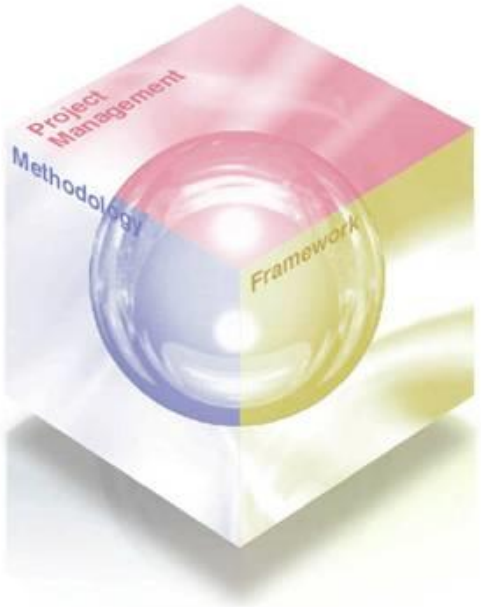
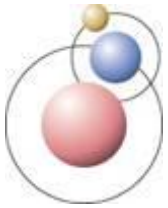
TERASOLUNA® Server Framework for Java（Web版）
アーキテクチャ説明書
第2.0.6.2版



NTT DATA

株式会社 NTTデータ

TERASOLUNA®



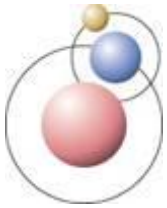
本ドキュメントを使用するにあたり、以下の規約に同意していただく必要があります。同意いただけない場合は、本ドキュメント及びその複製物の全てを直ちに消去又は破棄してください。

1. 本ドキュメントの著作権及びその他一切の権利は、NTTデータあるいはNTTデータに権利を許諾する第三者に帰属します。
2. 本ドキュメントの一部または全部を、自らが使用する目的において、複製、翻訳、翻案することができます。ただし本ページの規約全文、およびNTTデータの著作権表示を削除することはできません。
3. 本ドキュメントの一部または全部を、自らが使用する目的において改変したり、本ドキュメントを用いた二次的著作物を作成することができます。ただし、「参考文献:TERASOLUNA Server Framework for Java (Web版)アーキテクチャ説明書」あるいは同等の表現を、作成したドキュメント及びその複製物に記載するものとします。
4. 前2項によって作成したドキュメント及びその複製物を、無償の場合に限り、第三者へ提供することができます。
5. NTTデータの書面による承諾を得ることなく、本規約に定められる条件を超えて、本ドキュメント及びその複製物を使用したり、本規約上の権利の全部又は一部を第三者に譲渡したりすることはできません。
6. NTTデータは、本ドキュメントの内容の正確性、使用目的への適合性の保証、使用結果についての的確性や信頼性の保証、及び瑕疵担保義務も含め、直接、間接に被ったいかなる損害に対しても一切の責任を負いません。
7. NTTデータは、本ドキュメントが第三者の著作権、その他如何なる権利も侵害しないことを保証しません。また、著作権、その他の権利侵害を直接又は間接の原因としてなされる如何なる請求(第三者との間の紛争を理由になされる請求を含む。)に関しても、NTTデータは一切の責任を負いません。

本ドキュメントで使用されている各社の会社名及びサービス名、商品名に関する登録商標および商標は、以下の通りです。

- TERASOLUNAは、株式会社NTTデータの登録商標です。
- その他の会社名、製品名は、各社の登録商標または商標です。

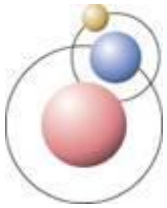
Copyright (c) 2007-2016 NTT DATA Corporation.



変更履歴

バージョン	日付	改訂箇所	改訂内容
2.0.2.0	2009/03/31	商標、説明	富士通Interstageに対応
〃	〃	依存するオープンソースライブラリ一覧	commons-jxpathのバージョンを1.2から1.3へ変更
〃	〃	メッセージ管理機能	メッセージ管理機能に関する記述を修正
〃	〃	例外ハンドリング機能	SystemExceptionHandlerの記述例にlogLevelを追加 DefaultExceptionHandlerに関する記述を追加
〃	〃	画面表示(カスタムタグ)機能	date, jdateタグにformat属性を追加。value属性の説明を修正
〃	〃	データベースアクセス機能	QueryRowHandleDAOに関する記述を追加
〃	2009/09/30	動作確認環境	新たに検証した動作確認環境のバージョンを追記
2.0.2.1	2010/02/26	全体	Bean定義例の書式を統一
〃	〃	全体	フッター情報を更新
2.0.3.0	2010/04/02	全体	バージョンを更新
2.0.3.1	2011/06/29	全体	バージョンを更新
〃	〃	動作確認環境	新たに検証した動作確認環境のバージョンを追記
2.0.4.0	2012/04/01	全体	バージョンを更新
〃	〃	任意選択モジュール	モジュールを追加
〃	〃	依存するオープンソースライブラリ一覧	ライブラリの名称・バージョン等を変更 モジュールの追加
2.0.5.0	2012/12/26	全体	バージョンを更新
〃	〃	全体	フッター情報を更新
〃	〃	概要	Springのバージョンを2.5から3.1.3へ変更
〃	〃	動作確認環境	新たに検証した動作確認環境のバージョンを追記
〃	〃	依存するオープンソースライブラリ一覧	commons-langのバージョンを2.3から2.5へ変更 Springのバージョンを2.5.6から3.1.3へ変更
2.0.5.1	2014/03/31	全体	バージョンを更新
〃	〃	全体	フッター情報を更新
〃	〃	動作確認環境	新たに検証した動作確認環境のバージョンを追記
〃	〃	必要モジュール、依存するオープンソースライブラリ一覧	ライブラリのバージョンを変更 モジュールの追加
〃	〃	コードリスト機能	国際化対応についての説明を追記
2.0.5.2	2014/05/23	全体	バージョンを更新
〃	〃	依存するオープンソースライブラリ一覧	Strutsのバージョンを1.2.9から1.2.9-sp1へ変更

Copyright (c) 2007-2016 NTT DATA Corporation.



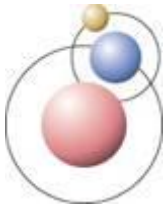
変更履歴

バージョン	日付	改訂箇所	改訂内容
2.0.5.3	2014/08/20	全体	バージョンを更新
〃	〃	依存するオープンソースライブラリ一覧	Strutsのバージョンを1.2.9-sp1から1.2.9-sp2へ変更
〃	〃	ライブラリの依存関係	新規追加
2.0.6.1	2015/07/02	全体	バージョンを更新
〃	〃	全体	フッター情報を更新
〃	〃	動作確認環境	新たに検証した動作確認環境のバージョンを追記
〃	〃	依存するオープンソースライブラリ一覧	ライブラリのバージョン等を変更
2.0.6.2	2016/08/31	全体	バージョンを更新
〃	〃	全体	フッター情報を更新
〃	〃	動作確認環境	新たに検証した動作確認環境のバージョンを追記
〃	〃	必要モジュール	terasoluna-ContentTypeLengthLimitFilterの箇所を削除
〃	〃	依存するオープンソースライブラリ一覧	ライブラリのバージョン等を変更

Copyright (c) 2007-2016 NTT DATA Corporation.

目次

Copyright (c) 2007-2016 NTT DATA Corporation.



はじめに

概要

- 本資料は、Webシステムを構築するためのサーバ側フレームワーク「TERASOLUNA Server Framework for Java（Web版）ver2.0.6.2」を解説した資料である。
- 本フレームワークは、Spring3.2.13、Struts1.2をベースに拡張を行ったフレームワークである。

Copyright (c) 2007-2016 NTT DATA Corporation.

はじめに

動作確認環境

対応JDK

- Oracle (Sun) JDK 5 / 6 / 7 / 8 *
- OpenJDK 6 / 7 */ 8 *
- HP JDK 5 / 6(HP-UX利用時)
- jRockit JDK 5 / 6(WebLogic利用時)
- IBM JDK 5 / 6 / 7(WebSphere利用時)
- 日立 JDK 5 / 6(Cosminexus利用時)
- 富士通 JDK 5(Interstage利用時)

対応WebAPサーバ

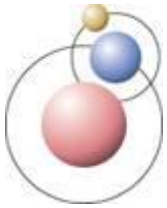
- Tomcat 5.5 / 6 / 7 */ 8 *
- WebLogic 9.2 / 10.0 / 10.3.x / 12c / 12.2.1 *
- WebSphere 6.1 / 7.0 / 8.5
- Cosminexus V7 / V8 / V9
- Interstage 9
- WebOTX 8
- JBOSS EAP 6 *

対応データベース

- Oracle 10g / 11g / 12c *
- PostgreSQL 8 / 9 *
- DB2 10

**付きが本バージョンで動作確認を行った環境*

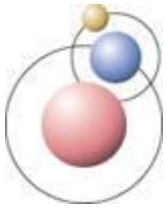
Copyright (c) 2007-2016 NTT DATA Corporation.



はじめに

- 必要モジュール
  - terasoluna-thin
    - Web版特有の機能を実装したモジュール。
  - terasoluna-commons
    - ユーティリティ機能を提供するモジュール。

Copyright (c) 2007-2016 NTT DATA Corporation.



はじめに

任意選択モジュール

- terasoluna-dao
  - DAOインタフェースを提供するモジュール。
- terasoluna-ibatis
  - ORマッピングツールiBatisを利用した、データベースアクセス機能を提供するモジュール。
- terasoluna-web-cr
  - blogic-io等の設定ファイルを削減する機能を提供するモジュール。
- terasoluna-taglibex
  - カスタムタグを強化するためのモジュール。
- terasoluna-commonscreenflow
  - 共通画面フローを提供するモジュール。
- teralib-exception
  - 共通例外機能を提供するモジュール。
- teralib-filter
  - フィルターを強化するためのモジュール。
- teralib-log
  - デバッグログ出力機能を提供するためのモジュール。

Copyright (c) 2007-2016 NTT DATA Corporation.

はじめに

依存するオープンソースライブラリー覧(1/2)

オープンソースライブラリ名	バージョン	TERASOLUNAモジュール(terasoluna-*)				
		commons	dao	ibatis	thin	ContentType LengthLimitFilter
antlr.jar	2.7.6				○	
aspectjweaver.jar	1.7.4				○	
commons-beanutils.jar	1.8.3	○			○	
commons-digester.jar	2.0				○	
commons-fileupload.jar	1.3.2				○	
commons-jxpath.jar	1.3	○			○	
commons-lang	2.5	○			○	
commons-logging.jar	1.1.3	○			○	○
commons-validator.jar	1.3.1				○	
easymock.jar	2.3		○(テスト用)			
mybatis.jar※1	2.3.5			○		
jakarta-oro.jar	2.0.8				○	
jsp-api.jar	2.0				○	
junit.jar	3.8.2	○(テスト用)	○(テスト用)	○(テスト用)	○(テスト用)	○(テスト用)
junit-addons.jar	1.4	○(テスト用)	○(テスト用)	○(テスト用)	○(テスト用)	○(テスト用)
log4j.jar	1.2.16				○	
mockrunner.jar	0.3.7	○(テスト用)			○(テスト用)	
servlet-api.jar	2.5	○	○		○	○
struts.jar※2	1.2.9-sp3				○	
aopalliance.jar	1.0		○	○	○	
spring-aop.jar	3.2.13.RELEASE	○	○			
spring-beans.jar	3.2.13.RELEASE	○	○		○	
spring-context.jar	3.2.13.RELEASE				○	
spring-core.jar	3.2.13.RELEASE					
spring-expression.jar	3.2.13.RELEASE				○	
spring-jdbc.jar	3.2.13.RELEASE	○		○	○	
spring-orm.jar	3.2.13.RELEASE			○		
spring-struts.jar	3.2.13.RELEASE				○	
spring-test.jar	3.2.13.RELEASE				○(テスト用)	
spring-tx.jar	3.2.13.RELEASE		○	○	○	
spring-web.jar	3.2.13.RELEASE				○	

※1 ver2.3.5へのバージョンアップによりibatisからmybatisに名称が変更
※2 struts.jar ver 1.2.9にパッチをあてて提供

Copyright (c) 2007-2016 NTT DATA Corporation.

はじめに

依存するオープンソースライブラリー覧(2/2)

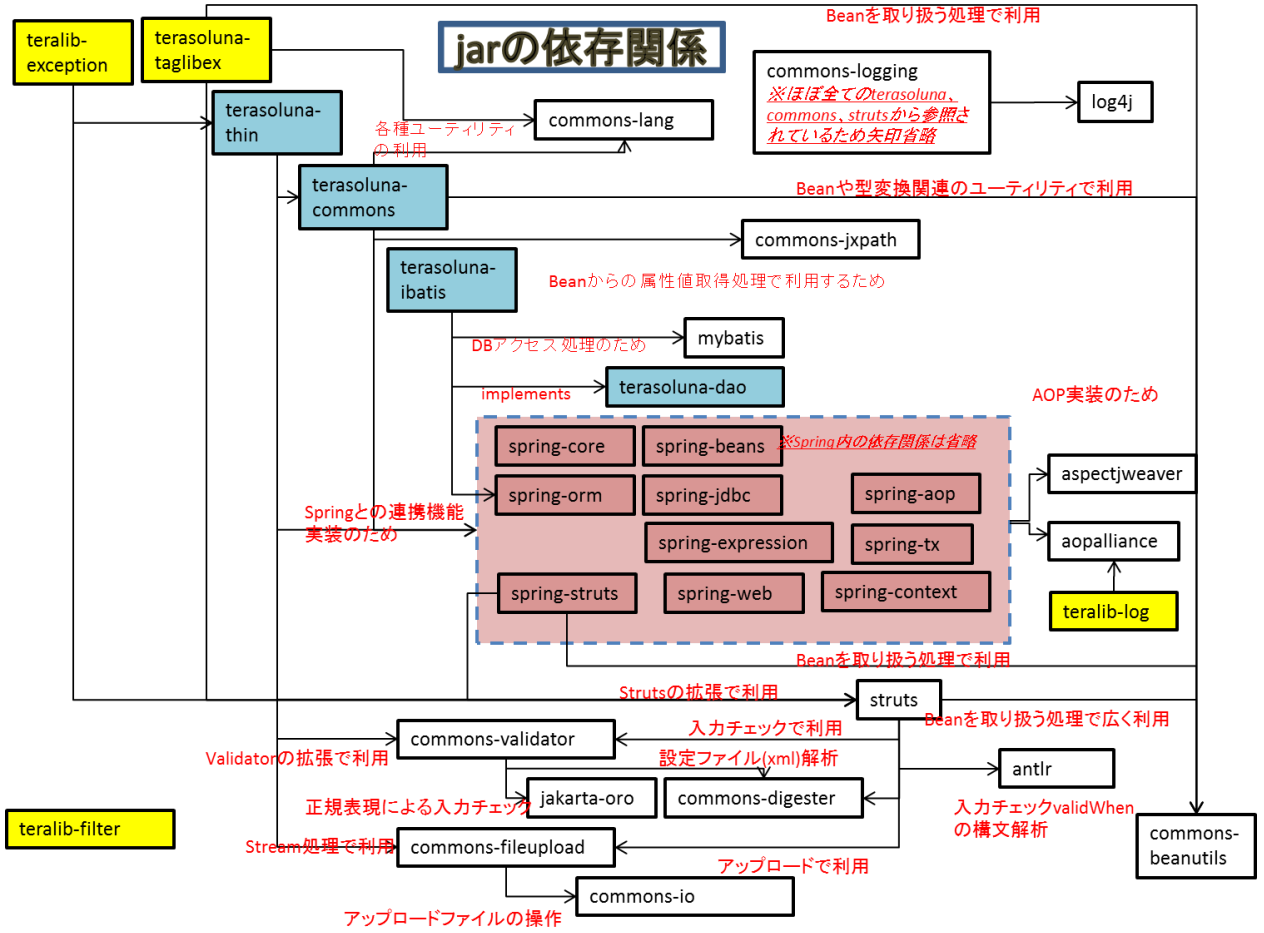
オープンソースライブラリ名	バージョン	TERASOLUNAモジュール(terasoluna-*/teralib-*)					
		web-cr	taglibex	common screenflow	exception	filter	log
antlr.jar	2.7.6	○					
aspectjweaver.jar	1.7.4						
commons-beanutils.jar	1.8.3	○			○		
commons-digester.jar	1.8	○					
commons-fileupload.jar	1.3.2	○					
commons-jxpath.jar	1.3	○					
commons-lang	2.5	○		○			
commons-logging.jar	1.1.3	○	○		○	○	○
commons-validator.jar	1.3.1	○					
easymock.jar	2.5.2	○(テスト用)					
mybatis.jar※1	2.3.5						
jakarta-oro.jar	2.0.8	○					
jsp-api.jar	2.0	○		○	○		
junit.jar	4.8.2	○(テスト用)	○(テスト用)	○(テスト用)	○(テスト用)	○(テスト用)	○(テスト用)
junit-addons.jar	1.4	○(テスト用)		○(テスト用)	○(テスト用)	○(テスト用)	
log4j.jar	1.2.16						
mockrunner.jar	0.3.7	○(テスト用)*2	○(テスト用)				
mockito.jar	1.8.2					○(テスト用)	
objenesis.jar	1.2				○		
servlet-api.jar	2.5	○		○	○	○	
struts.jar※3	1.2.9-sp3	○		○	○		
aopalliance.jar	1.0						○
spring-aop.jar	3.2.13.RELEASE	○					
spring-beans.jar	3.2.13.RELEASE	○				○	
spring-context.jar	3.2.13.RELEASE	○					
spring-core.jar	3.2.13.RELEASE	○					
spring-expression.jar	3.2.13.RELEASE	○					
spring-jdbc.jar	3.2.13.RELEASE						
spring-orm.jar	3.2.13.RELEASE					○	
spring-struts.jar	3.2.13.RELEASE						
spring-test.jar	3.2.13.RELEASE	○(テスト用)					
spring-tx.jar	3.2.13.RELEASE						
spring-web.jar	3.2.13.RELEASE	○					

※1 ver2.3.5へのバージョンアップによりibatisからmybatisに名称が変更
※2 web-crのみver0.4.2を利用
※3 struts.jar ver 1.2.9にパッチをあてて提供

Copyright (c) 2007-2016 NTT DATA Corporation.

はじめに

ライブラリの依存関係(1/2)

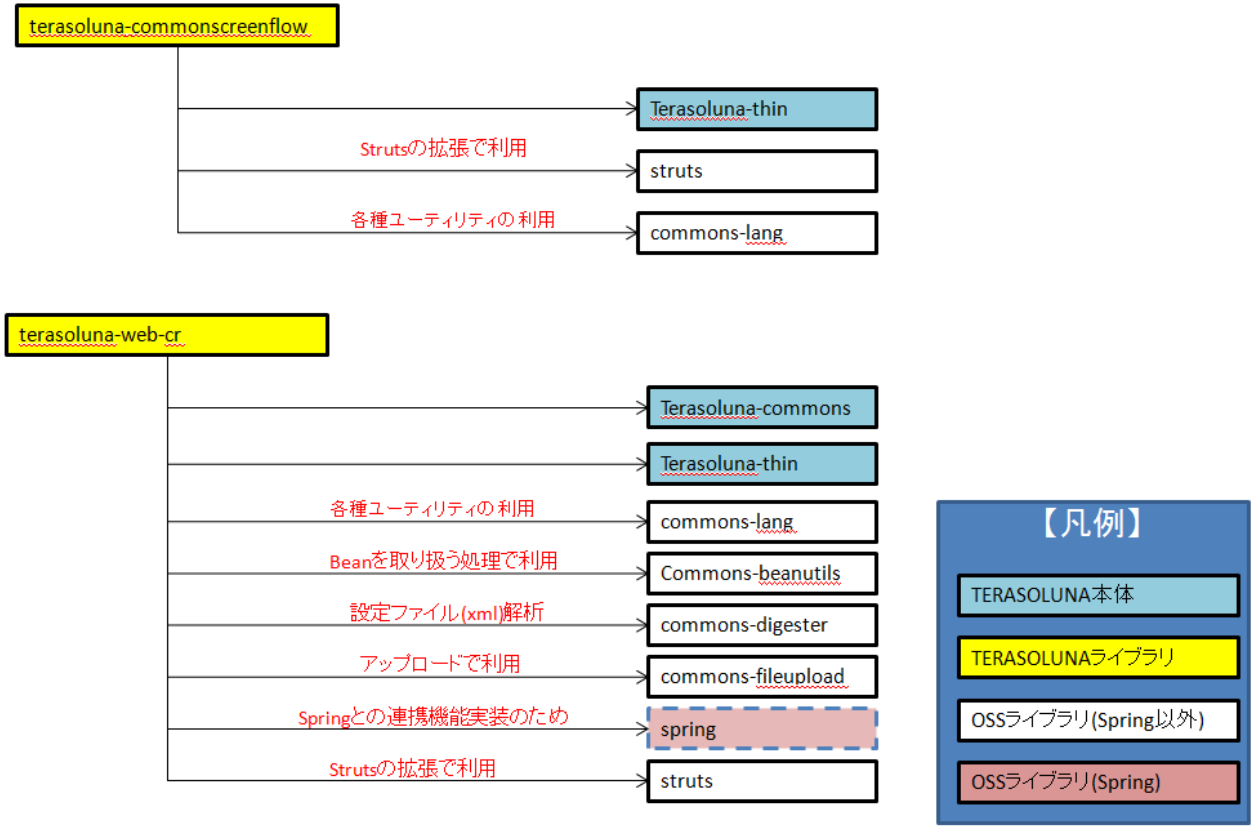


Copyright (c) 2007-2016 NTT DATA Corporation.

はじめに

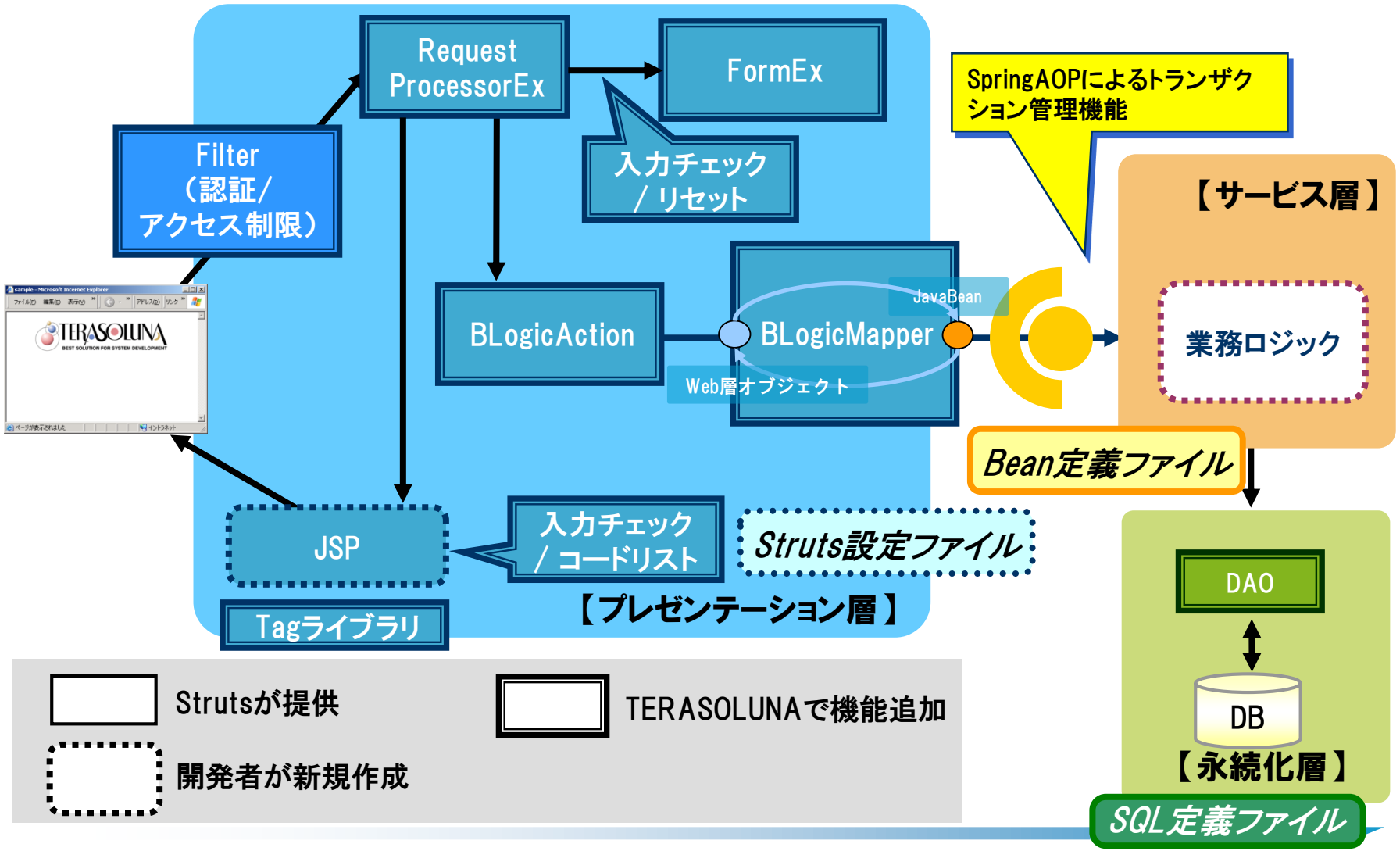
ライブラリの依存関係(2/2)

※図のシンプル化のため、以下のライブラリの依存関係のみ、本スライドに別記する。



Copyright (c) 2007-2016 NTT DATA Corporation.

アーキテクチャ概観（全体）



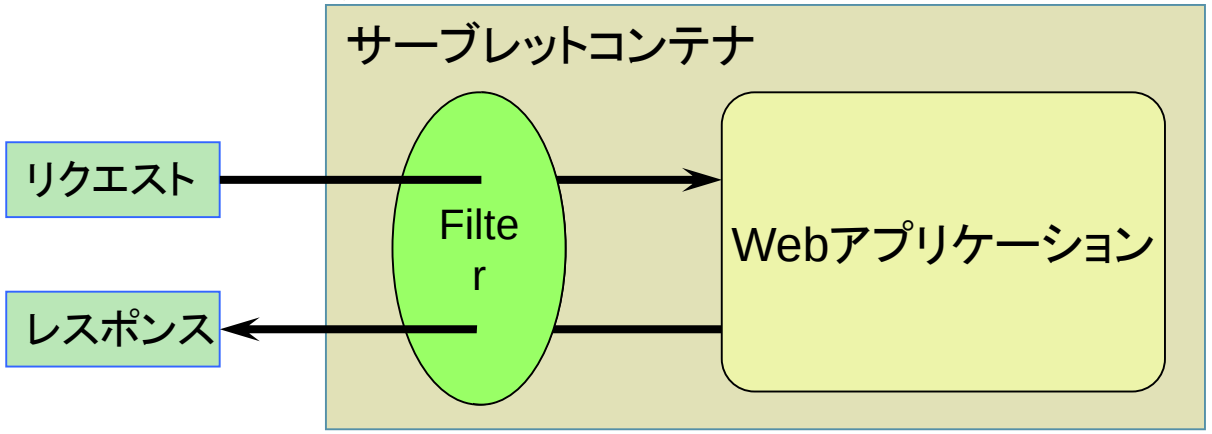
Copyright (c) 2007-2016 NTT DATA Corporation.

①認証・アクセス制御機能

認証・アクセス制御機能

- Filterインタフェースを利用した認証・アクセス制御機能を提供する。
  - ログオン済みチェック機能
  - アクセス権限チェック機能
  - サーバ閉塞チェック機能
  - 業務閉塞チェック機能
  - 拡張子直接アクセス禁止機能

参考）Filterの動作イメージ



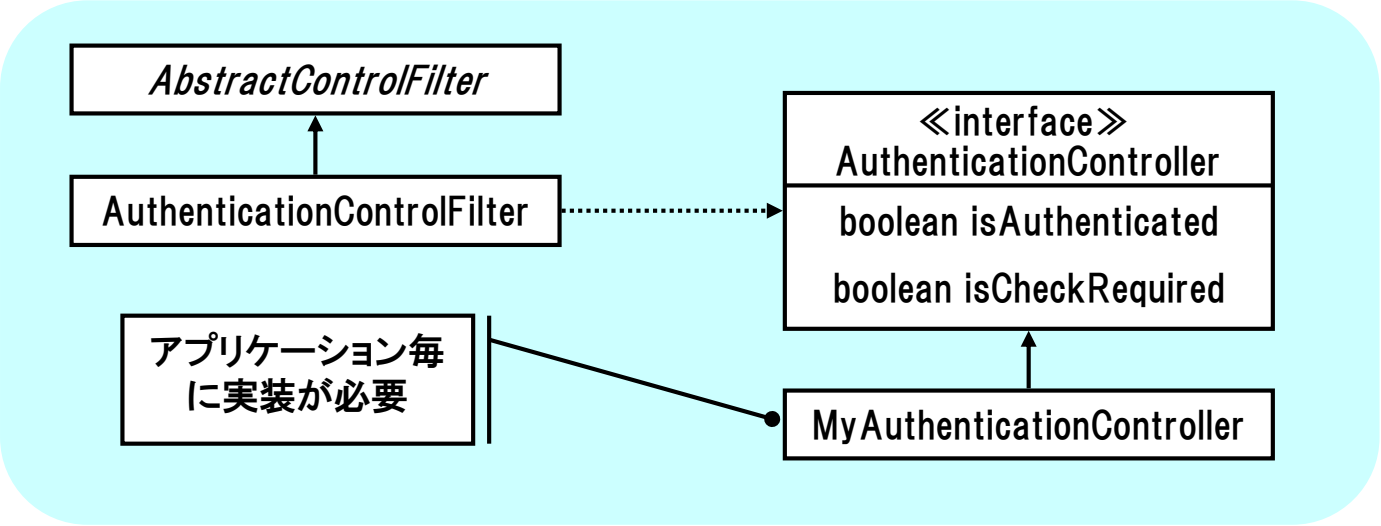
Copyright (c) 2007-2016 NTT DATA Corporation.

①認証・アクセス制御機能

ログオン済みチェック機能

◆ 概要

- アプリケーションにアクセスしたユーザがログオン済みであるかどうかを判定する。
- ログオン済みかどうかの判定はアプリケーションによって異なるため、アプリケーション毎にAuthenticationControllerインタフェースを実装したクラスを作成する必要がある。



Copyright (c) 2007-2016 NTT DATA Corporation.

①認証・アクセス制御機能

ログオン済みチェック機能

◆ 使用方法

- AuthenticationControllerインタフェースを実装する。
  - boolean isAuthenticated・・・ユーザがログオン済みか判定する。ログオンしていない場合はfalseを返却する。
  - boolean isCheckRequired・・・ログオン済みチェックが必要であるかを判定する。ログオン済みチェックが不要な場合(ログイン画面など)はfalseを返却する。
- web.xml、Bean定義ファイルに設定を記述する。

web.xml

```
<filter>
  <filter-name>authenticationControlFilter</filter-name>
  <filter-class>jp.terasoluna.fw.web.thin.AuthenticationControlFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>authenticationControlFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
<error-page>
  <exception-type>jp.terasoluna.fw.web.thin.UnauthenticatedException</exception-type>
  <location>/unauthenticatedError.jsp</location>
</error-page>
```

FWが提供するFilter

Bean定義ファイル

```
<bean id="authenticationController"
    class="jp.terasoluna…MyAuthenticationController" />
```

Bean定義ファイルにはAuthenticationControllerインタフェースを実装したクラスを記述する。BeanIdはauthenticationController。

認証エラーの時に発生する例外。エラーページの遷移先を設定すること。

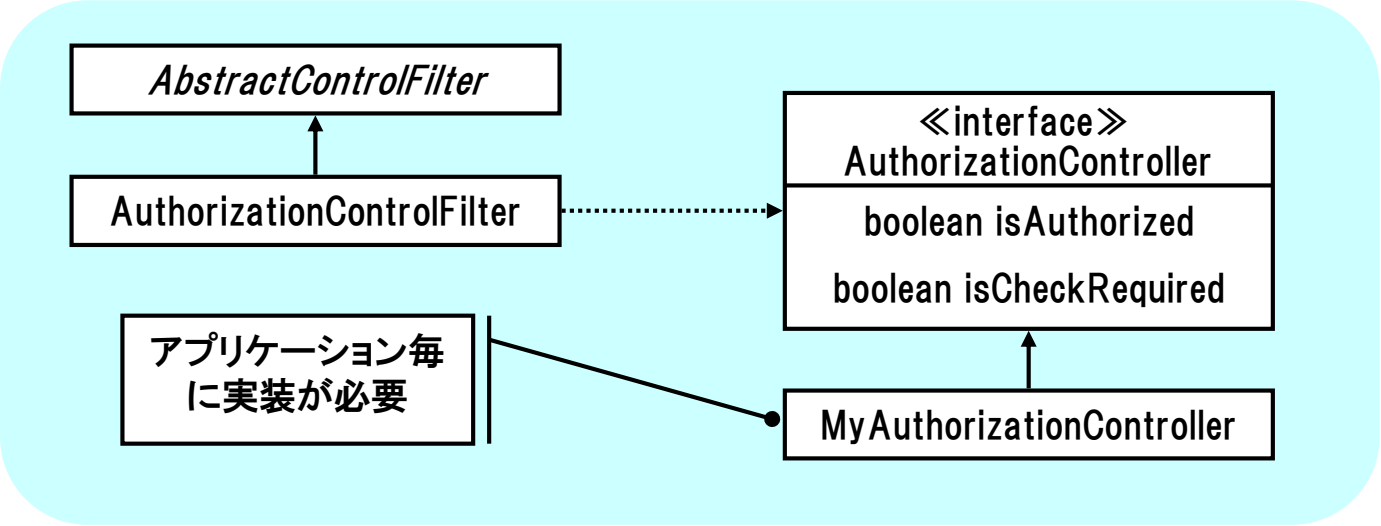
Copyright (c) 2007-2016 NTT DATA Corporation.

①認証・アクセス制御機能

アクセス権限チェック機能

概要

- ユーザが要求したリクエスト先にアクセス権限があるかどうかを判定する。
- アクセス権限を有するどうかの判定はアプリケーションによって異なるため、アプリケーション毎にAuthorizationControllerインタフェースを実装したクラスを作成する必要がある。



Copyright (c) 2007-2016 NTT DATA Corporation.

①認証・アクセス制御機能

アクセス権限チェック機能

使用方法

- AuthorizationControllerインタフェースを実装する。
  - boolean isAuthorized・・・アクセス権限を有するか判定する。権限がない場合はfalseを返却する。
  - boolean isCheckRequired・・・アクセス権限チェックが必要であるかを判定する。アクセス権限チェックが不要な場合(ログイン画面など)はfalseを返却する。
- web.xml、Bean定義ファイルに設定を記述する。

web.xml

```
<filter>
  <filter-name>authorizationControlFilter</filter-name>
  <filter-class>jp.terasoluna.fw.web.thin.AuthorizatinoControlFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>authorizationControlFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
<error-page>
  <exception-type>jp.terasoluna.fw.web.thin.UnauthorizedException</exception-type>
  <location>/unauthorizedError.jsp</location>
</error-page>
```

FWが提供するFilter

Bean定義ファイル

```
<bean id="authorizationController"
    class="jp.terasoluna…MyAuthorizationController" />
```

Bean定義ファイルにはAuthorizationControllerインタフェースを実装したクラスを記述する。BeanIdはauthorizationController。

アクセス権限エラーの時に発生する例外。エラーページの遷移先を設定すること。

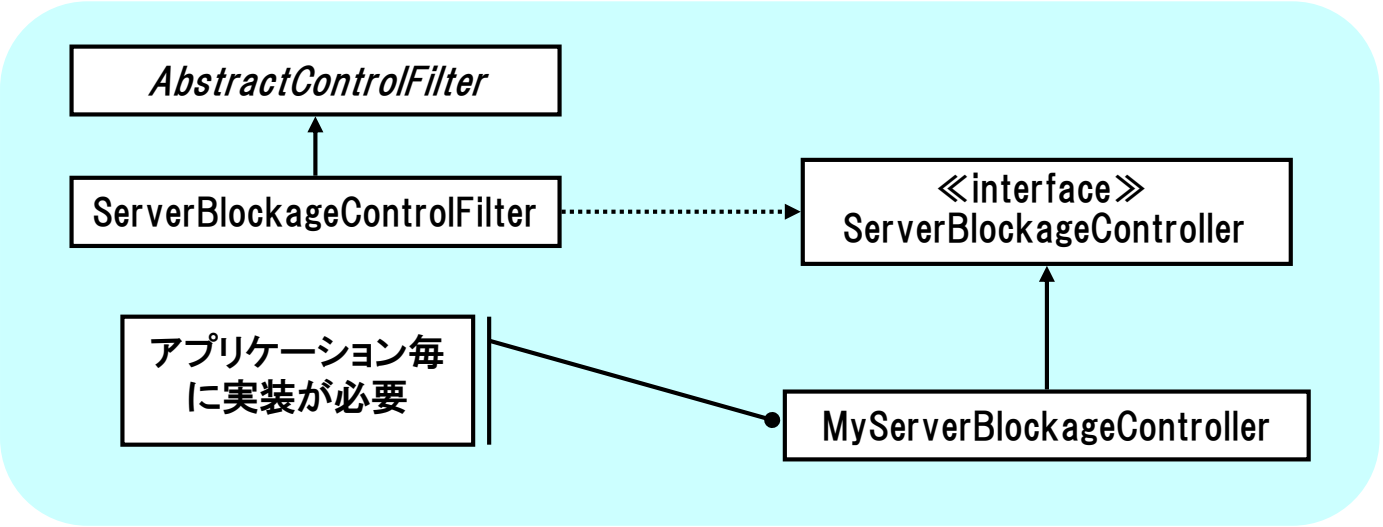
Copyright (c) 2007-2016 NTT DATA Corporation.

①認証・アクセス制御機能

サーバ閉塞チェック機能

概要

- アプリケーションが閉塞中であるかどうかを判定する。
- 閉塞中であるかどうかの判定はアプリケーションによって異なるため、アプリケーション毎にServerBlockageControllerインタフェースを実装したクラスを作成する必要がある。



Copyright (c) 2007-2016 NTT DATA Corporation.

①認証・アクセス制御機能

サーバ閉塞チェック機能

使用方法

- ServerBlockageControllerインタフェースを実装する。
  - void blockade・・・閉塞状態に遷移する。
  - boolean isBlockaded・・・閉塞状態かどうか判定する。
  - void preBlockade・・・予閉塞状態に遷移する。
  - boolean isPreBlockaded・・・予閉塞状態かどうか判定する。
  - void open・・・閉塞状態を解除する。
- web.xml、Bean定義ファイルに設定を記述する。

web.xml

```
<filter>
  <filter-name>serverBlockageControlFilter</filter-name>
  <filter-class>jp.terasoluna.fw.web.thin.ServerBlockageControlFilter </filter-class>
</filter>
<filter-mapping>
  <filter-name>serverBlockageControlFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
<error-page>
  <exception-type>jp.terasoluna.fw.web.thin.ServerBlockageException </exception-type>
  <location>/serverBlockaded.jsp</location>
</error-page>
```

FWが提供するFilter

Bean定義ファイル

```
<bean id="serverBlockageController"
    class="jp.terasoluna…MyServerBlockageController" />
```

Bean定義ファイルにはServerBlockageControllerインタフェースを実装したクラスを記述する。BeanIdはserverBlockageController。

サーバ閉塞時に発生する例外。エラーページの遷移先を設定すること。

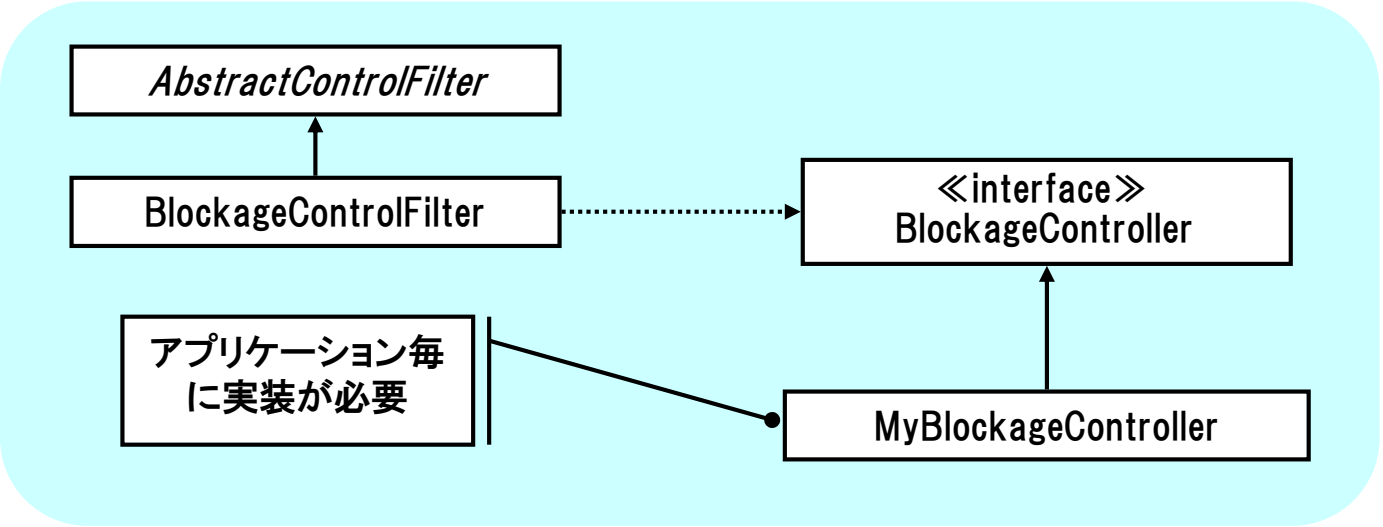
Copyright (c) 2007-2016 NTT DATA Corporation.

①認証・アクセス制御機能

業務閉塞チェック機能

◆ 概要

- 特定の業務が閉塞中であるかどうかを判定する。
- 閉塞中であるかどうかの判定はアプリケーションによって異なるため、アプリケーション毎にBlockageControllerインタフェースを実装したクラスを作成する必要がある。



Copyright (c) 2007-2016 NTT DATA Corporation.

①認証・アクセス制御機能

業務閉塞チェック機能

使用方法

- BlockageControllerインタフェースを実装する。
  - void blockade・・・閉塞状態に遷移する。
  - boolean isBlockaded・・・閉塞状態かどうか判定する。
  - void open・・・閉塞状態を解除する。
- web.xml、Bean定義ファイルに設定を記述する。

web.xml

```
<filter>
  <filter-name>blockageControlFilter</filter-name>
  <filter-class>jp.terasoluna.fw.web.thin.BlockageControlFilter </filter-class>
</filter>
<filter-mapping>
  <filter-name>blockageControlFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
<error-page>
  <exception-type>jp.terasoluna.fw.web.thin.BlockageException </exception-type>
  <location>/blockaded.jsp</location>
</error-page>
```

FWが提供するFilter

業務閉塞時に発生する例外。エラーページの遷移先を設定すること。

Bean定義ファイル

```
<bean id="blockageController"
    class="jp.terasoluna…MyBlockageController" />
```

Bean定義ファイルにはBlockageControllerインタフェースを実装したクラスを記述する。BeanIdはblockageController。

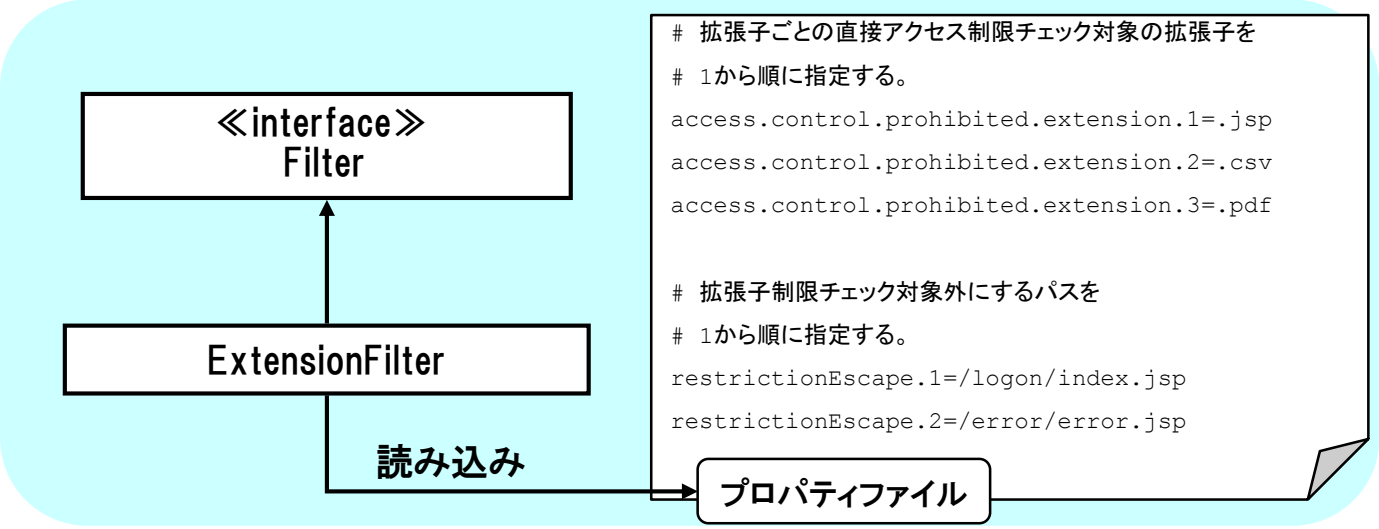
Copyright (c) 2007-2016 NTT DATA Corporation.

①認証・アクセス制御機能

拡張子直接アクセス禁止機能

◆ 概要

- クライアントからのアクセスが特定の拡張子である場合に、アクセスを禁止する。
- jspやpdf、csvファイル等への直接アクセスを禁止したい場合に利用する。
- 直接アクセスを禁止する拡張子はプロパティファイルに設定する。



Copyright (c) 2007-2016 NTT DATA Corporation.

①認証・アクセス制御機能

拡張子直接アクセス禁止機能

◆ 使用方法

- プロパティファイルに直接アクセスを禁止する拡張子、チェックを行わないパスを記述する。

```
access.control.prohibited.extension.<通番>=直接アクセスを禁止する拡張子
例)access.control.prohibited.extension.1=.jsp
restrictionEscape.<通番>=直接アクセス禁止チェックを行わないパス
例） restrictionEscape.1=/logon/index.jsp
```

※上記設定だと、クライアントからのアクセスが/*.jspの場合はエラーとなるが、/logon/index.jspの場合のみチェックを行わない。

- web.xmlに設定を記述する。

web.xml

```
<filter>
  <filter-name>extensionFilter</filter-name>
  <filter-class>jp.terasoluna.fw.web.thin.ExtensionFilter </filter-class>
</filter>
<filter-mapping>
  <filter-name>extensionFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

FWが提供するFilter

Copyright (c) 2007-2016 NTT DATA Corporation.

②ユーザ情報保持機能

ユーザ情報保持機能

- ログオン中のユーザの情報を保持する機能
  - 抽象クラス、UserValueObjectを継承して、クラスを作成する。必要な属性を定義すること。
  - プロパティファイルにUserValueObject継承クラスを設定する。

```
public class MyUserValueObject extends UserValueObject {
    /**ユーザ名。*/
    private String userName = null;
    /**ユーザロール*/
    private String userRole = null;
    /**
     * ユーザ名を返却する。
     * @return ユーザ名
     */
    public String getUserName() {
        return userName;
    }
   ・・・（以下略）
}
```

UserValueObject実装クラス

```
user.value.object=jp.terasoluna.MyUserValueObject
```

プロパティファイル

UserValueObject#createUserValueObject()でプロパティファイルに記述したクラスのインスタンスが作成される。

Copyright (c) 2007-2016 NTT DATA Corporation.

③コードリスト機能

コードリスト機能

◆ コードリストとは？

- 同一の目的で利用される「名前=値」のペアの集まりで、アプリケーション中で変更される頻度がない（極めて少ない）。
- 例）和暦年号

```
wareki.meiji=明治
wareki.taisho=大正
wareki.showa=昭和
wareki.heisei=平成
```

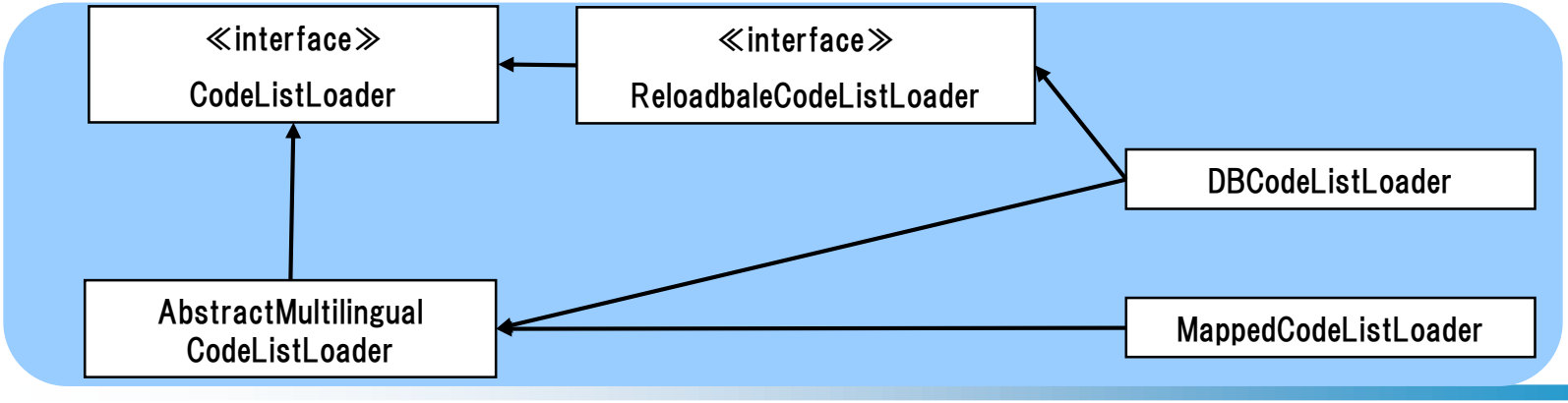
◆ コードリスト機能

- コードリストの定義方法、読み込み方法、表示・使用方法を提供する。

Copyright (c) 2007-2016 NTT DATA Corporation.

③コードリスト機能

- 概要
  - CodeListLoader・・・コードリストを読み込み/保持する全てのクラスが実装するインタフェース。
  - AbstractMultilingualCodeListLoader・・・国際化対応したCodeListLoaderを実装するための抽象クラス
  - MappedCodeListLoader・・・設定ファイルベースでコードリストを読み込むクラス。
  - ReloadableCodeListLoader・・・更新可能なコードリストを保持するクラスが実装するインタフェース。
  - DBCodeListLoader・・・データベースからコードリストを読み込むクラス。
  - CodeBean・・・コードリストの一つの要素を表すクラス。



Copyright (c) 2007-2016 NTT DATA Corporation.

③コードリスト機能

- MappedCodeListLoader
  - SpringFrameworkのBean定義ファイルで設定されることを前提としたCodeListLoader実装クラス。
  - 以下のようにBean定義ファイルに定義する。

```
<bean id="codeList"
      class="jp.terasoluna.fw.web.codelist.MappedCodeListLoader"
      init-method="load">
  <property name="codeListMap">
    <map>
      <entry key="001">
        <value>value001</value>
      </entry>
      <entry key="002">
        <value>value002</value>
      </entry>
      <entry key="003">
        <value>value003</value>
      </entry>
    </map>
  </property>
</bean>
```

codeListMap属性にコードリスト情報をMap形式で設定する。

Bean定義ファイル

CodeBean[]	
001	value001
002	value002
003	value003

codeListMap属性にロケール毎のコードリスト情報を設定することで国際化対応も可能である。

```
<property name="codeListMap">
   <map>
     <entry key="ja">
       <map>
         <entry key="001">
           <value>value001</value>
         </entry>
         <entry key="002">
           <value>value002</value>
         </entry>
       </map>
     </entry>
     <entry key="en_US">
        (略)
     </entry>
   </map>
 </property>
```


Copyright (c) 2007-2016 NTT DATA Corporation.

③コードリスト機能

- DBCodeListLoader
  - データベースからコードリストの情報を読み込むReloadCodeListLoader実装クラス。reloadメソッドを実行することでコードリストの内容を更新することができる。
  - 読み込む対象のデータソースはSpringで定義されたデータソースを使用する。
  - コードリストを取得するSQLはBean定義ファイルから設定する。

Bean定義ファイル

```
<bean id="codeList"
       class="jp.terasoluna.fw.web.codelist.DBCodeListLoader"
       init-method="load">
   <property name="dataSource"
             ref="TerasolunaDataSource"/>
   <property name="sql">
     <value>SELECT KEY, VALUE FROM CODE_LIST ORDER BY KEY</value>
   </property>
</bean>
```

Springで定義される
データソース

コードリストを取得するSQL文

  - 対象テーブルにロケール情報(言語コード、国コードなど)のカラムを追加することで国際化対応することも可能である。

Copyright (c) 2007-2016 NTT DATA Corporation.

③コードリスト機能

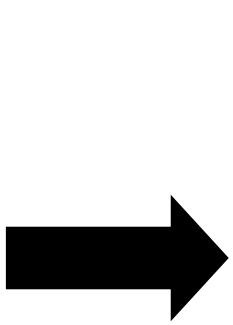
◆ コードリストの表示

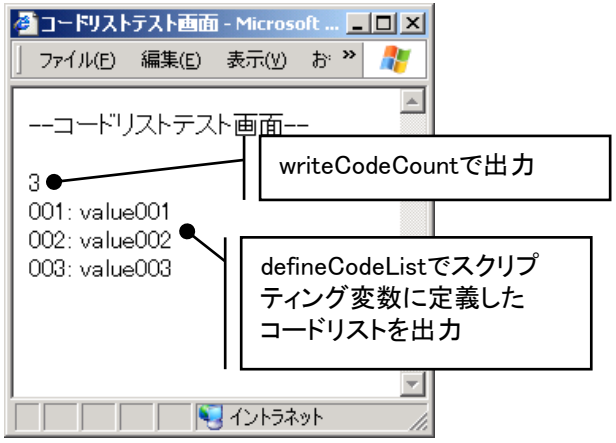
- コードリストをJSP上で扱いやすくするため以下のタグライブラリが用意されている。
  - defineCodeListタグ・・・CodeListLoaderのBeanIdをキーとしてコードリストをスクリプティング変数として定義する。
  - writeCodeCountタグ・・・ CodeListLoaderのBeanIdをキーとしてコードリストの件数を画面に出力する。
  - writeCodeValueタグ・・・Code値をキーにして表示名を画面に出力する。

```
<%@ taglib prefix="t"  uri="/WEB-INF/terasoluna.tld"%>
<html><head><title>コードリストテスト画面</title></head>
<body>
--コードリストテスト画面--<br><br>
<t:writeCodeCount id="codeList" /><br/>
<t:defineCodeList id="codeList"/>
<logic:iterate id="list" name="codeList">
  <bean:write name="list" property="id"/>:
  <bean:write name="list" property="name"/><br>
</logic:iterate>
</body>
</html>
```

一致させる

JSP

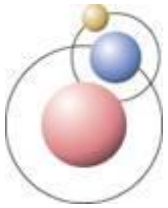





```
<bean id="codeList"
  class="jp.terasoluna.fw.web.codelist.MappedCodeListLoader"
  init-method="load">
 ・・・(以下略)
```

Bean定義ファイル

Copyright (c) 2007-2016 NTT DATA Corporation.

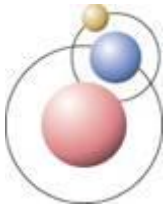


④RequestProcessor拡張機能

RequestProcessor拡張機能

- TERASOLUNAが提供するRequestProcessorExはSpringFrameworkのDelegationgRequestProcessorを継承している。
- 以下の機能を拡張して提供する。
  - 業務毎アクションフォーム切り替え機能
  - processPopulate抑止機能

Copyright (c) 2007-2016 NTT DATA Corporation.



④RequestProcessor拡張機能

- 業務毎アクションフォーム切り替え機能
  - セッションスコープのアクションフォームを使用し、業務が切り替わった際に前の業務のアクションフォームをセッションから削除する。この機能の実行は、
    - アクションフォームのスコープはセッション
    - 業務毎にアクションフォームを定義
    - アクションフォーム名の先頭に『_』(アンダースコア)をつける
    - アプリケーションのパスが業務毎に論理的に分かれている

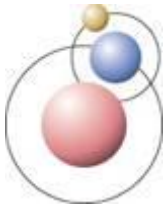
  上記の条件を満たす必要がある。

Struts設定ファイル

```
<!-- form1業務で使用するアクションフォーム -->
<form-bean name="_form1" type="jp.terasoluna.xxx.FormBean1" />
<!-- form2業務で使用するアクションフォーム -->
<form-bean name="_form2" type="jp.terasoluna.xxx.FormBean2" />
<action path="/form1/hoge" name="_form1" scope="session"
                parameter="/form1/index.jsp" />
<action path="/form2/hoge" name="_form2" scope="session"
                parameter="/form2/index.jsp" />
```

パスの最初の階層に業務名を入れることで、どの業務に所属するかフレームワークが判定する。

Copyright (c) 2007-2016 NTT DATA Corporation.



④RequestProcessor拡張機能

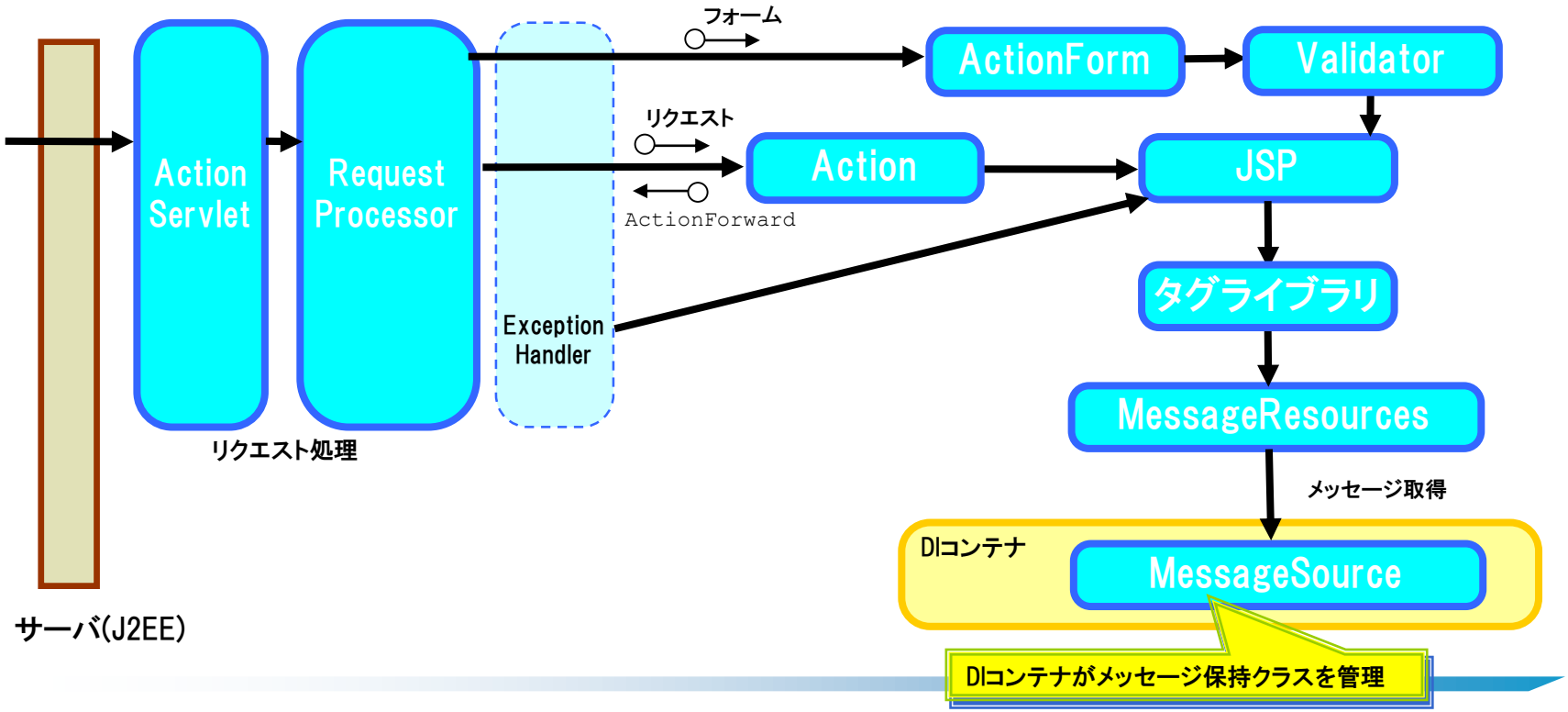
- ProcessPopulate抑止機能
  - ビジネスロジックの実行により、アクションフォームの値が変更された場合、その後の同一リクエスト内ではProcessPopulateを実行しない。
  - ビジネスロジックの実行により変更されたアクションフォームの値が、リクエストパラメータで再度上書きされるのを防ぐため。

Copyright (c) 2007-2016 NTT DATA Corporation.

⑤メッセージ管理機能

メッセージ管理機能とは

- フレームワーク内で使用されるメッセージ(エラーメッセージ等)を保持・取得する機能
  - リソースバンドルメッセージ機能・・・メッセージをプロパティから読み込む
  - DBメッセージ機能　　・・・メッセージをデータベースから読み込む



Copyright (c) 2007-2016 NTT DATA Corporation.

⑤メッセージ管理機能

関連クラス

- SpringMessageResources、SpringMessageResourcesFactory
  - SpringのMessageSourceからメッセージを取得する。

Struts設定ファイル

```
<message-resources
factory="jp.terasoluna.fw.web.struts.util.SpringMessageResourcesFactory"
parameter="messageSource">
```

parameter属性に何も設定しない場合、デフォルトのBean名として"messageSource"を自動で関連付ける。
ただし、parameter属性自体を省略することは不可能。

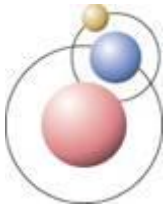
- MessageSource
  - メッセージを保持するクラスが実装すべきインタフェース
  - Springでは「messageSource」というIDのBean定義がメッセージクラスとして認識される
- ResourceBundleMessageSource
  - リソースバンドルを利用したMessageSource実装クラス
    - » プロパティファイルからメッセージを取得する(複数ファイル可)
    - » 国際化対応可能
    - » Webアプリケーション起動中のメッセージリロードはできない

Bean定義ファイル

```
<bean id="messageSource"
      class="org.springframework.context.support.ResourceBundleMessageSource">
  <property name="basenames" value="applicationResources,errors"/>
</bean>
```

読み込むメッセージプロパティファイルをカンマ連結で設定する。メッセージの優先順位は先に記述したものほど高い。通常は、『モジュール毎のメッセージ』→『業務共通メッセージ』→『システムメッセージ』の順に記述すれば良い。

Copyright (c) 2007-2016 NTT DATA Corporation.



⑤メッセージ管理機能

- DataSourceMessageSource
  - データベースを利用したMessageSource実装クラス
    - » データベースからメッセージを取得する
    - » 国際化対応可能
    - » Webアプリケーション起動中のメッセージリロードが可能

メッセージ取得クラス（DBMessageResourceDAOインタフェース実装クラス）を設定（DI）する必要がある。
（デフォルト実装として
DBMessageResourceDAOImplを提供している）

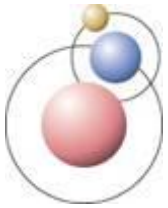
Bean定義ファイル

```
<bean id="messageSource"
      class="jp.terasoluna.fw.message.DataSourceMessageSource">
  <property name="DBMessageResourceDAO" ref="dBMessageResourceDAO"/>
</bean>

<bean id="dBMessageResourceDAO"
      class="jp.terasoluna.fw.message.DBMessageResourceDAOImpl">
  <property name="dataSource" ref="dataSource"/>
</bean>
```

DBからメッセージを読み込むためのDAO

デフォルトのSQLは
SELECT CODE,MESSAGE FROM MESSAGES
テーブル名 = MESSAGES
メッセージコードを格納するカラム名 = CODE
メッセージ本文を格納するカラム名 = MESSAGE

Copyright (c) 2007-2016 NTT DATA Corporation.

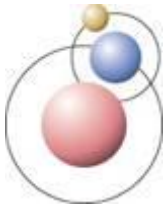


⑥ActionForm拡張機能

ActionForm拡張機能

- 以下の機能を提供する。
  - アクションフォーム基底クラス
  - 自動リセット機能
  - 業務スコープのアクションフォーム
  - ヘルパメソッドの提供
    - リスト、配列項目の要素数自動制御
  - populate抑止機能

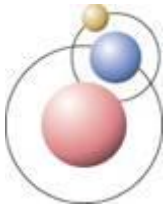
Copyright (c) 2007-2016 NTT DATA Corporation.



⑥ActionForm拡張機能

- ActionFormのスコープ
  - ActionFormを保存するスコープは業務処理の特性に合わせて決定する。
    - 一画面でしか使用しないデータを持つ場合はrequestスコープ
    - 複数画面でデータを持ちまわる必要がある場合はsessionスコープ
      sessionスコープのアクションフォームは必要なくなった時点で削除する必要がある。
  - アクションフォーム名の先頭に"_"を付けたsessionスコープのアクションフォームはセッション内に一つしか存在できない。
    - ユーザが同時に複数の業務を操作しないことが保証される場合は、"_"付のアクションフォームは有効である。(アクションフォームを消すタイミングを意識する必要がない)
      ※ユーザが複数業務を同時に操作する可能性がある場合は、"_"付のアクションフォームは非推奨である。

Copyright (c) 2007-2016 NTT DATA Corporation.

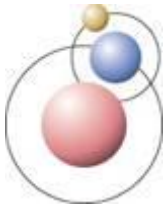


⑥ActionForm拡張機能

- 動的アクションフォームと静的アクションフォームのどちらを使うべきか？
  - 静的アクションフォームはユーザがメソッドを定義する為、型の安全性が保証される。
  - 動的アクションフォームはプロパティが存在しない場合、型の不一致などが発生したときは、実行時にエラーが発生する。（静的アクションフォームはコンパイル時にチェックできる）
  - 動的アクションフォームは静的アクションフォームに比べクラスの実装が不要な為、開発コストは低い。
  - 性能は一概にどちらが上とは言えない。一般的に動的アクションフォームは初期化時に時間がかかる。静的アクションフォームはフレームワークから使用する場合はリフレクションを使用するので動的アクションフォームより値の取得/設定には比較的時間がかかる。
    ※実際には属性の数や使用条件で異なる為、アプリケーション毎に異なる。

→静的アクションフォームのほうが安全性は高い。

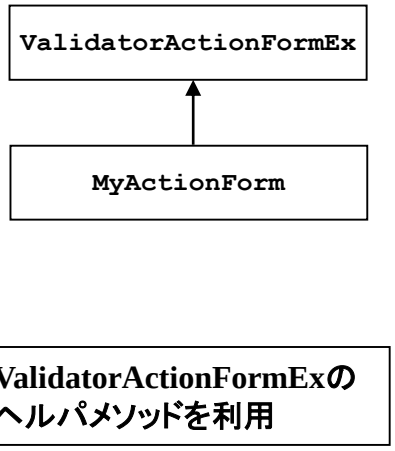
Copyright (c) 2007-2016 NTT DATA Corporation.



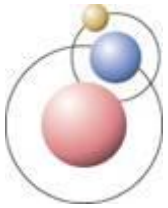
⑥ActionForm拡張機能

- ValidatorActionFormEx
  - ValidatorActionFormのサブクラス
  - 開発者はValidatorActionFormExを継承したクラスを作成し、必要なプロパティ、アクセサメソッドを定義する必要がある
  - 配列/Listプロパティのアクセサメソッド用のヘルパメソッドを定義している（ヘルパメソッドについては後述）
- 実装例

```
public class MyActionForm extends ValidatorActionFormEx {
    private String[] param1 = new String[]{};
    public String getParam1(int index) {
        return (String) super.getIndexedValue("param1", index);
    }
    public void setParam1(int index, String value) {
        super.setIndexedValue("param1", index, value);
    }
}
```



Copyright (c) 2007-2016 NTT DATA Corporation.



⑥ActionForm拡張機能

- DynaValidatorActionFormEx
  - DynaValidatorActionFormのサブクラス
  - 開発者はクラスを実装する必要がなく、Struts設定ファイルに必要なプロパティを定義するだけでよい実装例

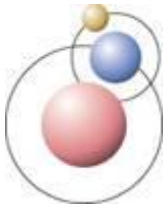
- 実装例

```
<struts-config>
  <form-beans>
      <form-bean name="dynaFormBean"
        type="jp.terasoluna.fw.web.struts.form.DynaValidatorActionFormEx" >
        <form-property name="param1" type="java.lang.String"/>
        <form-property name="param2" type="java.lang.String[]"/>
      </form-bean>
  </form-beans>
```

必要なプロパティを定義する

DynaValidatorActionFormExを指定

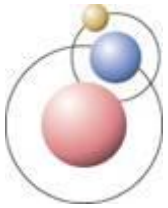
Copyright (c) 2007-2016 NTT DATA Corporation.



⑥ActionForm拡張機能

- FormExインタフェース
  - ValidatorActionFormEx、DynaValidatorActionFormExが実装するインタフェース
  - フレームワーク内部でアクションフォームを操作する際、二つのアクションフォームの差異を意識せずに操作するために用意されている
  - 開発者がこのインタフェースを操作することはない

Copyright (c) 2007-2016 NTT DATA Corporation.



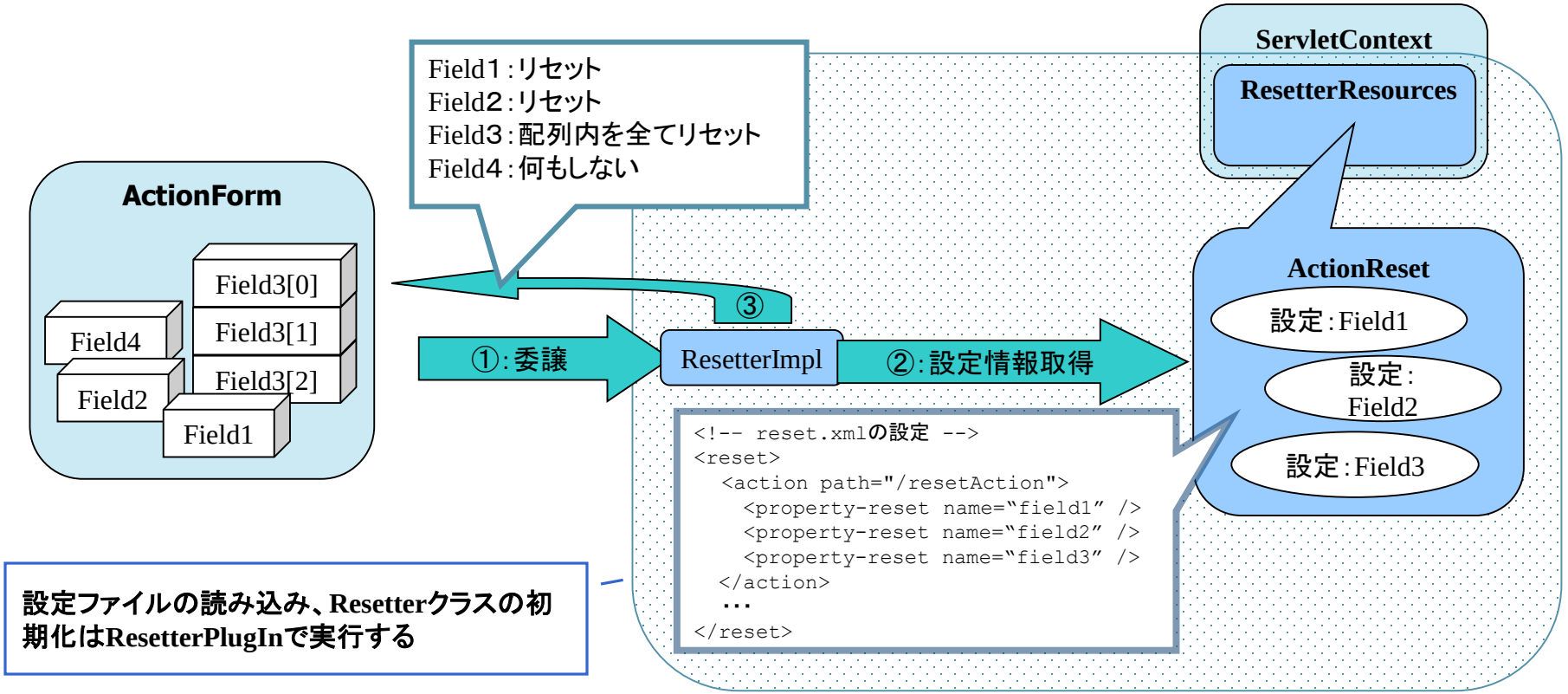
⑥ActionForm拡張機能

- 自動リセット機能
  - ActionFormのreset()メソッドを拡張し、リセット処理をResetterインタフェース実装クラスに委譲する仕組み
  - デフォルトのResetterImplでは設定ファイルを利用することでプロパティのリセットを宣言的に行うことができる
  - HTMLのチェックボックスやセレクトボックスなど、値の反映前に一度フィールド値の初期化が必要なフィールドに対してリセットを行うことを目的として提供する
  - 配列、Listプロパティの指定した範囲のみリセットを行うことも可能
  - TERASOLUNAフレームワークが提供するのは
    - Resetter・・・リセット処理を実行するインタフェース
    - ResetterImpl・・・デフォルトのResetter実装クラス
    - ResetterPlugIn・・・リセット設定ファイル、Resetterクラスを初期化するプラグイン
    - リセット設定ファイル・・・開発者が必要に応じ、指定されたフォーマットで記述する設定ファイル

Copyright (c) 2007-2016 NTT DATA Corporation.

⑥ActionForm拡張機能

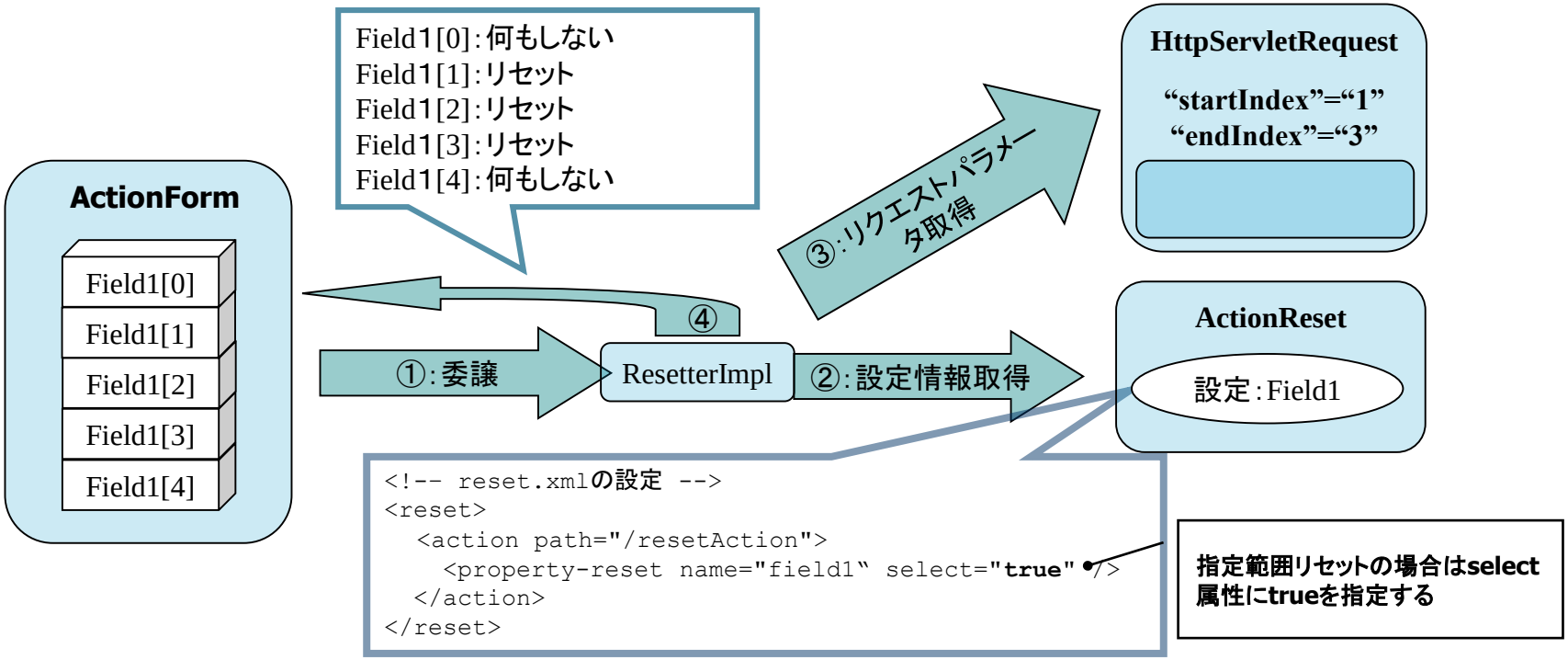
- リセット処理のイメージ(ResetterImpl)
  - reset.xmlで指定されたフィールドをリセットする。



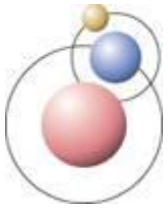
Copyright (c) 2007-2016 NTT DATA Corporation.

⑥ActionForm拡張機能

- 指定範囲リセット処理のイメージ(ResetterImpl)
  - 指定されたIndexをリセット
    - リセット対象範囲はリクエストパラメータのstartIndex、endIndexで指定する
    - 対象となるフィールドがList/配列以外の場合は、通常のリセット処理を行なう



Copyright (c) 2007-2016 NTT DATA Corporation.



⑥ActionForm拡張機能

- ResetterPlugIn
  - リセット設定ファイル、Resetterクラスの初期化を行う
  - Strutsのプラグイン機能として提供（モジュール毎に設定が必要）

プラグイン設定例（Struts設定ファイル）

```
<!-- Resetter plugin -->
<plug-in className="jp.terasoluna.fw.web.struts.plugins.ResetterPlugIn">
    <set-property property="resetter"
                  value="jp.terasoluna.fw.web.xxx.ResetterImpl"/>
    <set-property property="resources" value="/WEB-INF/reset.xml"/>
</plug-in>
```

ResetterPlugInクラスを指定

Resetterクラスを指定

リセット設定ファイルのパスを指定

Copyright (c) 2007-2016 NTT DATA Corporation.

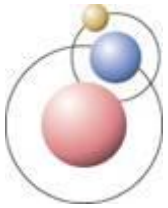
⑥ActionForm拡張機能

- リセット設定ファイル
  - ResetterImpl用の設定ファイル。自動的にリセット処理を行う
  - アクションパス毎にリセット対象のプロパティを設定する
  - 形式

```
<reset>
    <action path="アクションのパス名">
       <property-reset name="フィールド名" select="セレクト属性"/>
    </action>
</reset>
```

    - *アクションのパス名・・・*アクション単位でreset処理を行なうため、アクションのパス名を設定
    - *フィールド名・・・*Formのプロパティ名を設定。
    - *セレクト属性・・・*指定範囲リセット機能を使用する場合は"true"を設定(デフォルト:false)
  - ファイル名は任意。パスはResetterPlugInのプロパティに指定する。(前ページ参照)

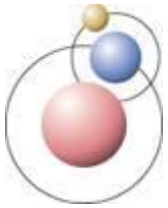
Copyright (c) 2007-2016 NTT DATA Corporation.



⑥ActionForm拡張機能

- 業務スコープのアクションフォーム
  - Strutsがサポートするアクションフォームのスコープはrequest/sessionだが、業務スコープでアクションフォームを使用することを可能としている
  - 業務スコープとは？
    - 実際にアクションフォームが保存されるのはsession
    - アクションフォーム名の先頭に"_"を付与し、session内で"_"付のアクションフォームの唯一性をフレームワークが保証する
    - アクションフォームは業務単位で作成する
      →業務単位でアクションフォームが切り替わるように見せかけている

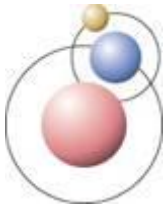
Copyright (c) 2007-2016 NTT DATA Corporation.



⑥ActionForm拡張機能

- ヘルパメソッド
  - アクションフォームの基底クラスであるDyna/ValidatorActionFormExに便利なメソッドを定義
  - 件数取得メソッド
    - 配列/List項目の件数を取得する
    - 配列/Listのヘルパメソッド
  - setIndexedValueメソッド
    - 配列/Listの初期化処理を省略可能、配列/Listのインデックス範囲を超えていた場合動的にサイズを増やす
    - DynaValidatorActionFormExの場合、set(String, int, Object)メソッドも上記と同じ処理を行う
  - getIndexedValueメソッド
    - インデックスが範囲を超えていた場合、nullを返却する
    - DynaValidatorActionFormExの場合、get(String, int)メソッドも上記と同じ処理を行う

Copyright (c) 2007-2016 NTT DATA Corporation.



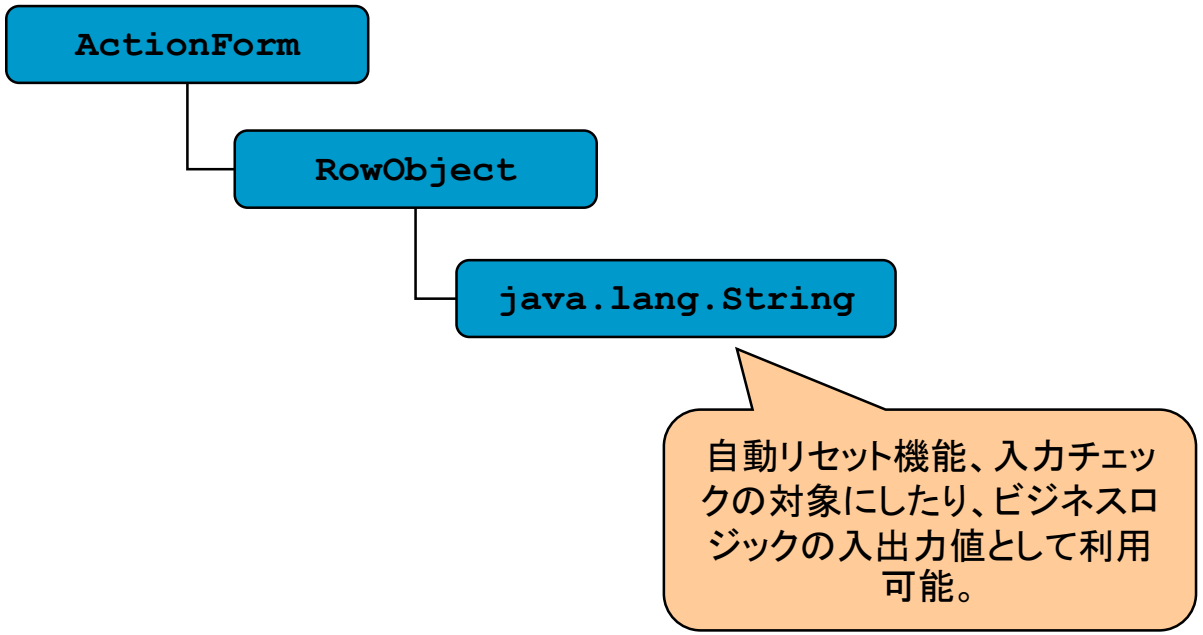
⑥ActionForm拡張機能

- Populate抑止機能
  - ビジネスロジックの実行後、アクションフォームの値が変更された場合にフラグを立て、processPopulateを行わない
  - StrutsのRequestProcessorのprocessPopulateメソッドはサーバー内部のフォワード処理でも実行される
    →ビジネスロジックで変更したアクションフォームの値がリクエストパラメータによって上書きされる可能性がある
  - 一度ビジネスロジックでアクションフォームの値が変更されると、その後processPopulateメソッドの処理は自動的にキャンセルされる

Copyright (c) 2007-2016 NTT DATA Corporation.

⑥ActionForm拡張機能

- ActionForm実装時の補足
  - TERASOLUNAフレームワーク for J2EE(Struts版)では対応していない、プロパティのネストが可能である。



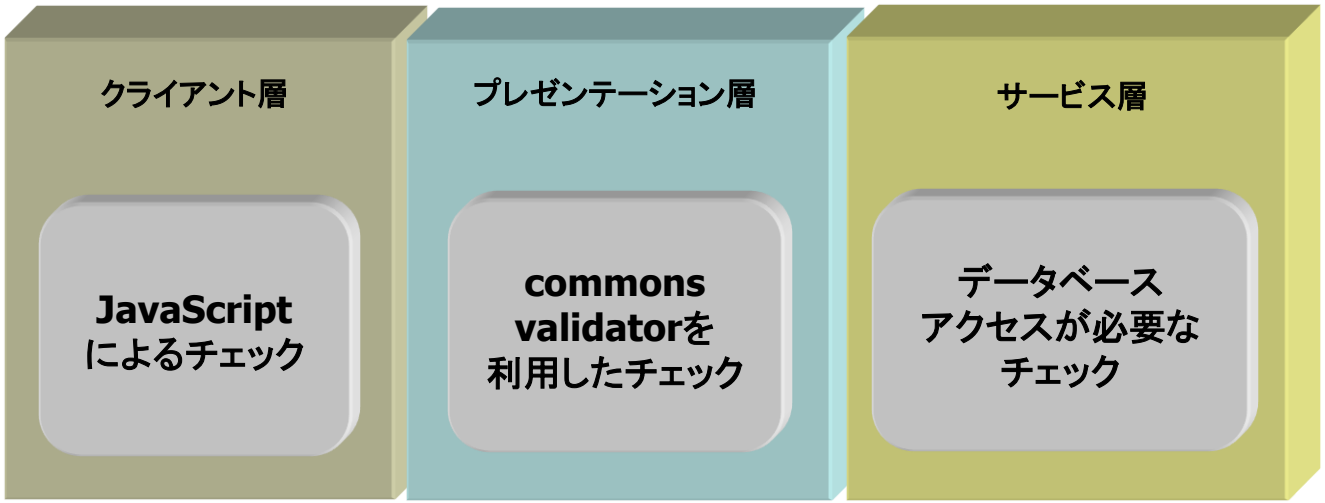
Copyright (c) 2007-2016 NTT DATA Corporation.

⑦入力チェック機能

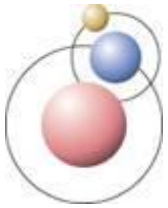
入力チェック機能

概要

- Strutsの入力チェック機能を拡張し、業務開発で必要になる典型的な入力チェックルールを提供する
- データベースアクセスが必要なチェックはサービス層でチェックを行う



Copyright (c) 2007-2016 NTT DATA Corporation.



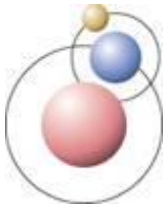
⑦入力チェック機能

解説

◆提供する入力チェックの種類

ルール名	概要	Struts提供		TERASOLUNAで拡張	
		クライアント	プレゼンテーション	クライアント	プレゼンテーション
required	必須入力チェック	○	○	-	-
requiredif	相関必須チェック	-	○	-	-
validwhen	相関チェック	-	○	-	-
minlength	最小入力桁チェック	○	○	-	-
maxlength	最大入力桁チェック	○	○	-	-
mask	正規表現チェック	○	○	-	-
byte	byte型形式チェック	○	○	-	-
short	short型形式チェック	○	○	-	-
integer	int型形式チェック	○	○	-	-
long	long型形式チェック	-	○	-	-
float	float型形式チェック	○	○	-	-

Copyright (c) 2007-2016 NTT DATA Corporation.

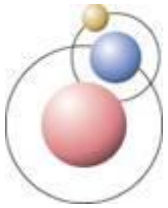


⑦入力チェック機能

◆提供する入力チェックの種類

ルール名	概要	Struts提供		TERASOLUNAで拡張	
		クライアント	プレゼンテーション	クライアント	プレゼンテーション
double	double型形式チェック	-	○	-	-
date	date型形式チェック	○	○	-	-
intRange	int型範囲チェック	○	○	-	-
floatRange	float型範囲チェック	○	○	-	-
doubleRange	double型範囲チェック	-	○	-	-
creditCard	クレジットカード番号 形式チェック	○	○	-	-
email	E-mailアドレス 形式チェック	○	○	-	-
url	URL形式チェック	-	○	-	-
alphaNumericString	半角英数字文字列チェック	-	-	○	○
hankakuKanaString	半角カナ文字列チェック	-	-	○	○

Copyright (c) 2007-2016 NTT DATA Corporation.

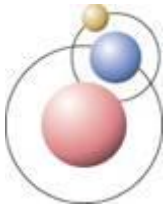


⑦入力チェック機能

◆ 提供する入力チェックの種類

ルール名	概要	Struts提供		TERASOLUNAで拡張	
		クライアント	プレゼンテーション	クライアント	プレゼンテーション
hankakuString	半角文字列チェック	-	-	○	○
zenkakuString	全角文字列チェック	-	-	○	○
zenkakuKanaString	全角カナ文字列チェック	-	-	○	○
capAlphaNumericString	大文字英数字文字列チェック	-	-	○	○
number	数値チェック	-	-	○	○
numericString	数字文字列チェック	-	-	○	○
prohibited	禁止文字列チェック	-	-	-	○
stringLength	文字列長チェック	-	-	○	○
byteLength	byte列長チェック	-	-	-	○
byteRange	byte列長範囲チェック	-	-	-	○
dateRange	date型範囲チェック	-	-	○	○
multiFieldCheck	複数フィールドチェック	-	-	-	○

Copyright (c) 2007-2016 NTT DATA Corporation.



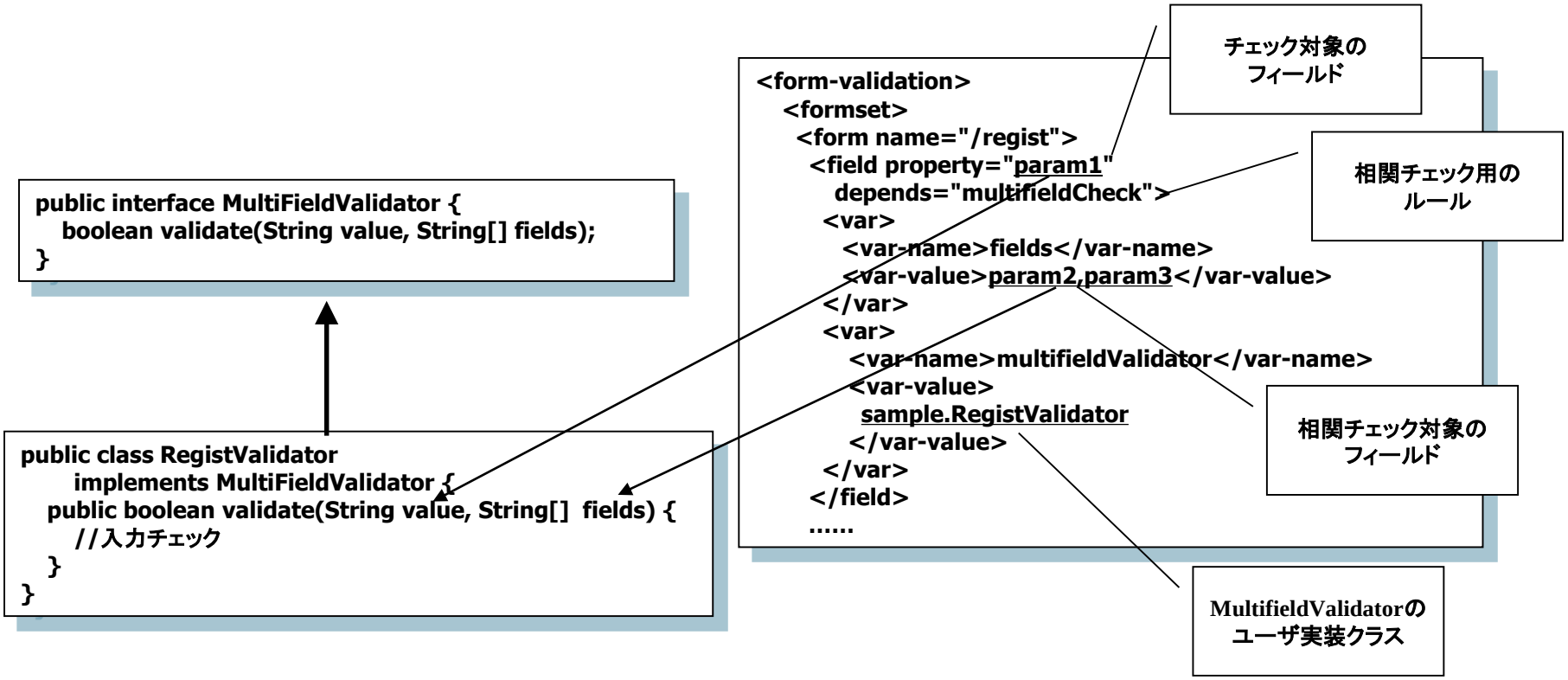
⑦入力チェック機能

- multiFieldCheck
  - TERASOLUNA独自の相関フィールドチェックとしてmultiFieldCheckを提供する
  - requiredif、validWhenでは対応できないような複数フィールドの相関チェックを行う。
  - データベースアクセスを伴うようなチェックはmultiFieldCheckではなく、サービス層でチェックを行う。
  - インタフェースのみを提供し、実際のチェックのロジックはユーザが実装する。

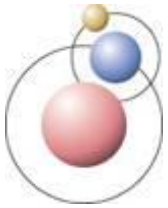
Copyright (c) 2007-2016 NTT DATA Corporation.

⑦入力チェック機能

- 相関チェック使用方法
  - フレームワークが提供するMultiFieldValidatorを実装する。



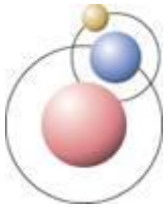
Copyright (c) 2007-2016 NTT DATA Corporation.



⑦入力チェック機能

- JavaScriptによる入力チェック機能
  - タグライブラリ拡張クラス、JavascriptValidatorTagExを提供
  - TERASOLUNAで提供するルールをクライアントで実行するためのタグライブラリ。
  - 属性、使用方法は<html:javascript>タグと同様。
  - JavascriptValidatorTagExではFieldCheckExより全角カナ、半角カナのリストを取得し、Javascriptの変数としてレンダリングを行っている。

Copyright (c) 2007-2016 NTT DATA Corporation.



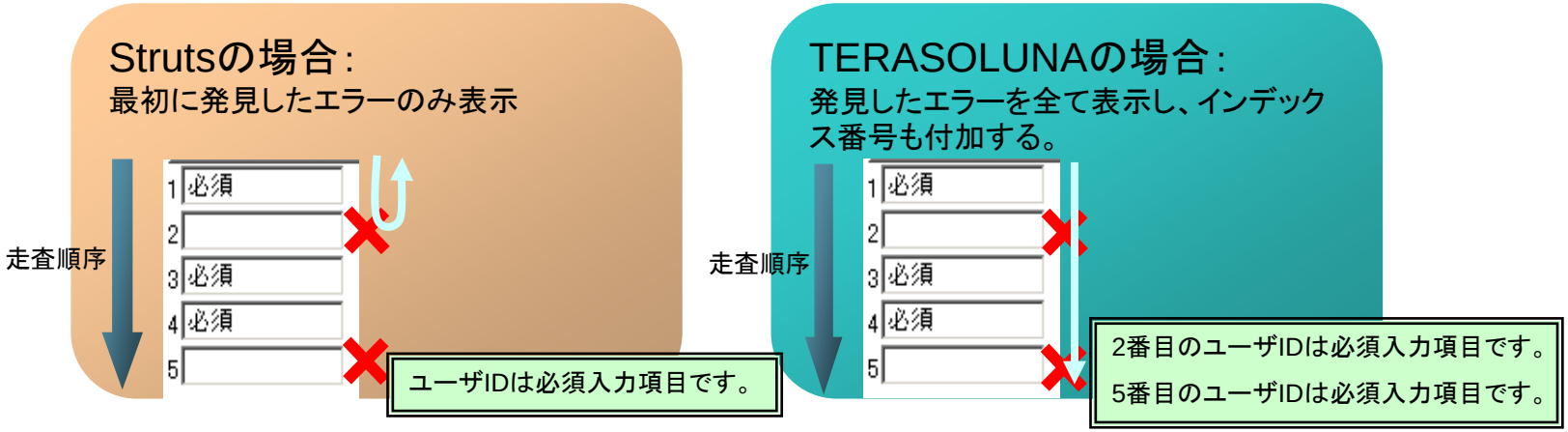
⑦入力チェック機能

- 配列・コレクション型フィールドの検証機能
  - Validatorフレームワークで提供している検証ルールは、配列・コレクション型のフォームプロパティを走査中に検証エラーが発生した場合、以降のフォームプロパティは検証されない。
  - TERASOLUNAでは、全ての要素を検証し、複数の検証エラー表示を可能にする。さらに、インデックス表示を行い、どのフィールドで検証エラーが発生したのかを明示的にエラー表示することが可能となる。
    - この機能を用いる為には、FieldChecksEx#validateArraysIndex()メソッドを実行する。
    - 但し、フィールドに対して複数ルールを組み合わせて検証を行なう場合は、一度に表示できるルールは1種類のみとなる。

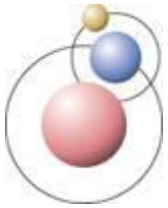
Copyright (c) 2007-2016 NTT DATA Corporation.

⑦入力チェック機能

- 配列、コレクション型フィールドの必須チェックのイメージ



Copyright (c) 2007-2016 NTT DATA Corporation.



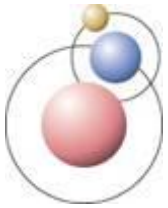
⑧アクション拡張機能

アクション拡張機能

概要

- StrutsのActionクラスを拡張し、アプリケーションで必要となる典型的なActionクラスを提供する。
- フレームワークのAction基底クラスとしてActionExを提供する。
- ActionクラスはDIコンテナで管理する。

Copyright (c) 2007-2016 NTT DATA Corporation.



⑧アクション拡張機能

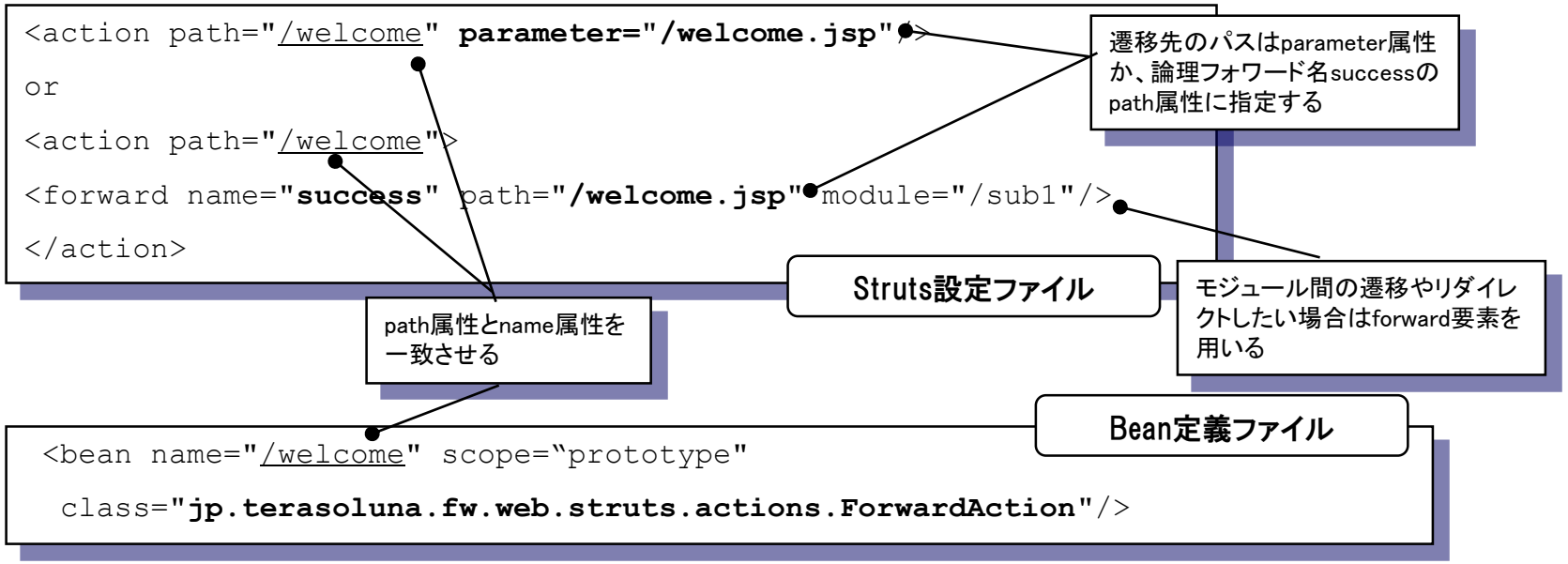
- ActionEx
  - フレームワークが提供するアクションの基底クラス
  - フレームワークが提供するアクションクラスは全てこのクラスからの派生クラス
  - 業務アプリケーションで新規にアクションを実装する場合も、ActionExを継承することを推奨する。
  - 以下の機能を提供する。
    - トランザクショントークンチェック機能
      - » Struts設定ファイルの宣言ベースでトランザクショントークンのチェック、トランザクショントークンの保存を実行する
    - メッセージ、エラー追加機能
      - » セッション上に既にメッセージ、エラーが存在する場合は追加処理を行う。

Copyright (c) 2007-2016 NTT DATA Corporation.

⑧アクション拡張機能

◆ ForwardAction

- JSPやその他の物理的/論理的リソースへのフォワード処理のみを実行するアクション。
- 画面表示はJSPに直接アクセスせず、このアクション経由で実行すること。

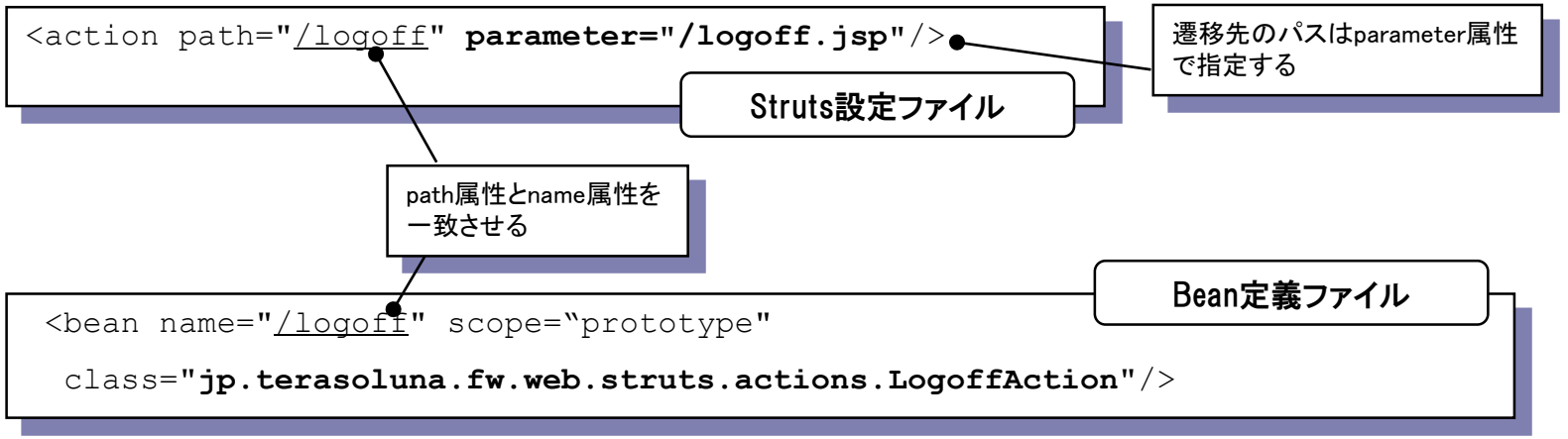


Copyright (c) 2007-2016 NTT DATA Corporation.

⑧アクション拡張機能

LogoffAction

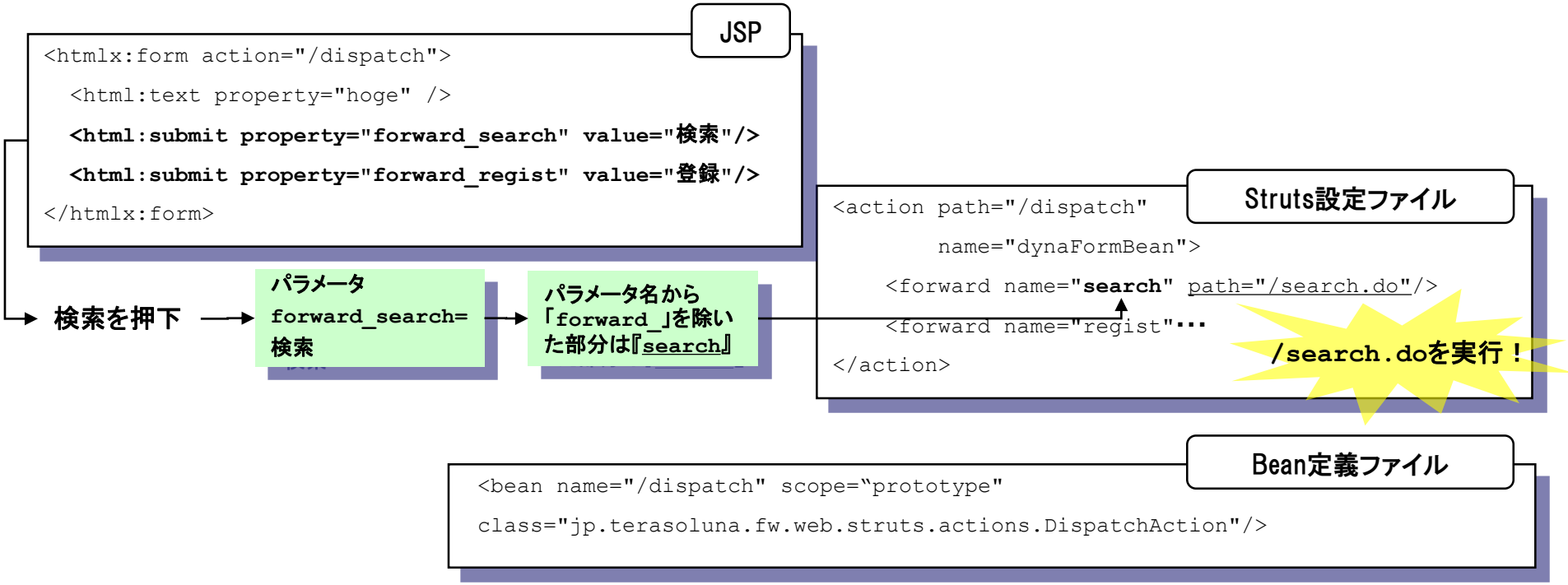
- ログオフの業務処理の基底となるアクションクラス
- セッションの無効化を行っている。その他の処理が必要な場合はこのクラスを継承すること。



Copyright (c) 2007-2016 NTT DATA Corporation.

⑧アクション拡張機能

- DispatchAction
  - 画面に複数のボタンがあるときに、ボタンごとの処理を振り分けるアクション
  - サーバ内部でボタンに紐付いたアクションにフォワードする

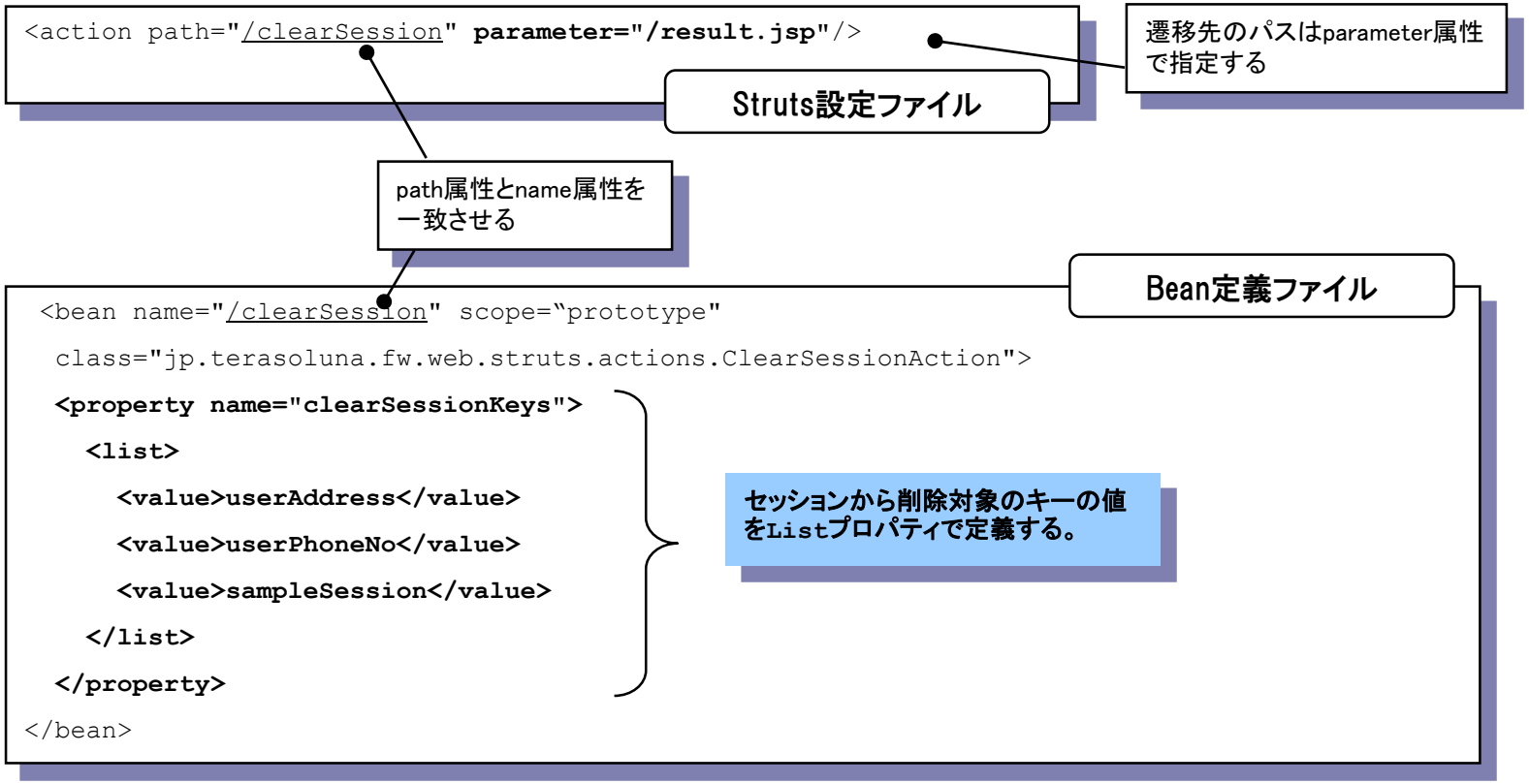


Copyright (c) 2007-2016 NTT DATA Corporation.

⑧アクション拡張機能

ClearSessionAction

- セッションから指定されたキーの値を削除する。



Copyright (c) 2007-2016 NTT DATA Corporation.

⑧アクション拡張機能

- MakeSessionDirectoryAction
  - ユーザ固有のディレクトリを作成するアクションクラス
  - 帳票の一時保存ディレクトリなどに使用できる
  - ディレクトリ名はセッションIDを基にした文字列
  - HttpSessionListenerImplを設定しておけば、ログオフ時にユーザディレクトリは自動的に削除される

```
<action path="/mkSessionDir" parameter="/result.jsp"/>
```

Struts設定ファイル

遷移先のパスはparameter属性で指定する

Bean定義ファイル

```
<bean name="/mkSessionDir" scope="prototype"
  class="jp.terasoluna.fw.web.struts.actions.MakeSessionDirectoryAction">
</bean>
```

```
session.dir.base=/tmp/sessions
```

プロパティファイル

ユーザディレクトリを作成するルートディレクトリ。

Copyright (c) 2007-2016 NTT DATA Corporation.

⑧アクション拡張機能

ReloadCodeListAction

- 指定されたコードリストを更新する

Struts設定ファイル

```
<action path="/reload" parameter="/result.jsp"/>
```

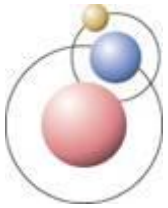
遷移先のパスはparameter属性で指定する

Bean定義ファイル

```
 <bean name="/reload" scope="prototype"
  class="jp.terasoluna.fw.web.struts.actions.ReloadCodeListAction">
  <property name="codeListLoader"
            ref="codeList"/>
</bean>
<bean id="codeList"
       class="jp.terasoluna.fw.web.codelist.DBCodeListLoader"
       init-method="load">
   <property name="dataSource" ref="TerasolunaDataSource"/>
</bean>
```

更新対象のRealoadableCodeListLoaderを指定する

Copyright (c) 2007-2016 NTT DATA Corporation.



⑧アクション拡張機能

- AbstractBLogicAction
  - ビジネスロジックを実行するアクションの基底クラス
  - 設定ファイルの内容からビジネスロジックの入力/出力処理をAbstractBLogicMapperクラスに委譲する
  - サービス層で生成されたメッセージオブジェクトをStrutsのメッセージオブジェクトに変換する
- BLogicAction
  - BLogicインタフェースを実装したビジネスロジッククラスを実行するアクションクラス

※ビジネスロジックの実行の詳細は次章を参照

Copyright (c) 2007-2016 NTT DATA Corporation.

⑨例外ハンドリング機能

例外ハンドリング機能

- Strutsの例外処理機能を利用した例外ハンドラクラスを提供する
  - SystemExceptionHandler
    - 発生したSystemExceptionからメッセージキーを取得し、メッセージを取得してSystemExceptionクラスに設定する
    - 業務ロジックなどでSystemExceptionが発生した場合は、この例外ハンドラクラスを使用すれば、メッセージの取得が自動的に行われる

```
<struts-config>
  <global-exceptions>
    <exception key="some.key" path="/system-error"
      type="jp.terasoluna.fw.exception.SystemException"
      className="jp.terasoluna.fw.web.struts.action.ExceptionConfigEx"
      handler="jp.terasoluna.fw.web.struts.action.SystemExceptionHandler">
      <set-property property="module" value="/exp"/>
      <set-property property="logLevel" value="fatal"/>
    </exception>
  </global-exceptions>
<struts-config>
```

keyに指定したメッセージキーでActionMessageを保存。
path属性で指定したパスへ遷移する。

typeはSystemException
classNameはExceptionConfigEx
handlerにSystemExceptionHandlerを設定

モジュール分割をしている場合は、set-propertyでモジュール名を設定する。
logLevelにはログ出力する際のログレベルを設定する。

Copyright (c) 2007-2016 NTT DATA Corporation.

⑨例外ハンドリング機能

- DefaultExceptionHandler
  - ログ出力のログレベルが指定可能なログハンドラクラス
  - 業務ロジックなどで発生したSystemException以外の例外を対象とする。

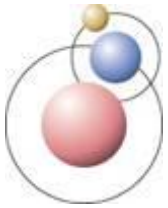
```
<struts-config>
  <global-exceptions>
    <exception key="some.key" path="/system-error"
      type="java.lang.Exception"
      className="jp.terasoluna.fw.web.struts.action.ExceptionConfigEx"
      handler="jp.terasoluna.fw.web.struts.action.DefaultExceptionHandler">
      <set-property property="module" value="/exp"/>
      <set-property property="logLevel" value="fatal"/>
    </exception>
  </global-exceptions>
<struts-config>
```

keyに指定したメッセージキーでActionMessageを保存。
path属性で指定したパスへ遷移する。

typeはjava.lang.Exceptionなどを指定する
classNameはExceptionConfigEx
handlerにDefaultExceptionHandlerを設定

モジュール分割をしている場合は、set-propertyでモジュール名を設定する。
logLevelにはログ出力する際のログレベルを設定する。

Copyright (c) 2007-2016 NTT DATA Corporation.



⑩ビジネスロジック実行機能

ビジネスロジック実行機能

◆ビジネスロジックを実行方式を提供する

- ビジネスロジッククラスは
  - BLogicインタフェースを実装
  - POJO

  のいずれかの方式を選択することができる

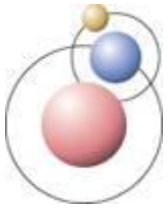
◆Web層⇔サービス層の入出力設定

- 設定ファイルベースで層間の入出力を自動化することが可能

◆トランザクション管理

- SpringFrameworkによるトランザクション管理を採用
- 開発者がトランザクション関連APIを意識することなく、宣言的にトランザクションの管理が可能

Copyright (c) 2007-2016 NTT DATA Corporation.



⑩ビジネスロジック実行機能

◆ビジネスロジックの実行方式①

- BLogicインタフェースを使用
  - BLogicインタフェースを実装し、ビジネスロジックを作成する

```
public interface BLogic<P> {
    BLogicResult execute(P params);
}
```

  - BLogicインタフェースの入力値
    - » POJOで実装する。
    - » 画面仕様、ビジネスロジックの仕様に従い必要なプロパティを定義する
  - BLogicインタフェースの出力値
    - » BLogicResultクラスを使用する。
    - » 結果オブジェクト、メッセージ情報、エラー情報、結果文字列を設定することが可能

Copyright (c) 2007-2016 NTT DATA Corporation.

⑩ビジネスロジック実行機能

- BLogicインタフェースの実装例

```
public class AddBLogic implements BLogic<AddBLogicParam> {
    public BLogicResult execute(AddBLogicParam params) {
        int value1 = params.getValue1();
        int value2 = params.getValue2();
        AddBLogicResult result = new AddBLogicResult();
        result.setResult(value1 + value2);
        BLogicResult bResult = new BLogicResult();
        bResult.setResultObject(result);
        BLogicMessages messages = new BLogicMessages();
        messages.add("message", new BLogicMessage("blogic.message"));
        bResult.setMessages(messages);
        bResult.setResultString("success");
        return bResult;
    }
}
```

入力値クラスはGenericで型を指定する

入力値クラスから入力値を取得する

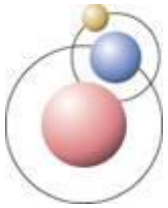
出力オブジェクト。ここではPOJOで実装。Map型も使用可

出力オブジェクトを出力値に設定

出力値にメッセージを設定

ビジネスロジックの処理結果を論理的な文字列として設定

Copyright (c) 2007-2016 NTT DATA Corporation.



⑩ビジネスロジック実行機能

- 設定ファイル
  - BLogicActionクラスを使用する

Struts設定ファイル

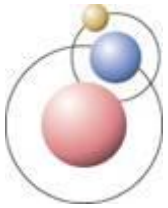
```
<action path="/add"
        name="addForm">
    <forward name="success" path="/add.jsp" />
</action>
```

Bean定義ファイル

```
<bean name="/add" scope="prototype"
    class="jp.terasoluna.fw.web.struts.actions.BLogicAction">
  <property name="businessLogic"
            ref="addBLogic"/>
</bean>
<bean id="addBLogic" class="jp.terasoluna.xxx.blogic.AddBLogic" />
```

BLogicActionの businessLogicプロパティに BLogic実装クラスをDIする

Copyright (c) 2007-2016 NTT DATA Corporation.



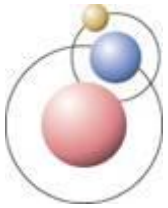
⑩ビジネスロジック実行機能

◆ビジネスロジックの実行方式②

- ビジネスロジックをPOJOで実装する
  - ビジネスロジックのポータビリティ(TERASOLUNA以外でも使用できる)を保つ場合はPOJOで実装する必要がある
  - POJOはTERASOLUNAのAPIに依存しないように実装する
  - ビジネスロジックの入出力値
    - » ビジネスメソッドのインタフェースには必要な値のみを指定する
  - Web層(Struts、TERASOLUNA)とのブリッジのためAbstractBLogicActionを実装したアクションクラスを作成する

```
public abstract class AbstractBLogicAction<P> extends ActionEx {
    public abstract BLogicResult doExecuteBLogic(P param) throws Exception;
}
```


Copyright (c) 2007-2016 NTT DATA Corporation.



⑩ビジネスロジック実行機能

- AbstractBLogicAction、POJOの実装例

```
public class AddAction extends AbstractBLogicAction<AddBLogicParam> {

    private AddService addService = null;

    public BLogicResult doExecuteBLogic(AddBLogicParam param) throws Exception {
        int value1 = param.getValue1();
        int value2 = param.getValue2();
        int result = addService.add(value1, value2);
        Map<String, Integer> map = new HashMap<String, Integer>();
        map.put("result", result);
        BLogicResult bResult = new BLogicResult();
        bResult.setResultObject(map);
        bResult.setResultString("success");
        return bResult;
    }
}
```

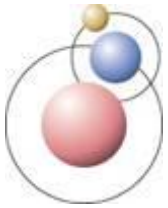
doExecuteBLogicの引数の型はGenericsで指定する

POJOで実装したビジネスロジック。DIコンテナで設定する。※ここではgetter/setterは省略

ビジネスロジックの呼び出し。Actionクラスの中ではビジネスロジックは記述しないこと。

doExecuteBLogicの戻り値はBLogicインタフェースの戻り値と同様。結果オブジェクトはここではMapを使用しているが、POJOも可。

Copyright (c) 2007-2016 NTT DATA Corporation.



⑩ビジネスロジック実行機能

- AbstractBLogicAction、POJOの実装例

```
public class AddBService {
    public int add(int value1, int value2) {
        return value1 + value2;
    }
}
```

Web層、プレゼンテーション層、Struts、TERASOLUNAに依存しないように実装することで、このビジネスロジックがどんなアプリケーションでも使用可能なクラスとなる

- 設定ファイル

```
<action path="/add"
        name="addForm">
    <forward name="success" path="/add.jsp" />
</action>
```

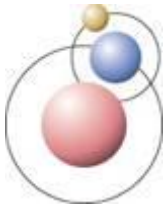
Struts設定ファイル

Bean定義ファイル

```
<bean name="/add" scope="prototype"
      class="jp.terasoluna.xxx.action.AddAction">
  <property name="addBLogic"
            ref="addBLogic"/>
</bean>

<bean id="addBLogic" class="jp.terasoluna.xxx.blogic.AddBLogic"
```

AddActionにPOJOのビジネスロジックをDIしている

Copyright (c) 2007-2016 NTT DATA Corporation.



⑩ビジネスロジック実行機能

◆ ビジネスロジックの入出力設定

- 設定ファイルに入出力の設定を記述することで、Web層⇔サービス層間のデータの変換を自動化する

- 形式

要素			要素数	属性	必須	説明
blogic-io			-	-	○	ルート要素。
	blogic-params		1	bean-name	-	入力値の設定要素。bean-name属性には入力値クラスを指定。入力がない場合は、設定不要。
		set-property	n	property	○	入力値のweb層での物理名。
				blogic-property	-	入力値のサービス層での物理名。bean-nameクラスの属性名となる。
				source	○	入力値の取得元。デフォルトではrequest,session,form,applicationが指定可。
	blogic-result		1	-	-	出力値の設定要素。
		set-property	n	property	○	出力値のweb層での物理名。
				blogic-property	-	出力値のサービス層での物理名。結果オブジェクトの属性名となる。
				dest	○	出力値の反映先。デフォルトではrequest,session,formが指定可。

Copyright (c) 2007-2016 NTT DATA Corporation.

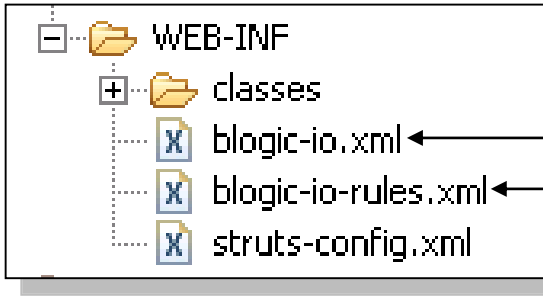
⑩ビジネスロジック実行機能

- ビジネスロジック入出力設定の初期化
  - Strutsのプラグインの機能を利用し、初期化する。
  - 設定例

Struts設定ファイル

```
<plug-in
  className="jp.terasoluna.fw.web.struts.plugins.BLogicIOPlugIn">
  <set-property property="resources" value="/WEB-INF/blogic-io.xml"/>
  <set-property property="digesterRules" value="/WEB-INF/blogic-io-rules.xml"/>
  <set-property property="mapperClass"
    value="jp.terasoluna.fw.service.thin.BLogicMapper"/>
</plug-in>
```

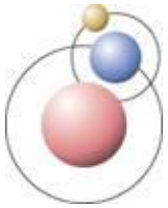
ビジネスロジック入出力処理を行うクラス。BLogicMapperはフレームワークが提供するデフォルトクラス。



ビジネスロジック入出力設定ファイル。ファイル名は任意。

ビジネスロジック入出力設定ファイルをパースするためのルールファイル。必要がない限りはフレームワークが提供するファイルをそのまま使って問題ない。

Copyright (c) 2007-2016 NTT DATA Corporation.



⑩ビジネスロジック実行機能

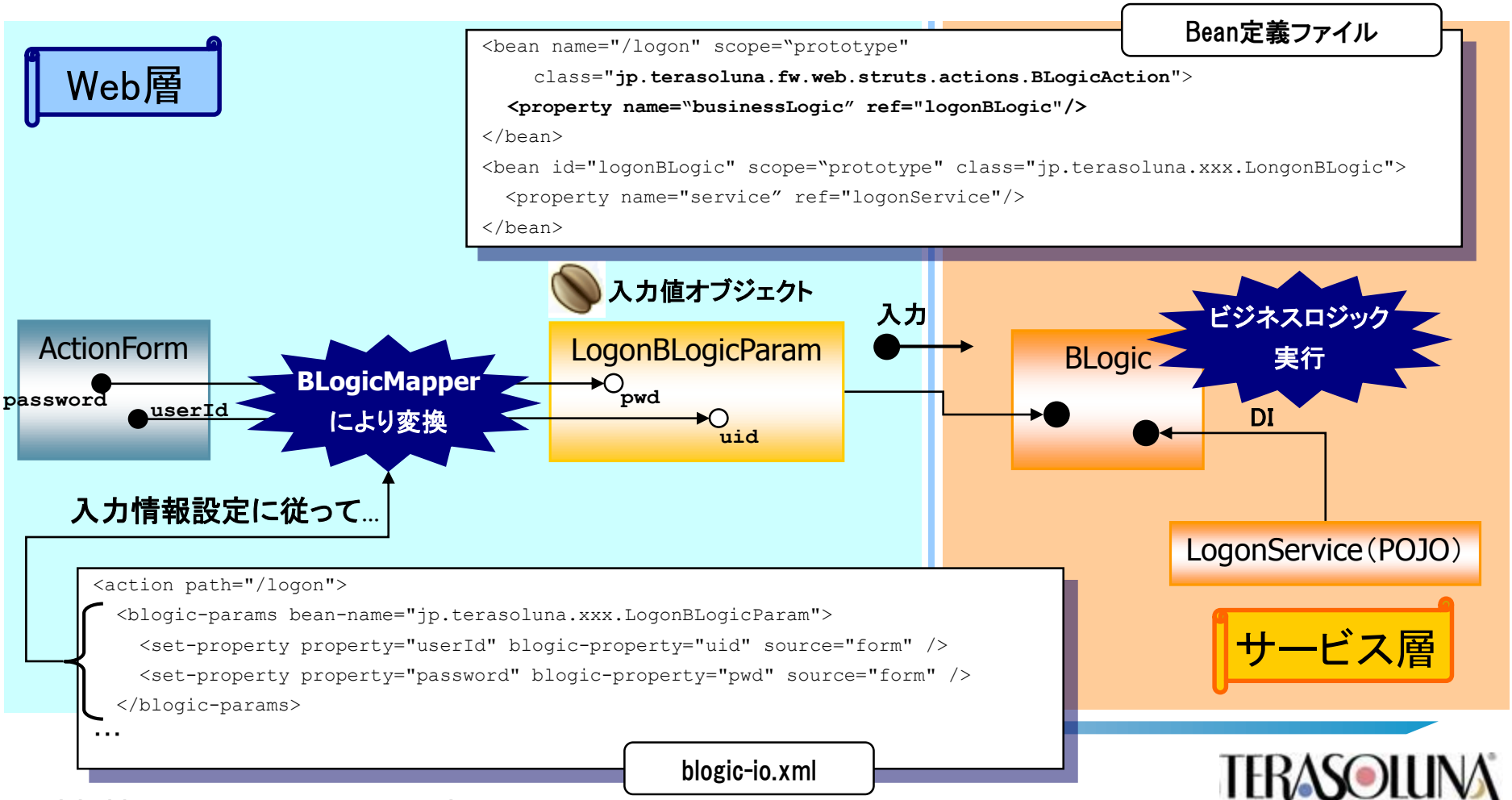
- ビジネスロジック入出力処理
  - AbstractBLogicMapperを実装したクラスで処理する
  - 実装クラスはプラグインの設定ファイルに記述(前頁参照)
  - リクエスト、セッション、アクションフォームに対して単純な入出力処理を行えるBLogicMapperクラスをデフォルトとして提供する
  - AbstractBLogicMapper実装クラスはビジネスロジック入出力ファイルのsource属性、dest属性に設定された文字列が示すオブジェクトと、入出力オブジェクトの変換を行う
    - 入力処理メソッド
      - getValueFrom<*source属性に指定された文字列*>
    - 出力処理メソッド
      - setValueTo<*dest属性に指定された文字列*>
  - デフォルトのBLogicMapperではsource属性にrequest、session、form、application、dest属性にrequest、session、form指定可能。それぞれHttpServletRequest、HttpSession、ActionForm、ServletContextとの入出力が可能である。

Copyright (c) 2007-2016 NTT DATA Corporation.

⑩ビジネスロジック実行機能

◆ビジネスロジックの入力設定例とイメージ

※ここではBLogicを使用する

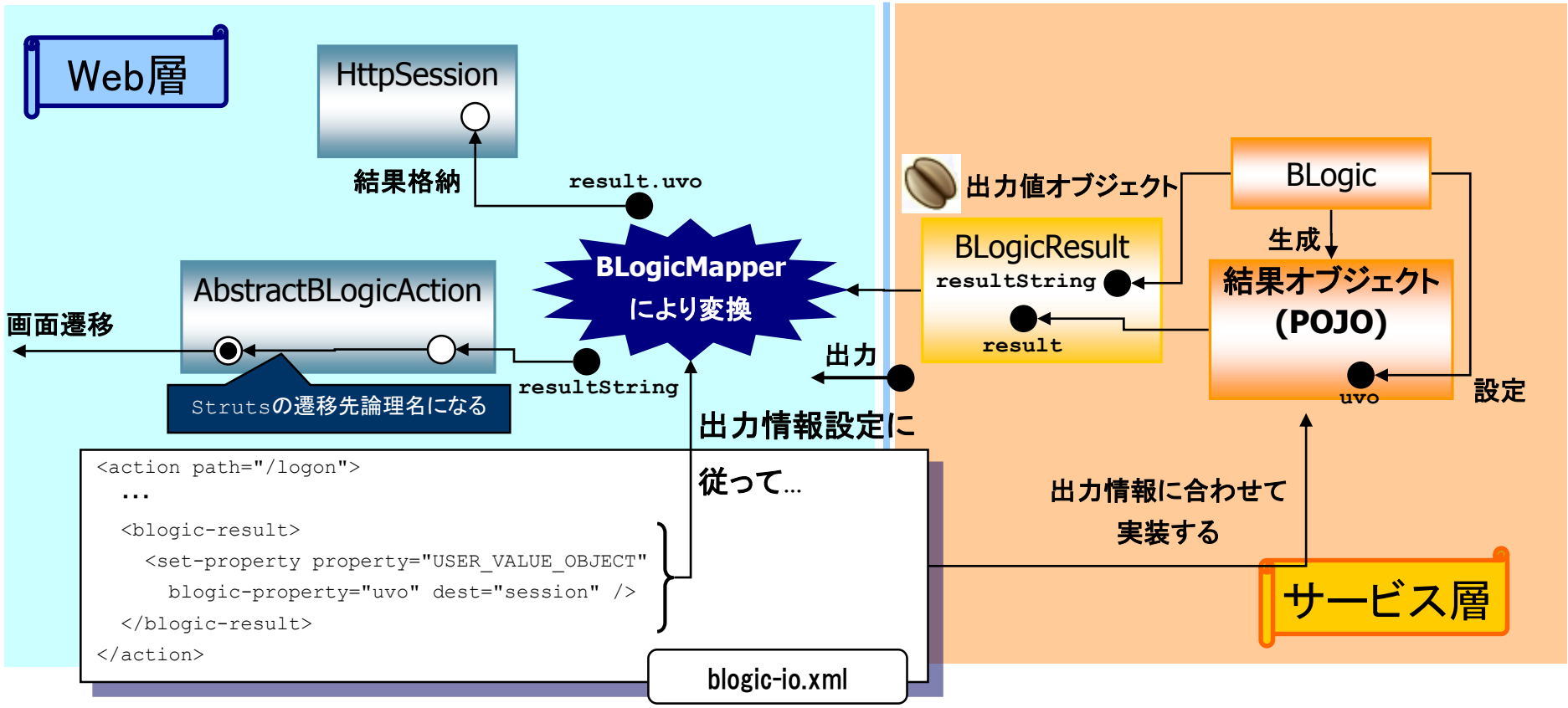


Copyright (c) 2007-2016 NTT DATA Corporation.

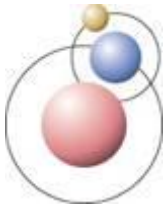
⑩ビジネスロジック実行機能

- 出力時

  ※ここではBLogicを使用する



Copyright (c) 2007-2016 NTT DATA Corporation.

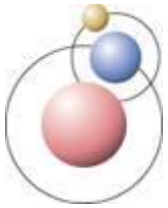


⑪トランザクション管理

トランザクション管理

- コミット・ロールバックはフレームワークが行う。
  - AOPを利用したトランザクションを行うため、開発者がトランザクションコードを実装する必要がない。
  - サービス開始時にトランザクションが開始され、終了時にコミットされる。例外時はロールバックを行う。
  - 入力値の業務妥当性エラーなどでトランザクションを明示的にロールバックする必要がある場合は例外をスローする。
  - ビジネスロジックから例外をスローせずにロールバックを行う必要がある場合は、ユーティリティクラスを利用する。
- サービス層をトランザクション境界とし、1サービス・1トランザクションとする

Copyright (c) 2007-2016 NTT DATA Corporation.



⑪トランザクション管理

トランザクション管理イメージ例(Bean定義ファイル) 1/3

```
<?xml version="1.0" encoding="UTF-8" ?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xsi:schemaLocation="
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-2.5.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-2.5.xsd">
```

Bean定義ファイルをスキーマベースで記述するための設定を行う。

AOPの設定に使うaopスキーマの設定、およびトランザクションの設定に使うtxスキーマの設定を行う。

それぞれの名前空間に対応するスキーマロケーションを空白区切りで指定する。

Copyright (c) 2007-2016 NTT DATA Corporation.

⑪トランザクション管理

トランザクション管理イメージ例(Bean定義ファイル) 2/3

```
<!-- 単一のJDBCデータソース向けのトランザクションマネージャ -->
<bean id="transactionManager" class="org .…(中略)
….DataSourceTransactionManager">
  <property name="dataSource" ref="dataSource"/>
</bean>

<!- トランザクションの設定 -->
<tx:advice id="transactionInterceptor">
  <tx:attributes>
    <tx:method name="insert*"
               propagation="REQUIRED"
               rollback-for="java.lang.Exception"/>
    <tx:method name="update*"
               propagation="REQUIRED"
               rollback-for="java.lang.Exception"/>
    <tx:method name="*"
               propagation="REQUIRED"
               rollback-for="java.lang.Exception"
               read-only="true"/>
  </tx:attributes>
</tx:advice>
```

トランザクションマネージャのBeanIDを transactionManager に設定しておくと、tx スキーマから自動的に利用される。

Springで提供されているタグ。
宣言的トランザクションを行うための設定をする。

<tx:method/>要素で
どのメソッドにどのようなトランザクション管理を行うのかを指定する。

Springのデフォルト設定では、検査例外がスローされてもロールバックされない。
例のように、rollback-for属性に例外クラスを指定するとロールバックされる。

Copyright (c) 2007-2016 NTT DATA Corporation.

⑪トランザクション管理

トランザクション管理イメージ例(Bean定義ファイル) 3/3

```
 <!- AOPの設定 -->
<aop:config>
   <aop:pointcut id="blogicBeans"  expression="bean(*BLogic)"/>
   <aop:pointcut id="serviceBeans" expression="bean(*Service)"/>
   <aop:advisor
        pointcut-ref="blogicBeans"
        advice-ref="transactionInterceptor"/>
   <aop:advisor
        pointcut-ref="serviceBeans"
        advice-ref="transactionInterceptor"/>
</aop:config>
```

Springで提供されているタグ。
AOPの設定を行う。

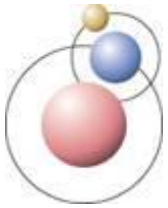
インタセプタを適用するBeanIDを指定し、ポイントカットIDを付与する。

ポイントカットIDとインタセプタを結び付ける。

```
<!-- 業務オブジェクトの定義。 -->
<bean id="sumService"
  class="jp.terasoluna.sample2.service.impl.SumServiceImpl"/>
</bean>
```

業務ロジック実装者は赤点線と同様の設定を行う

Copyright (c) 2007-2016 NTT DATA Corporation.

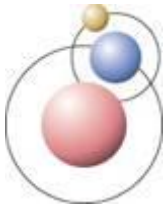


⑫データベースアクセス機能

データベースアクセス機能

- データアクセスオブジェクトのインタフェースを提供する
  - QueryDAO・・・参照系のデータベースアクセスを行うDAOインタフェース
  - UpdateDAO・・・更新系のデータベースアクセスを行うDAOインタフェース
  - StoredProcedureDAO・・・ストアドプロシージャを実行するDAOインタフェース
  - QueryRowHandleDAO・・・参照系のデータベースアクセスの結果を1件ずつ処理するためのDAOインタフェース
- RDBMS製品やO/Rマッピングツールに依存する処理はDAOの実装クラス内に隠蔽する
- DAOのデフォルト実装としてiBatisを用いた実装を提供する
- データベースコネクションのトランザクション管理はサービス層でSpringAOPによる宣言的トランザクション管理を用いるため、開発者が意識する必要はない

Copyright (c) 2007-2016 NTT DATA Corporation.



⑫データベースアクセス機能

◆ サービス層でのデータアクセスの実装

- DAOインスタンスはDIコンテナからビジネスロジックにDIする

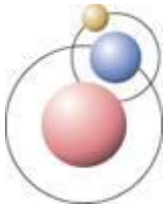
ビジネスロジック

```
public class QueryBLogic {
    private QueryDAO dao = null; // インタフェースの型でDAOを宣言
    ・・・daoのgetter/setterはここでは省略
    public Object dbAccess() {
        SelectUserArgBean bean = new SelectUserArgBean(); //プレースホルダ置換用のPOJO
        ・・・省略
        SelectUserResult result =
            dao.queryForObject("select.user", bean, SelectUserResult.class);
        //第一引数は実行する設定のID(DAOの実装に依存する)
        //第二引数はSQLなどにプレースホルダがある場合の置換文字列を格納したPOJO。不要であればnull。
        //第三引数は戻り値のクラス
        ・・・省略
    }
}
```

Bean定義ファイル

```
<bean id="addBLogic" scope="prototype"
  class="jp.terasoluna.xxx.blogic.AddBLogic" >
  <property name="dao" ref="queryDAO"/>
</bean>
<bean id="queryDAO" class="jp.terasoluna.xxx.MyQueryDAOImpl"/>
```


Copyright (c) 2007-2016 NTT DATA Corporation.



⑫データベースアクセス機能

- iBatisを用いたDAOのデフォルト実装
  - iBatisの機能を利用した以下のクラスを提供する
    - QueryDAOiBatisImpl
    - UpdateDAOiBatisImpl
    - StoredProcedureDAOiBaitsImpl
    - QueryRowHandleDAOImpl

Copyright (c) 2007-2016 NTT DATA Corporation.

⑫データベースアクセス機能

- 設定
  - Bean定義ファイル・・・データソース、sqlMapClientの設定を行う

Bean定義ファイル

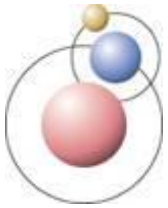
```
<!-- データソースの設定 -->
<bean id="TerasolunaDataSource"
      class="org.springframework.jdbc.datasource.DriverManagerDataSource"
      destroy-method="close">
  <property name="driverClassName" value="jdbc.driverClassName=oracle.jdbc.OracleDriver"/>
  <property name="url" value="jdbc:oracle:thin:@hostname:1521:sid "/>
  <property name="username" value="oracle"/>
  <property name="password" value="password"/>
</bean>

<!-- iBATIS定義 -->
<bean id="sqlMapClient"
      class="org.springframework.orm.ibatis.SqlMapClientFactoryBean">
  <property name="configLocation" value="WEB-INF/sqlMapConfig.xml"/>
  <property name="dataSource" ref="TerasolunaDataSource"/>
</bean>
```

sqlMapClient

iBatis設定ファイルへのパス

データソースの設定

Copyright (c) 2007-2016 NTT DATA Corporation.



⑫データベースアクセス機能

- 設定
  - Bean定義ファイル(続き)・・・DAOの設定を行う

Bean定義ファイル

```
<!-- DAO定義 -->
<bean id="queryDAO"
        class="jp.terasoluna.fw.dao.ibatis.QueryDAOiBatisImpl">
    <property name="sqlMapClient" ref="sqlMapClient"/>
</bean>
```

ここで定義したDAOをビジネスロジックにDIする

DAOにsqlMapClientをDIする

（UpdateDAOiBatisImpl、StoredProcedureDAOiBatisImplの場合も同様）

  - sqlMapConfig.xmlの設定
    - » 特別な定義は不要。sqlMap.xmlへのパスだけを設定する。
    - » Bean定義ファイルのsqlMapClientの設定に記述したパスと一致させる

sqlMapClient.xml

```
<sqlMapConfig>
    <sqlMap resource="sqlMap.xml"/>
</sqlMapConfig>
```

データベースアクセスごとの設定はsqlMap.xmlに記述する

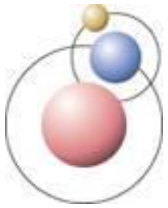
  - sqlMap.xmlの設定
    - » 実行するデータアクセス毎の設定を記述する
    - » iBatisの仕様どおりに記述する

sqlMap.xml

```
<select id="getUser"
        resultClass="jp.terasoluna.xxx.UserBean">
  SELECT ID, NAME, AGE FROM USERLIST WHERE ID = #VALUE#
</select>
```

データアクセスの結果を保持するクラス名を設定する

Copyright (c) 2007-2016 NTT DATA Corporation.

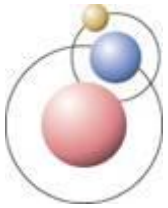


⑫データベースアクセス機能

- 以下の機能はiBatisの機能を用いて提供する。
  - 動的SQL実行
    - where句が条件によって異なる場合

```
<statement id="someName" resultMap="account-result" >
  select * from ACCOUNT
  <dynamic prepend="where">
    <isGreaterThan prepend="and" property="id" compareValue="0">
      ACC_ID = #id#
    </isGreaterThan>
    <isNotNull prepend="and" property="lastName">
      ACC_LAST_NAME = #lastName#
    </isNotNull>
  </dynamic>
 order by ACC_LAST_NAME
</statement>
```


Copyright (c) 2007-2016 NTT DATA Corporation.

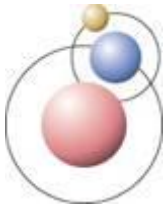


⑫データベースアクセス機能

- 以下の機能はiBatisの機能を用いて提供する。（続き）
  - 動的SQL実行
    - In句の?やORの数が配列の要素数により不特定になる場合はiterateを使用

```
1  <!-- prepend: AND や OR などを条件式の先頭に付加する-->
2  <!– open: イテレーションの先頭に付加される文字列 -->
3  <!– conjunction: AND や OR などを各イテレーションの先頭に付加する文字列 -->
4  <iterate prepend=”AND” property=”userNameList”  open=”(” close=”)” conjunction=”OR”>
5    username=#userNameList[]#
6  </iterate>
```


Copyright (c) 2007-2016 NTT DATA Corporation.



⑫データベースアクセス機能

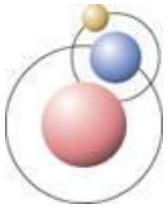
- 以下の機能はiBatisの機能を用いて提供する。（続き）
  - 指定件数取得
    - 取得開始インデックスと取得件数を指定する。
    - ページ遷移のある一覧表示などに利用可能

```
// daoはQueryDAOインタフェース型のフィールド。
// daoのフィールド定義とsetter/getterが別途あるがここでは省略
List<UserBean> bean
        = dao.executeForObjectList(
             "getUser",null, 20, 10);
// または
UserBean[] bean
        = dao.executeForObjectArray(
             "getUser",null, UserBean.class, 20, 10);
```

      - 上記の例ではSQLの条件に一致するデータの21件目から10件を取得する。

※java.sql.ResultSet#next()を利用して指定件数になるまでループするため性能劣化の原因になりうる。性能劣化が著しい場合はSQLチューニングを行う。

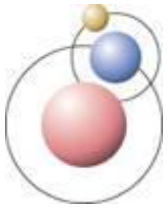
Copyright (c) 2007-2016 NTT DATA Corporation.



⑫データベースアクセス機能

- 以下の機能はiBatisの機能を用いて提供する。（続き）
  - データキャッシュ機能
    - キャッシュの有効時間、指定SQL実行タイミングでのキャッシュデータフラッシュが可能
      - » 時間指定が無い場合は永続的にデータを保存
    - 結果データはstaticなHashMapに格納される（Collectionsクラスにより同期化されている）
      - » VMに一つの格納領域となるため、クラスタ環境では指定SQL実行でのデータフラッシュは使用が困難
    - キャッシュタイプは4タイプある
      - » MEMORY：Weak（java.lang.ref.WeakReference）,Soft（java.lang.ref.SoftReference）, Strong（iBatis独自のstaticクラス）を子要素で設定する。Strongは明示的にキャッシュを消さない限り生存しつづける
      - » LRU：最近、最も使われていないものから消されていく
      - » FIFO：First In First Out
      - » OSCACHE：OpenSymphonyプロジェクトのOSCacheを利用してのキャッシュ

Copyright (c) 2007-2016 NTT DATA Corporation.



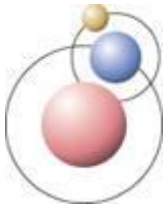
⑫データベースアクセス機能

- 以下の機能はiBatisの機能を用いて提供する。（続き）
  - データキャッシュ機能

【sql-config.xml ファイル】

```
<!－24時間キャッシュし、insert,update,delete でデータを更新
<  cacheModel id="product cache" type="LRU" readOnly=”true” serialize=”false”>
   <flushInterval hours="24"/>
    <flushOnExecute statement="insertProduct"/>
    <flushOnExecute statement="updateProduct"/>
    <flushOnExecute statement="deleteProduct"/>
  <property name=”cache-size” value=”1000” />
</cacheModel>
<statement id=”getProductList” cacheModel=”product-cache”>
    select * from PRODUCT where PRD_CAT_ID = #value#
</statement>
```


Copyright (c) 2007-2016 NTT DATA Corporation.

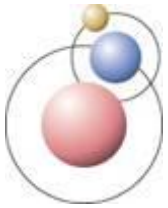


⑬画面表示(カスタムタグ)機能

terasoluna-thinで提供するカスタムタグ

- アクセス権限チェック機能
  - <t:ifAuthorized>
  - <t:ifAuthorizedBlock>
- サーバ閉塞チェック機能
  - <t:ifPreBlockade>
- カレンダー入力機能
  - <t:inputCalendar>
- 文字列表示機能
  - <t:write>
- 日付変換機能
  - <t:date>
- 和暦日付変換機能
  - <t:jdate>
- Decimal表示機能
  - <t:decimal>
- トリム機能
  - <t:rtrim>
  - <t:ltrim>
  - <t:trim>
- 文字列切り取り機能
  - <t:left>
- コードリスト定義機能
  - <t:defineCodeList>
- コードリスト件数出力機能
  - <t:writeCodeCount>
- 指定コードリスト値表示機能
  - <t:writeCodeValue>

Copyright (c) 2007-2016 NTT DATA Corporation.



⑬画面表示(カスタムタグ)機能

- terasoluna-thinで提供するカスタムタグ
  - スタイルクラス切り替え機能
    - <ts:changeStyleClass>
  - メッセージ表示機能
    - <ts:errors>
    - <ts:messages>
  - キャッシュ避けformタグ機能
    - <ts:form>
  - キャッシュ避けリンク機能
    - <ts:link>
  - フォームターゲット指定機能
    - <ts:submit>
  - メッセージポップアップ機能
    - <ts:messagesPopup>
    - <ts:body>
  - エラーメッセージチェック機能
    - <ts:ifErrors>
    - <ts:ifNotErrors>
  - クライアントチェック拡張機能
    - <ts:javascript>
  - 一覧表示関連機能
    - 一覧表示方法
      - <logic:iterate>
    - ページリンク機能
      - <ts:pageLinks>

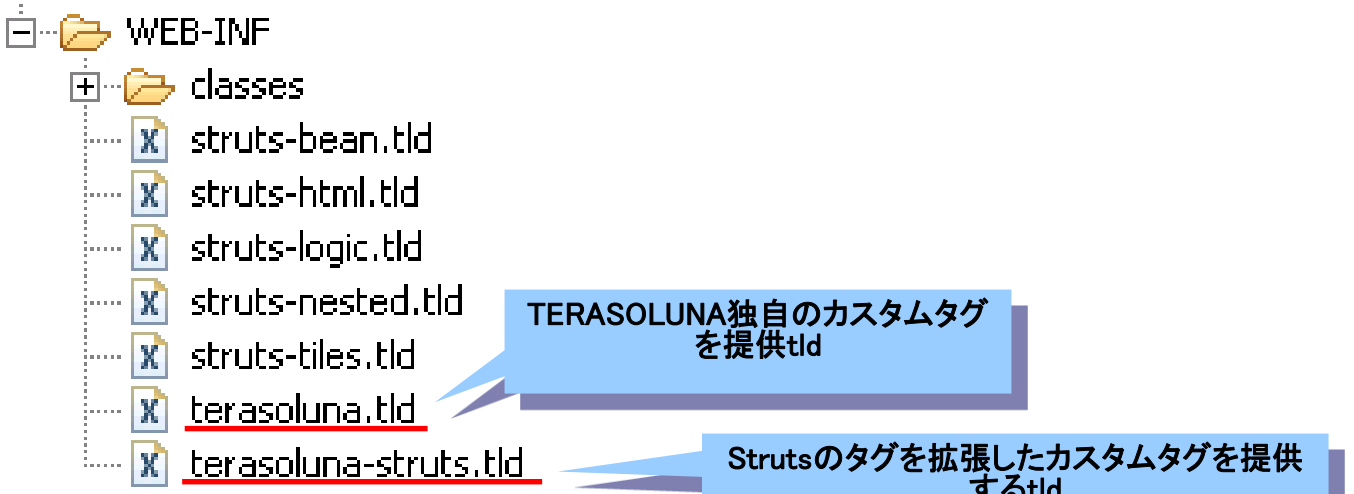
Copyright (c) 2007-2016 NTT DATA Corporation.

⑬画面表示(カスタムタグ)機能 ～カスタムタグの設定

◆ タグの設定

- フレームワークで提供するカスタムタグ
  - TERASOLUNA独自のカスタムタグ
  - Struts提供のタグを拡張したもの

の二つの種類に分かれている。設定ファイル(tld)が別ファイルなので、使用する場合はそれぞれのtldファイルを用意する。

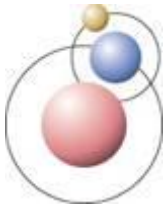


Copyright (c) 2007-2016 NTT DATA Corporation.

⑬画面表示(カスタムタグ)機能 ～アクセス権限チェック機能(1/3)

- 概要
  - 現在のログオンユーザが指定されたパスへのアクセス権限をもっているかどうかを判別して表示/非表示を切り替える。
- 解説
  - <t:ifAuthorized>
    - path属性で指定されたパスに対してアクセス権がある場合にのみ、タグのボディ部分を評価する。アクセス権のチェックは、AuthorizationControllerが行う。
  - <t:ifAuthorizedBlock>
    - <t:ifAuthorized>要素の結果をblockId毎に制御する為の要素で、blockIdで<t:ifAuthorized>要素と紐付けられ、ボディ内を表示するかどうかを判定する。
      - この要素を入れ子状にすることで、アクセス権限毎の制御を柔軟に行なう事が可能になる。入れ子状にする場合は、親要素のblockId属性と子要素のparentBlockId属性が紐付けられ、ボディ内を表示するかどうかを判定する。

Copyright (c) 2007-2016 NTT DATA Corporation.



⑬画面表示(カスタムタグ)機能 ～アクセス権限チェック機能(2/3)

◆ <t:ifAuthorized>要素の使用例

```
<t:ifAuthorized path="/pathToSomewhere">
     指定されたパスへのアクセス権限がある場合、出力される。
</t:ifAuthorized>
```

◆ <t:ifAuthorizedBlock>要素の使用例

```
<t:ifAthorizedBlock blockId="ABC" >
     ボディ内の blockId で紐付けられた
     IfAuthorizedBlockTag が表示される場合のみ表示される。
     <t:ifAthorizedBlock blockId="EFG" parentBlockId="ABC" >
      ボディ内の blockId で紐付けられた IfAuthorizedTag
      が表示される場合のみ表示される。
         <t:ifAthorized path="/sample1/test.do blockId="EFG" >
              指定されたパスへのアクセス権限がある場合、出力される。
         </t:ifAthorized>
     </t:ifAthorizedBlock>
</t:ifAthorizedBlock>
```

※AuthorizationControllerの設定は①認証・アクセス制限機能の頁を参照

Copyright (c) 2007-2016 NTT DATA Corporation.

⑬画面表示(カスタムタグ)機能 ～アクセス権限チェック機能(3/3)

◆ <t:ifAuthorized>要素の属性一覧

属性	必須	概要
path	○	対象となるpathを指定する。
blockId	-	このタグの親となる<htmlx:IfAuthorizedBlock>要素と紐付ける為のblockIdを指定する。

◆ <t:ifAuthorizedBlock>要素の属性一覧

属性	必須	概要
blockId	○	対象となるblockIdを指定する。
parentBlockId	-	このタグの親となる<htmlx:IfAuthorizedBlock>要素と紐付ける為のblockIdを指定する。

Copyright (c) 2007-2016 NTT DATA Corporation.

⑬画面表示(カスタムタグ)機能 ～サーバ閉塞チェック機能

- 概要
  - サーバが予閉塞または閉塞状態かどうかを判別して表示/非表示を切り替える。
- 解説
  - <t:ifPreBlockade>
    - サーバが閉塞状態又は予閉塞状態の場合にのみ、タグのボディ部分を出力する。サーバ閉塞のチェックは、ServerBlockageControllerが行う。
  - 使用例

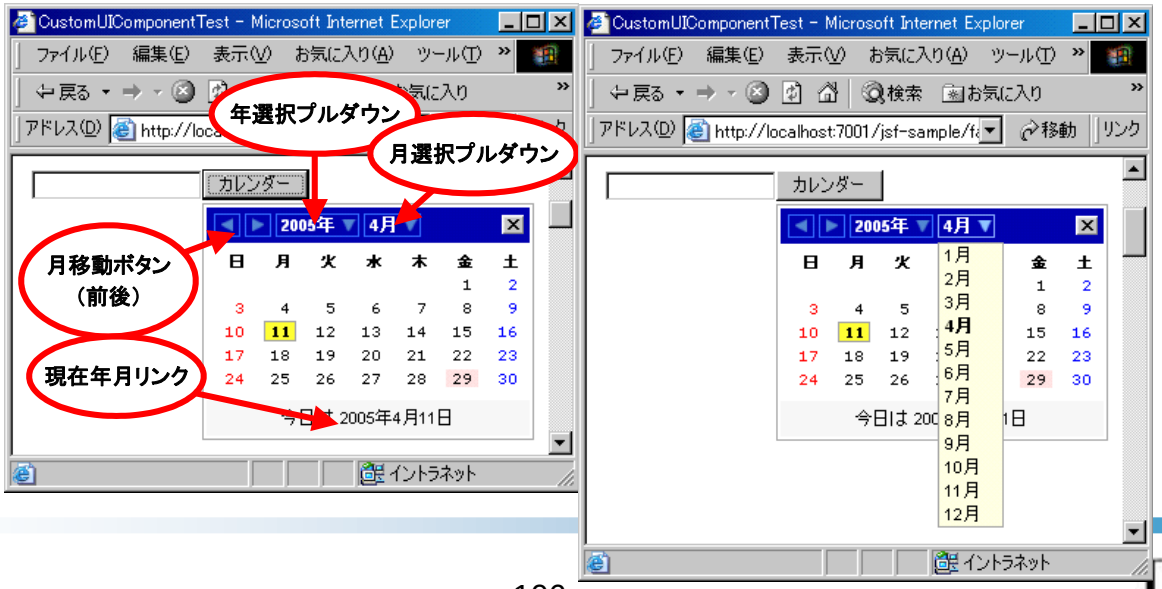
```
<t:ifPreBlockade>
    ... // サーバが閉塞状態又は予閉塞状態の場合にのみの表示項目等
</t:ifPreBlockade>
```

※AuthorizationControllerの設定は①認証・アクセス制限機能の頁を参照

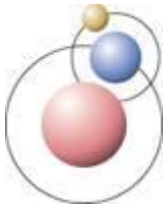
Copyright (c) 2007-2016 NTT DATA Corporation.

⑬画面表示(カスタムタグ)機能 ～カレンダー入力機能(1/6)

- 概要
  - 指定されたテキストフィールドに対して、カレンダー入力機能を提供する。
- 解説
  - <t:inputCalendar>
    - JavaScriptを使用したカレンダー画面を表示し選択した日付を、指定された入力フィールドに入力する。



Copyright (c) 2007-2016 NTT DATA Corporation.



⑬画面表示(カスタムタグ)機能 ～カレンダー入力機能(2/6)

- 機能詳細
  - 設定ファイル
    - 国際化対応のため、メッセージリソースファイルを設定ファイルとして利用する
    - メッセージリソースファイルはクラスパス直下に「calendar.properties」というファイル名で作成する。
  - 休日定義
    - メッセージリソースファイルに以下のように記述することでカレンダーの休日を指定することができる。

```
calendar.holiday.1=0,1,1,元旦
calendar.holiday.2=0,2,11,建国記念日
calendar.holiday.3=0,4,29,みどりの日
calendar.holiday.4=2005,1,10,成人の日
calendar.holiday.5=2005,3,20,春分の日
calendar.holiday.6=2005,9,19,敬老の日
calendar.holiday.7=2005,9,23,秋分の日
calendar.holiday.8=2005,10,10,体育の日
```

メッセージリソースキーは、「calendar.holiday.」部分を固定として、 その後に、「1」から連番を振ることとする。

パラメータは、「年」「月」「日」「休日概要」を「,(カンマ)」で 区切って記述することとする。

毎年同じ日付の休日定義は「年」を「0」と 指定することで毎年と認識する。

Copyright (c) 2007-2016 NTT DATA Corporation.

⑬画面表示(カスタムタグ)機能 ～カレンダー入力機能(3/6)

◆ ボタン表示文字列変更

- メッセージリソースファイルに以下のように記述することでカレンダーを表示するボタンの文字列を変更することができる。
- メッセージリソースキーは、「calendar.button.string」固定とする。デフォルトは「Calendar」となる。

```
calendar.button.string=カレンダー
```

◆ スタイルプレフィックス変更

- メッセージリソースファイルに以下のように記述することでカレンダーにて使用するスタイルシートのプレフィックス、および画像ファイルのプレフィックスを変更することができる。
- メッセージリソースキーは、「calendar.style.themeprefix」固定とする。デフォルトは「BlueStyle」となる。

```
calendar.style.themeprefix=WhiteStyle
```


Copyright (c) 2007-2016 NTT DATA Corporation.

⑬画面表示(カスタムタグ)機能 ～カレンダー入力機能(4/6)

◆ 現在日付表示文字列変更

- メッセージリソースファイルに以下のように記述することでカレンダーの下部に表示される現在日付に付与する文字列を 変更することができる。
- メッセージリソースキーは、「calendar.today.string」固定とする。デフォルトは「Today is 」となる。

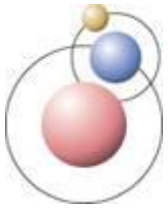
```
calendar.today.string=Today is
```

◆ カレンダー画像保存場所変更

- メッセージリソースファイルに以下のように記述することでカレンダー入力機能にて使用する画像の保存場所を変更することができる。
- 最後は「/」で終わる必要がある。画像の保存場所は変更可能だが、画像ファイルの名前は変更することができない。
- メッセージリソースキーは、「calendar.img.dir」固定とする。デフォルトは「img/calendar/」となる。

```
calendar.img.dir=image/
```


Copyright (c) 2007-2016 NTT DATA Corporation.



⑬画面表示(カスタムタグ)機能 ～カレンダー入力機能(5/6)

◆ スタイルシート保存場所変更

- メッセージリソースファイルに以下のように記述することでカレンダー入力機能にて使用するスタイルシートの保存場所を 変更することができる。最後は「/」で終わる必要がある。
- この機能で使用するスタイルシートのファイル名は、「<プレフィックス> + InputCalendar.css」である。
- メッセージリソースキーは、「calendar.stylesheet.dir」固定とする。デフォルトは「css/」となる。

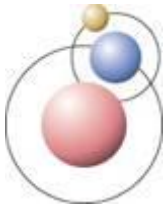
```
calendar.stylesheet.dir=stylesheet/
```

◆ 外部JavaScriptファイル保存場所変更

- メッセージリソースファイルに以下のように記述することでカレンダー入力機能にて使用する外部JavaScriptの保存場所を 変更することができる。最後は「/」で終わる必要がある。
- この機能で使用するJavaScriptのファイル名は、「InputCalendar.js」である。
- メッセージリソースキーは、「calendar.javascript.dir」固定とする。デフォルトは「js/」となる。

```
calendar.javascript.dir=javascript/
```


Copyright (c) 2007-2016 NTT DATA Corporation.



⑬画面表示(カスタムタグ)機能 ～カレンダー入力機能(6/6)

◆ 使用例

```
<html:text name="_dynamicForm" property="date1" maxlength="10" size="15" />
<t:inputCalendar for="date1" format="yyyy/MM/dd" />
```

◆ 属性一覧

属性	必須	概要
for	○	選択した日付を入力する入力フィールドを指定する。
format	-	カレンダーのフォーマットを指定する。 指定できる日付形式は「y(年)」「M(月)」「d(日)」、区切文字としては「/」「-」「.」「半角スペース」のいずれかである。また、区切り文字は、一文字のみを使用すること。「yyyy/MM-dd」のように複数の区切り文字を使用することはできない。
formatKey	-	カレンダーのフォーマットをメッセージリソースから 取得するためのキー値を指定する。

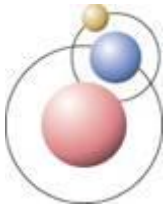
Copyright (c) 2007-2016 NTT DATA Corporation.

⑬画面表示(カスタムタグ)機能 ～文字列表示機能(1/3)

- 概要
  - 指定したbeanプロパティの値を変換し、表示する。
- 解説
  - <t:write>
    - 指定したbeanプロパティの値を取り出し、Stringとして現在のJspWriter に与える。また、属性により以下の変換を行う。
      - nullもしくは空文字を“ ”に置換
      - 半角スペースを“ ”に置換
      - 改行コードを
 に置換
      - 改行文字を無視
  - 使用例

```
primaryField = " あいえうお¥n かきくけこ"
<t:write name="htmlxForm"
                property="primaryField" />
↓出力結果
 あいうえお<br> かきくけこ
```


Copyright (c) 2007-2016 NTT DATA Corporation.

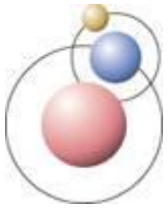


⑬画面表示(カスタムタグ)機能 ～文字列表示機能(2/3)

◆ 属性一覧

属性	必須	概要
filter	-	この属性がtrueにセットされる場合、表現されたプロパティ値は HTML内でセンシティブな文字のために フィルターされる。そしてこのような全ての文字は、等価な文字で置き換えられる。デフォルトでは、フィルタリングが行われる。無効にするためには、この属性に明示的に false をセットする必要がある。
replaceNullToNbsp	-	この属性がtrueにセットされ、指定したbeanプロパティの値が空文字及び、nullの場合 を出力する。無効にするためには、この属性に明示的に false をセットする必要がある。
replaceSpToNbsp	-	この属性がtrueにセットされ、指定したbeanプロパティの値に1Byteコードのスペースが存在する場合 に置換する。無効にするためには、この属性に明示的に false をセットする必要がある。
replaceLFtoBR	-	この属性がtrueにセットされる場合、指定したbeanプロパティの値の改行コードもしくは復帰文字が に に置換される。無効にするためには、この属性に明示的に false をセットする必要がある。
ignore	-	この属性がtrueにセットされ、name と scope属性で指定した bean が存在しない場合、なにもせずにリターンする。デフォルト値は false (このタグ ライブラリの中のほかのタグと 矛盾しないように実行時例外がスローされる)。

Copyright (c) 2007-2016 NTT DATA Corporation.

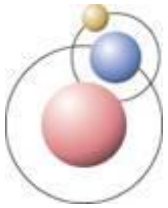


⑬画面表示(カスタムタグ)機能 ～文字列表示機能(3/3)

◆ 属性一覧

属性	必須	概要
name	○	property (指定がある場合) によって指定した値を 取り出すために、プロパティがアクセスされる bean の属性名を指定する。property が指定されない場合、この bean 自身の値が表現される。
property	-	name によって指定した bean 上でアクセスされる プロパティの名前を指定する。この値はシンプル、インデックス付き、またはネストされたプロパティ参照式になる。指定されない場合は、name によって識別された bean は それ自身を表現する。指定したプロパティがヌルを戻す場合、何も表現されない。
scope	-	name によって指定した bean を取り出すために検索された 可変スコープを指定する。指定されない場合、PageContext.findAttribute() によって適用されたデフォルトのルールが適用される。
fillColumn	-	fillColumnによって指定された文字数で区切り、区切った終端に を付与する。文字数の数え方は半角でも、全角でも 1つの文字とみなす。
addBR	-	この属性がtrueにセットされる場合、プロパティ値の末尾に を付与する。デフォルトはfalse。

Copyright (c) 2007-2016 NTT DATA Corporation.

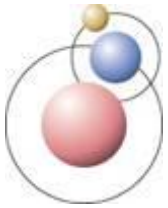


⑬画面表示(カスタムタグ)機能 ～日付変換機能(1/3)

- 概要
  - 指定された形式に従って日付・時刻をフォーマットする。
- 解説
  - <t:date>
    - pattern属性で指定された出力形式の文字列をjava.text.SimpleDateFormatクラスの時刻パターン文字列として解釈し、フォーマットする。時刻パターン文字列の詳細については、java.text.SimpleDateFormatクラスのドキュメントを参照のこと。
  - 使用例

```
<t:date name="form0001"
            property="field001"
            pattern="yyyy/MM/dd hh:mm aaa"/>

↓上記の出力結果

2005/7/14 11:24 PM
```


Copyright (c) 2007-2016 NTT DATA Corporation.

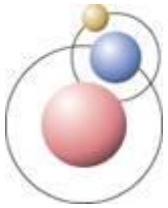


⑬画面表示(カスタムタグ)機能 ～日付変換機能(2/3)

◆ 属性一覧

属性	必須	概要
id	-	フォーマットされた文字列をレスポンスへ出力せずに、スクリプティング変数にセットする際に指定する。フォーマットされた文字列をスクリプティング変数にセットする場合には、filter 属性の指定に関わらずHTML 特殊文字はエスケープされない。
filter	-	フォーマットされた文字列を出力する際に、HTML特殊文字を エスケープするかどうかを指定する。ただし、id 属性が 指定されていた場合には、無視される。
ignore	-	name 属性で指定した beanが 見つからなかったときに無視するかどうかを指定する。false を指定すると、beanが見つからなかったときに JspException が投げられる。
name	-	フォーマット対象の文字列をプロパティに持つbeanの名前。property属性が指定されていなかったときは、name属性で指定されたインスタンスがフォーマットの対象となる。この場合、そのインスタンス自身がjava.util.Date型であるか、あるいはjava.lang.String型(かつyyyy/MM/dd hh:mm:ssの形式となっているもの)のどちらかである必要がある。value属性が指定されていた場合は無視される。
property	-	name 属性で指定された bean においてアクセスされるプロパティの名前。value 属性が 指定されていた場合には無視される。
scope	-	name 属性で指定された bean を検索する際のスコープ。
value	-	フォーマットする文字列。文字列は、format属性で指定した形式となっている必要がある。 value 属性を指定した場合には、name 属性、および property 属性は無視される。

Copyright (c) 2007-2016 NTT DATA Corporation.



⑬画面表示(カスタムタグ)機能 ～日付変換機能(3/3)

◆ 属性一覧

属性	必須	概要
pattern	-	フォーマットする出力形式。pattern 属性で指定した出力形式は、DateFormatterTagBase クラスのサブクラスで解釈される。詳細は、サブクラスのドキュメントを 参照のこと。
patternKey	-	フォーマットのパターンをメッセージリソースのキーで指定する。pattern属性に値が記述されている場合は、pattern属性が優先となる。
format	-	value属性で日付を指定する際の日付時刻のフォーマット。デフォルト値は“yyyy/MM/dd HH:mm:ss”

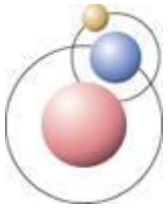
Copyright (c) 2007-2016 NTT DATA Corporation.

⑬画面表示(カスタムタグ)機能 ～和暦日付変換機能(1/4)

- 概要
  - 日付時刻データを和暦としてフォーマットする。
- 解説
  - <t:jdate>
    - 日付時刻のデータのフォーマットを行う際に、和暦の元号(“昭和”、“S”など)や、和暦の年(西暦の “2002” 年ではなく、平成“14”年 など)、および曜日の日本語表記(“月曜日”、“月” など)に変換する。
  - 使用例

```
<t:jdate name="form0001"
            property="field001"
            pattern="GGGGyy年MM月dd日(EEEE) hh時mm分ss秒"/>

↓上記の出力結果

平成17年07月14日(木曜日) 11時24分31秒
```


Copyright (c) 2007-2016 NTT DATA Corporation.



⑬画面表示(カスタムタグ)機能 ～和暦日付変換機能(2/4)

◆ 和暦の設定

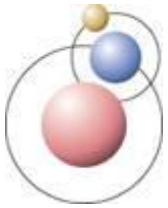
- 和暦のデータはDateUtilクラス経由でプロパティファイルから取得する。
- 以下の形式でプロパティファイルに設定する

```
wareki.gengo.ID.name=元号名
wareki.gengo.ID.roman=元号のローマ字表記
wareki.gengo.ID.startDate=元号法施行日(西暦:yyyy/MM/dd形式)
```

- 以下は一般的な設定例である

```
wareki.gengo.0.name = 平成
wareki.gengo.0.roman = H
wareki.gengo.0.startDate = 1989/01/08
wareki.gengo.1.name = 昭和
wareki.gengo.1.roman = S
wareki.gengo.1.startDate = 1926/12/25
wareki.gengo.2.name = 大正
wareki.gengo.2.roman = T
wareki.gengo.2.startDate = 1912/07/30
wareki.gengo.3.name = 明治
wareki.gengo.3.roman = M
wareki.gengo.3.startDate = 1868/09/04
```


Copyright (c) 2007-2016 NTT DATA Corporation.



⑬画面表示(カスタムタグ)機能 ～和暦日付変換機能(3/4)

◆ 属性一覧

属性	必須	概要
id	-	フォーマットされた文字列をレスポンスへ出力せずに、スクリプティング変数にセットする際に指定する。フォーマットされた文字列をスクリプティング変数にセットする場合には、filter 属性の指定に関わらずHTML 特殊文字はエスケープされない。
filter	-	フォーマットされた文字列を出力する際に、HTML特殊文字を エスケープするかどうかを指定する。ただし、id 属性が 指定されていた場合には、無視される。
ignore	-	name 属性で指定した beanが 見つからなかったときに無視するかどうかを指定する。false を指定すると、beanが見つからなかったときに JspException が投げられる。
name	-	フォーマット対象の文字列をプロパティに持つbeanの名前。property属性が指定されていなかったときは、name属性で指定されたインスタンスがフォーマットの対象となる。この場合、そのインスタンス自身がjava.util.Date型であるか、あるいはjava.lang.String型(かつ"yyyy/MM/dd hh:mm:ss"の形式となっているもの)のどちらかである必要がある。value属性が指定されていた場合は無視される。
property	-	name 属性で指定された bean においてアクセスされるプロパティの名前。value 属性が 指定されていた場合には無視される。
scope	-	name 属性で指定された bean を検索する際のスコープ。
value	-	フォーマットする文字列。文字列は、format属性で指定した形式となっている必要がある。value 属性を指定した場合には、name 属性、および property 属性は無視される。
pattern	-	フォーマットする出力形式。pattern 属性で指定した出力形式は、DateFormatterTagBase クラスのサブクラスで解釈される。詳細は、サブクラスのドキュメントを 参照のこと。

Copyright (c) 2007-2016 NTT DATA Corporation.

⑬画面表示(カスタムタグ)機能 ～和暦日付変換機能(4/4)

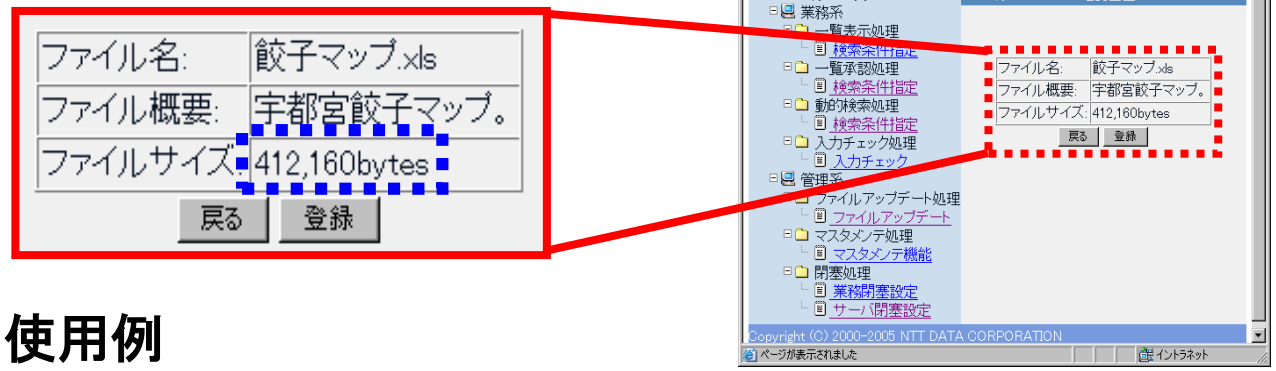
◆ 属性一覧

属性	必須	概要
format	-	value属性で日付を指定する際の日付時刻のフォーマット。デフォルト値は“yyyy/MM/dd HH:mm:ss”

Copyright (c) 2007-2016 NTT DATA Corporation.

⑬画面表示(カスタムタグ)機能 ～Decimal表示機能(1/3)

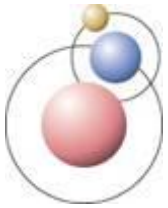
- 概要
  - 符号、および小数点付き数値をフォーマットして出力、あるいはスクリプティング変数として定義する。
- 解説
  - <t:decimal>



  - 使用例

```
<td nowrap>ファイルサイズ:</td>
<td nowrap>
  <t:decimal name="_fileForm" property="fileSize" pattern="###,###bytes"/>
</td>
```


Copyright (c) 2007-2016 NTT DATA Corporation.



⑬画面表示(カスタムタグ)機能 ～Decimal表示機能(2/3)

◆ 属性一覧

属性	必須	概要
id	-	フォーマットされた文字列をレスポンスへ出力せずに、スクリプティング変数にセットする際に指定する。フォーマットされた文字列をスクリプティング変数にセットする場合には、filter 属性の指定に関わらずHTML 特殊文字はエスケープされない。
filter	-	フォーマットされた文字列を出力する際に、HTML特殊文字を エスケープするかどうかを指定する。ただし、id 属性が 指定されていた場合には、無視される。
ignore	-	name 属性で指定した beanが 見つからなかったときに無視するかどうかを指定する。false を 指定すると、beanが見つからなかったときに JspException が投げられる。
name	-	フォーマット対象の文字列をプロパティに持つbeanの名前。property 属性が指定されていなかったときには、name 属性で指定されたインスタンス がフォーマットの対象となる。この場合は、そのインスタンス自身が java.math.BigDecimal 型であるか、あるいは java.lang.String 型(かつ右側の空白除去後に BigDecimal のコンストラクタによって解釈可能であるもの) のどちらかである必要がある。value 属性が指定されていた場合には、無視される。
property	-	name 属性で指定された bean においてアクセスされるプロパティの名前。value 属性が 指定されていた場合には無視される。

Copyright (c) 2007-2016 NTT DATA Corporation.

⑬画面表示(カスタムタグ)機能 ～Decimal表示機能(3/3)

◆ 属性一覧

属性	必須	概要
scope	-	name 属性で指定されたbean を検索する際のスコープ。
value	-	フォーマットする文字列。文字列は、右側の空白除去後に BigDecimal のコンストラクタによって解釈可能である必要がある。value 属性を指定した場合には、name 属性、および property 属性は無視される。
pattern	○	フォーマットする出力形式。pattern 属性で指定した出力形式は、DecimalFormat クラスのパターンとして解釈される。詳細は、DecimalFormat クラスのドキュメントを参照のこと。
scale	-	丸め動作後の小数点以下桁数。n を指定した場合には、小数第 n + 1 位が丸められる。 丸めモードはround属性で指定する。round 属性が指定されていない場合は、四捨五入が行われる。
round	-	丸めモード。scale属性が 指定されている時、有効になる。ROUND_HALF_UP(四捨五入)、ROUND_FLOOR(切り捨て)、ROUND_CEILING (切り上げ)が 設定可能である。デフォルトはROUND_HALF_UP が実行される。これら3つの設定以外を指定した場合は、 IllegalArgumentExceptionがスローされる。

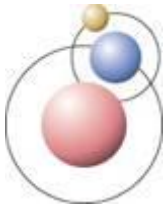
Copyright (c) 2007-2016 NTT DATA Corporation.

⑬画面表示(カスタムタグ)機能 ～トリム機能(1/2)

- 概要
  - 指定された文字列のスペースを削除する。
- 解説
  - <t:rtrim>
    - 文字列の右側のスペースを削除する。
    - 例:“_文字列_” ⇒削除⇒ “_文字列”
  - <t:ltrim>
    - 文字列の左側のスペースを削除する。
    - 例:“_文字列_” ⇒削除⇒ “文字列_”
  - <t:trim>
    - 文字列の左右両側のスペースを削除する。
    - 例:“_文字列_” ⇒削除⇒ “文字列”
  - 使用例

```
<t:rtrim name="form0001" property="field001" />
<t:ltrim name="form0001" property="field002" />
<t:trim  name="form0001" property="field003" />
```


Copyright (c) 2007-2016 NTT DATA Corporation.

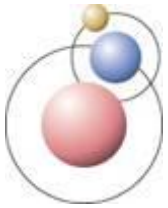


⑬画面表示(カスタムタグ)機能 ～トリム機能(2/2)

- <t:rtrim>、<t:ltrim>、<t:trim>要素の属性一覧

属性	必須	概要
id	-	フォーマットした文字列を出力せずに、スクリプティング変数 にセットする際に指定する。フォーマットされた文字列をスクリプティング 変数にセットする場合には、filter 属性の指定に関わらずHTML 特殊文字はエスケープされない。
filter	-	フォーマットされた文字列を出力する際に、HTML 特殊文字を エスケープするかどうかを指定する。ただし、id 属性が 指定されていた場合には、無視される。
ignore	-	name 属性で指定した beanが 見つからなかったときに無視するかどうかを指定する。false を 指定すると、bean が見つからなかったときに JspExceptionが投げられる。
name	-	フォーマット対象の文字列をプロパティに持つ bean の名前。property 属性が指定されて いなかったときには、name 属性で指定されたインスタンスの 文字列表現 toString() メソッドで返される文字列) がフォーマットの対象となる。value 属性が指定されていた場合には、無視される。
property	-	name 属性で指定された bean においてアクセスされるプロパティの名前。value 属性が 指定されていた場合には無視される。
scope	-	name 属性で指定された bean を検索する際のスコープ。
value	-	フォーマットする文字列。value 属性を 指定した場合には、name 属性、および property 属性は無視される。
replaceSpToNbsp	-	この属性がtrueにセットされ、 指定したbeanプロパティの値に1Byteコードのスペースが存在する場合 に置換する。 無効にするためにはこの属性に明示的に false をセットする必要がある。 ただし、id 属性が指定されていた場合には、無視される。

Copyright (c) 2007-2016 NTT DATA Corporation.

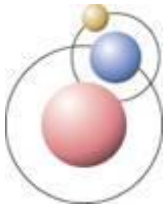


⑬画面表示(カスタムタグ)機能 ～文字列切り取り機能(1/2)

- 概要
  - 文字列の左端から指定された文字数分の文字列を切り出す。
- 解説
  - <t:left>
    - StringUtilクラスのsubstring()メソッドによって、文字列の左端から指定された文字数を切り出す。
  - 使用例

```
form0001.field001 = "Java write once, Run anywhere."

<t:left name="form0001"
            property="field001"
            length="10" />

↓上記の出力結果

"Java write"
```


Copyright (c) 2007-2016 NTT DATA Corporation.



⑬画面表示(カスタムタグ)機能 ～文字列切り取り機能(2/2)

◆ 属性一覧

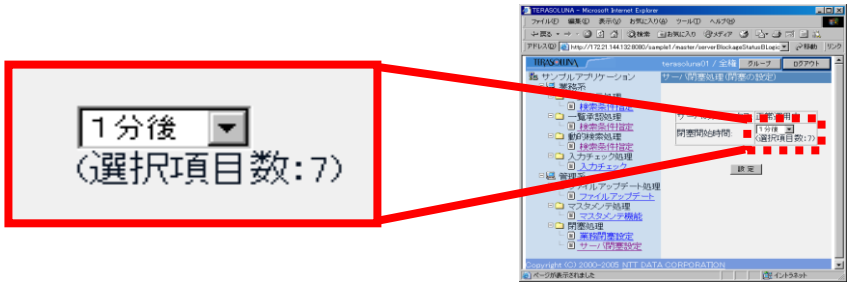
属性	必須	概要
id	-	フォーマットした文字列を出力せずに、スクリプティング変数 にセットする際に指定する。フォーマットされた文字列をスクリプティング 変数にセットする場合には、filter 属性の指定に関わらずHTML 特殊文字はエスケープされない。
filter	-	フォーマットされた文字列を出力する際に、HTML 特殊文字を エスケープするかどうかを指定する。ただし、id 属性が 指定されていた場合には、無視される。
ignore	-	name 属性で指定した beanが 見つからなかったときに無視するかどうかを指定する。false を 指定すると、bean が見つからなかったときに JspExceptionが投げられる。
name	-	フォーマット対象の文字列をプロパティに持つ bean の名前。property 属性が指定されて いなかったときには、name 属性で指定されたインスタンスの 文字列表現 toString() メソッドで返される文字列) がフォーマットの対象となる。value 属性が指定されていた場合には、無視される。
property	-	name 属性で指定された bean においてアクセスされるプロパティの名前。value 属性が 指定されていた場合には無視される。
scope	-	name 属性で指定された bean を検索する際のスコープ。
value	-	フォーマットする文字列。value 属性を 指定した場合には、name 属性、および property 属性は無視される。
replaceSpToNbsp	-	この属性がtrueにセットされ、 指定したbeanプロパティの値に1Byteコードのスペースが存在する場合 に置換する。 無効にするためにはこの属性に明示的に false をセットする必要がある。 ただし、id 属性が指定されていた場合には、無視される。

Copyright (c) 2007-2016 NTT DATA Corporation.

TERASOLUNA

⑬画面表示(カスタムタグ)機能 ～コードリスト定義機能(1/2)

- 概要
  - 読み込み済みのコードリストを、jsp内で利用する。
- 解説
  - <t:defineCodeList>
    - 指定したidのCodeListLoaderからコードリストを取得し、ページ属性のBeanとして定義する。

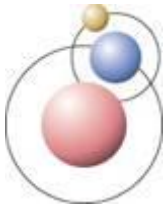


  - 使用例

```
<t:defineCodeList id="loader"/>
<html:select property="server_blockage_time">
  <html:options collection="loader" property="id" labelProperty="name" />
</html:select>
```

※コードリストの設定は③コードリスト機能を参照

Copyright (c) 2007-2016 NTT DATA Corporation.



⑬画面表示(カスタムタグ)機能 ～コードリスト定義機能(2/2)

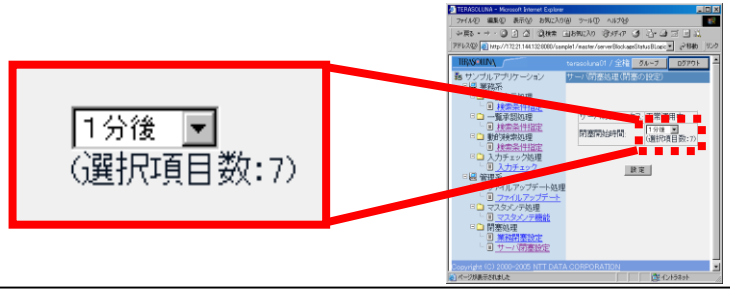
◆ 属性一覧

属性	必須	概要
id	○	この属性からコードリストを検索する。このタグ宣言以降、<logic:iterator>タグ、<html:options>タグなどでコードリストが参照できる。

Copyright (c) 2007-2016 NTT DATA Corporation.

⑬画面表示(カスタムタグ)機能 ～コードリスト件数出力機能(1/2)

- 概要
  - コードリストの要素数を出力する。
- 解説
  - <t:writeCodeCount>
    - 指定したidのCodeListLoaderからコードリストの要素数を取得する。

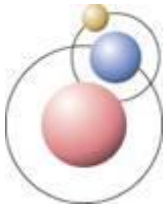


  - 使用例

```
<html:select property="server_blockage_time">
  <html:options collection="SQL_C0901" property="id" labelProperty="name" />
</html:select><br>
(選択項目数:<t:writeCodeCount id="loader"/>)
```

※コードリストの設定は③コードリスト機能を参照

Copyright (c) 2007-2016 NTT DATA Corporation.



⑬画面表示(カスタムタグ)機能 ～コードリスト件数出力機能(2/2)

◆ 属性一覧

属性	必須	概要
id	○	この属性で参照されるコードリストの要素数を出力する。コードリストが見つからない場合、0が返却される。

Copyright (c) 2007-2016 NTT DATA Corporation.

⑬画面表示(カスタムタグ)機能 ～指定コードリスト値表示機能(1/2)

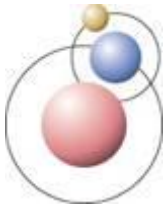
- 概要
  - 指定した値のコードリストの表示名を画面に出力する。
- 解説
  - <t:writeCodeValue>
    - CodeListLoaderのidと表示したいコード値を指定することで画面にコードリストの表示名を表示する。

  - 使用例

```
<tr>
  <th>客室タイプ</th>
  <td><t:writeCodeValue codeList="roomTypeCodeList"
        key="roomType01"/></td>
</tr>
```

※コードリストの設定は③コードリスト機能を参照

Copyright (c) 2007-2016 NTT DATA Corporation.



⑬画面表示(カスタムタグ)機能 ～指定コードリスト値表示機能(2/2)

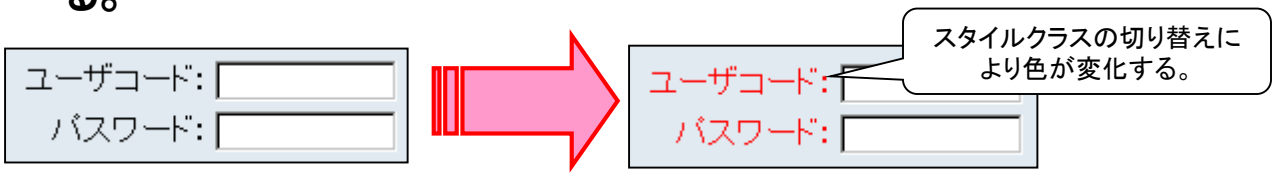
◆ 属性一覧

属性	必須	概要
codeList	○	出力対象のCodeBeanを保持しているCodeListLoaderインスタンスのBeanId。
key	-	取得したコードリストから値を取得するためのコード値を直接指定する。この値を省略した場合は必ず、nameとproperty属性を指定し、コード値を取得するBean名とプロパティ名を指定すること。
name	-	取得したコードリストから値を取得するためのコード値を保持するBeanの名前。key属性が指定されていた場合は、無効。
property	-	取得したコードリストから値を取得するためのコード値を保持するBeanのプロパティ。key属性が指定されていた場合は、無効。
scope	-	取得したコードリストから値を取得するためのコード値を保持するBeanが 存在するスコープ。

Copyright (c) 2007-2016 NTT DATA Corporation.

⑬画面表示(カスタムタグ)機能 ～スタイルクラス切り替え機能(1/2)

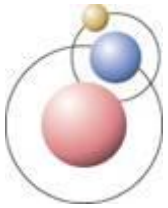
- 概要
  - エラー時に、スタイルシートのクラスを切り替える。
- 解説
  - <ts:changeStyleClass>
    - リクエストおよびセッションに指定されたフィールドに対するエラー情報が設定されているどうかにより、スタイルシートのクラスを切り替える。



  - 使用例

```
<td nowrap class='<ts:changeStyleClass name="userCode"
                                       default="ItemLabel" error="ErrorItem"/>'>
  ユーザコード:
</td>
```


Copyright (c) 2007-2016 NTT DATA Corporation.



⑬画面表示(カスタムタグ)機能 ～スタイルクラス切り替え機能(2/2)

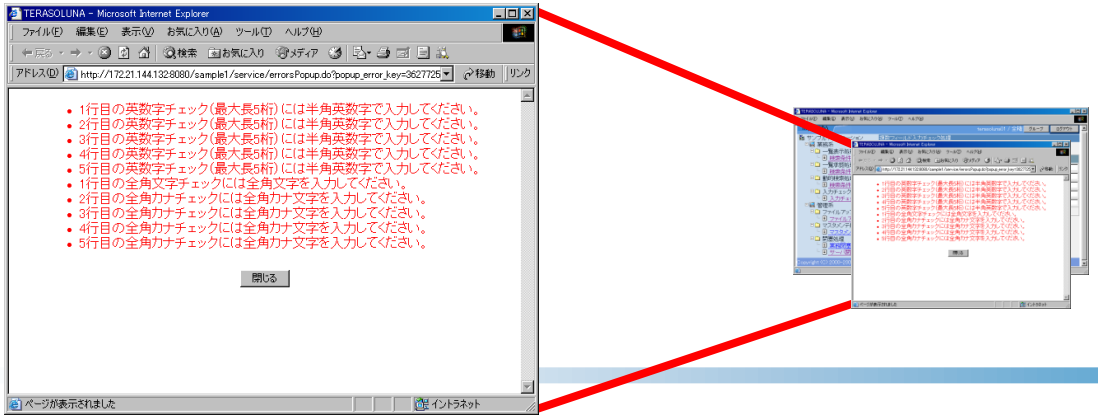
◆ 属性一覧

属性	必須	概要
name	○	エラー情報が設定されているかどうかを判定するフィールド名を指定する。
default	○	エラーがない場合のスタイルシートクラス名を指定する。
error	○	エラーがある場合のスタイルシートクラス名を指定する。

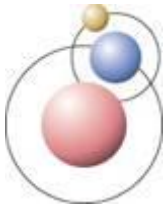
Copyright (c) 2007-2016 NTT DATA Corporation.

⑬画面表示(カスタムタグ)機能 ～メッセージ表示機能(1/5)

- 概要
  - メッセージまたはエラーメッセージ情報を表示する。
- 解説
  - <ts:errors>
    - StrutsのErrorsTagを拡張している。
      - セッションからリクエストに移しかえられたエラー情報の取得・表示を行う。
  - <ts:messages>
    - StrutsのMessageTagを拡張している。
      - セッションからリクエストに移しかえられた情報を取得する。



Copyright (c) 2007-2016 NTT DATA Corporation.



⑬画面表示(カスタムタグ)機能 ～メッセージ表示機能(2/5)

◆ <ts:errors>要素の使用例

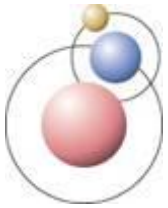
```
<ts:errors/>
<html:button property="forward_action" value="閉じる" onclick="window.close()"/>
```

◆ <ts:messages>要素の使用例

<ts:messages>要素は<ts:errors>要素と違い、メッセージを格納したBeanを定義するだけなので出力部分は実装しなければならない。

```
<ts:messages id="message" message="true">
  <bean:write name="message"/>
</ts:messages>
<html:button property="forward_action" value="閉じる" onclick="window.close()"/>
```


Copyright (c) 2007-2016 NTT DATA Corporation.

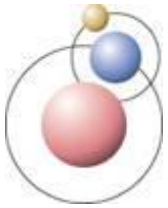


⑬画面表示(カスタムタグ)機能 ～メッセージ表示機能(3/5)

- <ts:errors>要素の属性一覧
  - <html:errors>要素のAPIと同様

- <ts:errors>要素の注意点
  - ポップアップ画面での表示は、このタグが使用されない限り、セッションから情報は削除されない。

Copyright (c) 2007-2016 NTT DATA Corporation.

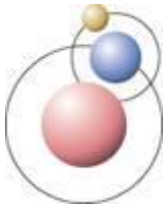


⑬画面表示(カスタムタグ)機能 ～メッセージ表示機能(4/5)

◆ <ts:messages>要素の属性一覧

属性	必須	概要
id	○	メッセージを格納したいBean名を指定する。
bundle	-	メッセージリソース名を指定する。ここを指定しない場合デフォルト のメッセージリソースとなる。
locale	-	出力メッセージのロケールを指定する。 ここを指定しない場合、デフォルトのロケールが使用される。
name	-	表示を行なうアクションメッセージのメッセージキーを個別に指定する。message属性の値をtrueに指定した場合は、必ず、Globals.MESSAGE_KEYが設定される。なお、設定が行なわれていない場合、Globals.ERROR_KEYが設定される。
property	-	表示を行なう(フォーム)プロパティ名を指定する。 ここが指定されない場合、プロパティ名に関わらず、 全てのアクションメッセージが表示される。
header	-	メッセージ本文一覧の前に出力されるヘッダメッセージキーを指定する。
footer	-	メッセージ本文一覧の後に出力されるヘッダメッセージキーを指定する。
message	-	値をtrueに指定したとき、name属性が、 Globals.MESSAGE_KEYとして設定される。

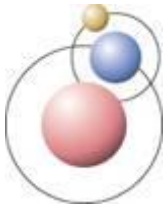
Copyright (c) 2007-2016 NTT DATA Corporation.



⑬画面表示(カスタムタグ)機能 ～メッセージ表示機能(5/5)

- <ts:messages>要素の注意点
  - ポップアップ画面での表示は、このタグが使用されない限り、セッションから情報は削除されない。

Copyright (c) 2007-2016 NTT DATA Corporation.

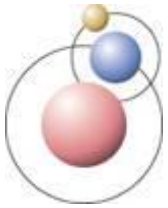


⑬画面表示(カスタムタグ)機能 ～キャッシュ避けformタグ機能(1/2)

- 概要
  - アクションURLにキャッシュ避け用ランダムIDを追加する。
- 解説
  - <ts:form>
    - Struts の提供する<ts:form>要素を拡張する。
    - 機能として、アクション URL にキャッシュ避け用ランダム ID を追加する。
  - 使用例

```
<table border="0">
 <ts:form action="/logonBLogic">
 <div align="center">

↓出力結果

<table border="0">
 <form name="_logonForm" method="post"
        action="/sample1/logon/logonBLogic.do?r=12306310836703916 70">
 <input type="hidden" name="org.apache.struts.taglib.html.TOKEN"
        value="6235eb8a1f477895315e96be95bcc7f2">
 <div align="center">
```

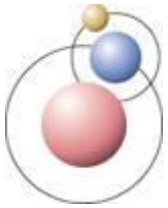

Copyright (c) 2007-2016 NTT DATA Corporation.



⑬画面表示(カスタムタグ)機能 ～キャッシュ避けformタグ機能(2/2)

- 属性一覧
  - <html:form>要素のAPIと同様

Copyright (c) 2007-2016 NTT DATA Corporation.



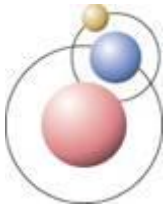
⑬画面表示(カスタムタグ)機能 ～キャッシュ避けリンク機能

- 概要
  - Strutsの提供する<html:link>タグを拡張する。
- 解説
  - <ts:link>
    - アクションURLにキャッシュ避け用ランダムIDを追加する。
  - 使用例
    - <ts:link>要素と同様に使用する。

```
<ts:link href="/hoge.do">href=/hoge.do</ts:link>

↓出力時に、ランダムIDを追加する。

<a href="/hoge.do?r=33365172649972688823">href=/hoge.jsp</a>
```

  - 属性一覧
    - <html:link>要素のAPIと同様

Copyright (c) 2007-2016 NTT DATA Corporation.



⑬画面表示(カスタムタグ)機能 ～フォームターゲット指定機能

- 概要
  - フォームのターゲットを指定する。
- 解説
  - <ts:submit>
    - target属性値を設定することで、フォームのターゲットを指定する。
  - 使用例

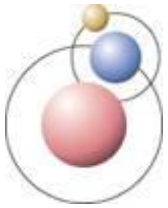
```
<ts:submit value="submit" target="rightFrame"/>
```

  - 属性一覧

属性	必須	概要
target	-	ターゲット先を指定する。

    - その他の属性は<html:submit>と同様である。

Copyright (c) 2007-2016 NTT DATA Corporation.

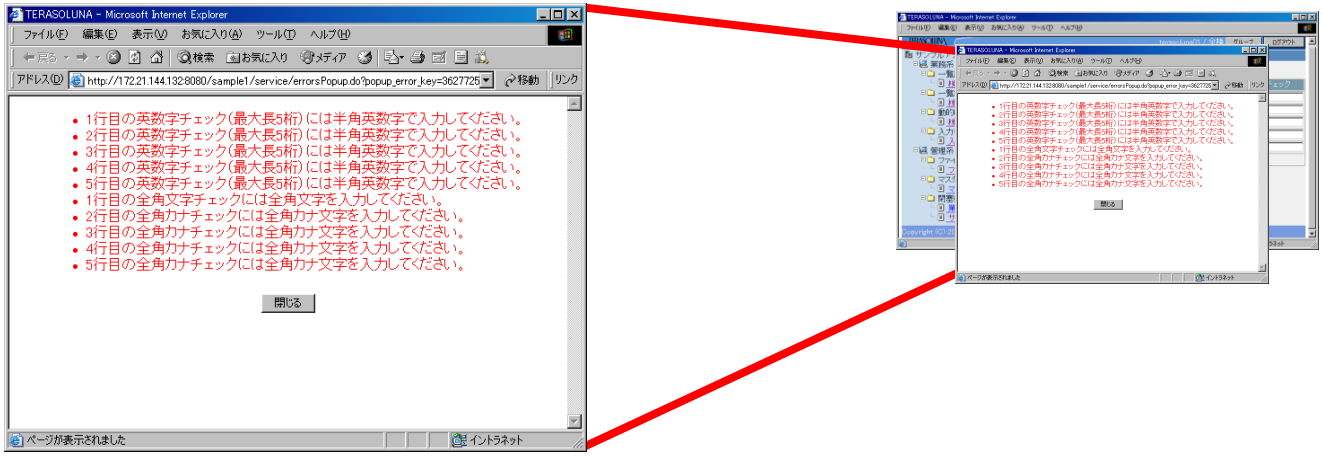


⑬画面表示(カスタムタグ)機能 ～メッセージポップアップ機能(1/4)

- 概要
  - メッセージまたはエラーメッセージをポップアップ画面に表示する。
  - PageContextに"ON_LOAD"をキーに埋め込まれたスクリプトをonLoadイベント処理に追加する。
- 解説
  - <ts:messagesPopup>
  - <ts:body>
    - このタグで生成するHTMLの<body>タグでは、onLoadイベント処理時のスクリプトとして、JavaScript関数の_onLoad_()を呼び出す。
    - JavaScript関数_onLoad_()の定義は、このタグで生成するため、HTML内に同名のJavaScriptを記述してはならない。

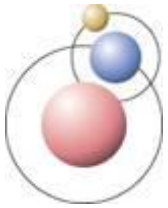
Copyright (c) 2007-2016 NTT DATA Corporation.

⑬画面表示(カスタムタグ)機能
～メッセージポップアップ機能(2/4)




```
<ts:messagesPopup popup="/service/errorsPopup.do" title="TERASOLUNA"/>
<ts:body styleClass="MainPage">
<% String script="alert('出力');";
   pageContext.setAttribute("ON_LOAD", script); %>
<ts:body>
↓上記の出力結果
<body onLoad="__onLoad__()">
  <script type="text/javascript">
  <!--
    function __onLoad__() {
      alert('出力');
    }
  //-->
  </script>
```


Copyright (c) 2007-2016 NTT DATA Corporation.

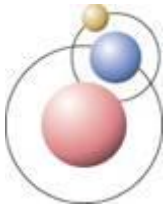


⑬画面表示(カスタムタグ)機能 ～メッセージポップアップ機能(3/4)

◆ <ts:messagesPopup>要素の属性一覧

属性	必須	概要
popup	○	ポップアップ画面で表示するURL。JavaScriptの window.open()の第一引数に対応する。
title	-	エラーを表示するポップアップ画面のタイトル。
param	-	JavaScript でポップアップ画面を開くときのパラメータ文字列。
paramType	-	JavaScript でポップアップ画面を開くときのパラメータ文字列を、ApplicationResources ファイルから取得する場合の リソースキー。
paramFunc	-	JavaScript でポップアップ画面を開くときの パラメータ文字列を取得する JavaScript 関数名。
windowId	-	開いたポップアップ画面を保持する JavaScript 変数名。

Copyright (c) 2007-2016 NTT DATA Corporation.

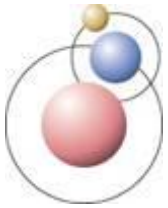


⑬画面表示(カスタムタグ)機能 ～メッセージポップアップ機能(4/4)

◆ <ts:body>要素の属性一覧

属性	必須	概要
onload	-	画面表示時に実行するJavaScript。
onunload	-	画面アンロード時に実行するJavaScript。
styleClass	-	スタイルシートのクラス名。
bgcolor	-	背景色。
background	-	背景に設定する画像
text	-	テキスト文字の色。
link	-	リンク部分の色。
vlink	-	既に選択されたリンク部分の色。
alink	-	選択中のリンク部分の色。

Copyright (c) 2007-2016 NTT DATA Corporation.

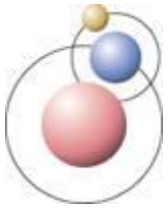


⑬画面表示(カスタムタグ)機能 ～エラーメッセージチェック機能

- 概要
  - リクエスト、またはセッションにエラー情報が設定されているかどうかを判別して表示/非表示を切り替える。
- 解説
  - <ts:ifErrors>
    - 入力チェックエラーがある場合、あるいは出力パラメータにエラー情報が設定されている場合にタグのボディ部分を評価する。
  - <ts:ifNotErrors>
    - 入力チェックエラーがなく、かつ出力パラメータにエラー情報が設定されて いない場合に、タグのボディ部分を評価する。
  - 使用例

```
<ts:ifErrors>
    ... // エラーがある場合の表示項目等
</ts:ifErrors>
<ts:ifNotErrors>
   ... // エラーがない場合の表示項目等
</ts:ifNotErrors>
```


Copyright (c) 2007-2016 NTT DATA Corporation.



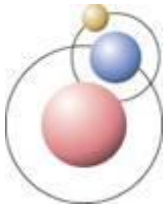
⑬画面表示(カスタムタグ)機能 ～クライアントチェック拡張機能

- 概要
  - クライアントでの入力チェック時に検証に使用する値をjavascriptの変数として出力する。
- 解説
  - <ts:javascript>
    - TERASOLUNAが提供する日本語系の検証ルールに必要な変数をjavascriptの変数として出力する
  - 使用例

```
<meta http-equiv="Content-Type" content="text/html; charset=Windows-31J">
<title>入力チェック(クライアント)テスト画面</title>
<ts:javascript formName="/clientValidate"/>
</head>
<body>
```

  - 属性一覧
    - <html:javascript>と同様

Copyright (c) 2007-2016 NTT DATA Corporation.



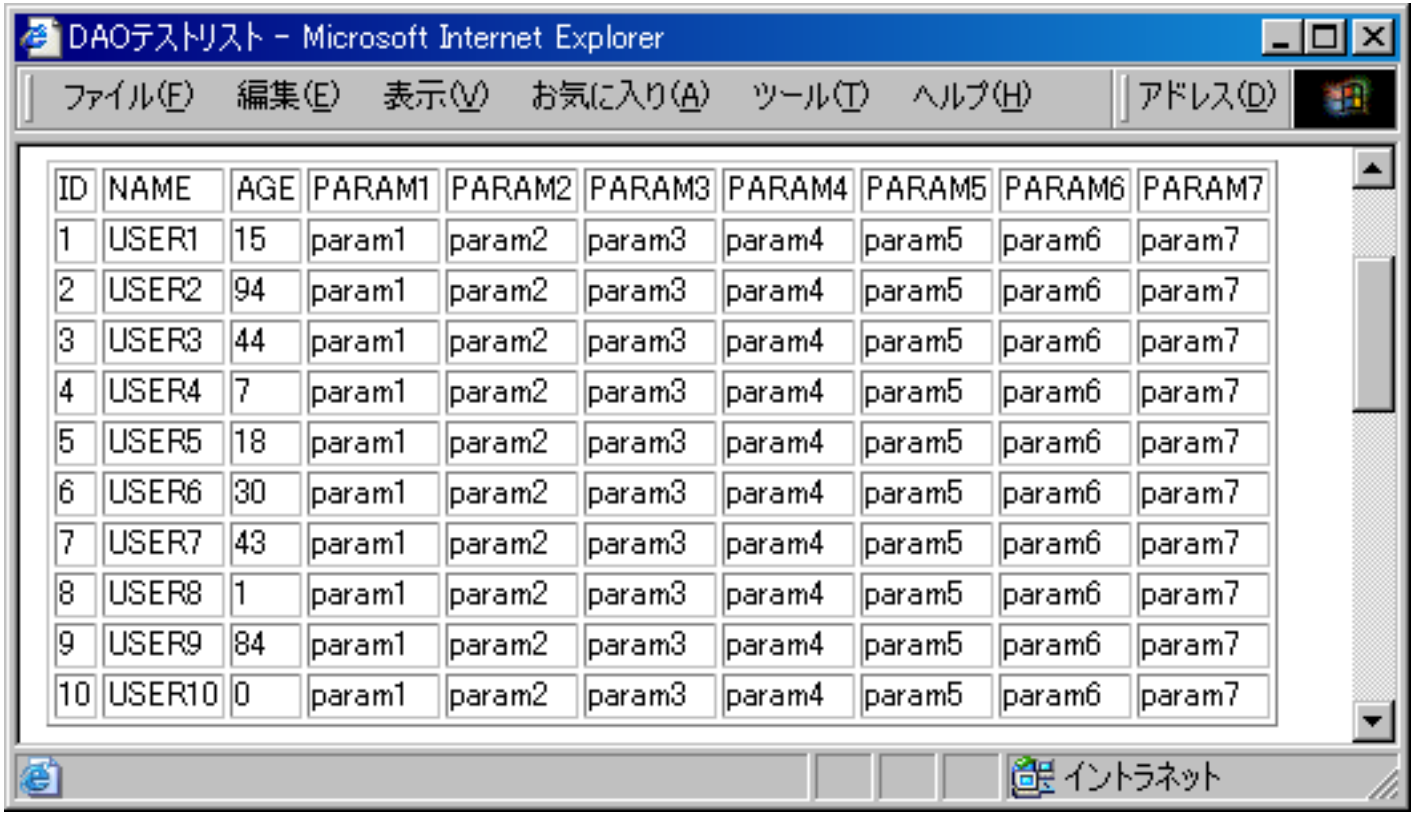
⑬画面表示(カスタムタグ)機能 ～一覧表示関連機能(1/14)

- 一覧表示の方法
  - Strutsの一覧表示機能<logic:iterate>要素を使用する。
  - 解説
    - 属性にて指定されたBeanから一覧情報を取得して、取得した一覧情報分のループを回す。
    - 指定するBeanは、Collection、ArrayList、Vector、Enumeration、Iterator、Map、HashMap、Hashtable、TreeMap、配列などである。
    - 繰り返し項目(HTMLテーブル内の<TR>～</TR>など)を囲むように記述する。

Copyright (c) 2007-2016 NTT DATA Corporation.

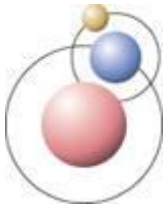
⑬画面表示(カスタムタグ)機能 ～一覧表示関連機能(2/14)

- 一覧表示の方法
  - 一覧表示機能を使用したJSPの実行例



ID	NAME	AGE	PARAM1	PARAM2	PARAM3	PARAM4	PARAM5	PARAM6	PARAM7
1	USER1	15	param1	param2	param3	param4	param5	param6	param7
2	USER2	94	param1	param2	param3	param4	param5	param6	param7
3	USER3	44	param1	param2	param3	param4	param5	param6	param7
4	USER4	7	param1	param2	param3	param4	param5	param6	param7
5	USER5	18	param1	param2	param3	param4	param5	param6	param7
6	USER6	30	param1	param2	param3	param4	param5	param6	param7
7	USER7	43	param1	param2	param3	param4	param5	param6	param7
8	USER8	1	param1	param2	param3	param4	param5	param6	param7
9	USER9	84	param1	param2	param3	param4	param5	param6	param7
10	USER10	0	param1	param2	param3	param4	param5	param6	param7

Copyright (c) 2007-2016 NTT DATA Corporation.



⑬画面表示(カスタムタグ)機能 ～一覧表示関連機能(3/14)

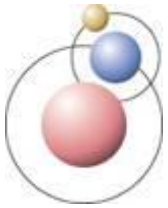
一覧表示の方法

一覧表示機能の実装例

```
<table border="1" frame="box">
    <tr>
        <td>ID</td>
        <td>NAME</td>
        <td>AGE</td>
        <td>PARAM1</td>
        ......
        <td>PARAM7</td>
    </tr>
    <logic:iterate id="userBean" name="dynaFormBean" property="userBeans">
    <tr>
        <td><bean:write name="userBean" property="id"/></td>
        <td><bean:write name="userBean" property="name"/></td>
        <td><bean:write name="userBean" property="age"/></td>
        <td><bean:write name="userBean" property="param1"/></td>
        ......
        <td><bean:write name="userBean" property="param7"/></td>
    </tr>
    </logic:iterate>
</table>
```

繰り返される。

Copyright (c) 2007-2016 NTT DATA Corporation.



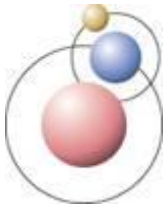
⑬画面表示(カスタムタグ)機能 ～一覧表示関連機能(4/14)

一覧表示の方法

<logic:iterate>要素の属性一覧

属性	必須	概要
id	○	一覧情報の1行分の情報を保持したエレメントBeanの名前を指定する。 指定した名前のBeanを使用してボディ内で表示を行う。
type	-	id属性で指定したエレメントBeanのタイプを指定する。
name	-	コレクションを保持しているBeanを指定する。(例:アクションフォーム名)
property	-	name属性で指定したBeanのプロパティを指定する。
scope	-	name属性で指定したBeanを取得するスコープを指定する。
collection	-	コレクションを実行時式で指定する。
length	-	ボディの繰り返し回数を指定する。
offset	-	表示する一覧情報の開始行を指定する。
indexId		ボディ内で使用する現在表示されている行のインデックス名を指定する。

Copyright (c) 2007-2016 NTT DATA Corporation.



⑬画面表示(カスタムタグ)機能 ～一覧表示関連機能(5/14)

- ページリンク機能
  - 機能概要
    - <logic:iterate>要素によって定義された一覧のページ遷移のリンクを表示する。
  - 解説
    - <ts:pageLinks>
    - ページリンク機能を使用する場合は、アクションフォームに以下のプロパティを用意する必要がある。
      - 表示行数を保持するプロパティ
      - 表示開始インデックスを保持するプロパティ
      - 一覧情報全行数を保持するプロパティ

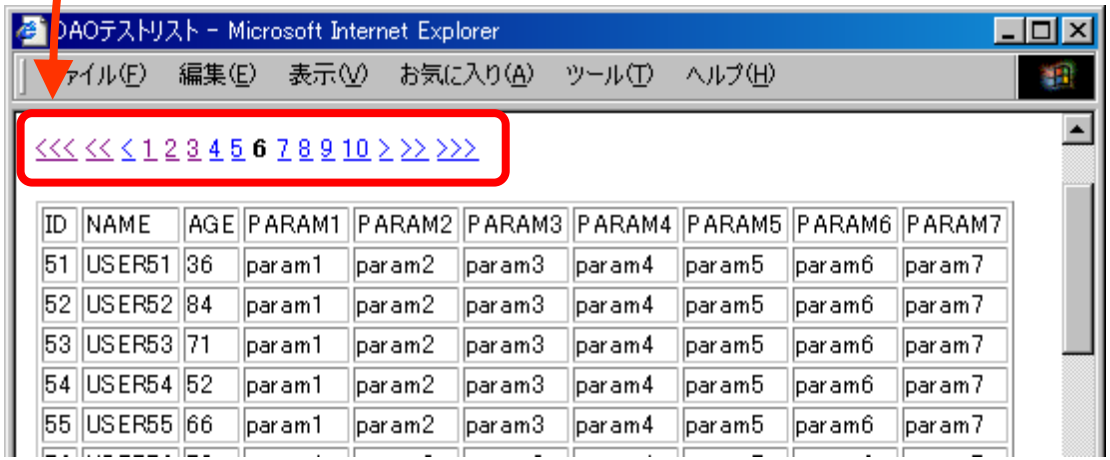
Copyright (c) 2007-2016 NTT DATA Corporation.

⑬画面表示(カスタムタグ)機能 ～一覧表示関連機能(6/14)

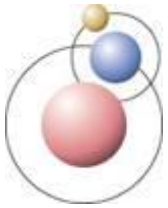
ページリンク機能

使用例

```
<ts:pageLinks action="/list" name="dynaFormBean" rowProperty="row"
                    totalProperty="totalCount" indexProperty="startIndex" />
<table border="1" frame="box">
    <logic:iterate id="userBean" name="dynaFormBean" property="userBeans">
    ......
    </logic:iterate>
</table>
```



Copyright (c) 2007-2016 NTT DATA Corporation.



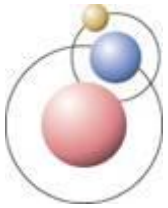
⑬画面表示(カスタムタグ)機能 ～一覧表示関連機能(7/14)

ページリンク機能

属性一覧(1/2)

属性	必須	概要
id	-	この属性に文字列が指定された場合、ページリンクの出力先を画面ではなくページコンテキストに保存する。この属性はページコンテキストに保存するキーとなる。
action	△	一覧表示画面の表示を行うアクションパス名を指定する。 submit属性がfalseの場合は必須属性となる。
name	-	表示行数、開始行インデックス、一覧情報全行数を取得するBeanを指定する。
rowProperty	-	表示行数のプロパティを指定する。
indexProperty	-	開始行インデックスのプロパティを指定する。
totalProperty	-	全行数のプロパティを指定する。
scope	-	name属性で指定したBeanを取得するスコープを指定する。
submit	-	リンクではなく、サブミットを行う場合はtrueを指定する。デフォルトはfalse。
forward	-	TERASOLUNAのDispatchActionを使用してフォワードによる振り分けを行う場合に使用する。trueを指定するとevent属性に設定された値のHiddenタグを出力する。また、サブミット時に、そのHiddenタグのvalue属性に“forward_pageLinks”を設定する。デフォルトはfalse。

Copyright (c) 2007-2016 NTT DATA Corporation.



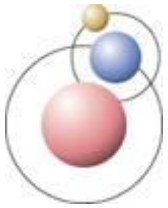
⑬画面表示(カスタムタグ)機能 ～一覧表示関連機能(8/14)

ページリンク機能

属性一覧(2/2)

属性	必須	概要
event	-	forward属性がtrueのときに有効となる属性で、TERASOLUNAのDispatchActionを使用してフォワードによる振り分けを行う場合に使用する。この属性に指定した名前のHiddenタグが生成される。デフォルトは"event"となる。
resetIndex	-	trueに設定すると指定範囲リセットを行うためのstartIndexとendIndexのHiddenタグを出力する。
currentPageIndex	-	対応する一覧の現在ページ数をページコンテキストに保存する際のキーとなる。デフォルトは"currentPageIndex"となる。
totalPageCount	-	対応する一覧の総ページ数をページコンテキストに保存する際のキーとなる。デフォルトは"totalPageCount"となる。

Copyright (c) 2007-2016 NTT DATA Corporation.

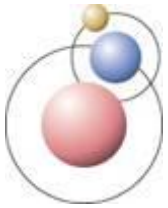


⑬画面表示(カスタムタグ)機能 ～一覧表示関連機能(9/14)

- ページリンク機能
  - 一覧情報を常にデータベースから取得する使用例
    - JSP

```
<ts:pageLinks action="/list" name="dynaFormBean" rowProperty="row"
                    totalProperty="totalCount" indexProperty="startIndex" />
<table border="1" frame="box">
    <logic:iterate id="userBean" name="dynaFormBean" property="userBeans">
    <tr>
        <td><bean:write name="userBean" property="id"/></td>
        <td><bean:write name="userBean" property="name"/></td>
        <td><bean:write name="userBean" property="age"/></td>
        <td><bean:write name="userBean" property="param1"/></td>
        <td><bean:write name="userBean" property="param2"/></td>
        <td><bean:write name="userBean" property="param3"/></td>
        <td><bean:write name="userBean" property="param4"/></td>
        <td><bean:write name="userBean" property="param5"/></td>
        <td><bean:write name="userBean" property="param6"/></td>
        <td><bean:write name="userBean" property="param7"/></td>
    </tr>
    </logic:iterate>
</table>
<ts:pageLinks action="/list" name="dynaFormBean" rowProperty="row"
                    totalProperty="totalCount" indexProperty="startIndex" />
```


Copyright (c) 2007-2016 NTT DATA Corporation.



⑬画面表示(カスタムタグ)機能 ～一覧表示関連機能(10/14)

- ページリンク機能
  - 一覧情報を常にデータベースから取得する使用例
    - Struts設定ファイル

```
<form-beans>
  <form-bean name="dynaFormBean"
    type="org.apache.struts.action.DynaActionForm" >
    <form-property name="userBeans"
                   type="jp.terasoluna.xxx.blogic.UserBean[]"/>
    <form-property name="row"
                   type="java.lang.String" initial="10"/>
    <form-property name="startIndex"
                   type="java.lang.String" initial="0"/>
    <form-property name="totalCount"
                   type="java.lang.String"/>
  </form-bean>
</form-beans>
<action path="/list"
        type="jp.terasoluna.xxx.action.ListAction"
        name="dynaFormBean" scope="session">
  <forward name="success" path="/listSRC.do"/>
</action>
<action path="/listSRC"
        type="org.apache.struts.actions.ForwardAction"
        parameter="/daoTestList.jsp">
</action>
```

一覧情報を保持したBeanの配列

1ページに表示する行数。初期値あり。

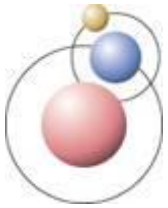
表示開始インデックス。初期値あり。

一覧情報全件数。

ページ単位に一覧情報を取得するビジネスロジックを呼び出す。

画面遷移のみ

Copyright (c) 2007-2016 NTT DATA Corporation.

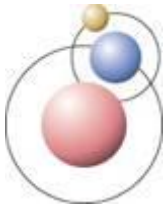


⑬画面表示(カスタムタグ)機能 ～一覧表示関連機能(11/14)

- ページリンク機能
  - 一覧情報を常にデータベースから取得する使用例
    - ビジネスロジック

```
DynaActionForm dynaForm = (DynaActionForm) form;
String strIndex = (String) dynaForm.get("startIndex");
String strRow = (String) dynaForm.get("row");
int startIndex = 0;
int row = 10;

//intへの変換処理
......

String totalCount
    = dao.executeForObject("getUserCount", null, String.class);

UserBean[] bean = dao.executeForObjectList("getUserList", null,
        UserBean.class, startIndex, row);

dynaForm.set("totalCount", totalCount);
dynaForm.set("userBeans", bean);
```


Copyright (c) 2007-2016 NTT DATA Corporation.



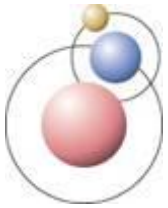
⑬画面表示(カスタムタグ)機能 ～一覧表示関連機能(12/14)

- ページリンク機能
  - 一覧情報をアクションフォームから取得する使用例
    - JSP

```
<ts:pageLinks action="/listSRC" name="dynaFormBean" rowProperty="row"
                    totalProperty="totalCount" indexProperty="startIndex" />
<table border="1" frame="box">
    <bean:define id="startIndex" name="dynaFormBean"
                 property="startIndex" type="java.lang.String" />
    <logic:iterate id="userBean" name="dynaFormBean" length="10"
                   property="userBeans" offset="<%=startIndex%>">
    <tr>
        <td><bean:write name="userBean" property="id"/></td>
        ......
        <td><bean:write name="userBean" property="param7"/></td>
    </tr>
    </logic:iterate>
</table>
<ts:pageLinks action="/listSRC" name="dynaFormBean" rowProperty="row"
                    totalProperty="totalCount" indexProperty="startIndex" />
```


Copyright (c) 2007-2016 NTT DATA Corporation.

TERASOLUNA



⑬画面表示(カスタムタグ)機能 ～一覧表示関連機能(13/14)

- ページリンク機能
  - 一覧情報を常にアクションフォームから取得する使用例
    - Struts設定ファイル

```
<form-beans>
  <form-bean name="dynaFormBean"
    type="org.apache.struts.action.DynaActionForm" >
      ......
  </form-bean>
</form-beans>
<action path="/list"
        type="jp.terasoluna.xxx.action.ListAction"
        name="dynaFormBean" scope="session">
  <forward name="success" path="/listSRC.do"/>
</action>
<action path="/listSRC"
        type="org.apache.struts.actions.ForwardAction"
        name="dynaFormBean" scope="session"
        parameter="/daoTestList.jsp">
</action>
```

最初に呼び出されて一覧情報を全件取得するビジネスロジックを呼び出す。

ページリンク機能にて呼び出される画面表示アクション。name属性で対応するアクションフォームを指定する必要がある。

Copyright (c) 2007-2016 NTT DATA Corporation.

⑬画面表示(カスタムタグ)機能 ～一覧表示関連機能(14/14)

ページリンク機能

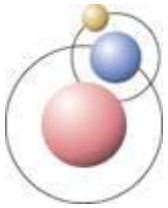
id、currentPageIndex、totalPageCount属性の使用例

- JSP

```
<ts:pageLinks id="pageLink" action="/listSRC" name="dynaFormBean"
    rowProperty="row" totalProperty="totalCount" indexProperty="startIndex"
    currentPageIndex="nowPage" totalPageCount="totalPage" />
<table border="1" frame="box">
    <logic:iterate id="userBean" name="dynaFormBean" property="userBeans">
    <tr>
        <td><bean:write name="userBean" property="id"/></td>
        ......
        <td><bean:write name="userBean" property="param7"/></td>
    </tr>
    </logic:iterate>
</table>
<bean:write name="pageLink" filter="false" />
<bean:write name="nowPage" />
<bean:write name="totalPage" />
```

PageLinksタグが、ページコンテキストに格納したリンク(<a>タグ)、現在ページ数、一覧情報の全件数を<bean:write>タグを使用して、出力する。

Copyright (c) 2007-2016 NTT DATA Corporation.

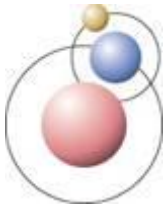


⑭ユーティリティ機能

ユーティリティ機能

- 業務開発で頻繁に使用する処理をユーティリティクラスとして提供する
- terasoluna-commons
  - ClassUtil
  - DateUtil
  - FileUtil
  - HashUtil
  - PropertyUtil
  - StringUtil
  - JndiSupport
    - DefaultJndiSupport

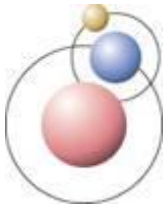
Copyright (c) 2007-2016 NTT DATA Corporation.



⑭ユーティリティ機能

- terasoluna-commons
  - ClassUtil
    - create(String)、create(String, Object[])
      - 完全修飾クラス名の文字列からインスタンスを生成する
  - DateUtil
    - dateToWarekiString(String, Date)
      - Dateインスタンスを和暦として指定のフォーマットに変換する
    - getSystemTime()
      - システム時刻を取得する
    - getWarekiGengoName(Date)
      - 指定された日付の和暦元号を取得する。
    - getWarekiGengoRoman(Date)
      - 指定された日付の和暦元号のローマ字表記(短縮形)を取得する。
    - getWarekiYear(Date)
      - 指定された日付の和暦年を取得する。

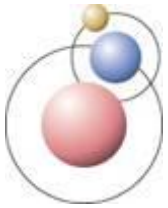
Copyright (c) 2007-2016 NTT DATA Corporation.



⑭ユーティリティ機能

- terasoluna-commons
  - FileUtil
    - getSessionDirectory(String)
      - 指定されたセッションIDに対応するディレクトリを取得する。
    - getSessionDirectoryName(String)
      - 指定されたセッションIDに対応するディレクトリ名を取得する。
    - makeSessionDirectory(String)
      - 指定されたセッションIDに対応するディレクトリを作成する。
    - removeSessionDirectory(String)
      - 指定されたセッションIDに対応するディレクトリを削除する。
    - rmdirs(File)
      - 指定されたディレクトリを削除する。

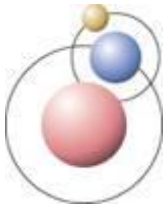
Copyright (c) 2007-2016 NTT DATA Corporation.



⑭ユーティリティ機能

- terasoluna-commons
  - HashUtil
    - hash(String, String)
      - 指定されたアルゴリズムで文字列のハッシュ値を取得する。
    - hashMD5(String)
      - MD5アルゴリズムで文字列のハッシュ値を取得する。
    - hashSHA1(String)
      - SHA1アルゴリズムで文字列のハッシュ値を取得する。

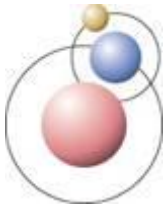
Copyright (c) 2007-2016 NTT DATA Corporation.



⑭ユーティリティ機能

- terasoluna-commons
  - PropertyUtil
    - addPropertyFile(String)
      - 指定されたプロパティファイルを追加で読み込む。
    - getPropertiesValues(Properties, Enumeration<String>)
      - キー一覧に対し、プロパティより取得した値を取得する。
    - getPropertiesValues(String, String)
      - プロパティファイル名、部分キー文字列を指定することにより値セットを取得する。
    - getProperty(String)
      - 指定されたキーのプロパティを取得する。
    - getPropertyNames()
      - プロパティのすべてのキーのリストを取得する。
    - getPropertyNames(String)
      - 指定されたプリフィックスから始まるキーのリストを取得する。
    - loadProperties(String)
      - 指定したプロパティファイル名で、プロパティオブジェクトを取得する。

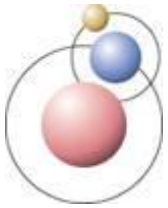
Copyright (c) 2007-2016 NTT DATA Corporation.



⑭ユーティリティ機能

- terasoluna-commons
  - StringUtil
    - capitalizeInitial(String)
      - 指定された文字列の頭文字を大文字にする。
    - getExtension(String)
      - 指定された文字列から末尾の拡張子を取得する。
    - hankakuToZenkaku(String)
      - 半角文字列を全角文字列に変換する。
    - zenkakuToHankaku(String)
      - 全角文字列を半角文字列に変換する。
    - parseCSV(String)
      - CSV形式の文字列を文字列の配列に変換する。
    - toLikeCondition(String)
      - 検索条件文字列をLIKE述語のパターン文字列に変換する。
    - trim(String)
      - 文字列の両側のホワイトスペースを削除する。

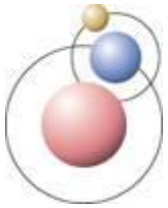
Copyright (c) 2007-2016 NTT DATA Corporation.



⑭ユーティリティ機能

- terasoluna-commons
  - JndiSupport
    - Jndi APIの操作を行うクラス。
    - インタフェースにアクセスすることで、APサーバーが動作していなくてもテストが可能
    - lookup(String)
      - » 指定された名前のオブジェクトを取得する
    - rebind(String, Object)
      - » 名前をオブジェクトにバインドして、既存のバインディングを上書きする。
    - unbind(String)
      - » 指定されたオブジェクトをアンバインドする。

Copyright (c) 2007-2016 NTT DATA Corporation.



⑭ユーティリティ機能

- terasoluna-commons
  - DefaultJndiSupport
    - Springの機能を利用して実装されたデフォルトのJndiSupport実装クラス
    - Bean定義ファイルに設定して利用する

```
<bean id="jndiSupport" scope="prototype"
  class="jp.terasoluna.fw.web.jndi.DefaultJndiSupport" >
</bean>
```

Bean定義ファイル

※テスト時はBean定義ファイルを切り替え、ダミー実装のJndiSupportクラスを定義する

Copyright (c) 2007-2016 NTT DATA Corporation.