ラベル表示をした上でマウスオーバー時はツールチップで通常のラベルを表示します。 各カラムの値(value値)に、AAA:BBB:CCC:DDD という値を設定できます。これは、\$1, \$2, \$3, \$4 に割り当てなおして、QUERYを実行します。また、\$1 は、本来の値として、メニューの初期値設定等に使用します。上記の例では、AAA が値で、それ以降は、引数になります。 又、\$0には自分自身のカラム名を割り当てます。 この機能を使用すれば、動的メニューを行ごとに条件を変えて作成することが

バージョン 属性名称 属性クラス	内容
ATTGLASS	
形式サンプル	
	<p>可能になります。</p> <p>例：select KEY, LABEL from xx where KUBUN=' \$2' and CDK=' \$3'</p> <p>さらに、元の文字列“AAA:BBB:CCC:DDD”は、\$0 に割り当てられます。割り当てがない変数は、“” (ゼロ文字列) として、扱われます。</p> <p>編集パラメータに“SEQ”と記述することで正方向にしか選べないシークメニューを実現できます。これにより、シーケンスにステータスを順に挙げていくような、プルダウンメニューを作成することが出来ます。(逆に戻れないメニュー)</p> <p>カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト 毎に1つ作成されます。</p>
org.opengion.plugin.column.Editor_DBRADIO	
5.5.4.2 editor DBRADIO	<p>DBRADIO エディターは、カラムの編集パラメーターのSQL文の実行結果より、動的にラジオボタンを作成して編集する場合に使用するエディタークラスです。</p> <p>編集パラメータには、ラジオボタンを作成するための、SQL文を記述します。 このSQL文は、select KEY, LABEL from xx ... という構文で、KEY部分とLABEL部分が選択されます。各カラムの意味は次のようになります。</p> <p>第1カラム (必須) : ラジオボタンのキー (値)</p> <p>第2カラム : ラベル (指定されない場合は、ラベルリソースの短縮ラベルを使用します)</p> <p>第3カラム : クラス そのオプションに色づけなどを行う為の指定します。 NULL (または、ゼロ文字列) の場合は、適用されません。</p> <p>第4カラム : この値は 'false' 又は '0' である場合にそのラジオボタンを選択不可にします。 NULL (または、ゼロ文字列) の場合は、選択可能になります。</p> <p>各カラムの値 (value値) に、AAA:BBB:CCC:DDD という値を設定できます。これは、\$1, \$2, \$3, \$4 に割り当てなおして、QUERYを実行します。また、\$1 は、本来の値として、メニューの初期値設定等に使用します。上記の例では、AAA が値で、それ以降は、引数になります。</p> <p>又、\$Gには自分自身のカラム名を割り当てます。 この機能を使用すれば、動的メニューを行ごとに条件を変えて作成することが可能になります。</p> <p>例：select KEY, LABEL from xx where KUBUN=' \$2' and CDK=' \$3'</p> <p>さらに、元の文字列“AAA:BBB:CCC:DDD”は、\$0 に割り当てられます。割り当てがない変数は、“” (ゼロ文字列) として、扱われます。</p> <p>このエディタはeventColumnに対応していません。</p> <p>カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト 毎に1つ作成されます。</p>
org.opengion.plugin.column.Editor_DBRADIO	
5.5.4.2 editor DECIMAL	<p>DECIMAL エディターは、カラムのデータをDecimal (10進数、少数) 表示する場合に使用するクラスです。</p> <p>編集パラメータに与えられた文字列は、java.text.DecimalFormat を使用してフォーマットされます。 フォーマット変換前に、カンマなどの数値変換時にエラーになる情報を削除しておきます。 標準のフォーマットは“#,##0.0”です。 default値が設定されていない場合の初期値は0.0にフォーマット処理をしたものです。</p> <p>フォーマットルールは、java.text.DecimalFormat を参照願います。</p> <p>カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト 毎に1つ作成されます。</p>
org.opengion.plugin.column.Editor_DECIMAL	
5.5.4.2 editor ENTCLM	<p>動的カラムのEntryカラムを編集する場合に使用するエディタークラスです。</p> <p>Editor_ENTCLM は、Editor_EntryColumn の略で、Editor_COLUMN.java を強化した形で作成します。 これは、引数の値をキーに、DBColumn を動的に作成する機能になります。</p> <p>Editor_COLUMN との違いは、こちらは、行ではなく、Entry形式のカラムを作成するところです。つまり、行番号は、関係ありません。 通常は、カラム名_行番号 をキーとするテキストフィールドなどを出力しますが、Editor_ENTCLM は、名前そのものをキーとするテキストフィールドなどを出力します。 あと、カラム引数 (: で区切られた値) が使えます。 通常は、値 (Value) に、カラム名のみをセットしますが、コロン (:) で、区切ってパラメータを渡せます。</p> <p>カラム名:値:must:Length:Label:Editor:DBType:EditParam の順番です。</p> <p>コロンの数だけ分離しますが、数は少なくとも良いが並び順は、必須です。</p> <p>通常、このままでは、リソースに存在することが前提ですが、編集パラメータに、SAVE=TRUE というキーワードをセットすると、個々に作成した値を元に、ResourceManager に、動的に作成した LabelData を追加する機能を持たせます。 この、LabelData は、通常のLabelDataLoaderのプールではなく、ResourceManagerで、個別に管理されるため、特殊な方法を使わないと値を取り出すことはできません。 このキャッシュされたラベルを用いることで、columnCheckのエラーメッセージのラベルを動的に書き換えたラベルで表示することができます。</p> <p>さらに、編集パラメータに、QUERY=... というキーワードをセットすると、そのSQL分を実行して、コードリソースを作成します。 こちらは、SAVE=TRUE の場合のみ実行され、コードリソースのキャッシュにセットされます。</p> <p>このエディタはeventColumnに対応していません。</p> <p>カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト 毎に1つ作成されます。</p>
org.opengion.plugin.column.Editor_ENTCLM	
5.5.4.2 editor HIDDEN	<p>カラムのデータをHIDDENで編集する場合に使用するエディタークラスです。</p> <p>つまり、編集して欲しくないが、データは、飛ばしたい場合に使用します。</p> <p>カラムの表示に必要な属性は、DBColumn オブジェクト より取り出します。 このクラスは、DBColumn オブジェクト 毎に1つ作成されます。</p>
org.opengion.plugin.column.Editor_HIDDEN	

バージョン 属性名称 属性クラス	内容
ATTGLASS 形式サンプル	
5.5.4.2 editor HTML	<p>HTMLタグを含むデータを編集する場合に使用するエディタークラスです。</p> <p>データベースに対して、加工なしデータを登録しています。HTMLタグを含むデータは、DBTypeのチェックにより、通常は登録できません。これは、クロスサイトスクリプティング対策の為です。DBTypeを指定することで、HTMLタグの登録が可能になります。しかし、エンジンでは、検索時処理を優先させる為、検索時には加工していません。つまり、HTMLタグを含むデータをそのまま表示させると、HTMLのコードと解釈され、レイアウトが崩れることになります。HTMLエディターは、データのHTMLタグをエスケープすることで、表示上は、HTMLコードがそのまま表示/編集できる状態にします。</p> <p>カラムの表示に必要な属性は、DBCColumn オブジェクト より取り出します。このクラスは、DBCColumn オブジェクト毎に1つ作成されます。</p>
org.opengion.plugin.column.Editor_HTML	
5.5.4.2 editor INDBMENU	<p>INDBMENU エディターは、カラムの表示パラメーターのSQL文を実行結果より、作成したプルダウンメニューと、テキストフィールドによる入力の両方をサポートする、編集に使用するクラスです。</p> <p>JavaScript によりテキストフィールドとメニュー(コンボボックス)を重ね合わせて表示しておき、メニューで選択した値を、テキストフィールドに設定しています。このエディタを使用するには、jsp/common/inputMenu.js を予め使用できるように設定しておく必要があります。</p> <p>このエディタはeventColumnに対応していません。</p> <p>カラムの表示に必要な属性は、DBCColumn オブジェクト より取り出します。このクラスは、DBCColumn オブジェクト毎に1つ作成されます。</p>
org.opengion.plugin.column.Editor_INDBMENU	
5.5.4.2 editor INMENU	<p>INMENU エディターは、コードリソースに対応したプルダウンメニューと、テキストフィールドによる入力の両方をサポートする、編集に使用するクラスです。</p> <p>JavaScript によりテキストフィールドとメニュー(コンボボックス)を重ね合わせて表示しておき、メニューで選択した値を、テキストフィールドに設定しています。このエディタを使用するには、jsp/common/inputMenu.js を予め使用できるように設定しておく必要があります。</p> <p>このエディタはeventColumnに対応していません。</p> <p>カラムの表示に必要な属性は、DBCColumn オブジェクト より取り出します。このクラスは、DBCColumn オブジェクト毎に1つ作成されます。</p>
org.opengion.plugin.column.Editor_INMENU	
5.5.4.2 editor MENU	<p>MENU エディターは、カラムのデータをコードリソースに対応したプルダウンメニューで編集する場合に使用するクラスです。</p> <p>カラムの表示に必要な属性は、DBCColumn オブジェクト より取り出します。このクラスは、DBCColumn オブジェクト毎に1つ作成されます。</p>
org.opengion.plugin.column.Editor_MENU	
5.5.4.2 editor NUMBER	<p>NUMBER エディターは、カラムのデータを数字編集する場合に使用するクラスです。</p> <p>カラムの表示に必要な属性は、DBCColumn オブジェクト より取り出します。このクラスは、DBCColumn オブジェクト毎に1つ作成されます。</p>
org.opengion.plugin.column.Editor_NUMBER	
5.5.4.2 editor PASSWD	<p>PASSWD エディターは、パスワード情報(*****)として編集する場合に使用するクラスです。</p> <p>パスワード編集とは、すべての入力値を、***** に置き換えて画面表示します。</p> <p>カラムの表示に必要な属性は、DBCColumn オブジェクト より取り出します。このクラスは、DBCColumn オブジェクト毎に1つ作成されます。</p>
org.opengion.plugin.column.Editor_PASSWD	
5.5.4.2 editor PN	<p>PN エディターは、カラムのデータをカラムのデータを品番情報(11桁の文字列を3-5-3編集)する場合に使用するクラスです。</p> <p>カラムの表示に必要な属性は、DBCColumn オブジェクト より取り出します。このクラスは、DBCColumn オブジェクト毎に1つ作成されます。</p>
org.opengion.plugin.column.Editor_PN	
5.5.4.2 editor QUERY	<p>QUERY エディターは、編集パラメータで指定された SQL文の実行結果をテキストエリアに表示するクラスで、元のValue を、\$1 として使用可能です。又、\$Cで自身のカラム名が使用可能です。</p> <p>カラムの表示に必要な属性は、DBCColumn オブジェクト より取り出します。このクラスは、DBCColumn オブジェクト毎に1つ作成されます。</p>
org.opengion.plugin.column.Editor_QUERY	
5.5.4.2 editor RADIO	<p>RADIO エディターは、カラムのデータをコードリソースに対応したラジオボタンで編集する場合に使用するクラスです。ラジオボタンは、ラベルがない状態で表示される為、運用時には、ラベルに値(選択肢のラベル)を作成して、columnEditor 等で登録して置いてください。</p> <p>一覧検索明細登録画面等で、ラベル表示が必要な場合は、編集パラメータに、“useLabel”と記述しておくことで、ラベルを出力することが可能です。</p> <p>このエディタはeventColumnに対応していません。</p> <p>カラムの表示に必要な属性は、DBCColumn オブジェクト より取り出します。このクラスは、DBCColumn オブジェクト毎に1つ作成されます。</p>
org.opengion.plugin.column.Editor_RADIO	
5.5.4.2 editor RADIO2	<p>RADIO2 エディターは、カラムのデータをチェックボックスで編集する場合に使用するクラスです。</p> <p>このエディターは、RADIOとは異なり、コードリソースでの制御など、ラジオボタン特有の制御を全く行いません。特性としては、typeがradioであるという1点を除いて、TEXT エディターと同じです</p> <p>カラムの表示に必要な属性は、DBCColumn オブジェクト より取り出します。このクラスは、DBCColumn オブジェクト毎に1つ作成されます。</p>
org.opengion.plugin.column.Editor_RADIO2	
5.5.4.2 editor TEXT	<p>TEXT エディターは、カラムのデータをテキストフィールドで編集する場合に使用するクラスです。</p> <p>カラムの表示に必要な属性は、DBCColumn オブジェクト より取り出します。このクラスは、DBCColumn オブジェクト毎に1つ作成されます。</p>
org.opengion.plugin.column.Editor_TEXT	
	<p>TEXTAREA エディターは、カラムのデータをテキストエリアで編集する場合に使用するクラスです。</p>

バージョン 属性名称 属性クラス	内容
ATTGLASS 形式サンプル	
5.5.4.2 editor TEXTAREA	従来との違いは、cols 属性の最大値を、検索時(query画面)では、HTML_COLUMNS_MAXSIZE を、登録時(result画面)では、HTML_VIEW_COLUMNS_MAXSIZE を使用します。 エリアの大きさは、デフォルト値を使用するか、編集パラメータに、x, y形式で指定します。 カラムの表示に必要な属性は、DBCColumn オブジェクト より取り出します。 このクラスは、DBCColumn オブジェクト毎に1つ作成されます。 なお、このクラスには、初めの文字が「(アンダーバー)」の場合は書込み禁止になる仕様が一標準的に導入されています。
org.opengion.plugin.column	Editor TEXTAREA
5.5.4.2 editor UPLOAD	UPLOAD エディターは、ファイルアップロードを行う場合に使用する編集用クラスです。 ファイルアップロードで、行ごとに指定できる機能を追加します。 新しいファイルを、カラム名_NEW で指定することで、事前に指定されたファイル名に強制変更後にアップロード可能です。 つまり、inputタグのtype="file" で指定するカラム名(例:file)と、対応する新ファイル名(例:file_NEW)を作成し、新ファイル名を hidden 等で渡します。 さらに、form の送信先は、forward.jsp ではなく、実際に処理したい画面に直接送信します。これは、ファイルアップロードが、enctype="multipart/form-data" のため、forward.jsp で処理できない為です。 ちなみに、ORACLE の場合、カラム名にFILE は使えません。これは予約語になっています。 下記サンプルJSP(34CheckIn2)でも、UFILE にしています。 カラムの表示に必要な属性は、DBCColumn オブジェクト より取り出します。 このクラスは、DBCColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column	Editor UPLOAD
5.5.4.2 editor WRITABLE	先頭1文字目が、アンダーバー() の場合に、書込み禁止属性()を強制的に付与するクラスです。 ・データベースに書き込むときには、通常のアンダーバー無しの文字列とします。 このエディタはeventColumnに対応していません。 カラムの表示に必要な属性は、DBCColumn オブジェクト より取り出します。 このクラスは、DBCColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column	Editor WRITABLE
5.5.4.2 editor YM	YM エディターは、カラムのデータを日付(年/月)編集する場合に使用するクラスです。 元の値が8桁の場合は先頭6桁にsubstringされます。 カラムの表示に必要な属性は、DBCColumn オブジェクト より取り出します。 このクラスは、DBCColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column	Editor YM
5.5.4.2 editor YMD	YMD エディターは、カラムのデータを日付(年/月/日)編集する場合に使用するクラスです。 カラムの表示に必要な属性は、DBCColumn オブジェクト より取り出します。 このクラスは、DBCColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column	Editor YMD
5.5.4.2 editor YMD2	YMD エディターは、カラムのデータを日付(年/月/日)編集する場合に使用するクラスです。 YMD2はカレンダーのポップアップボタンが付属するタイプです。 このエディタはeventColumnに対応していません。 カラムの表示に必要な属性は、DBCColumn オブジェクト より取り出します。 このクラスは、DBCColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column	Editor YMD2
5.5.4.2 editor YMDH	YMDH エディターは、カラムのデータを日時(年/月/日 時:分:秒)編集する場合に使用するクラスです。 カラムの表示に必要な属性は、DBCColumn オブジェクト より取り出します。 このクラスは、DBCColumn オブジェクト毎に1つ作成されます。
org.opengion.plugin.column	Editor YMDH
5.5.4.2 dbType (Interface)DBType	データベースのカラム属性のうち、静的論理属性を規定する情報を保持しているオブジェクトです。 このインターフェースを実装したクラスは、カラムの名前、ラベル、桁数、種類(文字、数字、日付)やタイプ(全角、半角、大文字、小文字)の情報を持っています。さらに、そのカラムの表示方法(CellRenderer)や、編集方法(CellEditor)を静的に割り当てる事が可能です。 これらは、カラムの静的情報であって、そのカラムに対する値を保持したり各アプリケーション毎に変更される情報は持っていません。すべての画面で同一の情報を保持しています。 データを登録する場合の最低限の整合性チェックは、このクラスを用いて行います。 チェック方法は、種類やタイプで異なりますが、それらはこのインターフェースを実装したサブクラス毎に異なります。
org.opengion.hayabusa.db	DBType
5.5.4.2 dbType ALL	半角/全角混在の一般的な制限のない文字列を扱う為の、カラム属性を定義します。 Char または Varchar2 属性に対応する文字型クラスで、DBType_XK との違いは、valueCheck で、クロスサイトスクリプティングチェックを行わない為、' &kt;' , ' >' などのデータを、直接データベースに登録することが可能です。 よって、このDBTypeを使用する場合は、必ず、Editor_HTML 、Renderer_HTML を使用して、' &kt;' , ' >' などのデータをエスケープ表示させてください。 タイプチェックとして、以下の条件を判定します。 ・文字列長は、Byte換算での文字数との比較 ・文字パラメータの 正規表現チェック
org.opengion.plugin.column	DBType ALL
5.5.4.2 dbType CRYPT	半角/全角混在の一般的な制限のない暗号化された文字列を扱う為の、カラム属性を定義します。 ログイン認証のパスワードなどは、MD5などのハッシュコードに変換する方式が使えます。 これは、非可逆変換なので、変換後の文字列は、元に戻すことは出来ません。 一般には、この非可逆変換が使用できるのであれば、漏洩に対しては安全です。 このクラスの暗号化は秘密キーによる可逆変換なので、変換方式と秘密キーが判ると元に戻すことが可能です。それでも、何もしないよりははるかにましです。 データベース等へ登録したデータを戻して利用したいが、そのまま抜き出されるのは困る場合に、使用できます。 なお、暗号化するため、元データの整合性ではなく暗号化された後のデータでの整合性がチェックされます。
org.opengion.plugin.column	DBType CRYPT
5.5.4.2 dbType	文字列の厳密な日付属性(年/月/日)の半角の日付を扱う為の、カラム属性を定義します。 YYYYMMDD に対応している必要があります。 これは、YMD に対して、日付の整合性チェックを厳密に行います。 valueAdd(String value)による日付の加算時にも、正式な日付で加算されます。

バージョン 属性名称 属性クラス	内容
ATTCLASS 形式サンプル	
DATE	<p>タイプチェックとして、以下の条件を判定します。</p> <ul style="list-style-type: none"> ・文字列長は、直接計算で文字数との比較 ・日付使用文字チェック「('0' > c '9' < c)以外」エラー ・YYYYMMDD 日付フォーマットでの変換
org.opengion.plugin.column.DBType_DATE	DATA_DEFAULT カラムで、内容の整合性を整えます。
5.5.4.2 dbType DD	<p>DATA_DEFAULT (初期値) カラムは、データベース上に設定されていますが、DB定義スクリプトや、データベースの種類によって、本来必要な形式で取得できないときがあります。</p> <p>ここでは、初期値の後ろに コメントがある場合は、削除し、データそのものに、シングルクオートをはずします。</p> <p>定義情報出力時には、カラムの属性(数値型、文字型)</p> <p>また、postgreSQL 対応として、::属性情報が入るので、削除します。</p> <p>シーケンス使用時にも、nextval が自動的にセットさえますが、削除します。</p> <p>このクラスは、valueAction メソッドで、action="VALSET" のみ動作します。</p>
org.opengion.plugin.column.DBType_DD	文字列の時間属性(時:分:秒)の半角の時間を扱う為の、カラム属性を定義します。
5.5.4.2 dbType HMS	<p>HMMSS に対応している必要があります。</p> <p>(HMMのデータでも利用可能です)</p> <p>ただし、日付の整合性チェックは行いませんが、valueAdd(String value) での日付の加算時には、正式な日付データにて加算します。</p> <p>タイプチェックとして、以下の条件を判定します。</p> <ul style="list-style-type: none"> ・文字列長は、直接計算で文字数との比較 ・日付使用文字チェック「('0' > c '9' < c)以外」エラー ・文字パラメータの 正規表現チェック
org.opengion.plugin.column.DBType_HMS	全角のみで構成される文字列を扱う為の、カラム属性を定義します。
5.5.4.2 dbType K	<p>全角文字とは、「c < 0x7f (0xff65 <= c && c < 0xffa0) 以外」の文字で構成される文字列のことです。</p> <p>タイプチェックとして、以下の条件を判定します。</p> <ul style="list-style-type: none"> ・全角文字チェック「c < 0x7f (0xff65 <= c && c < 0xffa0) 以外」エラー ・文字列長は、Byte換算での文字数との比較 ・文字パラメータの 正規表現チェック
org.opengion.plugin.column.DBType_K	全角+改行(CR, LF)のみで構成される文字列を扱う為の、カラム属性を定義します。
5.5.4.2 dbType KCL	<p>全角文字とは、「c < 0x7f (0xff65 <= c && c < 0xffa0) 以外」の文字で構成される文字列のことです。</p> <p>タイプチェックとして、以下の条件を判定します。</p> <ul style="list-style-type: none"> ・全角文字チェック「c < 0x7f (0xff65 <= c && c < 0xffa0) 以外」かつ改行C以外「c != 0x0a && c != 0x0d以外」エラー ・文字列長は、Byte換算での文字数との比較 ・文字パラメータの 正規表現チェック
org.opengion.plugin.column.DBType_KCL	DBType_KX は、全角/半角混在のChar または Varchar2 属性に対応するクラスで一般的な制限のない文字列カラム属性を定義します。
5.5.4.2 dbType KX	DBType_XK との違いは、デフォルトで IME(仮名漢字変換)を ON にします。
org.opengion.plugin.column.DBType_KX	パスワード情報など、重要な情報のハッシュコード(MD5)を扱う為の、カラム属性を定義します。
5.5.4.2 dbType MD5	<p>パスワード情報など、重要な情報のハッシュコードに、MD5 があります。このクラスは、MessageDigestにより、MD5 でハッシュした文字を作成します。</p> <p>値としては、標準の X と同じ半角文字列「c < 0x20 c > 0x7e 以外」でのみ処理することが出来ます。</p> <p>タイプチェックとして、以下の条件を判定します。</p> <ul style="list-style-type: none"> ・文字列長は、Byte換算での文字数との比較 ・半角文字列チェック「c < 0x20 c > 0x7e 以外」エラー ・文字パラメータの 正規表現チェック ・クロスサイトスクリプティングチェック
org.opengion.plugin.column.DBType_MD5	Unicode文字列の値を HTML のエスケープ記号(&#xZZZZ;)に変換する、カラム属性を定義します。
5.5.4.2 dbType NVAR	<p>SJIS(JA16SJIS) で作成されたデータベースに、(NVARCHAR2)を使用して中国語等を登録するのは非常に複雑でかつ、リスクが大きい処理になります。</p> <p>ORACLE殿でも、自信を持つては勧められない機能とのコメントを頂いています。</p> <p>そこで、HTMLでのエスケープ文字を使用して、Unicodeを文字列化して登録する為のDBType として、新規に作成します。</p> <p>ここでは、入力文字を、キャラクタ(char)型に分解し、(&#xZZZZ;)に変換していきます。</p> <p>よって、通常に1文字(Shift-JISで2Byte, UTF-8で3Byte)が、8Byteになります。</p> <p>この変換された文字列を、HTML上でそのまま取り出すと、元のUnicode文字に戻る為、通常のShift-JISでは、扱えない文字(中国語など)でも表示可能になります。</p> <p>(NVARCHAR2)を使用して中国語等を登録する場合の文字コード変換</p> <p>APレイヤ ネットレイヤ DBレイヤ 文字入力 → 全てDBのCHARセットに変換 → NCHARの分だけUTFに変換 (UTF) (S-jis) (UTF)</p> <p>タイプチェックは、HTMLのエスケープ記号(&#xZZZZ;)に変換後に実施されます。</p> <p>データベースへの登録には、半角英数字になります。</p> <ul style="list-style-type: none"> ・文字列長は、Byte換算での文字数との比較 ・半角文字列チェック「c < 0x20 c > 0x7e 以外」エラー ・文字パラメータの 正規表現チェック ・クロスサイトスクリプティングチェック
org.opengion.plugin.column.DBType_NVAR	IBOASで実装していた、半角文字(カナ含む)名称用記号を扱う為の、カラム属性を定義します。
5.5.4.2 dbType OASNM	<p>使用可能文字は、半角英数大小文字、スペース、半角カタカナ、.,+,-,(,),*,#/,...% です。(,は含みません)</p> <p>半角カタカナのUnicode範囲は、(c >= 0xff65 && c <= 0xff9f) を有効とします。</p> <p>タイプチェックとして、以下の条件を判定します。</p> <ul style="list-style-type: none"> ・文字列長は、Byte換算での文字数との比較

バージョン 属性名称 属性クラス	内容
ATTGLASS	
形式サンプル	
	<ul style="list-style-type: none"> ・半角文字+半角カタカナ+特殊名称チェック ・文字パラメータの 正規表現チェック
org.opengion.plugin.column.DBType_OASNM	情報機器事業部向け、品番情報の文字列を扱う為の、カラム属性を定義します。
5.5.4.2 dbType PN	<p>品番情報は、3-5-3 品番情報として扱われます。また、それぞれのフィールドにおいて、使用可能文字(例えば、Rev文字列の制限)などを考慮しています。</p> <p>電気品番(1桁目が' D ' 5桁目が' 8 ' OR ' 9 ')は、一番最後の桁をリビジョンUP (ABCEFGHJKLMNPSTUVWY) □□□-□□□□□□-□□■</p> <p>機械品番(上記以外の品番)は、10桁目をリビジョンUPする。 □□□-□□□□□□-□■ (0123456789ABCEFGHJKLMNPSTUVWY)</p> <p>タイプチェックとして、以下の条件を判定します。</p> <ul style="list-style-type: none"> ・文字列長は、Byte換算での文字数との比較 ・半角文字列チェック「 c < 0x20 c > 0x7e 以外」エラー ・文字パラメータの 正規表現チェック ・クロスサイトスクリプティングチェック
org.opengion.plugin.column.DBType_PN	半角少数付き数字の NUMBER を扱う為の、カラム属性を定義します。
5.5.4.2 dbType R	<p>' 0 ' ~ ' 9 ' , ' - ' , ' . ' のみ構成されている数字型カラム属性を定義します。 さらに、カンマ',' が含まれていても OK とします。ただし、データからは取り除きます。</p> <p>タイプチェックとして、以下の条件を判定します。</p> <ul style="list-style-type: none"> ・ 0 ~ 9、マイナス(-)および小数点(.)を許可 ・整数部の長さチェック ・少数部の長さチェック ・小数点の位置チェック ・符号の位置チェック ・文字パラメータの 正規表現チェック
org.opengion.plugin.column.DBType_R	半角数字の NUMBER を扱う為の、カラム属性を定義します。
5.5.4.2 dbType S9	<p>' 0 ' ~ ' 9 ' , ' - ' のみ構成されている数字型カラム属性を定義します。 さらに、カンマ',' が含まれていても OK とします。ただし、データからは取り除きます。</p> <p>タイプチェックとして、以下の条件を判定します。</p> <ul style="list-style-type: none"> ・ 0 ~ 9 およびマイナス(-)を許可 ・整数部の長さチェック ・符号の位置チェック ・文字パラメータの 正規表現チェック
org.opengion.plugin.column.DBType_S9	一般的な半角文字列を扱う為の、カラム属性を定義します。
5.5.4.2 dbType X	<p>半角文字列とは、「 c < 0x20 c > 0x7e 以外」でのみ構成された文字列のことです。</p> <p>タイプチェックとして、以下の条件を判定します。</p> <ul style="list-style-type: none"> ・文字列長は、Byte換算での文字数との比較 ・半角文字列チェック「 c < 0x20 c > 0x7e 以外」エラー ・文字パラメータの 正規表現チェック ・クロスサイトスクリプティングチェック
org.opengion.plugin.column.DBType_X	半角数字の NUMBER を扱う為の、カラム属性を定義します。
5.5.4.2 dbType X9	<p>' 0 ' ~ ' 9 ' , ' - ' , ' . ' , ' , ' のみ構成されている文字型カラム属性を定義します。 S9 と異なり、カンマ',' が含まれていても OK とし、データからも取り除きません。</p> <p>タイプチェックとして、以下の条件を判定します。</p> <ul style="list-style-type: none"> ・ 0 ~ 9、マイナス(-)、小数点(.)およびカンマ(,)を許可 ・文字列の長さチェック ・小数点の位置チェック ・符号の位置チェック ・文字パラメータの 正規表現チェック
org.opengion.plugin.column.DBType_X9	半角文字+半角カタカナを扱う為の、カラム属性を定義します。
5.5.4.2 dbType XH	<p>半角文字+半角カタカナとは、X 属性に半角カタカナを加えた、「 c < 0x20 (c > 0x7e && c < 0xff64) (c >= 0xffa0) 以外」でのみ構成された文字列のことです。</p> <p>タイプチェックとして、以下の条件を判定します。</p> <ul style="list-style-type: none"> ・文字列長は、Byte換算での文字数との比較 ・半角文字+半角カタカナチェック ・文字パラメータの 正規表現チェック ・クロスサイトスクリプティングチェック
org.opengion.plugin.column.DBType_XH	半角文字+半角カタカナの大文字のみに制限された文字列を扱う為の、カラム属性を定義します。
5.5.4.2 dbType XHU	<p>ホスト送信用なので、半角カタカナ小文字を半角カタカナ大文字に変換します。 通常の半角文字や半角カタカナを入力すると、大文字に変換後に、チェックを行います。 半角カタカナの小文字とは、「ア,イ,ウ,エ,オ,カ,キ,ク,ケ,コ」のことで、これらの文字列を、「A,I,U,E,O,K,KI,KU,KE,CO」に置き換えます。</p> <p>タイプチェックとして、以下の条件を判定します。</p> <ul style="list-style-type: none"> ・文字列長は、Byte換算での文字数との比較 ・半角文字+半角カタカナチェック ・文字パラメータの 正規表現チェック ・クロスサイトスクリプティングチェック
org.opengion.plugin.column.DBType_XHU	半角/全角混在の一般的な制限のない半角優先文字列を扱う為の、カラム属性を定義します。
5.5.4.2 dbType XK	<p>DBType_ALL との違いは、valueCheck で、クロスサイトスクリプティングチェックを行う為、' < ' , ' > ' などのデータは登録できません。 DBType_KX との違いは、デフォルトで IME(仮名漢字変換)は OFF になります。</p> <p>タイプチェックとして、以下の条件を判定します。</p> <ul style="list-style-type: none"> ・文字列長は、Byte換算での文字数との比較 ・文字パラメータの 正規表現チェック

バージョン 属性名称 属性クラス	内容
ATTGLASS 形式サンプル	
	・クロスサイトスクリプティングチェック
org.opengion.plugin.column.DBType_XK	
5.5.4.2 dbType XKZ	<p>半角/全角混在のクラスですが、半角カタカナのみを通さない文字列を扱う為の、カラム属性を定義します。</p> <p>半角カタカナの定義は、。から`までです。 半角カタカナを入力すると、全角カタカナに変換して、登録しています。 この変換処理に、KanaFilter#han2zen を使用しています。これは、 『CJKV日中韓越情報処理』のフィルタールゴリズムで濁点や半濁点の正しい処理も含まれています。</p> <p>タイプチェックとして、以下の条件を判定します。</p> <ul style="list-style-type: none"> ・文字列長は、Byte換算での文字数との比較 ・半角文字+全角カタカナチェック「c < 0x20 (c >= 0xff61 && c <= 0xff9f) 以外」エラー ・文字パラメータの 正規表現チェック ・クロスサイトスクリプティングチェック
org.opengion.plugin.column.DBType_XKZ	
5.5.4.2 dbType XL	<p>半角小文字の英数字の文字列を扱う為の、カラム属性を定義します。</p> <p>入力文字列は、trim された後、小文字化(toLowerCase)された後、タイプチェックされます。 半角カタカナは半角小文字に分類されない為、対象外(使用できない)です。</p> <p>タイプチェックとして、以下の条件を判定します。</p> <ul style="list-style-type: none"> ・文字列長は、Byte換算での文字数との比較 ・半角小文字チェック「c < 0x20 c > 0x7e ('A' <= c && c <= 'Z') 以外」エラー ・文字パラメータの 正規表現チェック ・クロスサイトスクリプティングチェック
org.opengion.plugin.column.DBType_XL	
5.5.4.2 dbType XLU9	<p>半角英数字のみの文字列を扱う為の、カラム属性を定義します。</p> <p>使用可能文字は、0-9, A-Z, a-z です。(, は含みません) 小数点、カンマ、半角スペース などを含みません。</p> <p>タイプチェックとして、以下の条件を判定します。</p> <ul style="list-style-type: none"> ・文字列長は、直接計算で文字数との比較 ・半角文字チェック「0-9, A-Z, a-z」の範囲 ・文字パラメータの 正規表現チェック
org.opengion.plugin.column.DBType_XLU9	
5.5.4.2 dbType XU	<p>半角大文字の英数字の文字列を扱う為の、カラム属性を定義します。</p> <p>入力文字列は、trim された後、大文字化(toUpperCase)された後、タイプチェックされます。 半角カタカナは半角大文字に分類されない為、対象外(使用できない)です。</p> <p>タイプチェックとして、以下の条件を判定します。</p> <ul style="list-style-type: none"> ・文字列長は、Byte換算での文字数との比較 ・半角大文字チェック「c < 0x20 c > 0x7e ('a' <= c && c <= 'z') 以外」エラー ・文字パラメータの 正規表現チェック ・クロスサイトスクリプティングチェック
org.opengion.plugin.column.DBType_XU	
5.5.4.2 dbType XU9	<p>半角英数大文字のみの文字列を扱う為の、カラム属性を定義します。</p> <p>使用可能文字は、0-9, A-Z です。(, は含みません) 小数点、カンマ、半角スペース などを含みません。</p> <p>タイプチェックとして、以下の条件を判定します。</p> <ul style="list-style-type: none"> ・文字列長は、直接計算で文字数との比較 ・半角文字チェック「0-9, A-Z」の範囲 ・文字パラメータの 正規表現チェック
org.opengion.plugin.column.DBType_XU9	
5.5.4.2 dbType YM	<p>文字列の日付属性(年/月)の半角の日付を扱う為の、カラム属性を定義します。</p> <p>YYYYMM に対応している必要があります。 ただし、日付の整合性チェックは行いませんが、valueAdd(String value)による 日付(月)の加算時には、0000XX と 9999XX は、加算されず、それ以外は、 正式な日付で加算されます。</p> <p>タイプチェックとして、以下の条件を判定します。</p> <ul style="list-style-type: none"> ・文字列長は、直接計算で文字数との比較 ・日付使用文字チェック「('0' > c '9' < c) 以外」エラー ・文字パラメータの 正規表現チェック
org.opengion.plugin.column.DBType_YM	
5.5.4.2 dbType YM01	<p>文字列の開始日付属性を規定する半角文字列を扱う為の、カラム属性を定義します。 登録時に0000XXXXは00000101、9999XXXXは99991231、それ以外はYYYYMM01と変換します。</p> <p>YYYYMM01 または、0000XXXX , 9999XXXX を許可します。 また、入力データとして、YYYYMM 形式 は、許可しません。あらかじめ、 valueSet(String) でデータを変換しておく必要があります。 ただし、日付としての整合性チェックは行いません。</p> <p>タイプチェックとして、以下の条件を判定します。</p> <ul style="list-style-type: none"> ・文字列長は、直接計算で文字数との比較 ・日付使用文字チェック「('0' > c '9' < c) 以外」エラー ・文字パラメータの 正規表現チェック
org.opengion.plugin.column.DBType_YM01	
5.5.4.2 dbType YM31	<p>文字列の終了日付属性を規定する半角文字列を扱う為の、カラム属性を定義します。 登録時に0000XXXXは00000101、9999XXXXは99991231、それ以外はYYYYMM31と変換します</p> <p>YYYYMM31 または、0000XXXX , 9999XXXX を許可します。 また、入力データとして、YYYYMM 形式 は、許可しません。あらかじめ、 valueSet(String) でデータを変換しておく必要があります。 ただし、日付としての整合性チェックは行いません。</p> <p>タイプチェックとして、以下の条件を判定します。</p> <ul style="list-style-type: none"> ・文字列長は、直接計算で文字数との比較 ・日付使用文字チェック「('0' > c '9' < c) 以外」エラー ・文字パラメータの 正規表現チェック
org.opengion.plugin.column.DBType_YM31	
5.5.4.2 dbType YMD	<p>文字列の日付属性(年/月/日)の半角の日付を扱う為の、カラム属性を定義します。</p> <p>YYYYMMDD に対応している必要があります。 ただし、日付の整合性チェックは行いませんが、valueAdd(String value)による 日付の加算時には正式な日付で加算されます。</p> <p>タイプチェックとして、以下の条件を判定します。</p>

バージョン 属性名称 属性クラス	内容
ATTGLASS	
形式サンプル	
	<ul style="list-style-type: none"> ・文字列長は、直接計算で文字数との比較 ・日付使用文字チェック「('0' > c '9' < c)以外」エラー ・文字パラメータの 正規表現チェック
org.opengion.plugin.column.DBType_YMD	文字列の日付属性(年/月/日 時:分:秒)の半角の日付を扱う為の、カラム属性を定義します。
5.5.4.2 dbType YMDH	<p>YYYYMMDDHHMMSS に対応している必要があります。</p> <p>ただし、日付の整合性チェックは行いませんが、valueAdd(String value)での日付の加算時には、正式な日付データにて加算します。</p> <p>タイプチェックとして、以下の条件を判定します。</p> <ul style="list-style-type: none"> ・文字列長は、直接計算で文字数との比較 ・日付使用文字チェック「('0' > c '9' < c)以外」エラー ・文字パラメータの 正規表現チェック
org.opengion.plugin.column.DBType_YMDH	
5.5.4.2 tableFilter (Interface)TableFilter	TableFilter は、TableFilterTag で使用される DBTableModelの処理用のインターフェースです。適切な実装(サブクラス)を TableFilterTag に与えることで、DBTableModelの処理を適宜切り替えることが出来ます。
org.opengion.hayabusa.db.TableFilter	
5.5.4.2 tableFilter BIKO2CODE	<p>TableFilter_CLMSET は、TableFilter インターフェースを継承した、DBTableModel 処理用の実装クラスです。</p> <p>ここでは、DB定義書情報の備考欄を分解し、コードリソースとして登録可能な形に再構築します。入力カラムとしてBIKOが、出力カラムとして、CODE, CODENAME, SEQが必要です。それぞれのカラム名は、keys, valsの引数としても指定可能です。</p> <p>分解方法としては、まず備考欄を' ' (スペース)区切りに分解し、取り出したコード情報をDBTableModelの行として追加します。</p> <p>その上で、さらに取り出した値を' : 'で分解してコードとコード名称に分離します。</p> <p>順番(SEQ)については、備考欄に記載されている順番になります。</p> <p>また、BIKO, CODE, CODENAME, SEQ で指定したカラムが DBTableModel に存在しない場合は、処理そのものを無視します。その場合は、警告も出力されませんので、ご注意ください。</p>
org.opengion.plugin.table.TableFilter_BIKO2CODE	
5.5.4.2 tableFilter CLMSET	<p>TableFilter_CLMSET は、TableFilter インターフェースを継承した、DBTableModel 処理用の実装クラスです。</p> <p>ここでは、CLM, SYSTEM_ID, LANG より、カラムリソースのRENDERER, EDITOR, DBTYPE, BIKOを設定します。検索した DBTableModel の属性として、RENDERER, EDITOR, DBTYPE, BIKO という名称のカラムが必要です。</p> <p>引数として指定可能なのは、SYSTEM_ID, LANG のみです。</p> <p>CLM : カラムリソースのキーとなる値が設定されているカラム名を指定します。</p> <p>SYSTEM_ID: コードリソースの作成システムIDを指定します。無指定時は、ログイン時のリソースになります。</p> <p>LANG: ラベルリソースの言語を指定します。無指定時は、日本語になります。</p> <p>また、CLM, RENDERER, EDITOR, DBTYPE, BIKO で指定したカラムが DBTableModel に存在しない場合は、処理そのものを無視します。その場合は、警告も出力されませんので、ご注意ください。</p>
org.opengion.plugin.table.TableFilter_CLMSET	
5.5.4.2 tableFilter DBARG	<p>TableFilter_DBARG は、TableFilter インターフェースを継承した、DBTableModel 処理用の実装クラスです。</p> <p>ここでは、キーのCLM, LBL と、LANG より、ラベルリソースに変換した値を設定します。</p> <p>CLM : ラベルリソースのキーとなる値が設定されているカラム名を指定します。</p> <p>LBL : CLMで指定されたカラム名から値を取り出し、ラベルリソースに変換した結果をセットします。</p> <p>LANG: ラベルリソースの言語を指定します。無指定時は、日本語になります。</p> <p>CLM と LBL に同じカラムを指定すると、このフィルターを通過するたびに、変換後のラベルが再びキーとして変換しようと試みるようになります。不測の事態に陥る可能性があるため、その様な設定は出来なくなっています。</p> <p>また、CLM または LBL で指定したカラムが DBTableModel に存在しない場合は、処理そのものを無視します。その場合は、警告も出力されませんので、ご注意ください。</p>
org.opengion.plugin.table.TableFilter_DBARG	
5.5.4.2 tableFilter DBARG_OUT	<p>TableFilter_DBARG_OUT は、TableFilter インターフェースを継承した、DBTableModel 処理用の実装クラスです。</p> <p>ここでは、テーブル一覧の検索結果より、GF05 のテーブルカラム定義テーブルから必要な情報を取得し、テーブル作成スクリプトを作成します。</p> <p>出力ファイルは、テーブル名+"S.sql" という命名規則で作成します。</p> <p>検索では、(SYSTEM_ID, TBLSYU, TABLE_NAME, NAME_JA, TABLESPACE_NAME, INITIAL_EXTENT, NEXT_EXTENT, COMMENTS)の項目を取得する必要があります。</p>
org.opengion.plugin.table.TableFilter_DBARG_OUT	
5.5.4.2 tableFilter DBSELECT	<p>TableFilter_DBSELECT は、TableFilter インターフェースを継承した、DBTableModel 処理用の実装クラスです。</p> <p>ここでは、Body部にかかれたSQLを実行した結果を、テーブルモデルにセットします。</p> <p>SQL文から取得されるカラム名とテーブルモデルのカラム名は一致している必要があります。</p> <p>検索結果のカラムがテーブルモデルに存在していない場合はエラーとなります。</p> <p>以下の属性を指定しないと、データが存在しない場合はNULLがセットされます。</p> <p>また、2行以上検索された場合でも、1行目のデータのみをセットします。</p> <p>[属性]</p> <p>INNER JOIN : データが存在しない場合、テーブルの該当行を削除します。</p> <p>APPEND : 2行以上検索された場合、データをアペンドします。</p> <p>SEPARATOR : APPENDする場合の区切り文字を指定します。</p> <p>Body部にかかれたSQLには[XXXX]形式の変数が指定できます。</p>
org.opengion.plugin.table.TableFilter_DBSELECT	
5.5.4.2 tableFilter DTYPE	<p>TableFilter_DTYPE は、TableFilter インターフェースを継承した、DBTableModel 処理用の実装クラスです。</p> <p>ここでは、キーの CLS_NAME, USE_LENGTH より、DTYPE の値を設定します。</p> <p>CLS_NAME は、VARCHAR2, NUMBER などのカラムの属性を表します。</p> <p>USE_LENGTH は、長さ(使用桁数)です。</p> <p>DTYPE は、X(10) や、S9(8) などの簡易型カラム属性です。</p> <p>エンジンを使用したシステムでは、この属性より、さらに詳細にカラムを定義する、DBTYPE 属性が、あります。将来的には、この定義を使用するように切り替えていく予定です。</p> <p>CLS_NAME, USE_LENGTH, DTYPE の カラム名については、初期値はこのままですが、keys, vals に指定すれば、別名についても使用することが可能です。</p>
org.opengion.plugin.table.TableFilter_DTYPE	

バージョン 属性名称 属性クラス	内容
ATTGLASS 形式サンプル	
5.5.4.2 tableFilter INDEX	TableFilter_INDEX は、TableUpda インターフェースを継承した、DBTableModel 処理用の実装クラスです。 ここでは、テーブル一覧の検索結果より、GF07 のインデックスカラム定義テーブルから必要な情報を取得し、インデックス作成スクリプトを作成します。 出力ファイルは、テーブル名+“I.sql” という命名規則で作成します。 検索では、(SYSTEM_ID, TBLSYU, TABLE_NAME, NAME_JA, TABLESPACE_NAME, INITIAL_EXTENT, NEXT_EXTENT, COMMENTS) の項目を取得する必要があります。
org.opengion.plugin.table.	TableFilter_INDEX
5.5.4.2 tableFilter INDEX_FIREBIRD	TableFilter_INDEX_FIREBIRD は、TableUpda インターフェースを継承した、DBTableModel 処理用の実装クラスです。とくに、FireBird用のインデックス作成スクリプトを作成します。 ここでは、テーブル一覧の検索結果より、GF07 のインデックスカラム定義テーブルから必要な情報を取得し、テーブル作成スクリプトを作成します。 出力ファイルは、テーブル名+“I.sql” という命名規則で作成します。 検索では、(SYSTEM_ID, TBLSYU, TABLE_NAME, NAME_JA, TABLESPACE_NAME, INITIAL_EXTENT, NEXT_EXTENT, COMMENTS) の項目を取得する必要があります。
org.opengion.plugin.table.	TableFilter_INDEX_FIREBIRD
5.5.4.2 tableFilter INDEX_HSQLDB	TableFilter_INDEX_HSQLDB は、TableUpda インターフェースを継承した、DBTableModel 処理用の実装クラスです。とくに、HSQLDB用のインデックス作成スクリプトを作成します。 ここでは、テーブル一覧の検索結果より、GF07 のインデックスカラム定義テーブルから必要な情報を取得し、テーブル作成スクリプトを作成します。 出力ファイルは、テーブル名+“I.sql” という命名規則で作成します。 検索では、(SYSTEM_ID, TBLSYU, TABLE_NAME, NAME_JA, TABLESPACE_NAME, INITIAL_EXTENT, NEXT_EXTENT, COMMENTS) の項目を取得する必要があります。
org.opengion.plugin.table.	TableFilter_INDEX_HSQLDB
5.5.4.2 tableFilter INDEX_MYSQL	TableFilter_INDEX_MYSQL は、TableUpda インターフェースを継承した、DBTableModel 処理用の実装クラスです。とくに、MySQL用のインデックス作成スクリプトを作成します。 ここでは、テーブル一覧の検索結果より、GF07 のインデックスカラム定義テーブルから必要な情報を取得し、テーブル作成スクリプトを作成します。 出力ファイルは、テーブル名+“I.sql” という命名規則で作成します。 検索では、(SYSTEM_ID, TBLSYU, TABLE_NAME, NAME_JA, TABLESPACE_NAME, INITIAL_EXTENT, NEXT_EXTENT, COMMENTS) の項目を取得する必要があります。
org.opengion.plugin.table.	TableFilter_INDEX_MYSQL
5.5.4.2 tableFilter INDEX_POSTGRE	TableFilter_INDEX_POSTGRE は、TableUpda インターフェースを継承した、DBTableModel 処理用の実装クラスです。とくに、PostgreSQL用のインデックス作成スクリプトを作成します。 ここでは、テーブル一覧の検索結果より、GF07 のインデックスカラム定義テーブルから必要な情報を取得し、テーブル作成スクリプトを作成します。 出力ファイルは、テーブル名+“I.sql” という命名規則で作成します。 検索では、(SYSTEM_ID, TBLSYU, TABLE_NAME, NAME_JA, TABLESPACE_NAME, INITIAL_EXTENT, NEXT_EXTENT, COMMENTS) の項目を取得する必要があります。
org.opengion.plugin.table.	TableFilter_INDEX_POSTGRE
5.5.4.2 tableFilter INDEX_SQLSERVER	TableFilter_INDEX_SQLSERVER は、TableUpda インターフェースを継承した、DBTableModel 処理用の実装クラスです。とくに、SQLServer用のインデックス作成スクリプトを作成します。 ここでは、テーブル一覧の検索結果より、GF07 のインデックスカラム定義テーブルから必要な情報を取得し、テーブル作成スクリプトを作成します。 出力ファイルは、テーブル名+“I.sql” という命名規則で作成します。 検索では、(SYSTEM_ID, TBLSYU, TABLE_NAME, NAME_JA, TABLESPACE_NAME, INITIAL_EXTENT, NEXT_EXTENT, COMMENTS) の項目を取得する必要があります。
org.opengion.plugin.table.	TableFilter_INDEX_SQLSERVER
5.5.4.2 tableFilter LABEL	TableFilter_LABEL は、TableFilter インターフェースを継承した、DBTableModel 処理用の実装クラスです。 ここでは、キーのCLM, LBL と、LANG より、ラベルリソースに変換した値を設定します。 SYSTEM_ID: ラベルリソースの作成システムIDを指定します。無指定時は、ログイン時のリソースになります。 LANG: ラベルリソースの言語を指定します。無指定時は、日本語になります。 CLM : ラベルリソースのキーとなる値が設定されているカラム名を指定します。 LBL : CLMで指定されたカラム名から値を取り出し、ラベルリソースに変換した結果をセットします。 CLM と LBL に同じカラムを指定すると、このフィルターを通過するたびに、変換後のラベルが再びキーとして変換しようと試みるようになります。不測の事態に陥る可能性があるため、その様な設定は出来なくなっています。 また、CLM または LBL で指定したカラムが DBTableModel に存在しない場合は、処理そのものを無視します。その場合は、警告も出力されませんので、ご注意ください。
org.opengion.plugin.table.	TableFilter_LABEL
5.5.4.2 tableFilter REPORTDATA	TableFilter_REPORTLAYOUT は、TableFilter インターフェースを継承した、DBTableModel 処理用の実装クラスです。 ここでは、指定された要求NOに対してGE51(帳票明細データ)をGE52(帳票レイアウトテーブル)の定義に従って、分割し、DBTableModelを生成します。
org.opengion.plugin.table.	TableFilter_REPORTDATA
5.5.4.2 tableFilter REPORTLAYOUT	TableFilter_REPORTLAYOUT は、TableFilter インターフェースを継承した、DBTableModel 処理用の実装クラスです。 ここでは、GE52(帳票レイアウトテーブル)の変更に、 GE52のSEQ, 開始位置 (GE54のSQLの再定義) 廃止 を行うための情報を生成しています。
org.opengion.plugin.table.	TableFilter_REPORTLAYOUT
5.5.4.2 tableFilter ROTATE	TableFilter_ROTATE は、TableFilter インターフェースを継承した、DBTableModel 処理用の実装クラスです。 ここではテーブルの回転、及びその逆回転を行います。 KEY_CLM : キーカラム(複数指定可) ROTATE_CLM : 回転するカラム VALUE_CLM : 回転カラムの値 REVERSE : 回転(false)・逆回転(true) (任意指定 初期値:false) MUST_CLM : 必須属性を定義するカラム (任意指定 初期値:false) DEF_CLM : 初期値を定義するカラム (任意指定) ※ それぞれに指定されたカラム名が存在しない場合は、処理されませんのでご注意下さい。 ①回転 キーカラムに指定された値が同じN行を1行として回転します。 (キーカラムの値がブレイクしたタイミングで、行を変更します) このN行に含まれる回転カラムの値がカラム名に、回転カラム値が各カラムの値になります。

バージョン 属性名称 属性クラス	内容
ATTGLASS 形式サンプル	
	<p>キーカラムは、カンマ区切りで複数指定可能です。</p> <p>生成されたテーブルモデルのカラムは、始めのMカラムがキーカラムに、その後のNカラムが回転されたカラムになります。</p> <p>また、元テーブルにMUST_CLMにより、各カラムの必須属性を定義することができます。(MUST属性は、'1' 又は 'true' の場合に必須になります。)</p> <p>②逆回転 回転時の逆の挙動になります。 “キーカラムに指定されたカラム以外”を回転カラムで指定されたカラムの値として分解します。 各回転カラムの値は、回転カラム値に指定されたカラムに格納されます。</p> <p>分解後のカラム数は、キーカラム数 + 2 (回転カラム、回転カラム値) になります。 また、行数は、(分解前の行数) x (回転カラム数) になります。</p>
org.opengion.plugin.table.TableFilter_ROTATE	TableFilter_SEQUENCE は、TableFilter インターフェースを継承した、DBTableModel 処理用の実装クラスです。
5.5.4.2 tableFilter SEQUERESET	<p>ここでは、テーブルモデルのシーケンス項目の値を再セットしています。 シーケンスの値は、通常10ずつ増加しますが、BREAK_CLMが指定されており、かつ、その項目の値がブレイクした場合は、100増加させます。 また、CLEAR_CLMが指定されている場合、そのカラムの値がキーブレイクした場合は、シーケンス値を初期化(10)にします。 SEQ_CLM : シーケンス項目 BREAK_CLM : キーブレイク項目(任意) CLEAR_CLM : シーケンス初期化項目 INCREMENT : シーケンスの増分 START_NO : シーケンスの初期値</p>
org.opengion.plugin.table.TableFilter_SEQUENCE	TableFilter_SEQUENCE は、TableUpda インターフェースを継承した、DBTableModel 処理用の実装クラスです。
5.5.4.2 tableFilter SEQUENCE	<p>ここでは、シーケンス一覧の検索結果より、GF09 のシーケンス定義テーブルから必要な情報を取得し、シーケンス作成スクリプトを作成します。</p> <p>この処理を実行するには、DBTableModelのカラムとして、 SEQNAME, INCREBY, STARTVAL, MINVAL, MAXVAL, FGCCYCLE, SUCACHE が必要です。</p>
org.opengion.plugin.table.TableFilter_SEQUENCE_FIREBIRD	TableFilter_SEQUENCE_FIREBIRD は、TableUpda インターフェースを継承した、DBTableModel 処理用の実装クラスです。
5.5.4.2 tableFilter SEQUENCE_FIREBIRD	<p>ここでは、シーケンス一覧の検索結果より、GF09 のシーケンス定義テーブルから必要な情報を取得し、シーケンス作成スクリプトを作成します。</p> <p>この処理を実行するには、DBTableModelのカラムとして、 SEQNAME, INCREBY, STARTVAL, MINVAL, MAXVAL, FGCCYCLE, SUCACHE が必要です。</p> <p>※Firebirdに対して生成されるスクリプトでは、INCREBY, STARTVAL, MINVAL, MAXVAL, FGCCYCLE, SUCACHEは無視されます。</p>
org.opengion.plugin.table.TableFilter_SEQUENCE_HSQLDB	TableFilter_SEQUENCE_HSQLDB は、TableUpda インターフェースを継承した、DBTableModel 処理用の実装クラスです。
5.5.4.2 tableFilter SEQUENCE_HSQLDB	<p>ここでは、シーケンス一覧の検索結果より、GF09 のシーケンス定義テーブルから必要な情報を取得し、シーケンス作成スクリプトを作成します。</p> <p>この処理を実行するには、DBTableModelのカラムとして、 SEQNAME, INCREBY, STARTVAL, MINVAL, MAXVAL, FGCCYCLE, SUCACHE が必要です。</p> <p>※HSQLDBに対して生成されるスクリプトでは、MINVAL, MAXVAL, FGCCYCLE, SUCACHEは無視されます。</p>
org.opengion.plugin.table.TableFilter_SEQUENCE_MYSQL	TableFilter_SEQUENCE_MYSQL は、TableUpda インターフェースを継承した、DBTableModel 処理用の実装クラスです。
5.5.4.2 tableFilter SEQUENCE_MYSQL	<p>ここでは、シーケンス一覧の検索結果より、GF09 のシーケンス定義テーブルから必要な情報を取得し、シーケンス作成スクリプトを作成します。</p> <p>この処理を実行するには、DBTableModelのカラムとして、 SEQNAME, INCREBY, STARTVAL, MINVAL, MAXVAL, FGCCYCLE, SUCACHE が必要です。</p> <p>※MySQLに対して生成されるスクリプトでは、INCREBY, MINVAL, MAXVAL, FGCCYCLE, SUCACHEは無視されます。 なお、MySQLではシーケンスはサポートされていないため、シーケンス名と同じ名前のシーケンス採番テーブルを生成し、これを1ずつ加算してアップデートすることで、シーケンスをエミュレートします。</p>
org.opengion.plugin.table.TableFilter_SEQUENCE_POSTGRE	TableFilter_SEQUENCE_POSTGRE は、TableUpda インターフェースを継承した、DBTableModel 処理用の実装クラスです。
5.5.4.2 tableFilter SEQUENCE_POSTGRE	<p>ここでは、シーケンス一覧の検索結果より、GF09 のシーケンス定義テーブルから必要な情報を取得し、シーケンス作成スクリプトを作成します。</p> <p>この処理を実行するには、DBTableModelのカラムとして、 SEQNAME, INCREBY, STARTVAL, MINVAL, MAXVAL, FGCCYCLE, SUCACHE が必要です。</p> <p>※PostgreSQLに対して生成されるスクリプトでは、SUCACHEは無視されます。</p>
org.opengion.plugin.table.TableFilter_SEQUENCE_SQLSERVER	TableFilter_SEQUENCE_SQLSERVER は、TableUpda インターフェースを継承した、DBTableModel 処理用の実装クラスです。
5.5.4.2 tableFilter SEQUENCE_SQLSERVER	<p>ここでは、シーケンス一覧の検索結果より、GF09 のシーケンス定義テーブルから必要な情報を取得し、シーケンス作成スクリプトを作成します。</p> <p>この処理を実行するには、DBTableModelのカラムとして、 SEQNAME, INCREBY, STARTVAL, MINVAL, MAXVAL, FGCCYCLE, SUCACHE が必要です。</p> <p>※SQLServerに対して生成されるスクリプトでは、INCREBY, MINVAL, MAXVAL, FGCCYCLE, SUCACHEは無視されます。</p>

バージョン 属性名称 属性クラス	内容
ATTGLASS 形式サンプル	
	なお、SQLServerではシーケンスはサポートされていないため、シーケンス名と同じ名前のシーケンス探番テーブルを生成し、これを1ずつ加算してアップデートすることで、シーケンスをエミュレートします。(この実装は未テストです)
org.opengion.plugin.table. 5.5.4.2 tableFilter TABLE	TableFilter_SEQUENCE SQLSERVER TableFilter_TABLE は、TableFilter インターフェースを継承した、DBTableModel 処理用の実装クラスです。 ここでは、テーブル一覧の検索結果より、GF05 のテーブルカラム定義テーブルから必要な情報を取得し、テーブル作成スクリプトを作成します。 出力ファイルは、テーブル名+"\$.\$sql" という命名規則で作成します。 検索では、(SYSTEM_ID, TBLSYU, TABLE_NAME, NAME_JA, TABLESPACE_NAME, INITIAL_EXTENT, NEXT_EXTENT, COMMENTS) の項目を取得する必要があります。
org.opengion.plugin.table. 5.5.4.2 tableFilter TABLE_FIREBIRD	TableFilter_TABLE_FIREBIRD は、TableUpda インターフェースを継承した、DBTableModel 処理用の実装クラスです。とくに、Firebird用のDB定義スクリプトを作成します。 ここでは、テーブル一覧の検索結果より、GF05 のテーブルカラム定義テーブルから必要な情報を取得し、テーブル作成スクリプトを作成します。 出力ファイルは、テーブル名+"\$.\$sql" という命名規則で作成します。 検索では、(SYSTEM_ID, TBLSYU, TABLE_NAME, NAME_JA, TABLESPACE_NAME, INITIAL_EXTENT, NEXT_EXTENT, COMMENTS) の項目を取得する必要があります。
org.opengion.plugin.table. 5.5.4.2 tableFilter TABLE_HSQLDB	TableFilter_TABLE_HSQLDB は、TableUpda インターフェースを継承した、DBTableModel 処理用の実装クラスです。とくに、HSQLDB用のDB定義スクリプトを作成します。 ここでは、テーブル一覧の検索結果より、GF05 のテーブルカラム定義テーブルから必要な情報を取得し、テーブル作成スクリプトを作成します。 出力ファイルは、テーブル名+"\$.\$sql" という命名規則で作成します。 検索では、(SYSTEM_ID, TBLSYU, TABLE_NAME, NAME_JA, TABLESPACE_NAME, INITIAL_EXTENT, NEXT_EXTENT, COMMENTS) の項目を取得する必要があります。
org.opengion.plugin.table. 5.5.4.2 tableFilter TABLE_MYSQL	TableFilter_TABLE_MYSQL は、TableUpda インターフェースを継承した、DBTableModel 処理用の実装クラスです。とくに、MySQL用のDB定義スクリプトを作成します。 ここでは、テーブル一覧の検索結果より、GF05 のテーブルカラム定義テーブルから必要な情報を取得し、テーブル作成スクリプトを作成します。 出力ファイルは、テーブル名+"\$.\$sql" という命名規則で作成します。 検索では、(SYSTEM_ID, TBLSYU, TABLE_NAME, NAME_JA, TABLESPACE_NAME, INITIAL_EXTENT, NEXT_EXTENT, COMMENTS) の項目を取得する必要があります。
org.opengion.plugin.table. 5.5.4.2 tableFilter TABLE_POSTGRE	TableFilter_TABLE_POSTGRE は、TableUpda インターフェースを継承した、DBTableModel 処理用の実装クラスです。とくに、PostgreSQL用のDB定義スクリプトを作成します。 ここでは、テーブル一覧の検索結果より、GF05 のテーブルカラム定義テーブルから必要な情報を取得し、テーブル作成スクリプトを作成します。 出力ファイルは、テーブル名+"\$.\$sql" という命名規則で作成します。 検索では、(SYSTEM_ID, TBLSYU, TABLE_NAME, NAME_JA, TABLESPACE_NAME, INITIAL_EXTENT, NEXT_EXTENT, COMMENTS) の項目を取得する必要があります。
org.opengion.plugin.table. 5.5.4.2 tableFilter TABLE_SQLSERVER	TableFilter_TABLE_SQLSERVER は、TableUpda インターフェースを継承した、DBTableModel 処理用の実装クラスです。とくに、SQLServer用のDB定義スクリプトを作成します。 ここでは、テーブル一覧の検索結果より、GF05 のテーブルカラム定義テーブルから必要な情報を取得し、テーブル作成スクリプトを作成します。 出力ファイルは、テーブル名+"\$.\$sql" という命名規則で作成します。 検索では、(SYSTEM_ID, TBLSYU, TABLE_NAME, NAME_JA, TABLESPACE_NAME, INITIAL_EXTENT, NEXT_EXTENT, COMMENTS) の項目を取得する必要があります。
org.opengion.plugin.table. 5.5.4.2 tableFilter UNIQ_NAME	TableFilter_TABLE_UNIQ_NAME は、TableFilter インターフェースを継承した、DBTableModel 処理用の実装クラスです。 ここでは、NAME_IN, NAME_OUT, GROUP_KEY, TYPE より、名前を最短ユニーク化します。 例えば、氏名で、姓と名で、同姓の場合、姓(名)を付けることで、区別することができます。 NAME_IN : 名前のオリジナルのカラムを指定します。(必須) NAME_OUT : 変換後の名前を設定するカラムを指定します。NAME_INと同じでもかまいません。(必須) GROUP_KEY : 名前をユニークにするグループを指定するカラム名を指定します。(選択) グループはソートされている必要があります。内部的にはキーブレイク処理します。 TYPE : 処理の方法を指定します(初期値:1) 1:姓と名を分けます。重複分は、姓(名)形式で、ユニークになるまで、名の文字を増やします。 2:姓と名を分けます。1.と異なるのは、最初に見つけた重複分は、姓 のまま残します。 姓名の分離は、全角または、半角のスペースで区切ります。また、重複しなければ、(名)は付きません。 TYPE="2" の方式は、慣例的に、昔からいる社員は苗字そのまま、後から入社した人にだけ(名)を付けたい場合に、名前を入社年の古い順にならべることで、実現できます。
org.opengion.plugin.table. 5.5.4.2 selection (Interface)Selection	TableFilter_UNIQ_NAME データのコード情報を取り扱う共通インターフェースです。 コードのキーとラベルの情報から、HTMLのメニューやリストを作成するための オプションタグを作成したり、与えられたキーをもとに、チェック済みのオプションタグを作成したりします。
org.opengion.hayabusa.db. 5.5.4.2 selection CODE	Selection データのコード情報を取り扱うクラスです。 コードのキーとラベルの情報から、HTMLのメニューやリストを作成するための オプションタグを作成したり、与えられたキーをもとに、チェック済みのオプションタグを作成したりします。
org.opengion.hayabusa.db. 5.5.4.2 selection DB	Selection_CODE データのコード情報を取り扱うクラスです。 コードのキーとラベルの情報から、HTMLのメニューやリストを作成するための オプションタグを作成したり、与えられたキーをもとに、チェック済みのオプションタグを作成したりします。 QUERYの第1カラムは、選択キーになります。第2カラムはラベルです。ここまでは必須です。 第3カラムが存在する場合は、短縮カラムとして認識されます。存在しない場合は、短縮ラベルは使用しません。 メニュー作成用に、SELECT文を与えます。 SELECT 値,ラベル[,グループ][,クラス] FROM XXXX で指定され、値、ラベルまでは必須、グループは、optgroup に対して指定するラベルです。クラスは、そのオプションに色づけなどを行う為の指定です。 なお、グループ、クラス は、NULL(または、ゼロ文字列)の場合は、適用されません。)
org.opengion.hayabusa.db. Selection_DB	Selection_DB

バージョン
属性名称
属性クラス

内容

ATTCLASS
形式サンプル

5.5.4.2
selection
DBRADIO

データのコード情報を取り扱うクラスです。

コードのキーとラベルの情報から、HTMLのメニューやリストを作成するための オプションタグを作成したり、与えられたキーをもとに、チェック済みのオプションタグを作成したりします。
QUERYのそれぞれのカラムは、以下の意味を持ちます。
第1カラム(必須) : ラジオボタンのキー(値)
第2カラム : ラベル(指定されない場合は、ラベルリソースの短縮ラベルを使用します)
第3カラム : クラス そのオプションに色づけなどを行う為の指定します。
NULL(または、ゼロ文字列)の場合は、適用されません。
第4カラム : この値は'false'又は'0'である場合にそのラジオボタンを選択不可にします。
NULL(または、ゼロ文字列)の場合は、選択可能になります。

org.opengion.hayabusa.db.Selection_DBRADIO

5.5.4.2
selection
RADIO

データのコード情報を取り扱うクラスです。

コードのキーとラベルの情報から、HTMLのメニューやリストを作成するための オプションタグを作成したり、与えられたキーをもとに、チェック済みのオプションタグを作成したりします。

org.opengion.hayabusa.db.Selection_RADIO

5.5.4.2
viewFormType
CustomData

ヘッダ、フッタ、ボディを指定して作成する、自由レイアウトが可能な、カスタムテーブル表示クラスです。
このクラスは、ViewForm_HTMLFormatTable クラスの代替えとしても使用できます。
その場合は、thead のみ指定すれば、同じフォームが tbody にも適用されます。
これは、まさに、ViewForm_HTMLFormatTable と同じです。

AbstractViewForm により、setter/getterメソッドのデフォルト実装を提供しています。
各HTMLのタグに必要な setter/getterメソッドのみ、追加定義しています。

AbstractViewForm を継承している為、ロケールに応じたラベルを出力させる事が出来ます。

org.opengion.plugin.view.ViewForm_CustomData

5.5.4.2
viewFormType
HTMLAjaxTreeTable

JavaScript のツリー階層を持ったテーブル表示を行う、ツリーテーブル表示クラスです。

AbstractViewForm により、setter/getterメソッドのデフォルト実装を提供しています。
各HTMLのタグに必要な setter/getterメソッドのみ、追加定義しています。

AbstractViewForm を継承している為、ロケールに応じたラベルを出力させる事が出来ます。

org.opengion.plugin.view.ViewForm_HTMLAjaxTreeTable

5.5.4.2
viewFormType
HTMLCalendar

1ヶ月分のカレンダー形式で、検索結果を表示する、カレンダー表示クラスです。

AbstractViewForm により、setter/getterメソッドのデフォルト実装を提供しています。
各HTMLのタグに必要な setter/getterメソッドのみ、追加定義しています。

AbstractViewForm を継承している為、ロケールに応じたラベルを出力させる事が出来ます。

org.opengion.plugin.view.ViewForm_HTMLCalendar

5.5.4.2
viewFormType
HTMLCrossTable

クロス集計テーブル作成クラスです。

select dept.dname, emp.deptno, substrb(job,1,2) as X, job, mgr, sum(sal),count(*)
from emp,dept
where emp.deptno = dept.deptno
group by dept.dname, emp.deptno, cube(job, mgr)
order by emp.deptno, job, mgr;

HEAD1 : ヘッダー。前段と同じデータは表示させない。
HEAD2 : キーブレイクさせるカラム。また、前段と同じデータは表示させない。
HEAD3 : キーブレイクさせないカラム。また、前段と同じデータでも表示させる。
ROW : 行データのヘッダーになるカラム
COL : 列データのヘッダーになるカラム。下記のSUM1, SUM2の両方のヘッダーになる。
SUM1 : 列データの値になるカラム。
SUM2 : 列データの値になるカラム。

SUMカラムの数、キーブレイクのカラム名、グループ化するカラム名を指定することで、これらのクロス集計結果の表示方法を指定します。

breakColumn = "DEPTNO" キーブレイクのカラム名
noGroupColumns = "X" グループ化するカラム名
sumNumber = "2" SUMカラムの数
cubeXColumn = "JOB" CUBE計算の1つ目(X)カラムを指定
cubeYColumn = "MGR" CUBE計算の2つ目(Y)カラムを指定
cubeSortType = "NUMBER" CUBE Y の列ヘッダーのソート方法を指定
gokeiSortDir = "false" 合計カラムのソート方向を指定(初期値:ソートしない)
shokeiLabel = "SHOKEI" 列小計のカラムに表示するラベルID
gokeiLabel = "GOKEI" 列合計のカラムに表示するラベルID
useHeaderColumn="false" ヘッダーカラムにレンダラー、エディターを適用するかを指定
useClassAdd = "false" 各列情報のclass属性に、カラム名などを付与するかどうかを指定

各カラムの属性(HEAD, SUM等)を認識する方法

HEAD1 HEAD2 HEAD3 ROW COL SUM1 SUM2 という並びを認識する方法は、多数の前提条件を利用して、出来るだけ少ないパラメータで自動認識させています。
若干理解しにくいかもしれませんが、慣れてください。

前提条件 :
ROW, COL は、必ず1個ずつ存在する。
HEAD群、ROW, COL, SUM群 という並びになっている。
SUM群の数は、パラメータで指定する。
計算方法 :
HEAD数=カラム数(7)-SUM数(2)-1 (ROW, COL分) = 4 個 (0 ~ 3)
ROWアドレス=cubeXColumn 設定 (3) ※ アドレスは0から始まる為
COLアドレス=cubeYColumn 設定 (4)
SUMアドレス=HEAD数+1 ~ カラム数(7)-1 (5 ~ 6)

org.opengion.plugin.view.ViewForm_HTMLCrossTable

5.5.4.2
viewFormType
HTMLCustomTable

ヘッダ、フッタ、ボディを指定して作成する、自由レイアウトが可能な、カスタムテーブル表示クラスです。
このクラスは、ViewForm_HTMLFormatTable クラスの代替えとしても使用できます。
その場合は、thead のみ指定すれば、同じフォームが tbody にも適用されます。
これは、まさに、ViewForm_HTMLFormatTable と同じです。

AbstractViewForm により、setter/getterメソッドのデフォルト実装を提供しています。
各HTMLのタグに必要な setter/getterメソッドのみ、追加定義しています。

AbstractViewForm を継承している為、ロケールに応じたラベルを出力させる事が出来ます。

org.opengion.plugin.view.ViewForm_HTMLCustomTable

バージョン 属性名称 属性クラス	内容
ATTCLASS 形式サンプル	
5.5.4.2 viewFormType HTMLCustomTreeBOM	JavaScript のツリー階層を持ったテーブル表示を行う、ツリーテーブル表示クラスです。 AbstractViewForm により、setter/getterメソッドのデフォルト実装を提供しています。 各HTMLのタグに必要な setter/getterメソッドのみ、追加定義しています。 AbstractViewForm を継承している為、ロケールに応じたラベルを出力させる事が出来ます。
org.opengion.plugin.view.ViewForm.HTMLCustomTreeBOM	
5.5.4.2 viewFormType HTMLDynamic	各フィールド情報から、動的にカラムを作成する動的カラム一覧表示クラスです。 AbstractViewForm により、setter/getterメソッドのデフォルト実装を提供しています。 各HTMLのタグに必要な setter/getterメソッドのみ、追加定義しています。 AbstractViewForm を継承している為、ロケールに応じたラベルを出力させる事が出来ます。
org.opengion.plugin.view.ViewForm.HTMLDynamic	
5.5.4.2 viewFormType HTMLEntry	エントリ形式フォーム作成クラスです。 フォーマットを外部から指定することにより、自由にレイアウトを作成できます。 AbstractViewForm により、setter/getterメソッドのデフォルト実装を提供しています。 各HTMLのタグに必要な setter/getterメソッドのみ、追加定義しています。 AbstractViewForm を継承している為、ロケールに応じたラベルを出力させる事が出来ます。
org.opengion.plugin.view.ViewForm.HTMLEntry	
5.5.4.2 viewFormType HTMLFormatTable	ヘッダー部分のフォーマットに応じたテーブルを自動作成する、フォーマットテーブル作成クラスです。 AbstractViewForm により、setter/getterメソッドのデフォルト実装を提供しています。 各HTMLのタグに必要な setter/getterメソッドのみ、追加定義しています。 [XXXX]は、カラムを指定します。ラベル+入力フィールドをそれぞれtdで囲みます。 [#XXXX]は、対応するカラムのラベルを出力します。 [\$XXXX]は、対応するカラムのレンダラーを出力します。 [!XXXX]は、対応するカラムの値を出力します。 特殊記号の解釈は、HTMLFormatTextField系とHTMLFormatTable系で異なりますので ご注意ください。 AbstractViewForm を継承している為、ロケールに応じたラベルを出力させる事が出来ます。
org.opengion.plugin.view.ViewForm.HTMLFormatTable	
5.5.4.2 viewFormType HTMLFormatTextField	フォーマットを外部から指定して作成する自由レイアウトの、テキストフィールド表示クラスです。 AbstractViewForm により、setter/getterメソッドのデフォルト実装を提供しています。 各HTMLのタグに必要な setter/getterメソッドのみ、追加定義しています。 [XXXX]は、カラムを指定します。ラベル+入力フィールドをそれぞれtdで囲みます。 [#XXXX]は、テーブルタグのtdを使用せず、ラベルと入力フィールドを出力します。 [\$XXXX]は、ラベルもtdも出さずに、入力フィールドのみ出力します。 [!XXXX]は、値のみ出力します。 特殊記号の解釈は、HTMLFormatTextField系とHTMLFormatTable系で異なりますので ご注意ください。 AbstractViewForm を継承している為、ロケールに応じたラベルを出力させる事が出来ます。
org.opengion.plugin.view.ViewForm.HTMLFormatTextField	
5.5.4.2 viewFormType HTMLGanttTable	ガントチャート（テーブル形式）を作成する、ガントチャート表示クラスです。 AbstractViewForm により、setter/getterメソッドのデフォルト実装を提供しています。 各HTMLのタグに必要な setter/getterメソッドのみ、追加定義しています。 AbstractViewForm を継承している為、ロケールに応じたラベルを出力させる事が出来ます。
org.opengion.plugin.view.ViewForm.HTMLGanttTable	
5.5.4.2 viewFormType HTMLRotationTable	行と列を入れ替えて表示する、テーブル回転表示クラスです。 このクラスは、表示のみ実行可能です。旧ヘッダー部分は、第一カラムに表示されます。 このビューでは、行と列が入れ替わって表示している為、登録はできません。 AbstractViewForm により、setter/getterメソッドのデフォルト実装を提供しています。 各HTMLのタグに必要な setter/getterメソッドのみ、追加定義しています。 AbstractViewForm を継承している為、ロケールに応じたラベルを出力させる事が出来ます。
org.opengion.plugin.view.ViewForm.HTMLRotationTable	
5.5.4.2 viewFormType HTMLSeqCimTable	検索結果を自動的に表形式に変換する、テーブル作成クラスです。 ユーザー単位に表示するカラムの順番、表示可非を指定できるように対応します。 setColumnDisplay(final String columnName) に、指定された順番に 表示するというHTMLFormatTable の簡易版として用意します。 各HTMLのタグに必要な setter/getterメソッドのみ、追加定義しています。 AbstractViewForm を継承している為、ロケールに応じたラベルを出力させる事が出来ます。
org.opengion.plugin.view.ViewForm.HTMLSeqCimTable	
5.5.4.2 viewFormType HTMLSimpleList	検索結果を単純なリスト形式で表示するクラスです。 このクラスでは、検索結果を単純なリストで表示します。 表示のみでこの表示フォーマットを利用してデータ編集を行うことはできません。 各カラムのデータは、カンマによって連結され、またヘッダー部分も出力されません。 さらに各カラムの属性値に基づくclass属性等も一切出力されません。 AbstractViewForm により、setter/getterメソッドのデフォルト実装を提供しています。 各HTMLのタグに必要な setter/getterメソッドのみ、追加定義しています。 AbstractViewForm を継承している為、ロケールに応じたラベルを出力させる事が出来ます。
org.opengion.plugin.view.ViewForm.HTMLSimpleList	
5.5.4.2 viewFormType HTMLTable	検索結果を自動的に表形式に変換する、テーブル作成クラスです。 AbstractViewForm により、setter/getterメソッドのデフォルト実装を提供しています。 各HTMLのタグに必要な setter/getterメソッドのみ、追加定義しています。 AbstractViewForm を継承している為、ロケールに応じたラベルを出力させる事が出来ます。
org.opengion.plugin.view.ViewForm.HTMLTable	
5.5.4.2 viewFormType HTMLTextField	検索結果から、テキストフィールドタグを自動生成する、テキストフィールド作成クラスです。 AbstractViewForm により、setter/getterメソッドのデフォルト実装を提供しています。 各HTMLのタグに必要な setter/getterメソッドのみ、追加定義しています。 AbstractViewForm を継承している為、ロケールに応じたラベルを出力させる事が出来ます。
org.opengion.plugin.view.ViewForm.HTMLTextField	

バージョン 属性名称 属性クラス	内容
ATTGLASS 形式サンプル	
5.5.4.2 viewFormType HTMLTimeTable	<p>時間軸を持つタイムテーブルの表示を行うクラスです。</p> <p>パラメータが必要な場合は、ViewTimeTableParamTag を使用してください。</p> <p>パラメータが設定されていない場合は、ViewForm_HTMLTimeTable の初期値が使用されます。 (パラメータを使用するには、viewタグのuseParam 属性をtrueに設定する必要があります。)</p> <p>SELECT文は、日付、キー、備考、開始時刻、終了時刻、リンクが、必須項目で、この並び順は、完全に固定です。よって、カラム位置を指定する必要はありませんが、SELECT文を自由に設定することも出来ませんので、ご注意ください。</p> <p>この固定化に伴い、WRITABLE 指定も使用できません。 なお、日付、キー、備考 に関しては、columnDisplay 属性で、表示の ON/OFF 制御は可能です。 また、日付ブレイク、キーブレイクの設定で、カラム自体をテーブルの外に出すことが可能です。 (キーと備考はセットになっています。)</p> <p>タイムテーブルが空の場合のリンクを指定できます。(ViewTimeTableParam.NULL_LINK_CLM_ID) (ViewTimeTableParam の nullLinkColumn 属性) 指定しない場合は、空きのリンクは作成されません。 このリンクは、特殊で、引数に、パラメータを追加できますが、“(\$1)”、“(\$2)” で指定します。 この(\$1)、(\$2)は、開始時刻、終了時刻がセットされますが、SELECT文の固定カラムと同じ並び順ですが、DBTableModelの値を設定しているわけではありません。 空きの場合は、データ自体が存在しない場合と、日付、キー のみが 外部結合で生成されたレコードが実際に存在する場合がありますが、外部結合で生成されたレコードには、開始時刻、終了時刻はありません。(\$1) と(\$2)には、それぞれ、最小開始時刻と最大終了時刻をセットします。</p> <p>例として、&TMSTART=(\$1)&TMEND=(\$2) という文字列の (\$*) 部分を解析して割当ます。</p> <p>ブレイク処理を行うカラム I DをCSV形式でセットできます。(ViewTimeTableParam.BREAK_CLMS) (ViewTimeTableParam の breakClms 属性) これは、ブレイク毎にテーブルが分かれて、テーブルの先頭に、ブレイクした値が表示されます。 例えば、日付カラムをブレイクカラムとして設定すると、日付がブレイクするたび、日付をヘッダーに出して、テーブルを作成します。 ブレイクカラムは、CSV形式で複数指定できます。その場合は、複数指定のカラムの合成された値で、キーブレイクの判定を行います。(簡単に言うとOR判定になります。) なお、ブレイクカラムを指定した場合は、自動的に、noDisplay 属性にその値をセットします。</p>
org.opengion.plugin.view.ViewForm_HTMLTimeTable	JavaScript の ツリー階層を持ったテーブル表示を行う、ツリーテーブル表示クラスです。
5.5.4.2 viewFormType HTMLTreeBOM	<p>AbstractViewForm により、setter/getterメソッドのデフォルト実装を提供しています。 各HTMLのタグに必要な setter/getterメソッドのみ、追加定義しています。</p> <p>AbstractViewForm を継承している為、ロケールに応じたラベルを出力させる事が出来ます。</p>
org.opengion.plugin.view.ViewForm_HTMLTreeBOM	カンマ区切りダブルクォートゼロカンマファイル(CSV)形式書き込みクラスです。
5.5.4.2 writerClass CSV	DefaultTableWriter を継承していますので、ラベル、名前、データの出力部のみオーバーライドして、可変長カンマ区切り文字ファイルの出力機能を実現しています。
org.opengion.plugin.io.TableWriter_CSV	【廃止予定】加工なしカンマ区切りダブルクォートファイル(CSV)形式書き込みクラスです。 このクラスは、廃止予定です。代わりに、Data2 の セパレータをカンマに設定して使用してください。
5.5.4.2 writerClass CSV2	<p>DefaultTableWriter を継承していますので、ラベル、名前、データの出力部のみオーバーライドして、可変長カンマ区切り文字ファイルの出力機能を実現しています。</p> <p>このクラスは、文字型のデータ出力時の、先頭が0の場合のシングルコーテーションを出さないように変更したクラスです。(専用特殊クラス)</p>
org.opengion.plugin.io.TableWriter_CSV2	Calcファイルの書き出しクラスです。
5.5.4.2 writerClass Calc	DefaultTableWriter を継承していますので、ラベル、名前、データの出力部のみオーバーライドして、OpenOfficeのCalcファイルの出力機能を実現しています。
org.opengion.plugin.io.TableWriter_Calc	Calcファイルの書き出しクラスです。
5.5.4.2 writerClass CalcDef	<p>このクラスでは、通常の出カクラスと異なり、データ部分には、データではなく、{@カラム名_行番号}が出力されます。</p> <p>この出力結果は、通常、Calc帳票システムの雛形を作成するための、元情報として利用することを想定しています。</p>
org.opengion.plugin.io.TableWriter_CalcDef	Calcファイルの書き出しクラスです。
5.5.4.2 writerClass CalcDefAno	<p>このクラスでは、通常の出カクラスと異なり、以下のように出力されます。 ①データ 部分には、X(文字側)または9(数値型)をリソース定義の桁数分出力 ②各セルのコメント情報として{@ANO. カラム名_行番号}を出力</p> <p>この出力結果は、通常、Calc帳票システムの雛形を作成するための、元情報として利用することを想定しています。</p>
org.opengion.plugin.io.TableWriter_CalcDefAno	加工なし区切り文字指定データの書き出しクラスです。
5.5.4.2 writerClass Data	<p>DefaultTableWriter を継承して、データの出力部のみオーバーライドして、データそのものを加工なしで、出力します。 本来は、DefaultTableWriter の出力形態のはずですが、過去の互換性との関係で、なまデータを出力するクラスを、追加作成しました。</p>
org.opengion.plugin.io.TableWriter_Data	加工なしダブルクォート区切り文字指定データの書き出しクラスです。
5.5.4.2 writerClass Data2	<p>DefaultTableWriter を継承して、データの出力部のみオーバーライドして、データそのものを加工なしで、ダブルコーテーションで処理して出力します。 本来は、DefaultTableWriter の出力形態のはずですが、過去の互換性との関係で、なまデータを出力するクラスを、追加作成しました。 従来の CSV2 は、このクラスを使用してください。</p>
org.opengion.plugin.io.TableWriter_Data2	区切り文字指定(初期値:タブ)ゼロカンマファイルの書き出しクラスです。

バージョン 属性名称 属性クラス	内容
ATTCLASS	
形式サンプル	
5.5.4.2 writerClass Default	ラベル、名前、データの出力部のみオーバーライドすれば、各種出力フォーマットに合わせたサブクラスを実現する事が可能です。 ゼロカンマファイルとは、EXCELのゼロサプレス対策として、頭ゼロの文字型データを出力する時に、先頭にカンマ(,)を付けて、ゼロが削除(見えなくなる)現象を抑止しているファイルです。 このクラスは、可変長タブ区切り文字ファイルの出力機能を実現しています。
org.opengion.plugin.io.TableWriter_Default	ネイティブEXCELファイルの書き出しクラスです。
5.5.4.2 writerClass Excel	DefaultTableWriter を継承していますので、ラベル、名前、データの出力部のみオーバーライドして、Microsoft Excelファイルの出力機能を実現しています。 出力形式は、openXML形式にも対応しています。 出力ファイルの拡張子が、.xlsならExcel2003のバイナリ形式、.xlsxならExcel2007のopenXML形式で出力されます。
org.opengion.plugin.io.TableWriter_Excel	
5.5.4.2 writerClass Fixed	固定長文字ファイルの書き出しクラスです。 DefaultTableWriter を継承していますので、ラベル、名前、サイズ、データの出力部のみオーバーライドして、固定長文字ファイルの出力機能を実現しています。 なお、固定長出力されるのは、データ部のみで、ラベル、名前、サイズは、separator で指定された区切り記号で連結されて出力されます。
org.opengion.plugin.io.TableWriter_Fixed	
5.5.4.2 writerClass Properties	プロパティファイル形式(エンジン専用特殊形式)の書き出しクラスです。 Ver4 では、プロパティファイル形式をサポートしていません。 DefaultTableWriter を継承していますので、ラベル、名前、データの出力部のみオーバーライドして、プロパティファイルの出力機能を実現しています。
org.opengion.plugin.io.TableWriter_Properties	
5.5.4.2 writerClass T	Excelでの文字変換関数 =T("値") という文字列で書き出すクラスです。 DefaultTableWriter を継承して、データの出力部のみオーバーライドして、文字列カラム(クラス名VARCHAR2のカラム)に、=T("値") という文字列で出力するファイルの出力機能を実現しています。
org.opengion.plugin.io.TableWriter_T	
5.5.4.2 writerClass XML	TableWriter をXML形式で出力する為の実装クラスです。 DefaultTableWriter を継承していますので、ラベル、名前、データの出力部のみオーバーライドして、XML形式ファイルの出力機能を実現しています。 出力のXML形式は、拡張オラクル XDK形式のXMLファイルです。 オラクルXDK形式のXMLとは、下記のような ROWSET をトップとする ROW の集まりで1レコードを表し、各ROWには、カラム名をキーとするXMLになっています。 <pre> <ROWSET> <ROW num="1"> <カラム1>値1</カラム1> ... <カラムn>値n</カラムn> </ROW> ... <ROW num="n"> ... </ROW> </ROWSET> </pre> この形式であれば、XDK(Oracle XML Developer's Kit)を利用すれば、非常に簡単にデータベースとXMLファイルとの交換が可能です。 XDK (Oracle XML Developer's Kit) 拡張XDK形式とは、ROW 以外に、SQL処理用タグ(EXEC_SQL)を持つ XML ファイルです。これは、オラクルXDKで処理する場合、無視されますので、同様に扱うことが出来ます。この、EXEC_SQL は、それぞれの XMLデータをデータベースに登録する際に、SQL処理を自動的に流す為の、SQL文を記載します。この処理は、イベント毎に実行される為、その配置順は重要です。このタグは、複数記述することも出来ませんが、BODY部には、1つのSQL文のみ記述します。 <pre> <ROWSET tablename="GEXX" > <EXEC_SQL> 最初に記載して、初期処理(データクリア等)を実行させる。 delete from GEXX where YYYY </EXEC_SQL> <ROW num="1"> <カラム1>値1</カラム1> ... <カラムn>値n</カラムn> </ROW> ... <ROW num="n"> ... </ROW> <EXEC_SQL> 最後に記載して、項目の設定(整合性登録)を行う。 update GEXX set AA='XX' , BB='XX' where YYYY </EXEC_SQL> </ROWSET> </pre>
org.opengion.plugin.io.TableWriter_XML	
5.5.4.2 readerClass Calc	XMLパーサによる、OpenOffice.org Calcの表計算ドキュメントファイルを読み取る実装クラスです。 ①カラム名が指定されている場合 #NAMEで始まる行を検索し、その行のそれぞれの値をカラム名として処理します。 #NAMEで始まる行より以前の行については、全て無視されます。 また、#NAMEより前のカラム及び、#NAMEの行の値がNULL(カラム名が設定されていない)カラムも無視します。 読み飛ばされたカラム列に入力された値は取り込まれません。 また、#NAME行以降の#で始まる行は、コメント行とみなされ処理されません。 ②カラム名が指定されている場合 指定されたカラム名に基づき、値を取り込みます。 カラム名の順番と、シートに記述されている値の順番は一致している必要があります。 指定されたカラム数を超える列の値については全て無視されます。 #で始まる行は、コメント行とみなされ処理されません。

バージョン 属性名称 属性クラス	内容
ATTCLASS	
形式サンプル	
	また、いずれの場合も全くデータが存在していない行は読み飛ばされます。
org.opengion.plugin.io.TableReader_Calc	
5.5.4.2 readerClass Default	指定の区切り記号(初期値:タブ区切り) ファイルの読み取りクラスです。
org.opengion.plugin.io.TableReader_Default	名前、データの入力部のみオーバーライドすれば、各種入力フォーマットに合わせたサブクラスを実現する事が可能です。
5.5.4.2 readerClass Excel	POI による、EXCELバイナリファイルを読み取る実装クラスです。
org.opengion.plugin.io.TableReader_Excel	ファイル名、シート名を指定して、データを読み取ることが可能です。 第一カラムが # で始まる行は、コメント行なので、読み飛ばします。 カラム名の指定行で、カラム名が null の場合は、その列は読み飛ばします。
5.5.4.2 readerClass Fixed	入力形式は、openXML形式にも対応しています。 ファイルの内容に応じて、.xlsと.xlsxのどちらで読み取るかは、内部的に自動判定されます。
org.opengion.plugin.io.TableReader_Fixed	固定長ファイルの読み取りクラスです。
5.5.4.2 readerClass JExcel	NAMEは、先頭に、#NAME とすることで自動的にカラム名に対応付けます。 外部から、指定することも出来ます。(外部指定が優先) 固定長での読み取りでは、各行の先頭の行番号は、含めないで下さい。先頭より、データを埋めてください。
org.opengion.plugin.io.TableReader_JExcel	JExcelによるEXCELバイナリファイルを読み取る実装クラスです。
5.5.4.2 calDB GE13	ファイル名、シート名を指定して、データを読み取ることが可能です。 第一カラムが # で始まる行は、コメント行なので、読み飛ばします。 カラム名の指定行で、カラム名が null の場合は、その列は読み飛ばします。
org.opengion.plugin.calendar.CalendarQuery_GE13	カレンダーDB (GE13) の検索QUERYを定義したクラスです。
5.5.4.2 calDB HM20	QUERY は、このオブジェクトを、toString() して求めることとします。 本来は、これらのクラスの共通インターフェースを作成して、getQuery() などのメソッドを介して取得すべきですが、Object の共通クラスを利用することとします。
org.opengion.plugin.calendar.CalendarQuery_HM20	カレンダーDB (HM20) の検索QUERYを定義したクラスです。
5.5.4.2 calDB TP637	QUERY は、このオブジェクトを、toString() して求めることとします。 本来は、これらのクラスの共通インターフェースを作成して、getQuery() などのメソッドを介して取得すべきですが、Object の共通クラスを利用することとします。
org.opengion.plugin.calendar.CalendarQuery_TP637	カレンダーDB (TP637) の検索QUERYを定義したクラスです。
5.5.4.2 calDB TP652	QUERY は、このオブジェクトを、toString() して求めることとします。 本来は、これらのクラスの共通インターフェースを作成して、getQuery() などのメソッドを介して取得すべきですが、Object の共通クラスを利用することとします。
org.opengion.plugin.calendar.CalendarQuery_TP652	カレンダーDB (TP652) の検索QUERYを定義したクラスです。
5.5.4.2 calDB WI008	QUERY は、このオブジェクトを、toString() して求めることとします。 本来は、これらのクラスの共通インターフェースを作成して、getQuery() などのメソッドを介して取得すべきですが、Object の共通クラスを利用することとします。
org.opengion.plugin.calendar.CalendarQuery_WI008	カレンダーDB (WI008) の検索QUERYを定義したクラスです。
5.5.4.2 calDB ZY01	QUERY は、このオブジェクトを、toString() して求めることとします。 本来は、これらのクラスの共通インターフェースを作成して、getQuery() などのメソッドを介して取得すべきですが、Object の共通クラスを利用することとします。
org.opengion.plugin.calendar.CalendarQuery_ZY01	カレンダーDB (ZY01) の検索QUERYを定義したクラスです。
5.5.4.2 process BulkQuery	Process_BulkQueryは、データベースから読み取った内容を、一括処理するために、ParamProcess のサブクラス(Process_DBParam)にセットしたり、加工したりするFirstProcess と、ChainProcess のインターフェースを両方持った、実装クラスです。 このクラスは、上流から、下流への処理は、1度しか実行されません。 FirstProcess の検索結果は、Set オブジェクトとして、Process_DBParam に渡します。 ChainProcess は、その結果を取り出し、自分自身の処理結果と合せて加工します。 FirstProcess では、-action は、query のみです。 query は、指定のSQL文を実行し、結果のSetをParamProcessに設定します。 ChainProcess では、-action は、query、bulkSet、minus、intersect が指定できます。 query は、上記と同じです。 minus は、先のSetから、SQL文の実行結果を引き算し、結果Setを再設定します。 intersect は、先のSetから、SQL文の実行結果と重複する結果Setを再設定します。 bulkSet は、先のSetを取り出し、SQL文に加味して処理します。 流れ的には、query で検索し、minusまたはintersect でSetオブジェクトを加工し、bulkSet で利用します。例えば、ORACLEから、ユニークキーのSetを作成し、SQLServerのユニークキーをminusした結果を、ORACLEからDELETEすれば、不要なデータを削除するなどの処理が実行可能になります。 また、単純に、query だけを、チェーンすれば、単発のUPDATE文を実行することが可能です。 データベース接続先等は、ParamProcess のサブクラス(Process_DBParam)に設定された接続(Connection)を使用します。 DBID は、Process_DBParam の -configFile で指定する DBConfig.xml ファイルを使用します。 引数文字列中にスペースを含む場合は、ダブルコーテーション(“)で括って下さい。 引数文字列の 『=』 の前後には、スペースは挟めません。必ず、-key=value の様に繋げて下さい。 SQL文には、[@DATE.YMDH]等のシステム変数が使用できます。
org.opengion.fukurou.process.Process_BulkQuery	
Process_BulkQuery -action=query -dbid=DBGE -sql="select KEY from TABLE_X"	
-action=処理方法(必須) : 実行する処理方法を指定します	
-action=query 単なるSQL文を実行します。	
-action=bulkSet 実行したSQL文の結果を、Set オブジェクトに設定します。	
-action=minus Set オブジェクトと、ここでの実行結果の差分をとります。	

バージョン 属性名称 属性クラス	内容
ATTGLASS	
形式サンプル	
-action=intersect Set オブジェクトと、ここでの実行結果の積分をとります。 [-dbid=DB接続ID] : -dbid=DBGE (例: Process_DBParam の -configFile で指定する DBConfig.xml ファイルで規定) [-sql=検索SQL文] : -sql="select * from GEA08" [-sqlFile=検索SQLファイル] : -sqlFile=select.sql -sql= を指定しない場合は、ファイルで必ず指定してください。 [-sql_XXXX=固定値] : -sql_SYSTEM_ID=GE SQL文中の{@XXXX} 文字列を指定の固定値で置き換えます。 WHERE SYSTEM_ID=' {@SYSTEM_ID}' ⇒ WHERE SYSTEM_ID=' GE' [-bulkKey=XXXX] : -bulkKey=XXXX SQL文中の{@XXXX} 文字列をProcess_BulkQuery等で取得した値で置き換えます。 WHERE SYSTEM_ID IN ({@XXXX}) ⇒ WHERE SYSTEM_ID IN ('AA', 'BB', 'CC') [-bulkType=NUM STR] : -bulkType=STR Bulkの値を文字列に変換する場合に、数字型か、文字型を指定します。 数字型では、AA, BB, CC とし、文字型では、'AA', 'BB', 'CC' に変換します(初期値:STR)。 [-fetchSize=100] : フェッチする行数(初期値:100) [-display=false true] : 結果を標準出力に表示する(true)かしない(false)か(初期値:false[表示しない]) [-debug=false true] : デバッグ情報を標準出力に表示する(true)かしない(false)か(初期値:false[表示しない])	
5.5.4.2 process DBCountFilter	Process_DBCountFilter は、データベースの存在件数でフィルタリングする ChainProcess インターフェースの実装クラスです。 上流(プロセスチェーンのデータは上流から下流へと渡されます。)から受け取った LineModel を元に、データベースの存在チェックを行い、下流への処理を振り分けます。 具体的には、指定する SELECT 文は、必ず、『select count(*) from …』形式にして下さい。 検索カラムは、一つだけで、そこには数字が入ります。 データベース接続先等は、ParamProcess のサブクラス(Process_DBParam)に 設定された接続(Connection)を使用します。 引数文字列中にスペースを含む場合は、ダブルコーテーション(“)で括弧して下さい。 引数文字列の『=』の前後には、スペースは挟めません。必ず、-key=value の様に 繋げて下さい。
org.opengion.fukurou.process.Process_DBCountFilter Process_DBCountFilter -dbid=DBGE -sql="select count(*) from GEA03"	
[-dbid=DB接続ID] : -dbid=DBGE (例: Process_DBParam の -configFile で指定する DBConfig.xml ファイルで規定) [-sql=検索SQL文] : -sql="SELECT COUNT(*) FROM GEA03 WHERE SYSTEM_ID = [SYSTEM_ID] AND CLM = [CLM] AND FGJ = 'I'" [-sqlFile=検索SQLファイル] : -sqlFile=select.sql -sql や -sqlFile が指定されない場合は、エラーです。 [-count=スルー条件] : -count=[0 1 2] は、検索値に応じたスルー条件。 0: 0 件時にスルー(処理を継続) つまり、なければ継続 1: 1 件時にスルー(処理を継続) つまり、あれば継続 2: 2 件以上ある場合にスルー つまり、キー重複時に継続 [-display=false true] : 結果を標準出力に表示する(true)かしない(false)か(初期値:false[表示しない])	
5.5.4.2 process DBMerge	Process_DBMerge は、UPDATE と INSERT を指定し データベースを追加更新 する、ChainProcess インターフェースの実装クラスです。 上流(プロセスチェーンのデータは上流から下流へと渡されます。)から 受け取った LineModel を元に、DBTableModel 形式ファイルを出力します。 データベース接続先等は、ParamProcess のサブクラス(Process_DBParam)に 設定された接続(Connection)を使用します。 引数文字列中にスペースを含む場合は、ダブルコーテーション(“)で括弧して下さい。 引数文字列の『=』の前後には、スペースは挟めません。必ず、-key=value の様に 繋げて下さい。 SQL文には、{@DATE.YMDH} 等のシステム変数が使用できます。
org.opengion.fukurou.process.Process_DBMerge Process_DBMerge -dbid=DBGE -insertTable=GE41	
[-dbid=DB接続ID] : -dbid=DBGE (例: Process_DBParam の -configFile で指定する DBConfig.xml ファイルで規定) [-update=検索SQL文] : -update="UPDATE GE41 SET NAME_JA = [NAME_JA], LABEL_NAME = [LABEL_NAME] WHERE SYSTEM_ID = [SYSTEM_ID] AND CLM = [CLM]" [-updateFile=登録SQLファイル] : -updateFile=update.sql -update や -updateFile が指定されない場合は、エラーです。 [-update_XXXX=固定値] : -update_SYSTEM_ID=GE SQL文中の{@XXXX} 文字列を指定の固定値で置き換えます。 WHERE SYSTEM_ID=' {@SYSTEM_ID}' ⇒ WHERE SYSTEM_ID=' GE' [-insertTable=登録テーブルID] : INSERT文を指定する場合は不要。INSERT する場合のテーブルID [-insert=検索SQL文] : -insert="INSERT INTO GE41 (SYSTEM_ID, CLM, NAME_JA, LABEL_NAME) VALUES ([SYSTEM_ID], [CLM], [NAME_JA], [LABEL_NAME])" [-insertFile=登録SQLファイル] : -insertFile=insert.sql -insert や -insertFile や -table が指定されない場合は、エラーです。 [-insert_XXXX=固定値] : -insert_SYSTEM_ID=GE SQL文中の{@XXXX} 文字列を指定の固定値で置き換えます。 WHERE SYSTEM_ID=' {@SYSTEM_ID}' ⇒ WHERE SYSTEM_ID=' GE' [-const_XXXX=固定値] : -const_FGJ=1 LineModel のキー(const_に続く文字列)の値に、固定値を設定します。 キーが異なれば、複数のカラム名を指定できます。 [-commitCnt=commit処理指定] : 指定数毎にコミットを発行します。0 の場合は、終了までコミットしません。 [-display=false true] : 結果を標準出力に表示する(true)かしない(false)か(初期値:false[表示しない])	
5.5.4.2 process DBParam	Process_DBParam は、他のプロセスへ共通のデータベース接続を割り当てる為の、 ParamProcess インターフェースの実装クラスです。 DB接続 が必要な Process (DBCountFilter, DBMerge, DBReader, DBWriterなど)を 使用して処理する場合に、接続を指定することができます。 DBID(接続先) は、Process_DBParam の -configFile で指定する DBConfig.xml ファイルを使用します。
org.opengion.fukurou.process.Process_DBParam Process_DBParam -infoUSER=C00000 -infoPGID=GE1234 -configFile=DBConfig.xml	
-infoUSER=実行ユーザー : DB接続履歴取得用の実行ユーザー(例:C00000) -infoPGID=実行プログラムID : DB接続履歴取得用の実行プログラムID(例:GE1234) -configFile=実行プログラムID : DB接続情報設定 XMLファイル(例:DBConfig.xml)	
5.5.4.2	Process_DBReaderは、データベースから読み取った内容を、LineModel に設定後、 下流に渡す、FirstProcess インターフェースの実装クラスです。 データベースから読み取った内容より、LineModelを作成し、下流(プロセス チェーンは、チェーンしているため、データは上流から下流へと渡されます。)に 渡します。ここで指定できるのは、検索系SQL のみです。

バージョン 属性名称 属性クラス	内容
ATTGLASS 形式サンプル	
process DBReader	<p>データベース接続先等は、ParamProcess のサブクラス (Process_DBParam) に設定された接続 (Connection) を使用します。</p> <p>引数文字列中にスペースを含む場合は、ダブルコーテーション (") で括弧して下さい。 引数文字列の 『=』 の前後には、スペースは挟めません。必ず、-key=value の様に繋げてください。</p> <p>SQL文には、{@DATE.YMDH}等のシステム変数が使用できます。</p>
org.opengion.fukuroku.process.Process.DBReader Process_DBReader -dbid=DBGE -sql="select * from GEA08"	
<pre> [-dbid=DB接続ID] : -dbid=DBGE (例: Process_DBParam の -configFile で指定する DBConfig.xml ファイルで規定) [-sql=検索SQL文] : -sql="select * from GEA08" [-sqlFile=検索SQLファイル] : -sqlFile=select.sql -sql= を指定しない場合は、ファイルで必ず指定してください。 [-sql_XXXX=固定値] : -sql_SYSTEM_ID=GE SQL文中の{@XXXX}文字列を指定の固定値で置き換えます。 WHERE SYSTEM_ID=' {@SYSTEM_ID}' ⇒ WHERE SYSTEM_ID=' GE' [-fetchSize=100] : フェッチする行数 (初期値:100) [-display=false true] : 結果を標準出力に表示する (true)かしない (false)か (初期値:false[表示しない]) </pre>	
5.5.4.2 process DBWriter	<p>Process_DBWriter は、上流から受け取ったデータをデータベースに書き込む ChainProcess インターフェースの実装クラスです。</p> <p>上流 (プロセスチェーンのデータは上流から下流へと渡されます。) から受け取った LineModel を元に、データベースへの書き込みを行います。</p> <p>データベース接続先等は、ParamProcess のサブクラス (Process_DBParam) に設定された接続 (Connection) を使用します。</p> <p>引数文字列中にスペースを含む場合は、ダブルコーテーション (") で括弧して下さい。 引数文字列の 『=』 の前後には、スペースは挟めません。必ず、-key=value の様に繋げてください。</p> <p>SQL文には、{@DATE.YMDH}等のシステム変数が使用できます。</p>
org.opengion.fukuroku.process.Process.DBWriter Process_DBWriter -dbid=DBGE -table=GE41	
<pre> [-dbid=DB接続ID] : -dbid=DBGE (例: Process_DBParam の -configFile で指定する DBConfig.xml ファイルで規定) [-table=登録テーブルID] : SQL文を指定する場合は不要。INSERT する場合のテーブルID [-sql=検索SQL文] : -sql="UPDATE GE41 SET NAME_JA = [NAME_JA], LABEL_NAME = [LABEL_NAME] WHERE SYSTEM_ID = [SYSTEM_ID] AND CLM = [CLM]" [-sqlFile=登録SQLファイル] : -sqlFile=update.sql -sql や -sqlFile が指定されない場合は、-table で指定のテーブルに全カラム insert です。 [-sql_XXXX=固定値] : -sql_SYSTEM_ID=GE SQL文中の{@XXXX}文字列を指定の固定値で置き換えます。 WHERE SYSTEM_ID=' {@SYSTEM_ID}' ⇒ WHERE SYSTEM_ID=' GE' [-const_XXXX=固定値] : -const_FGJ=1 LineModel のキー (const_ に続く文字列) の値に、固定値を設定します。 キーが異なれば、複数のカラム名を指定できます。 [-omitClms=AAA, BBB, ...] : -omitClms=UNIQ, FGJ, DYSET -table 属性でINSERT文を自動作成する場合、取り除くカラム名を カンマ区切りで複数指定できます。 [-commitCnt=commit処理指定] : 指定数毎にコミットを発行します。0 の場合は、終了までコミットしません。 [-display=false true] : 結果を標準出力に表示する (true)かしない (false)か (初期値:false[表示しない]) </pre>	
5.5.4.2 process FileCopy	<p>Process_FileCopy は、上流から受け取った FileLineModel を処理する、ChainProcess インターフェースの実装クラスです。</p> <p>上流から受け取った FileLineModel の ファイルから、inPath の共通パス以下のファイルを、outPath の共通パス以下にコピーします。 コピーの種類は、バイナリか、テキストで、テキストの場合は、エンコード変換も行うことが可能です。 inPath と outPath が同じ、または、outPath が未設定の場合は、入力と出力が同じですので、自分自身のエンコード変換処理を行うことになります。</p> <p>コピーされるファイルのファイル名は、入力ファイル名と同一です。保存されるフォルダが異なります。(同一にすることも可能です。)</p> <p>上流プロセスでは、Name 属性として、『File』を持ち、値は、Fileオブジェクトである、Process_FileSearch を使用するのが、便利です。それ以外のクラスを使用する場合でも、Name属性と、File オブジェクトを持つ LineModel を受け渡しできれば、使用可能です。</p> <p>引数文字列中に空白を含む場合は、ダブルコーテーション (") で括弧して下さい。 引数文字列の 『=』 の前後には、空白は挟めません。必ず、-key=value の様に繋げてください。</p>
org.opengion.fukuroku.process.Process.FileCopy Process_FileCopy -inPath=入力共通パス -inEncode=Windows-31J -outPath=出力共通パス -outEncode=UTF-8	
<pre> -inPath=入力共通パス : 上流で検索されたファイルパスの共通部分 [-inEncode=入力エンコード] : 入力ファイルのエンコードタイプ [-outPath=出力共通パス] : 出力するファイルパスの共通部分 [-outEncode=出力エンコード] : 出力ファイルのエンコードタイプ [-binary=[false/true]] : trueは、バイナリファイルのコピー (初期値:false) [-changeCrLf=[false/true]] : trueは、バイナリファイルのコピー時にCR+LFに変換します (初期値:false) [-keepTimeStamp=[false/true]] : trueは、コピー元のファイルのタイムスタンプで作成します (初期値:false) [-display=[false/true]] : trueは、コピー状況を表示します (初期値:false) </pre>	
5.5.4.2 process FileFtp	<p>Process_FileFtp は、上流から受け取った FileLineModel を処理する、ChainProcess インターフェースの実装クラスです。</p> <p>上流から受け取った FileLineModel の ファイルから、localPath のローカル共通パスを remotePath のFTP共通パスに、PFT伝送します。(command=PUT 処理のみ) ファイルそのものの階層構造は、維持されるため、ローカルからFTPサーバーへのフォルダコピーに近いイメージになります。</p> <p>Process_FileCopy との違いは、ファイルのエンコード変換は行いません。ただし、FTP伝送での改行コードの変換は、-mode=ASCII で指定できます。 もうひとつ、Process_FileCopy では、inPath と outPath でのCOPY処理でしたが、このクラスでは、localPath と、remotePath でそれぞれの共通パスを指定します。</p> <p>上流プロセスでは、Name 属性として、『File』を持ち、値は、Fileオブジェクトである、Process_FileSearch を使用するのが、便利です。それ以外のクラスを使用する場合でも、Name属性と、File オブジェクトを持つ LineModel を受け渡し</p>

バージョン 属性名称 属性クラス	内容
ATTCLASS 形式サンプル	
	できれば、使用可能です。 引数文字列中に空白を含む場合は、ダブルコーテーション(“”) で括って下さい。 引数文字列の 『=』 の前後には、空白は挟めません。必ず、-key=value の様に 繋げてください。
org.opengion.fukurou.process.Process.FileFtp Process.FileFtp -host=FTPサーバー -user=ユーザー -pass=パスワード -localPath=ローカル共通パス -remotePath=FTP共通パス [-mode=[ASCII/BINARY]] [-passive=[true/false]]	
	-host=FTPサーバー : FTPサーバー -user=ユーザー : ユーザー -pass=パスワード : パスワード -localPath=ローカル共通パス : 上流で検索されたファイルパスのローカル側共通部分 -remotePath=FTP共通パス : 上流で検索されたファイルパスのFTP側共通部分 [-mode=[ASCII/BINARY]] : 扱うファイルの種類を指定します (初期値:ASCII) [-passive=[true/false]] : パッシブモード (ローカルからサーバーへ接続を張る) を利用するかどうか (初期値:true) (false:アクティブモード (通常のFTPの初期値) で通信します。) [-mkdirs=[true/false]] : 受け側ファイル (GET時:LOCAL、PUT時:FTPサーバー) にディレクトリを作成するかどうか (初期値:true) (false:ディレクトリが無ければ、エラーにします。) [-encode=エンコード名] : 日本語ファイル名などのエンコード名を指定します (初期値:UTF-8) [-timeout=タイムアウト [秒]] : Dataタイムアウト (初期値:600 [秒]) [-display=[false/true]] : trueは、検索状況を表示します (初期値:false) [-debug=[false/true]] : デバッグ情報を標準出力に表示する (true) かしない (false) か (初期値:false [表示しない])
5.5.4.2 process FileSearch	Process.FileSearch は、指定のフォルダ以下のファイルを一覧する、FirstProcess インターフェースと、ChainProcess インターフェースの実装クラスです。 指定の条件に合致するファイルを検索し、LineModel のサブクラスである、 FileLineModel オブジェクトを作成して、下流に渡します。 FileLineModel オブジェクトには、ファイル属性 (Level, File, Length, Modify) が設定されます。 引数文字列中にスペースを含む場合は、ダブルコーテーション(“”) で括って下さい。 引数文字列の 『=』 の前後には、スペースは挟めません。必ず、-key=value の様に 繋げてください。
org.opengion.fukurou.process.Process.FileSearch Process.FileSearch -start=d:/ -suffix=.jsp	
	-start=開始フォルダ : 検索を開始するフォルダ [-prefix=接頭辞] : File.... View.... など、指定の接頭辞で始まるファイルを検索 [-unprefix=不接頭辞] : File.... View.... など、指定の接頭辞で始まらないファイルを検索 [-suffix=接尾辞] : .txt,.java,.jsp.... など、指定の接尾辞で終わるファイルを検索 [-unsuffix=不接尾辞] : .txt,.java,.jsp.... など、指定の接尾辞で終わらないファイルを検索 [-instr=部分文字列] : ファイル名と一致する部分文字列を指定 [-uninstr=不部分文字列] : ファイル名と一致しない部分文字列を指定 [-equals=一致] : ファイル名と一致する文字列 (大文字小文字は区別しない) を指定 [-notequals=不一致] : ファイル名と一致しない文字列 (大文字小文字は区別しない) を指定 [-match=正規表現] : ファイル名と一致する正規表現を指定 [-unmatch=正規表現] : ファイル名と一致しない正規表現を指定 [-modify=YYYYMMDD] : 指定日付け以降に変更されたファイルを検索 YYYYMMDD : YYYYMMDD 形式での指定日の 00:00:00 を基準時刻 TODAY : 実行日の 00:00:00 を基準時刻 YESTERDAY : 実行日前日の 00:00:00 を基準時刻 LAST_WEEK : 実行日の先週 (7日前) 00:00:00 を基準時刻 MONTH : 実行月の 1日 00:00:00 を基準時刻 LAST_MONTH : 実行前月の 同日 00:00:00 を基準時刻 LAST_YEAR : 実行前年の 同月同日 00:00:00 を基準時刻 [-useDIR=[false/true]] : 判定をファイルだけでなく、ディレクトリでも行うかどうかを指定 (初期値:false) [-larger=サイズ (KByte)] : ファイルの大きさが指定のKバイト数より大きいファイルを検索 [-smaller=サイズ (KByte)] : ファイルの大きさが指定のKバイト数より小さいファイルを検索 [-maxLevel=最大階層数] : ディレクトリの階層を下がる最大数 (初期値:256) [-useLineCnt=行数計算] : ファイルの行数をカウントするかどうかを指定 (初期値:false) [-inPath=入力共通パス] : BIK0作成用のファイルパスから削除する部分 (文字数のみ) [-outPath=出力追加パス] : BIK0作成用のファイルパスに追加する部分 [-display=[false/true]] : trueは、検索状況を表示します (初期値:false)
5.5.4.2 process Grep	Process.Grep は、上流から受け取った FileLineModel から、文字列を見つけ出す ChainProcess インターフェースの実装クラスです。 正規表現の keyword を上流から受け取った FileLineModel から検索します。 見つかった対象ファイルから、指定の文字列を置換する場合は、-change か -changeFile で、keyword を置換する文字列を指定して下さい。 置換する文字列には、%t と %n の特殊文字が使用できます。 処理対象は、通常は、1行づつ読み取りながら処理を行います。存在チェックの場合は、 見つかった時点で処理を中止します。これは、該当箇所をピックアップするのではなく、 存在しているかどうかを判断して、あれば、下流に流すというのが目的だからです。 keyword を、改行を含む正規表現で、検索・置換する場合は、-useBulkRead 属性を true に設定してください。これは、入力ファイルを一括して読み込みます。 -ignoreCase は、正規表現の検索時にキーの大文字小文字を無視するように指定します。 -notEquals は、結果 (見つければtrue) を反転 (見つからなければtrue) します。 これは、行単位ではなく、ファイル単位に判定しますので、change 指定した場合 でも、対象行は、見つかった行です。ただし、下流に対して、見つからない 場合だけ処理を継続させます。 -inEncode は、入力ファイルのエンコード指定になります。 -outEncode は、出力ファイルのエンコードや、changeFileで指定の置換文字列ファイルの エンコード指定になります。(changeFile は、必ず 出力ファイルと同じエンコードです。) これらのエンコードが無指定の場合は、System.getProperty("file.encoding") で 求まる値を使用します。 -changeFile を使用することで、複数行の文字列に置換することが可能です。 -outfile では、処理を行ったファイル名一覧をセーブします。 上流 (プロセスチェーンのデータは上流から渡されます。) からのLineModel の ファイルオブジェクトより、指定の文字列が含まれているか検索します。 上流プロセスでは、Name 属性として、『File』を持ち、値は、Fileオブジェクト である、Process.FileSearch を使用するのが、便利です。それ以外のクラスを 使用する場合でも、Name属性と、File オブジェクトを持つ LineModel を受け渡し できれば、使用可能です。 引数文字列中に空白を含む場合は、ダブルコーテーション(“”) で括って下さい。 引数文字列の 『=』 の前後には、空白は挟めません。必ず、-key=value の様に 繋げてください。
org.opengion.fukurou.process.Process.Grep	

2012/07/13

バージョン 属性名称 属性クラス	内容
ATTCLASS 形式サンプル	
5.5.4.2 process LDAPReader	<p>Process_LDAPReaderは、LDAPから読み取った内容を、LineModel に設定後、下流に渡す、FirstProcess インターフェースの実装クラスです。</p> <p>LDAPから読み取った内容より、LineModelを作成し、下流(プロセスチェーンは、チェーンしているため、データは上流から下流へと渡されます。)に渡します。</p> <p>引数文字列中にスペースを含む場合は、ダブルコーテーション(“”) で括って下さい。 引数文字列の 『=』 の前後には、スペースは挟めません。必ず、-key=value の様に繋げてください。</p>
org.opengion.fukurou.process.Process_LDAPReader Process_LDAPReader -attrs=uid,cn,officeName,ou,mail,belongOUID -orderBy=uid -filter=((&(objectClass=person)((belongOUID=61200) (belongOUID=61100))))	
<pre> [-initctx=コンテキストファクトリ] : 初期コンテキストファクトリ (初期値:com.sun.jndi.Ldap.LdapCtxFactory) [-providerURL=サービスプロバイダリ] : サービスプロバイダリ (初期値:ldap://ldap.opengion.org:389) [-entrydn=取得元の名前] : 属性の取得元のオブジェクトの名前 (初期値:cn=inquiry-sys,o=opengion,c=JP) [-password=取得元のパスワード] : 属性の取得元のパスワード (初期値:*****) [-searchbase=コンテキストベース名] : 検索するコンテキストのベース名 (初期値:soouid=employeeuser,o=opengion,c=JP) [-searchScope=検索範囲] : 検索範囲。『OBJECT』『ONELEVEL』『SUBTREE』のどれか (初期値:SUBTREE) [-timeLimit=検索制限時間] : 結果が返されるまでのミリ秒数。0 の場合無制限 (初期値:0) [-attrs=属性の識別子] : エントリと一緒に返される属性の識別子。null の場合すべての属性 [-columns=属性のカラム名] : 属性の識別子に対する別名。識別子と同じ場合は『.』のみで区切る。 [-maxRowCount=最大検索数] : 最大検索数 (初期値:0[無制限]) [-match_XXX=正規表現] : 指定のカラムと正規表現で一致時のみ処理 (-match_LANG=ABC=[a-zA-Z]* など。) [-filter=検索条件] : 検索する LDAP に指定する条件 [-display=false true] : 結果を標準出力に表示する(true)かしない(false)か (初期値:false[表示しない]) </pre>	
5.5.4.2 process Logger	<p>Process_Logger は、画面出力、ファイルログ、エラーメールを管理する、ロギング関係の LoggerProcess インターフェースの実装クラスです。</p> <p>MainProcess で使用されるログと、各種 Process で使用されるディスプレイを管理します。また、エラー発生時の、メール送信機能も、ここで用意します。</p> <p>引数文字列中にスペースを含む場合は、ダブルコーテーション(“”) で括って下さい。 引数文字列の 『=』 の前後には、スペースは挟めません。必ず、-key=value の様に繋げてください。</p>
org.opengion.fukurou.process.Process_Logger Process_Logger -logFile=ABC.txt -dispFile=System.out	
<pre> [-logFile=ログ出力先指定] : -logFile=[ファイル名/System.out/System.err] (初期値:null) [-dispFile=画面出力先指定] : -dispFile=[ファイル名/System.out/System.err] (初期値:null) [-host=メールサーバ] : -host=メールサーバ [-from=送信From] : -from=送信元アドレス [-to=受信To] : -to=送信先アドレスをカンマ区切りで並べる [-charset=キャラクタセット] : -charset=メール送信時のキャラクタセット [ISO-2022-JP / Windows-31J] [-subject=タイトル] : -subject=タイトル [-message=本文雛形] : -message=本文雛形文章 [-msgFile=本文雛形ファイル] : -msgFile=本文を格納しているファイルのアドレス [-@XXXX=YYYY] : メッセージ本文の [XXXX] 文字列を、YYYY 文字列に変換します。 </pre>	
5.5.4.2 process StringUtil	<p>Process_StringUtil は、上流から受け取ったデータをStringUtilクラスの特定のメソッドでデータ変換する、GainProcess インターフェースの実装クラスです。</p> <p>上流(プロセスチェーンのデータは上流から下流へと渡されます。)からLineModel を元に、指定のカラムの文字を、変換します。</p> <p>現時点で利用できるStringUtil のメソッドは、下記の通りです。</p> <pre> urlEncode : UTF-8 で、URLエンコードを行う。 rTrim : 文字列の後ろのスペースを削除 htmlFilter : HTML上のエスケープ文字を変換 code39 : CODE39 の 文字列を作成(チェックデジット付き) getUnicodeEscape : HTML のエスケープ記号 (&#xZZZZ;)に変換 getReplaceEscape : HTML のエスケープ記号 (&#xZZZZ;)を戻す spanCut : 引数からspanタグを取り除く </pre> <p>HybsCryptography のメソッドも呼び出せます。</p> <pre> getMD5 : MessageDigestにより、MD5 でハッシュした文字に変換 encrypt : Hybs独自の暗号化を行います (Hybs内部設定の秘密鍵) decrypt : Hybs独自の復号化を行います (Hybs内部設定の秘密鍵) </pre> <p>引数文字列中にスペースを含む場合は、ダブルコーテーション(“”) で括って下さい。 引数文字列の 『=』 の前後には、スペースは挟めません。必ず、-key=value の様に繋げてください。</p>
org.opengion.fukurou.process.Process_StringUtil Process_StringUtil -action=getMD5 encrypt decrypt code39 getUnicodeEscape getReplaceEscape ... -keys=AA,BB,CC	
<pre> -action=ESC REV : StringUtilクラス特定のメソッド名を指定します(必須)。 urlEncode rTrim htmlFilter getMD5 code39 getUnicodeEscape getReplaceEscape spanCut -keys=AA,BB,CC : 変換するカラムをCSV形式で複数指定できます(必須)。 [-display=false true] : 結果を標準出力に表示する(true)かしない(false)か (初期値:false[表示しない]) </pre>	
5.5.4.2 process TableDiff	<p>Process_TableDiffは、ファイルから読み取った内容を、LineModel に設定後、下流に渡す、FirstProcess インターフェースの実装クラスです。</p> <p>DBTableModel 形式のファイルを読み取って、各行を LineModel にセットして、下流(プロセスチェーンのデータは上流から下流に渡されます。)に渡します。</p> <p>引数文字列中にスペースを含む場合は、ダブルコーテーション(“”) で括って下さい。 引数文字列の 『=』 の前後には、スペースは挟めません。必ず、-key=value の様に繋げてください。</p>
org.opengion.fukurou.process.Process_TableDiff Process_TableDiff -infile1=INFILE1 -infile2=INFILE2 -action=DIFF1 -encode=UTF-8 -columns=AA,BB,CC	
<pre> -infile1=入力ファイル名1 : 入力ファイル名1 -infile2=入力ファイル名2 : 入力ファイル名2 -action=比較結果の方法 : ONLY, DIFF, INTERSEC [-sep1=セパレータ文字] : 区切り文字1 (初期値:タブ) [-sep2=セパレータ文字] : 区切り文字2 (初期値:タブ) [-encode1=文字エンコード] : 入力ファイルのエンコードタイプ1 [-encode2=文字エンコード] : 入力ファイルのエンコードタイプ2 [-columns=読み取りカラム名] : 入力カラム名 (カンマ区切り) [-keyC1ms=比較するカラム名] : 比較する列の基準カラム名 (カンマ区切り) [-diffC1ms=比較するカラム名] : 比較するカラム名 (カンマ区切り) [-display=false true] : 結果を標準出力に表示する(true)かしない(false)か (初期値:false[表示しない]) [-debug=false true] : デバッグ情報を標準出力に表示する(true)かしない(false)か (初期値:false[表示しない]) </pre>	

バージョン 属性名称 属性クラス	内容
ATTCLASS 形式サンプル	
5.5.4.2 process TableFilter	<p>Process_TableFilter は、上流から受け取ったデータをフィルタする、ChainProcess インターフェースの実装クラスです。</p> <p>上流(プロセスチェーンのデータは上流から下流へと渡されます。)から受け取ったLineModel を元に、項目のフィルタリングを行います。条件が成立した場合は、下流に流します。複数の条件を指定できますが、すべて AND で判定されます。(設定条件すべてを満たす場合のみ、下流にデータを流します。)</p> <p>引数文字列中にスペースを含む場合は、ダブルコーテーション(“”) で括弧して下さい。引数文字列の 『=』 の前後には、スペースは挟めません。必ず、-key=value の様に繋げてください。</p>
org.opengion.fukurou.process.Process_TableFilter Process_TableFilter	
<pre> [-prefix_XXXX=接頭辞] : 項目名(XXXX) が、指定の接頭辞で始まる場合、条件成立。 [-suffix_XXXX=接尾辞] : 項目名(XXXX) が、指定の接尾辞で終わる場合、条件成立。 [-instr_XXXX=部分文字列] : 項目名(XXXX) が、指定の部分文字列と一致する場合、条件成立。 [-equals_XXXX=一致] : 項目名(XXXX) が、文字列と一致する場合、条件成立。 + CR + "文字列は、大文字小文字は区別しません。(equalsIgnoreCase)"; [-match_XXXX=正規表現] : 項目名(XXXX) が、正規表現と一致する場合、条件成立。 [-unmatch_XXXX=正規表現] : 項目名(XXXX) が、正規表現と一致しない場合、条件成立。 [-const_XXXX=固定値] : -const_FGJ=1 項目名(XXXX) に、固定値を設定します。 [-replace_XXXX=固定値] : -replace_BIK0="YYYY⇒ZZZZ" (元先指定は、⇒で区切ります。) 項目名(XXXX) の文字列から、YYYY という文字を ZZZZ に置換します。 [-display=false true] : 結果を標準出力に表示する(true)かしない(false)か(初期値:false[表示しない]) </pre>	
5.5.4.2 process TableReader	<p>Process_TableReaderは、ファイルから読み取った内容を、LineModel に設定後、下流に渡す、FirstProcess インターフェースの実装クラスです。</p> <p>DBTableModel 形式のファイルを読み取って、各行を LineModel にセットして、下流(プロセスチェーンのデータは上流から下流に渡されます。)に渡します。</p> <p>columns 属性は、#NAME で列カラムを外部から指定する場合に使用します。この属性とuseNumber属性は独立していますが、一般には、#NAME を指定する場合は、useNumber="true"として、行番号欄は使用しますし、外部から指定する場合は、useNumber="false"にして先頭から読み取ります。(自動セットではないので、必要に応じて設定してください) useNumber の初期値は、“true” です。</p> <p>引数文字列中にスペースを含む場合は、ダブルコーテーション(“”) で括弧して下さい。引数文字列の 『=』 の前後には、スペースは挟めません。必ず、-key=value の様に繋げてください。</p>
org.opengion.fukurou.process.Process_TableReader Process_TableReader -infile=INFILE -sep=, -encode=UTF-8 -columns=AA,BB,CC	
<pre> -infile=入力ファイル名 : 入力ファイル名 [-existCheck=存在確認] : ファイルが存在しない場合エラーにする(初期値:true) [-sep=セパレータ文字] : 区切り文字(初期値:タブ) [-encode=文字エンコード] : 入力ファイルのエンコードタイプ [-columns=読み取りカラム名] : 入力カラム名(カンマ区切り) [-useNumber=true false] : 行番号を使用する(true)か使用しない(false)か。 [-display=false true] : 結果を標準出力に表示する(true)かしない(false)か(初期値:false[表示しない]) </pre>	
5.5.4.2 process TableWriter	<p>Process_TableWriter は、上流から受け取ったデータをファイルに書き込むCainProcess インターフェースの実装クラスです。</p> <p>上流(プロセスチェーンのデータは上流から下流へと渡されます。)から受け取ったLineModel を元に、DBTableModel 形式ファイルを出力します。</p> <p>引数文字列中にスペースを含む場合は、ダブルコーテーション(“”) で括弧して下さい。引数文字列の 『=』 の前後には、スペースは挟めません。必ず、-key=value の様に繋げてください。</p>
org.opengion.fukurou.process.Process_TableWriter Process_TableWriter -outfile=OUTFILE -sep=, -encode=UTF-8 -append=true	
<pre> -outfile=出力ファイル名 : 出力ファイル名 [-sep=セパレータ文字] : 区切り文字(初期値:タブ) [-encode=文字エンコード] : 出力ファイルのエンコードタイプ [-append=false true] : 出力ファイルを、追記する(true)か新規作成する(false)か。 [-useHeader=true false] : ヘッダー情報(#NAME行)を出力する(true)か出力しない(false)か。 [-useNumber=true false] : 行番号を出力する(true)か出力しない(false)か。 [-useWquot=false true] : 出力データをダブルコーテーションで括弧する(true)かそのまま(false)か。 [-omitCTRL=false true] : コントロール文字を削除する(true)かそのまま(false)か。 [-const_XXXX=固定値] : -const_FGJ=1 LineModel のキー(const_ に続く文字列)の値に、固定値を設定します。 キーが異なれば、複数のカラム名を指定できます。 [-delKey_XXXX=文字列] : -delKey_File=G:¥¥webapps¥¥¥¥¥¥src LineModel のキー(delKey_ に続く文字列)の値から、指定の文字列を削除します。 キーが異なれば、複数のカラム名を指定できます。 [-insKey_XXXX=文字列] : -insKey_File=src¥¥¥¥¥¥ LineModel のキー(insKey_ に続く文字列)の値に、指定の文字列を追加します。 キーが異なれば、複数のカラム名を指定できます。 [-display=false true] : 結果を標準出力に表示する(true)かしない(false)か(初期値:false[表示しない]) </pre>	
5.5.4.2 process UnicodeEscape	<p>【廃止】 Process_UnicodeEscape は、上流から受け取ったデータをエスケープ変換するChainProcess インターフェースの実装クラスです。</p> <p>※ 5.0.0.2 (2009/09/15) このクラスは、廃止になりました。 類似機能に、Process_StringUtil を利用してください。 -action 引数に、getUnicodeEscape か、getReplaceEscape を指定することで、ESCAPE(変換)/REVERSE(戻し) の指定と同一の処理を行います。</p> <p>上流(プロセスチェーンのデータは上流から下流へと渡されます。)から受け取ったLineModel を元に、指定のカラムのUTF-8 文字コードと、HTML のエスケープ記号(&#xZZZZ;)文字との相互変換を行います。 入力文字を、キャラクタ(char)型に分解し、(&#xZZZZ;)に変換していきます。 よって、通常に1文字(Shift-JISで2Byte、UTF-8で3Byte)が、8Byteになります。 この変換された文字列を、HTML上でそのまま取り出すと、元のUnicode文字に戻る為、通常のShift-JISでは、扱えない文字(中国語など)でも表示可能になります。</p> <p>引数文字列中にスペースを含む場合は、ダブルコーテーション(“”) で括弧して下さい。</p>

バージョン 属性名称 属性クラス	内容
ATTCLASS 形式サンプル	
5.5.4.2 daemon Transfer_CB01	<p>このデーモンは、伝送定義マスタの読取方法が、旧伝送DB読取(CB01)の定義を対象として実行されます。 読取対象は、旧伝送DB(CB01)で、データコード、送り先、テキスト種別、状況='1'を条件に読み込まれます。 伝送定義マスタ上では、読取対象にて、以下の形式で定義する必要があります。 (データコード) (送り先) (テキスト種別) 例: '3 D9 B119' 処理実行後は、読み取ったヘッダーデータの状況を'2'に更新します。 但し、読取パラメーターに"NOUPDATE"を指定した場合、処理後の更新は行われません。 また、エラーが発生した場合はヘッダーデータの状況を'9'に更新します。</p> <p>トランザクションは、読取対象の単位になります。 同じ読取対象で、異なる実行方法、実行対象を定義した場合、同じデータに対して複数回処理が行われます。 しかし、この場合においても、トランザクションは読取対象の単位で生成されるため、複数回の処理の内、 1回でもエラーが発生した場合は、同じ読取対象でそれまでに処理した分についてもrollbackされます。</p> <p>また、この伝送デーモン(読取方法)は、旧伝送DB(CB01)に対するクエリ回数を減らすため、旧伝送DB(CB01)と 伝送定義マスタ(GE62)をJOINして一括でデータを取得しています。 このため、他の伝送デーモン(読取方法)とは読取部分の実装方法が異なっています。 具体的には、org.opengion.fukurou.transfer.TransferReadインターフェースを利用せずに、 このデーモン自体に読取及びステータス更新の処理を実装しています。</p> <p>※処理中に何らかのエラーが1度でも発生した場合、このデーモンは停止します。</p> <p>このクラスは、HybsTimerTask を継承した タイマータスククラスです。 startDaemon() がタイマータスクによって、呼び出されます。</p> <p>org.opengion.plugin.daemon.Daemon_Transfer_CB01</p>
5.5.4.2 daemon URLConnect	<p>【URLアクセス】 指定したパラメータでURLに接続します。 このクラスは、HybsTimerTask を継承した タイマータスククラスです。 startDaemon() がタイマータスクによって、呼び出されます。</p> <p>接続のためのパラメータは以下です url : 接続先URL(必須) proxyHost : プロキシのホスト名 proxyPort : プロキシのポート番号 useSystemUser : デフォルトのユーザ/パスワードを利用するか(初期値:true) trueの場合はSYSTEM:*****を利用します。</p> <p>authUserPass : ユーザとパスワードをUSER:PASSWORDの形で記述 keys : リクエストパラメータのキー(カンマ区切り) vals : リクエストパラメータの値(カンマ区切り) method : POSTかGETを指定(初期値:GET) debug : 接続したページを受信して、ログに書き出します(初期値:false)</p> <p>接続エラー時のログはファイル(SYS_LOG_URL)に出力されます。</p> <p>org.opengion.plugin.daemon.Daemon_URLConnect</p>