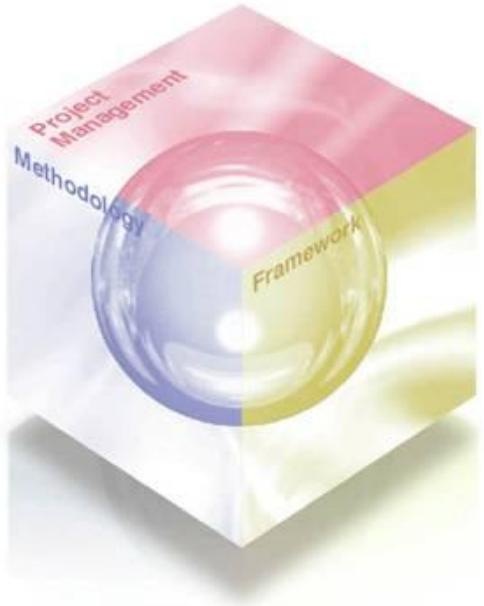
TERASOLUNA® Server Framework for Java Architecture Manual (Web Version)



NTT DATA Corporation
Technology Development Division
Software Engineering Promotion Center

TERASOLUNA®
BEST SOLUTION FOR SYSTEM DEVELOPMENT

NTT DATA

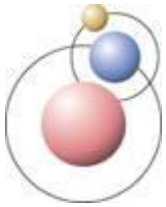
- In order to use this document, you are required to agree to abide by the following terms. If you do not agree with the terms, you must immediately delete or destroy this document and all its duplicate copies.

1. Copyrights and all other rights of this document shall belong to NTT DATA or third party possessing such rights.
2. This document may be reproduced, translated or adapted, in whole or in part for personal use. However, deletion of the terms given on this page and copyright notice of NTT DATA is prohibited.
3. This document may be changed, in whole or in part for personal use. Creation of secondary work using this document is allowed. However, “Reference document: TERASOLUNA POCKET BOOK” or equivalent documents may be mentioned in created document and its duplicate copies.
4. Document and its duplicate copies created according to Clause 2 may be provided to third party only if these are free of cost.
5. Use of this document and its duplicate copies, and transfer of rights of this contract to a third party, in whole or in part, beyond the conditions specified in this contract, are prohibited without the written consent of NTT Data.
6. NTT DATA shall not bear any responsibility regarding correctness of contents of this document, warranty of fitness for usage purpose, assurance for accuracy and reliability of usage result, liability for defect warranty, and any damage incurred directly or indirectly.
7. NTT DATA does not guarantee the infringement of copyrights and any other rights of third party through this document. In addition to this, NTT DATA shall not bear any responsibility regarding any claim (Including the claims occurred due to dispute with third party) occurred directly or indirectly due to infringement of copyright and other rights.

- Registered trademarks or trademarks of company name and service name, and product name of their respective companies used in this document are as follows.

  - Apache and Tomcat are the registered trademarks or trademarks of Apache Software Foundation.
  - Java, JDK, J2SE, J2EE, JSP, and Servlet are the registered trademarks or trademarks of Sun Microsystems, Inc. in the United States and other countries.
  - Oracle is a registered trademark or trademark of Oracle International Corp. in the United States and other countries.
  - TERASOLUNA is a registered trademark of NTT DATA Corporation.
  - Web Logic is a registered trademark or trademark of BEA Systems Inc..
  - Web Sphere is a registered trademark or trademark of IBM Corporation.
  - All other company names and product names are the registered trademarks or trademarks of their respective companies.

Table of Contents



Introduction

Outline

- This document describes the server framework “TERASOLUNA Server Framework for Java – Web Version" for creating Web applications.
- This framework is an extended framework based on Spring2.5 and Struts1.2

Introduction

Development Environment

JDK Version

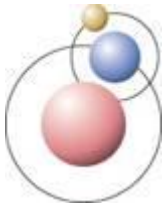
- JDK 5.0/6.0

Supported Web servers

- Tomcat 5.5/6.0
- WebLogic Server 9.2J/10.0/10.3
- WebSphere 6.1/7.0
- Cosminexus 07-60/08-00
- Interstage V9.1.0
- WebOTX V8.2

Supported Databases

- Oracle11g (11.1.0)
- Oracle10g (10.2.0)
- Oracle 9i (9.2.0)
- PostgreSQL 8.2/8.3/8.4



Introduction

Necessary modules

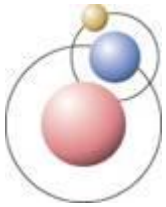
Pattern 1

- terasoluna-thin-server
  - *All the functionalities pertaining to the Web Version have been intergrated in this module.*
  - This single module provides the functions of the following modules: terasoluna-commons, terasoluna-thin, terasoluna-dao, terasoluna-ibatis.

Pattern 2

- terasoluna-thin
  - *This module provides functions peculiar to the Web Version.*

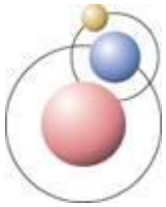
- terasoluna-commons
  - This module provides utility functions.



Introduction

- **Optional Modules**
  - terasoluna-dao
    - This module provides the DAO Interface
  - terasoluna-ibatis
    - This module provides the database access function using the OR-Mapping tool iBatis

※ ~~If you are~~ *While* using terasoluna-thin-server, the above modules are not necessary.



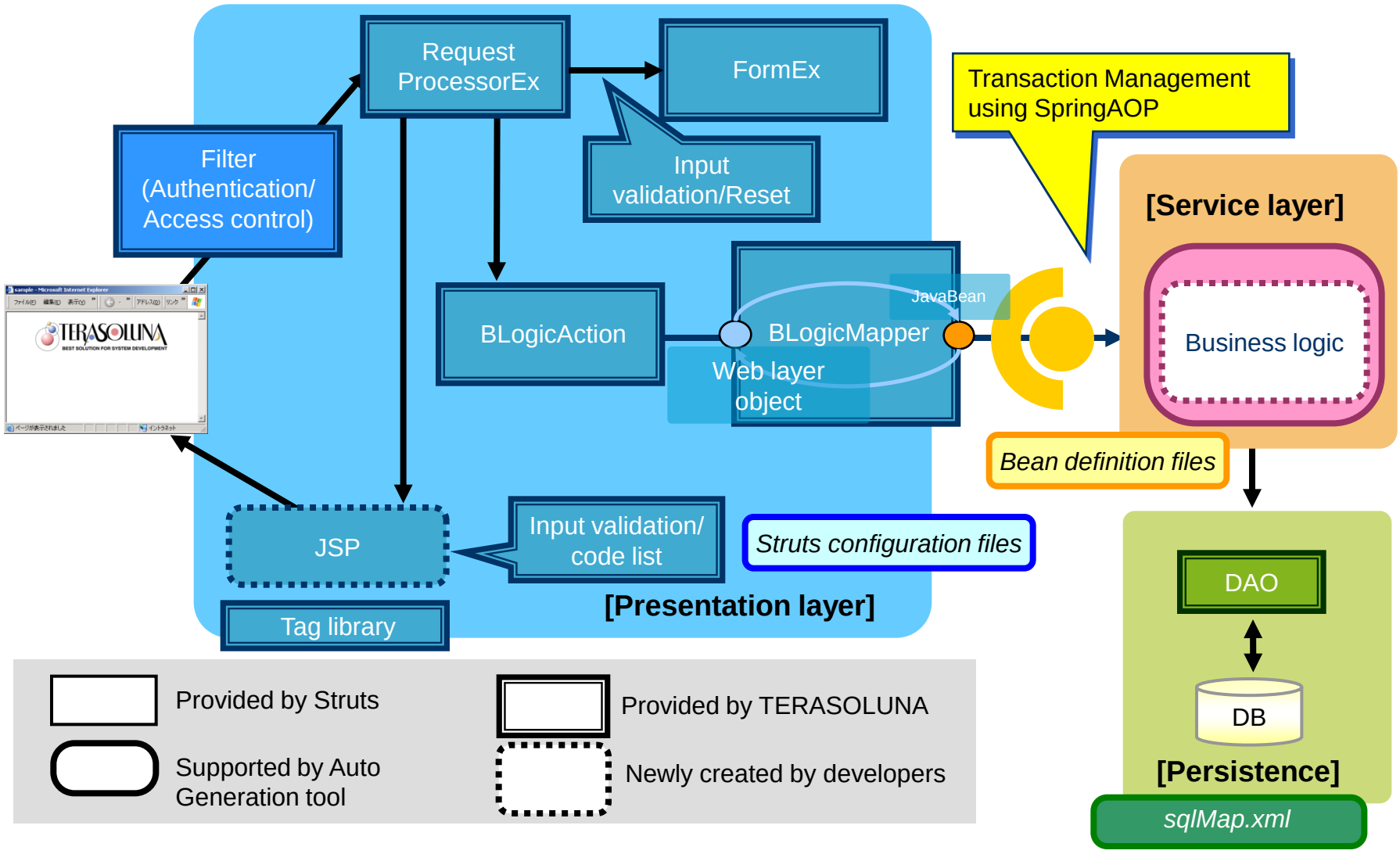
Introduction

List of Open Source Libraries being Used

Name of Open Source Library	Version	TERASOLUNA Module（terasoluna-*）			
		commons	dao	ibatis	thin
antlr.jar	2.7.6				○
cglib-nodep.jar	2.1_3	○	○	○	
commons-beanutils.jar	1.7.0	○			○
commons-dbcp.jar	1.2.2			○	
commons-digester.jar	1.8				○
commons-fileupload.jar	1.2				○
commons-jxpath.jar	1.3	○			○
commons-lang	2.3	○			○
commons-logging.jar	1.1.1	○			○
commons-pool.jar	1.3			○	
commons-validator.jar	1.3.1				○
easymock.jar	2.3		○（テスト用）		
ibatis.jar	2.3.4.726			○	
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）
junit-addons.jar	1.4	○（テスト用）	○（テスト用）	○（テスト用）	○（テスト用）
mockrunner-jdbc.jar	0.3.7	○（テスト用）			○（テスト用）
mockrunner-servlet.jar	0.3.7				○（テスト用）
mockrunner-struts.jar	0.3.7				○（テスト用）
mockrunner-tag.jar	0.3.7				○（テスト用）
servlet-api.jar	2.4	○	○		○
spring.jar	2.5.6.SEC01	○	○	○	○
spring-test.jar	2.5.6.SEC01				○（テスト用）
spring-webmvc-struts.jar	2.5.6.SEC01				○
struts.jar	1.2.9				○

TERASOLUNA

Architecture Overview

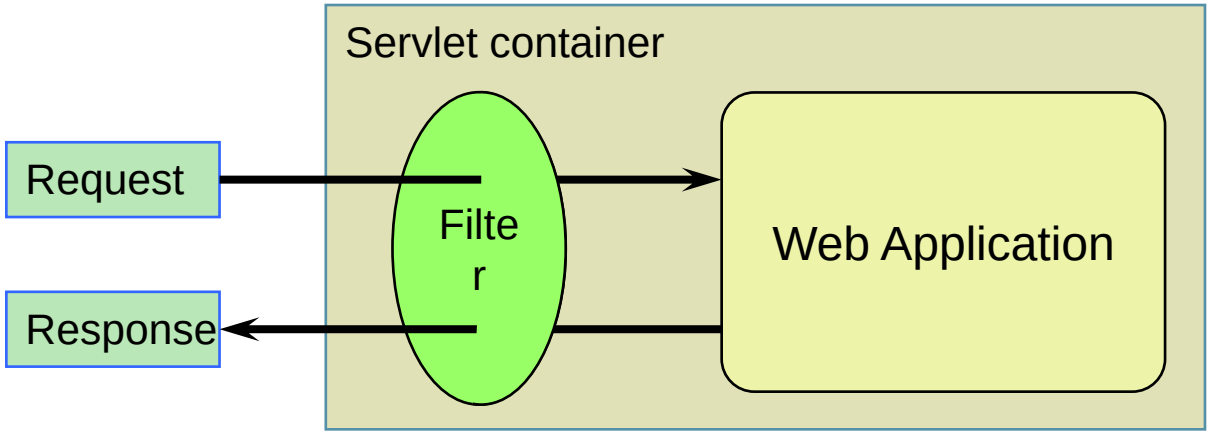


① Authentication/Access Control Function

Authentication/Access Control

- Provides authentication/access control using the Filter Interface.
  - Login Check
  - Authorization Check
  - Server Blockage Check
  - Business Logic Blockage Check
  - Prohibit direct access to Extensions

Reference: Filter execution

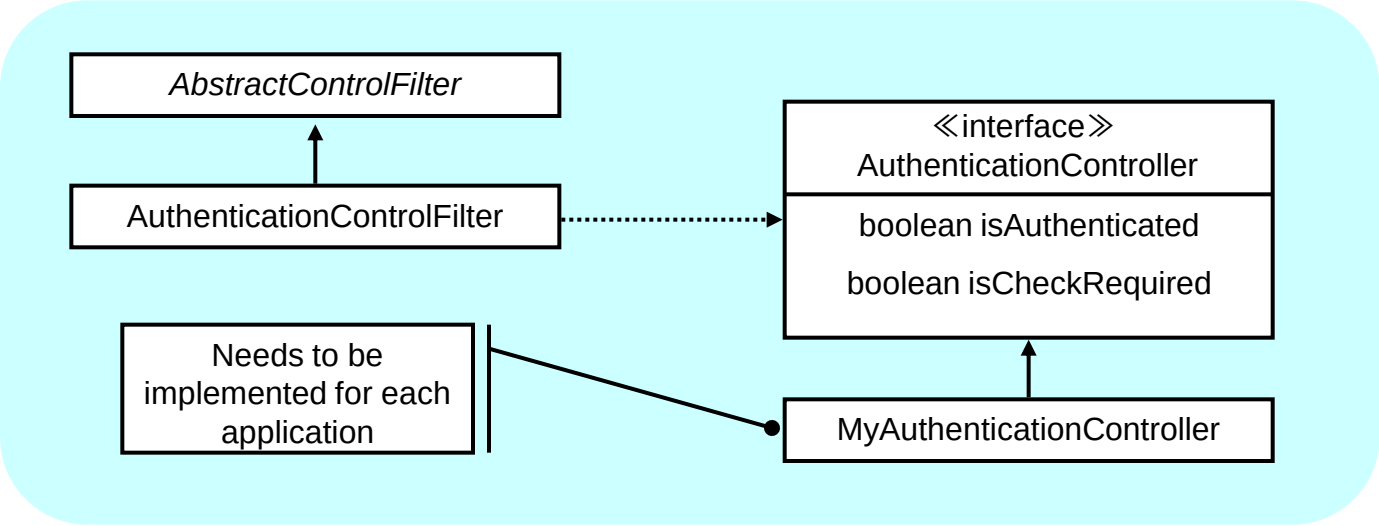


① Authentication/Access Control

Login Check

Outline

- Verifies Whether the user, who is accessing the Application, has logged In or not.
- As the authentication logic is application-dependent, it is required to create a class that implements the AuthenticationController Interface.



① Authentication/Access Control

Login Check

Usage:

- Implement AuthenticationController Interface.
  - boolean isAuthenticated…Returns true if the user has logged in. False otherwise.
  - boolean isCheckRequired…Returns true if login check is required. False otherwise. (Screens such as “Login”) return false.
- Edit the below described settings in web.xml and bean definition file.

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>
```

Filter provided by FW

Bean definition file

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

Set the class that implements the AuthenticationController interface. Bean ID is authenticationController.

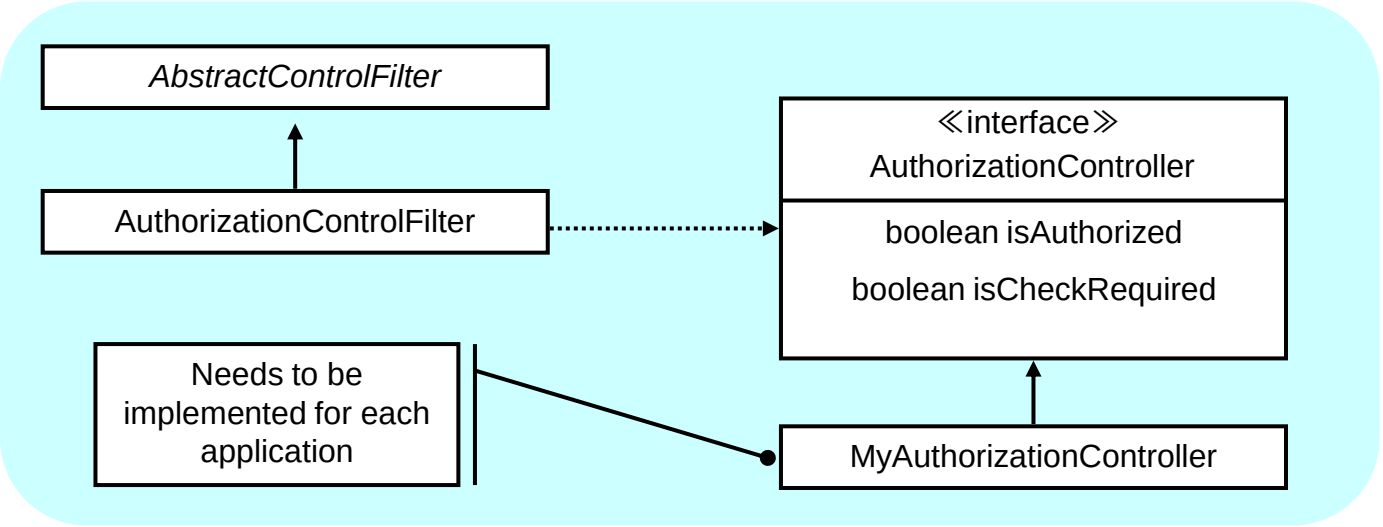
The exception class to be thrown in case of authentication error. The JSP page to be displayed should be set here.

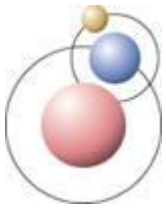
① Authentication/Access Control

Authorization Check

Outline

- Checks the Access Rights of the user before processing the user request.
- As the Authorization Check is Application-dependent , for each Application, it is required to create a class that implements the AuthorizationController Interface.





① Authentication/Access Control

Authorization Check

Usage:

- Implement AuthorizationController Interface.
  - boolean isAuthorized･･･. Returns true if user is authorized. False otherwise.
  - boolean isCheckRequired･･･Returns true if the Authorization Check is required. False otherwise. (Screens such as “Login”) return false.
- Edit the below described settings in web.xml and Bean definition files.

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>
```

Filter provided by FW

Bean definition file

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

Set the class that implements the AuthorizationController interface. Bean ID is authorizationController.

The exception class to be thrown in case of authorization error. The JSP page to be displayed should be set here.

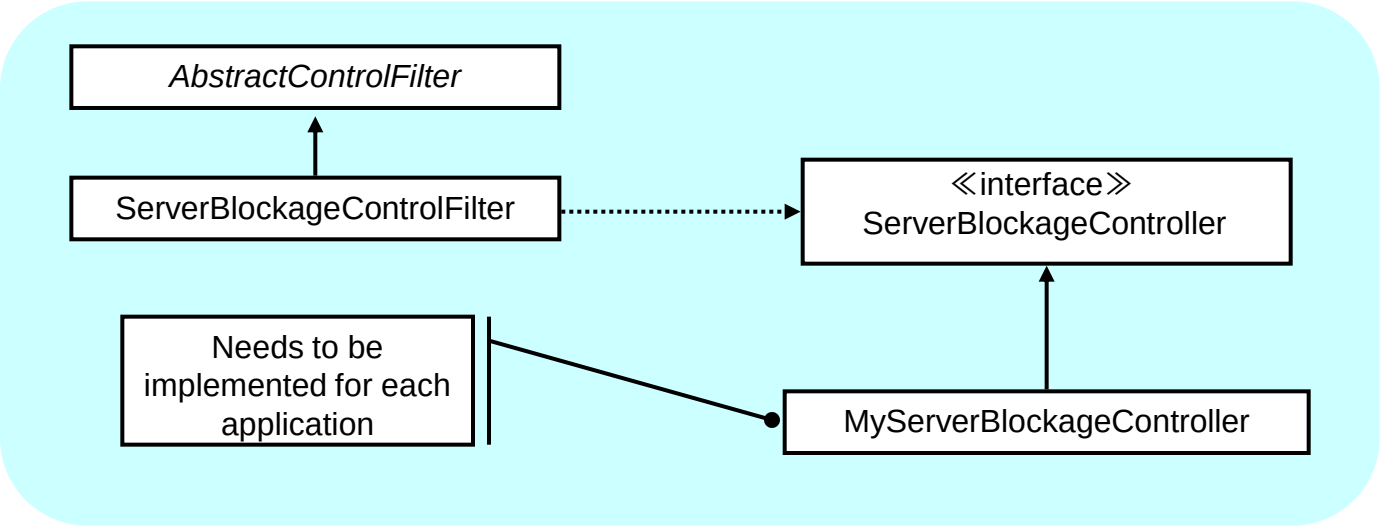
TERASOLUNA

① Authentication/Access Control

Server Blockage Check

Outline

- Determines whether the Application is blocked.
- As the Server Blockage Check is application-dependent, for each Application, it is required to create a class that implements the ServerBlockageController Interface.



①Authentication/Access Control

Server Blockage Check

Usage:

- Implement the ServerBlockageController Interface.
  - void blockade…Changes to blockage state.
  - boolean isBlockaded… Returns true if in blockage state. False otherwise.
  - void preBlockade… Changes to pre-blockage state.
  - boolean isPreBlockaded…Returns true if in pre-blockage state. False otherwise.
  - void open…Cancels the blockage state.
- Edit the below described settings in web.xml, Bean definition file.

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>
```

Filter provided by FW

Bean definition file

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

Set the class that implements the ServerBlockageController interface. Bean ID is serverBlockageController.

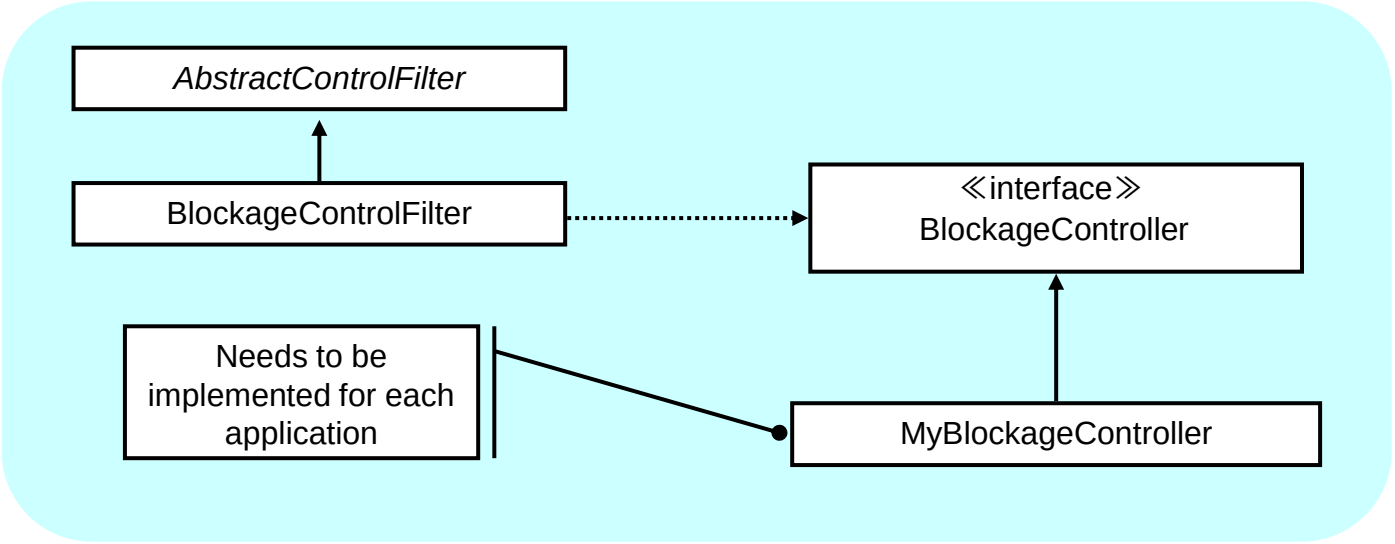
The exception class to be thrown in case of Blockage error. The JSP page to be displayed should be set here

① Authentication/Access Control

Business Blockage Check

Outline

- Determines whether a specific business functionality is blocked.
- As the Business Blockage Check is application-dependent, for each Application, it is required to create a class that implements the BlockageController Interface.



① Authentication/Access Control

Business Blockage Check

Usage:

- Implement BlockageController interface
  - void blockade…Changes to blockage state.
  - boolean isBlockaded…Returns true if in blockage state. False otherwise.
  - void open…Cancels the blockage state.
- Edit the below described settings in web.xml and Bean definition file.

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>
```

Filter provided by FW

Bean definition file

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

Set the class that implements the BlockageController interface.
Bean ID is BlockageController.

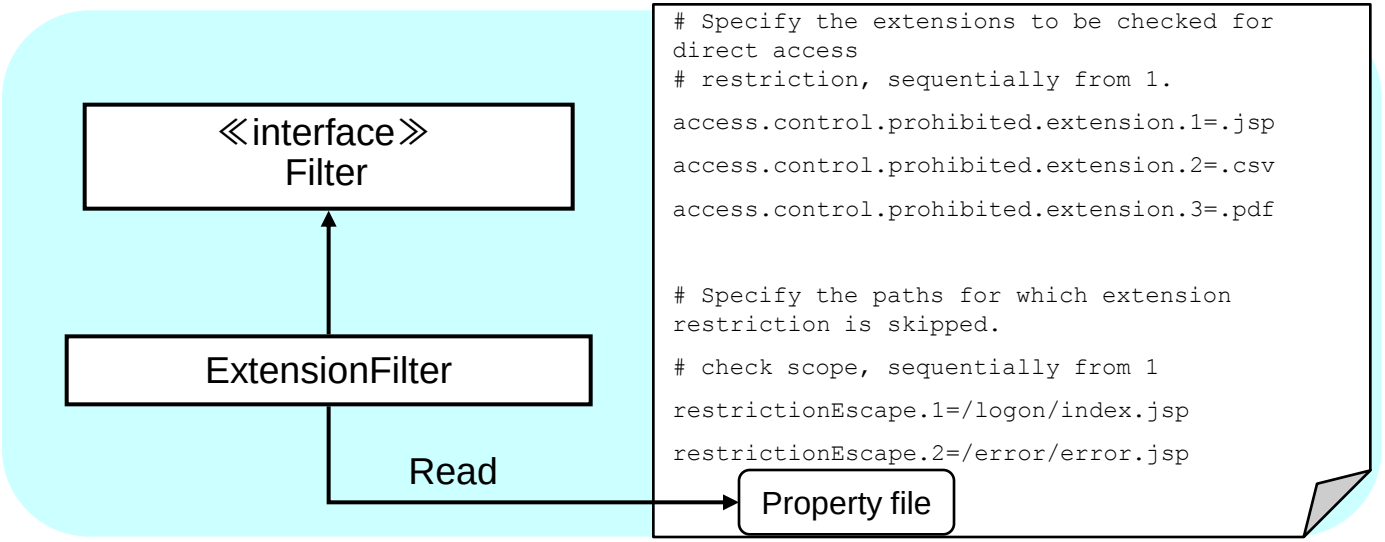
The exception class to be thrown in case of Blockage error. The JSP page to be displayed should be set here

① Authentication/Access Control

Restrict direct file access based on file extension

Outline

- Prohibit access if the client is not allowed to directly access files with certain extensions.
- Used if required to prohibit the direct access to jsp, pdf or csv etc files.
- The extensions for which direct access is prohibited, are set in a property file.



① Authentication/Access Control

Prohibit direct access to extensions

Usage:

- In property file, describe the extensions for which direct access is prohibited and paths for which the check is to be skipped.

```
access.control.prohibited.extension.<serial no.>=Extension for
which direct access is denied.
Example)access.control.prohibited.extension.1=.jsp
restrictionEscape.<serial no.>= Paths for which direct access
prohibition check is skipped.
Example) restrictionEscape.1=/logon/index.jsp
```

※As per above settings, error occurs when client accesses the URL /*.jsp, except /logon/index.jsp.

- Describe the settings in web.xml.

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>
```

Filter provided by FW

② User Information Retention

User Information Retention

- Functionality to retain the information of logged-on user
  - A class inheriting from abstract class UserValueObject and with the necessary attributes should be created.
  - The class inheriting from UserValueObject is specified in property file.

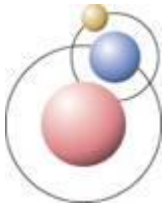
```
public class MyUserValueObject extends UserValueObject {
    /**User name.*/
    private String userName = null;
    /**User role*/
    private String userRole = null;
    /**
     * Return user name.
     * @return user name
     */
    public String getUserName() {
        return userName;
    }
   ･･･(Hereafter omitted)
}
```

UserValueObject implementation class

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

Property file

Instance of the class described in the property file is created by UserValueObject#createUserValueObject().



③ Code List Functionality

Code List Functionality

What is a code list?

- A group of “name=value” pairs used for a identical purposes. There is no (or extremely rare) chance of being changed within the application.
- Example: Japanese name of era

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

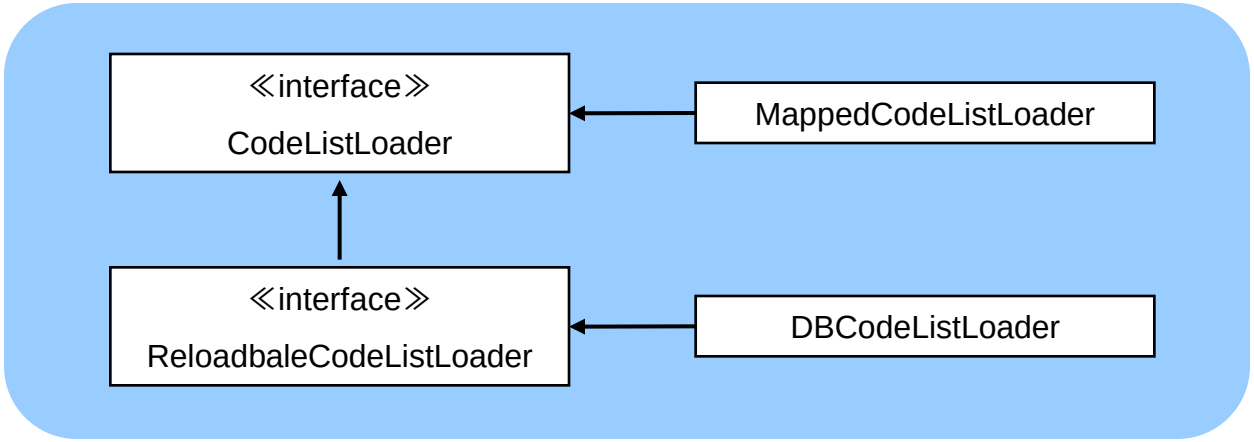
Code list functionality

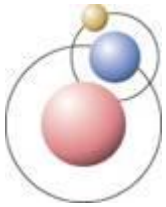
- Provides methods to define, read, display and use code lists.

③ Code List Functionality

Outline

- CodeListLoader… An Interface implemented by *all* classes that read/*retain* code lists.
- MappedCodeListLoader… A class that reads code lists based on a ~~setup~~ settings file.
- ReloadableCodeListLoader…An Interface implemented by classes that retain updatable code lists.
- DBCodeListLoader…A class that reads code lists from database.
- CodeBean…A class used to display an element from the code list.





③ Code List Functionality

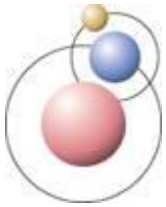
- **MappedCodeListLoader**
  - A Class implementing CodeListLoader is required to be set in Bean definition file of SpringFramework.
  - Settings in Bean definition file as follows:

```
<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>
```

Set the code list information in codeListMap attribute in MAP format.

CodeBean[]	
001	value001
002	value002
003	value003

Bean definition file



③ Code List Functionality

- **DBCodeListLoader**
  - The class that
    - Reads a code list from the database
    - Should implement ReloadCodeListLoader
  - Contents of a Code list can be updated by calling the reload method.
  - Uses the data-source specified in the Spring.
  - In the Bean definition file, declare the SQL, that retrieves the code list.

Bean definition file

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

SQL statement, which retrieves the code list in Spring

TERASOLUNA

③ Code List Functionality

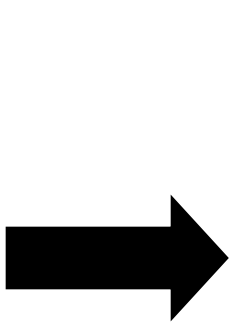
Code List Display

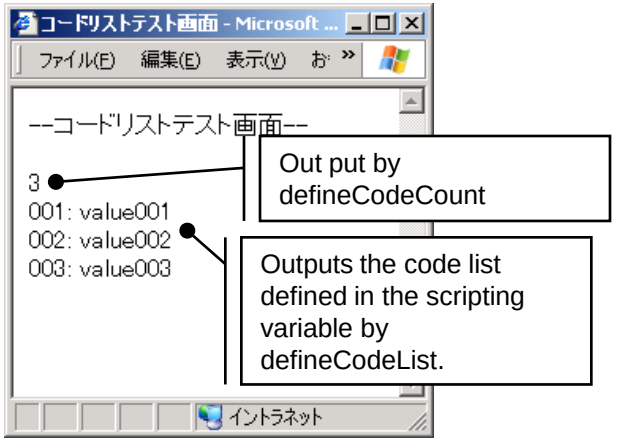
- A tag library is provided for easy handling of code lists on JSP.
  - defineCodeList Tag…Defines the code list as a scripting variable using the BeanId of the CodeListLoader as a key.
  - writeCodeCount Tag...Displays the code list record count using the BeanId of the CodeListLoader as Key.
  - writeCodeValue Tag...Writes the Display name on screen using the code value as key.

```
<%@ taglib prefix="t"  uri="/WEB-INF/terasoluna.tld"%>
<html><head><title>Code List Test Screen</title></head>
<body>
--Code List Test Screen--<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>
```

JSP

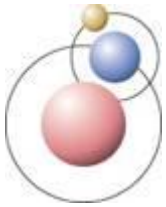
Use the same string






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

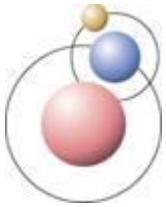
Bean definition file



④ RequestProcessor Extension

RequestProcessor Extension

- RequestProcessorEx provided by TERASOLUNA inherits the DelegatingRequestProcessor of SpringFramework.
- The following enhancements are provided:
  - Change of ActionForm based on change of Module.
  - Disabling the processPopulate function.



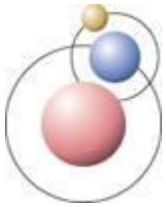
④ RequestProcessor Extension

- ***Change of ActionForm based on change of Module***
  - Action form of session scope is used and when the *Module* is switched, the action form of prior *Module* is deleted.
  - Steps to implement this function:
    - Define the scope of action form as “session”
    - Define action form for each *Module*.
    - Add “_” (underscore) at the beginning of action form name.
    - Divide the application path logically for each *Module*.

Struts configuration file

```
<!-- form1 Action form used in use case -->
<form-bean name="_form1" type="jp.terasoluna.xxx.FormBean1" />
<!-- form2 Action form used in use case -->
<form-bean name="_form2" type="jp.terasoluna.xxx.FormBean2" />
<action path="/form1/hoge" name="_form1" scope="session"
                parameter="/form1/index.jp" />
<action path="/form2/hoge" name="_form2" scope="session"
                parameter="/form2/index.jp" />
```

Framework gets the module name by parsing the first part of the path.



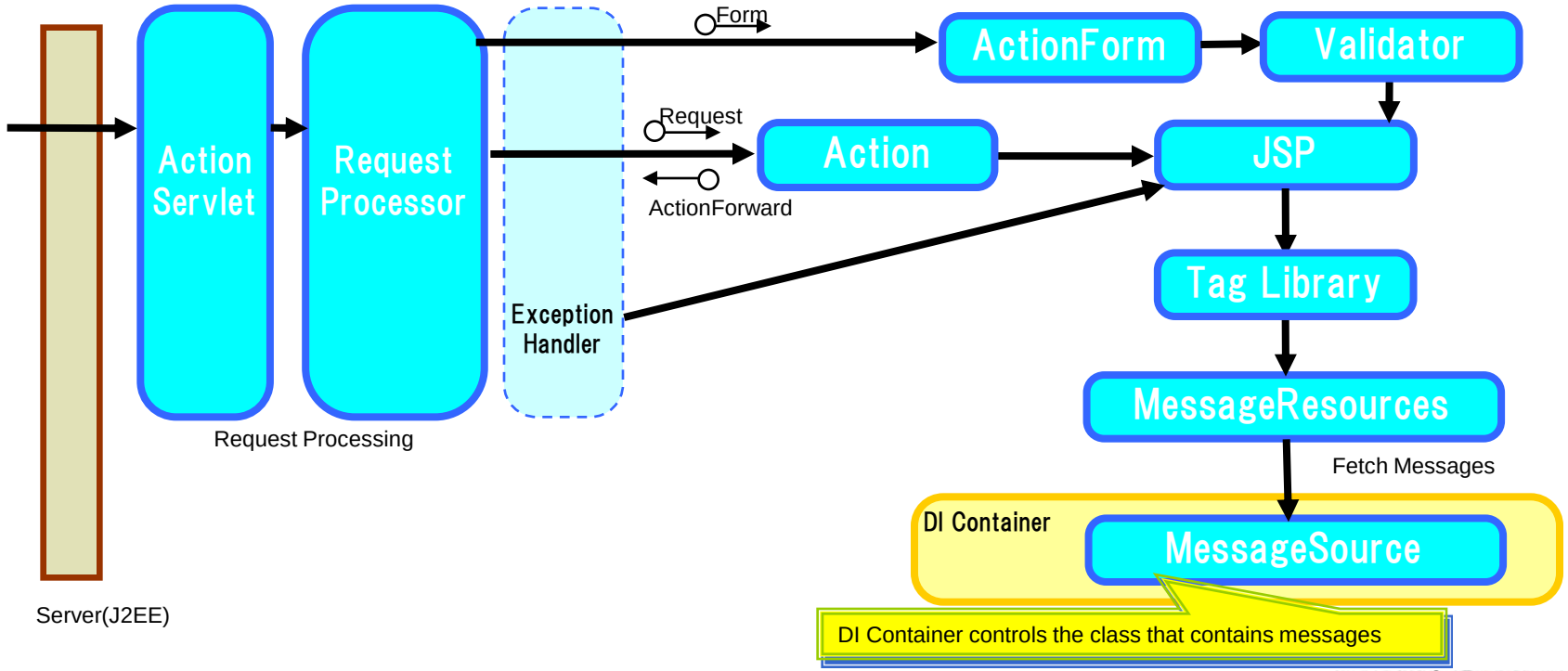
④ RequestProcessor Extension

- **Disabling processPopulate**
  - When ActionForm is updated by business logic, processPopulate is not executed in the same request.
  - This is done to prevent the modifications made to ActionForm by the business logic from being overwritten by the request parameters.

⑤ Message Management Function

Message Management

- Provides retention/fetch function of the messages(error messages etc) to be used in the Framework.
  - ResourceBundle Message Function･･･Read messages from the property file
  - DBMessage Function･･･Read Messages from the Database



⑤ Message Management Function

Related classes

- SpringMessageResources, SpringMessageResourcesFactory
  - Reads messages from messageSource of Spring

Struts-config file

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

If nothing is set in parameter property, then "messageSource" is used as the default bean name.

However , not specifying the "parameter" property itself is not possible.

- MessageSource
  - This Interface should be implemented by the class holding the messages
  - Bean Definition with the ID "messageSource " will be recognized as the message class in Spring
- ResourceBundleMessageSource
  - This Implementation of MessageSource uses the ResourceBundle
    - » Reads messages from property file(multiple files possible)
    - » Internationalization Support
    - » Message Loading in a running web application is not possible

Bean Definition File

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

Write comma-separated property file names. The priority of the files written first is higher. In general, Message s for each module → Messages common across the application → System Messages , should be the sequence of message files.

⑤ Message Management Function

- **DataSourceMessageSource**
  - MessageSource Implementation class uses Database
    - » Reads messages from database
    - » Internationalization Support
    - » Message Loading in a running web application is not possible

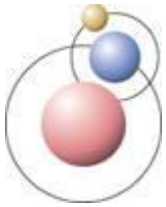
It is necessary to set (DI) the class for fetching messages (Class implementing DBMessageResourceDAO Interface)。(DBMessageResourceDAOImpl is provided as default implementation)

Bean Definition File

```
<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>
```

DAO for reading messages from DB

Default SQL is
SELECT CODE,MESSAGE FROM MESSAGES
Table name = MESSAGES
Column name with message code = CODE
Column name with message content = MESSAGE



⑤ ActionForm Extension Function

ActionForm Extension

- Following functionalities are provided:
  - Action form base class
  - Auto reset functionality
  - Action form with module scope
  - Helper methods
    - Automatically control the number of elements in a list, array
  - Disable the populate function

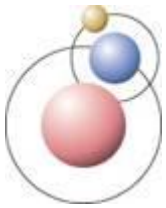
⑥ ActionForm Extension Functionality

ActionForm Scope

- The scope to retain the ActionForm is determined from the Business Process characteristics.
  - “request” scope is used when the data is to be used only in a single screen
  - “session” scope is used when the data is to be used in multiple screens.

    ActionForm having “session” scope should be deleted when it is not required.
- In a session, there *can exist* only one ActionForm having session scope and name with “_” attached at the beginning.
  - When it is confirmed that the user will not execute multiple modules simultaneously, the ActionForm having name attached with “_” can be used. (The ActionForm is then automatically destroyed by the framework.)

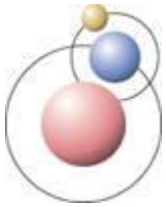
    ※However, if there is a possibility that the user operates multiple modules simultaneously, the ActionForm having name attached with “_” is not recommended.



⑥ ActionForm Extension Functionality

- **Which ActionForm should be used? (Dynamic or Static ?)**
  - Static ActionForm ensures Type Safety (as the user defines the methods).
  - Run-time error occurs in dynamic ActionForm when a property does not exist or when there is a type-mismatch. (Static ActionForm can be checked at compile-time)
  - Cost of development of a dynamic ActionForm is less compared to that of a static ActionForm (since implementation of the class is not necessary).
  - It is difficult to say whose performance is better.
    Generally, dynamic ActionForm take more time for initialization.
    Generally, static ActionForm take more time for getting/setting properties, (as reflection is used by framework)
    ※ Practically, Performance differs with every application, since the number of attributes and their usage conditions differ.

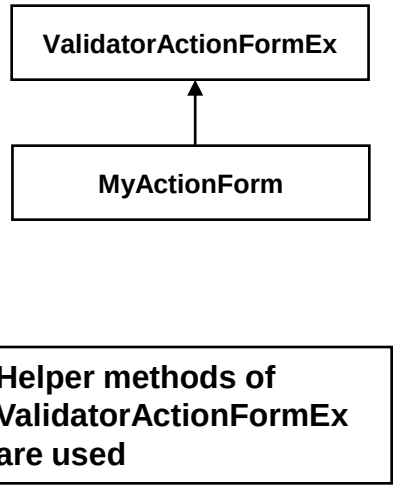
  → If the development cost of static ActionForm can be reduced by auto-code generation, then static ActionForm is recommended as they are type-safe.



⑥ ActionForm Extensions

- **ValidatorActionFormEx**
  - Subclass of ValidatorActionForm
  - Developer should create a class extending ValidatorActionFormEx, with the necessary properties and access methods
  - Array/List access methods should use helper methods (described later) declared in ValidatorActionFormEx.

```
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);
    }
}
```



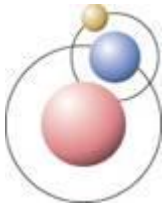
⑥ ActionForm Extensions

- **DynaValidatorActionFormEx**
  - Subclass of DynaValidatorActionForm
  - Below is an example of an implementation, where the developer need not implement a class. (only necessary properties are defined in Struts configuration file)
- **Implementation Example**

```
<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>
```

Necessary properties are defined

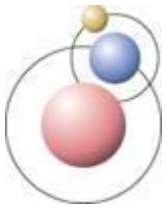
DynaValidatorActionFormEx is specified



⑥ ActionForm Extensions

- **FormEx Interface**
  - An Interface, implemented by ValidatorActionFormEx and DynaValidatorActionFormEx
  - Used internally within the framework, for processing the two ActionForms, without worrying of the difference between the two.
  - Developer need not implement this Interface.

TERASOLUNA



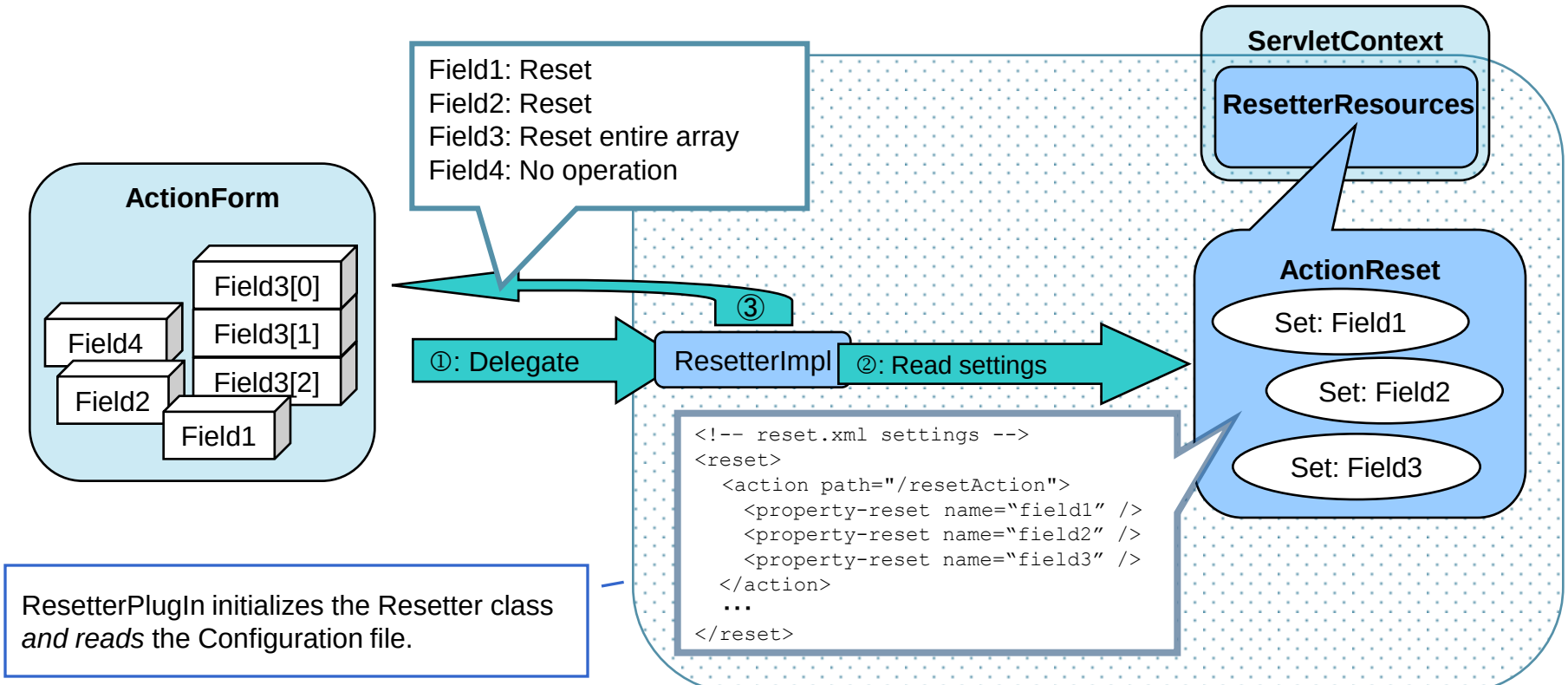
⑥ ActionForm Extensions

Auto-reset function

- The reset() *method* in ActionForm is overridden and the reset process is delegated to a class that implements the Resetter Interface.
- Using the default ResetterImpl, properties can be reset from a configuration file.
- To initialize fields such as HTML checkboxes, select boxes before setting a value
- For array/List properties, resetting within only a specified range is possible.
- TERASOLUNA framework provides the following items
  - Resetter ・・・An Interface, to invoke the reset process
  - ResetterImpl・・・Default Resetter Implementation Class
  - ResetterPlugIn ・・・A Plug-in to initialize Resetter class *and Reset configuration file*.
  - Reset configuration file ・・・Configuration file that must be prepared by the developer in the required format.

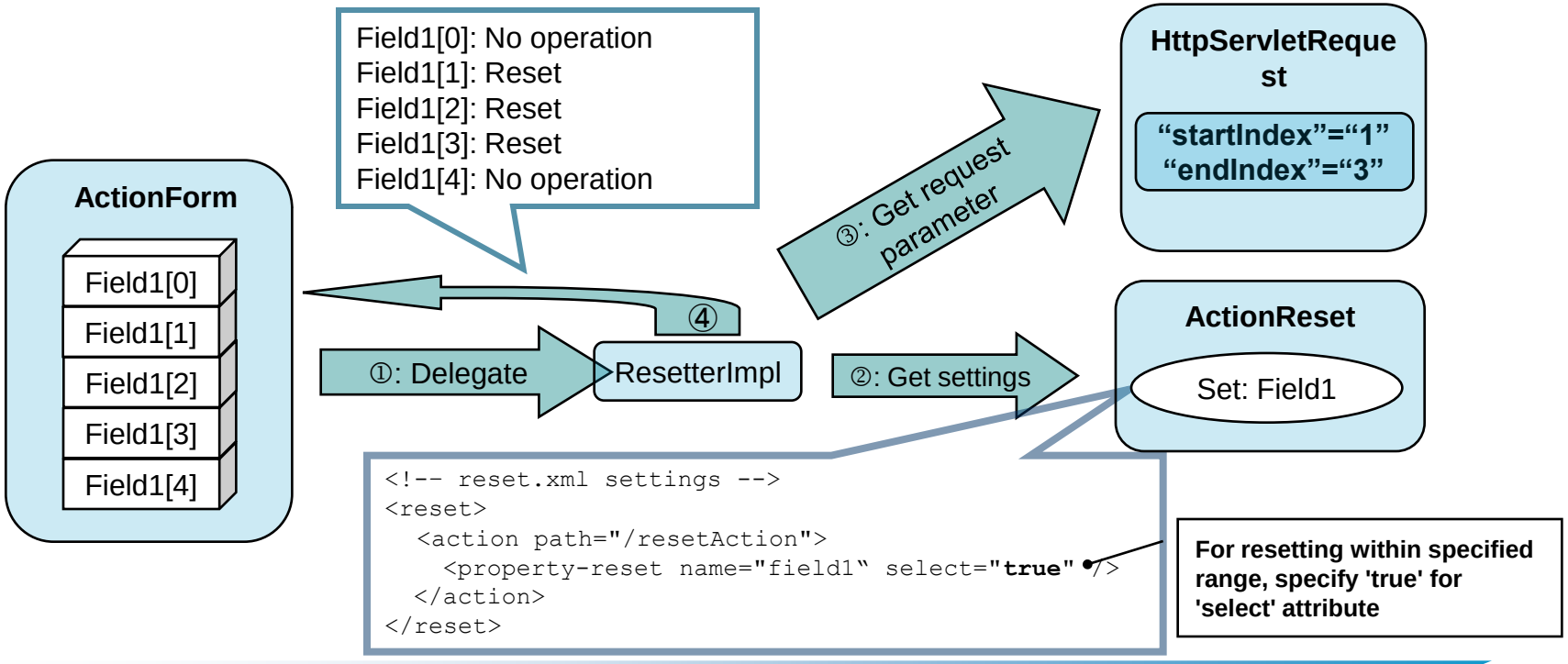
⑥ ActionForm Extensions

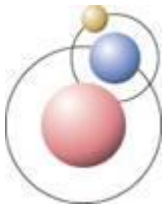
- **Figure showing Reset Process (ResetterImpl)**
  - Resets the fields specified in reset.xml



⑥ ActionForm Extensions

- **Figure showing the Reset process within specified range (ResetterImpl)**
  - Resets the specified Index
    - The reset range is specified by request parameters *startIndex* and *endIndex*.
    - Fields other than List/Array types are reset normally.





⑥ ActionForm Extensions

ResetterPlugIn

- Initializes the Resetter class from configuration file.
- It is provided as a PlugIn function for Struts (settings are mandatory for each module)

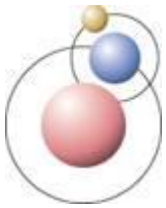
Example of PlugIn settings（Struts configuration file)

```
<!-- 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 class is specified

Resetter class is specified

Path of Reset configuration file is specified



⑥ ActionForm Extensions

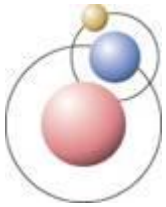
Reset configuration file

- Configuration file for ResetterImpl. Reset process is executed automatically.
- Set the property to be reset for every action path.
- Format

```
<reset>
    <action path="Action path">
       <property-reset name="Field name" select="select attribute"/>
    </action>
</reset>
```

  - *Action path*・・Set the action path, so that reset can be executed for that action.
  - *Field name*・・・Set the Form property name.
  - *Select attribute*・・・Set “true” to reset within specified range (default: false)
- File name can be freely decided. Specify the path in the property of ResetterPlugIn. (Refer to previous page)

TERASOLUNA

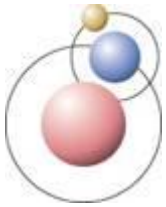


⑥ ActionForm Extensions

ActionForm having module scope

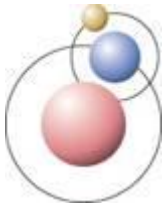
- The scope of an ActionForm supported by Struts is request/session. However, it is possible to use the Action form in Module scope.
- What is Module scope?
  - ‘session’ actually stores the Action form.
  - Attaching "_" at the beginning of Action form name, framework ensures the uniqueness of such a form in the session.
  - Action form is created for each module.
    →Shown as dynamic action form per module

TERASOLUNA



⑥ ActionForm Extensions

- **Helper method**
  - Useful methods are defined in Dyna/ValidatorActionFormEx (base class of ActionForm)
  - Get count method
    - Gets the size of array / List field
    - Helper methods for array / List
  - setIndexedValue method
    - Initialization of array/List can be skipped. The array/List dynamically increases size when the index range exceeds its size.
    - Above is also applicable to DynaValidatorActionFormEx, set(String, int, Object)
  - getIndexedValue method
    - When the index range exceeds, null is returned
    - Above is also applicable to DynaValidatorActionFormEx, get(String, int)

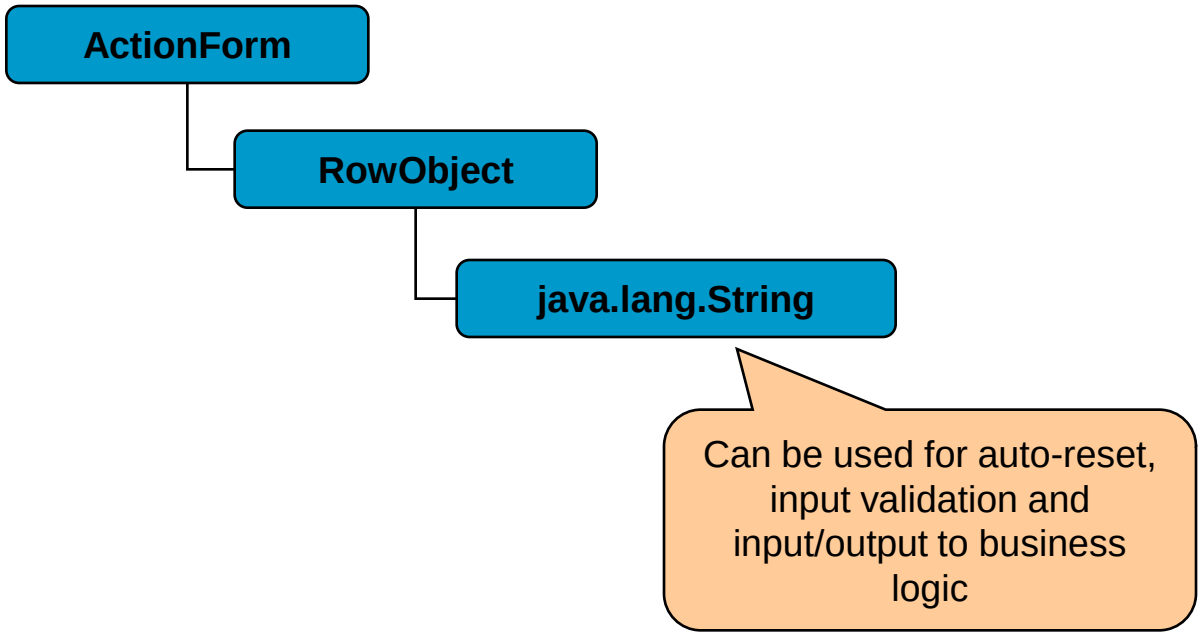


⑥ ActionForm Extensions

- **Control the Populate function**
  - A flag is set when the ActionForm is modified during execution of business logic, so that processPopulate does not execute later.
  - During the forward process on the server, processPopulate method of RequestProcessor in struts gets executed.
    - Action form value modified in business logic may be overwritten by the request parameters.
    - When the Action form value is modified in business logic, the above processPopulate process in Struts is automatically cancelled.

⑥ ActionForm Extensions

- Supplementary functions in ActionForm
  - Supports nested properties(not supported in TERASOLUNA Frame work for J2EE(Struts version)).

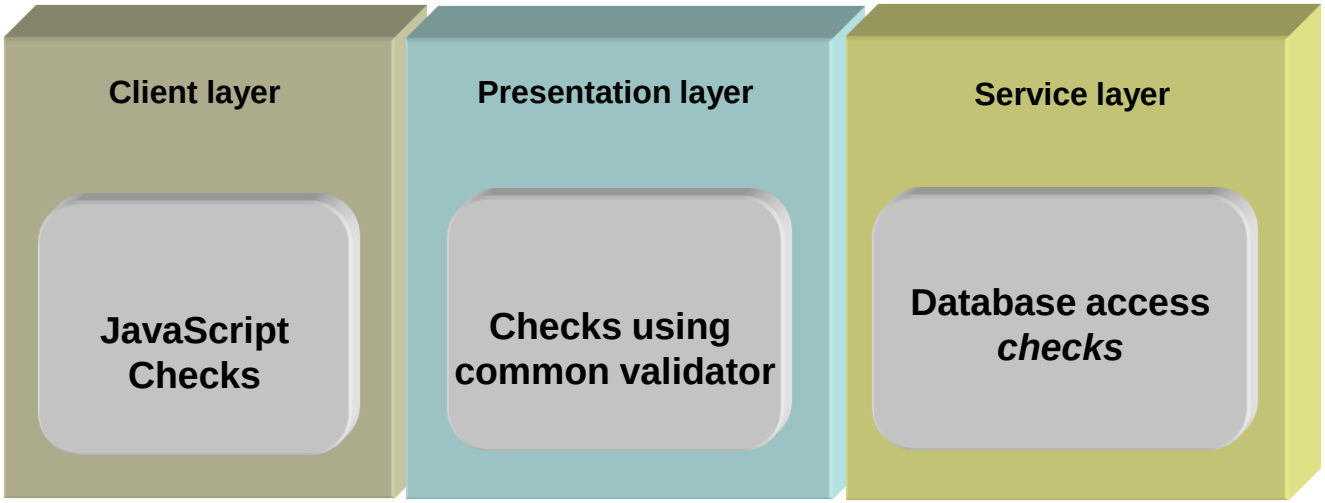


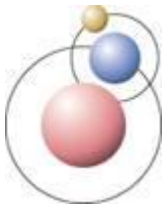
⑦ Input Validation

Input Validation

Outline

- Extends struts input validation and provides the typical validation rules that is required in business applications.
- Database access checks are executed in the service layer.



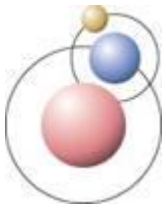


⑦ Input Validation

Description

- Types of input validation provided

	Outline	Provided by Struts		Extended by TERASOLUNA	
		Client	Presentation	Client	Presentation
required	Mandatory input check	○	○	-	-
requiredif	Correlation mandatory check	-	○	-	-
validwhen	Correlation check	-	○	-	-
minlength	Minimum input digit check	○	○	-	-
maxlength	Maximum input digit check	○	○	-	-
mask	Regular expression check	○	○	-	-
byte	byte type format check	○	○	-	-
short	short type format check	○	○	-	-
integer	int type format check	○	○	-	-
long	long type format check	-	○	-	-
float	float type format check	○	○	-	-



⑦ Input Validation

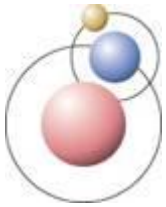
Types of input checks provided

	Outline	Provided by Struts		Extended by TERASOLUNA	
		Client	Presentation	Client	Presentation
double	double type format check	-	○	-	-
date	date type format check	○	○	-	-
intRange	int type range check	○	○	-	-
floatRange	float type range check	○	○	-	-
doubleRange	double type range check	-	○	-	-
creditCard	Credit card number format check	○	○	-	-
email	E-mail address format check	○	○	-	-
url	URL format check	-	○	-	-
alphaNumericString	Half-width alphanumeric string check	-	-	○	○
hankakuKanaString	Half-width Kana string check	-	-	○	○

⑦ Input Validation

- **Types of input checks provided.**

Rule	Outline	Provided by Struts		Extended by TERASOLUNA	
		Client	Presentation	Client	Presentation
hankakuString	Half-width string check	-	-	○	○
zenkakuString	Full-width string check	-	-	○	○
zenkakuKanaString	Full-width Kana string check	-	-	○	○
capAlphaNumericString	Capital Alphanumeric string check	-	-	○	○
number	Numerical check	-	-	○	○
numericString	Numeric string check	-	-	○	○
prohibited	Prohibited string check	-	-	-	○
stringLength	String length check	-	-	○	○
byteLength	Byte length check	-	-	-	○
byteRange	Byte length range check	-	-	-	○
dateRange	Date type range check	-	-	○	○
multiFieldCheck	Multiple field check	-	-	-	○

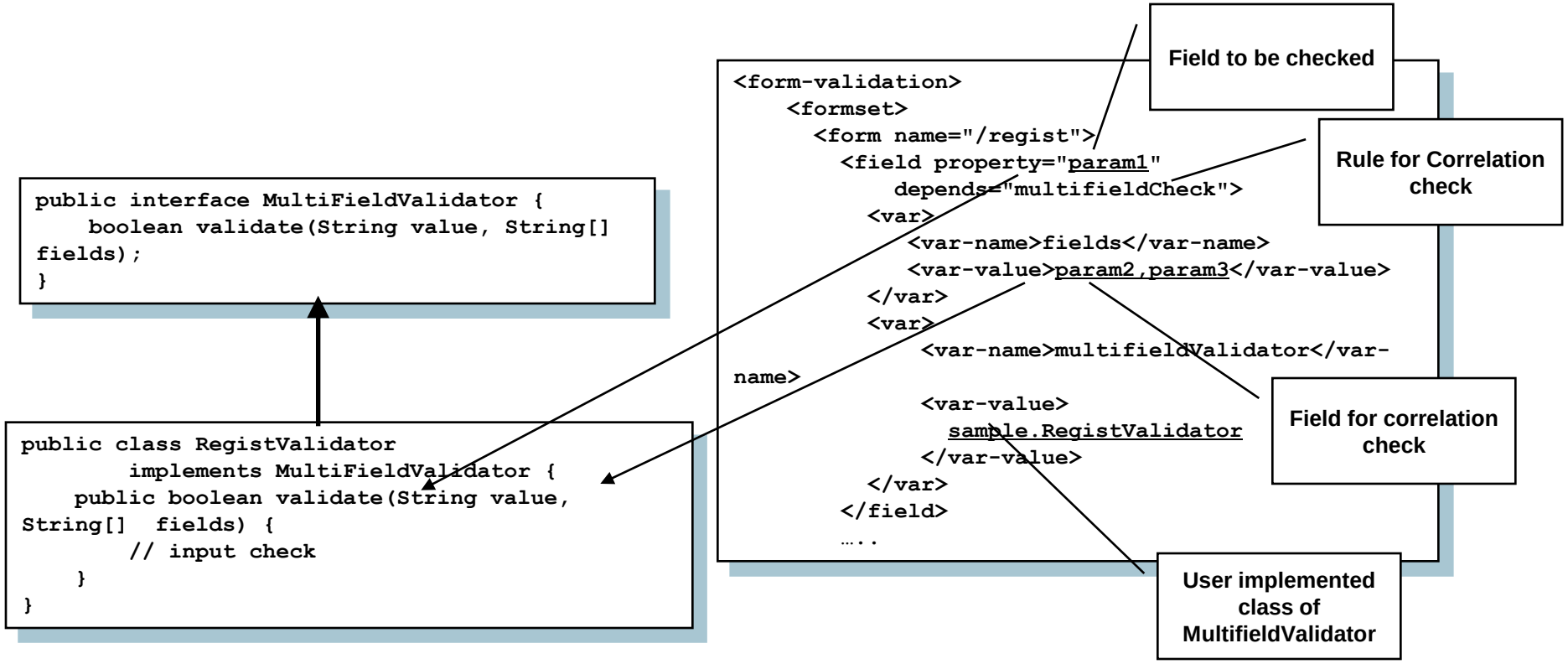


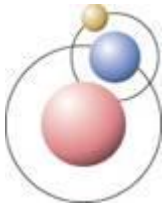
⑦ Input Validation

- **multiFieldCheck**
  - TERASOLUNA provides multiFieldCheck to validate the relation between multiple field values.
  - This is used for multiple fields for which requiredif, validWhen are not adequate.
  - *Checks based on Database access must be executed in Service Layer and not in multiFieldCheck*
  - Provides only the Interface, the developer must implement the actual validation logic.

⑦ Input Validation

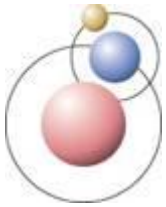
- **How to use the correlation check**
  - Implement *MultiFieldValidator* provided by framework.





⑦ Input Validation

- **Input validation by JavaScript**
  - Provides a tag library extension class, JavascriptValidationTagEX.
  - Provides a tag library for clients to execute rules provided by TERASOLUNA.
  - Attributes and usage is similar to the <html:javascript> tag.
  - In JavascriptValidatorTagEx, the entire list of full-width kana, half-width kana is fetched from FieldCheckEx and they are rendered using javascript variables.



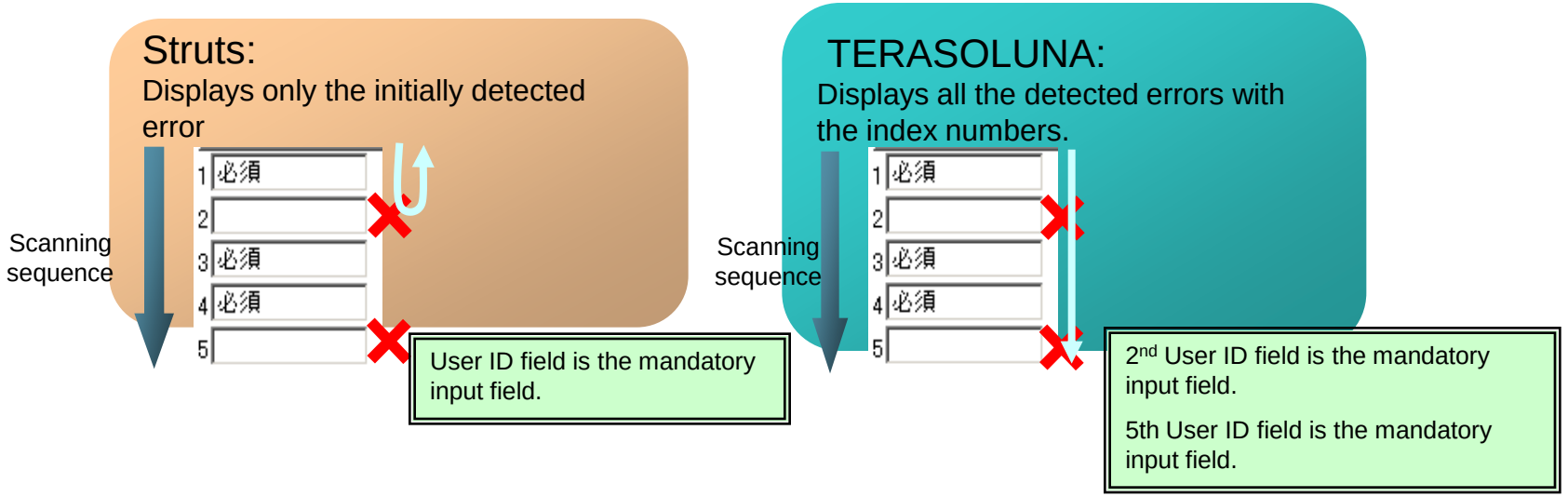
⑦ Input Validation

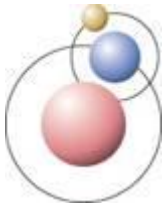
Validation for array/collection type fields

- In the validation rules provided by the validator framework, if there is an error validating array/collection types, validation of subsequent form properties is skipped.
- In TERASOLUNA, all properties are validated, and multiple error message can be displayed. The *index* number of the field *in the array* where error has occurred is also indicated.
  - Execute FieldChecksEx#validateArraysIndex() method for using this functionality.
  - However, when multiple validation rules are executed for a field, only one of them can be displayed at a time.

⑦ Input Validation

- Figure showing Mandatory check of array and collection type field





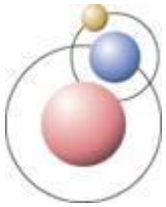
⑧ Action Extensions

Action Extensions

Outline

- Extends the Action class of Struts and provides the typical Action class necessary for an application.
- Provides ActionEx as the base Action class of framework.
- Action class is managed by DI container.

TERASOLUNA



⑧ Action Extensions

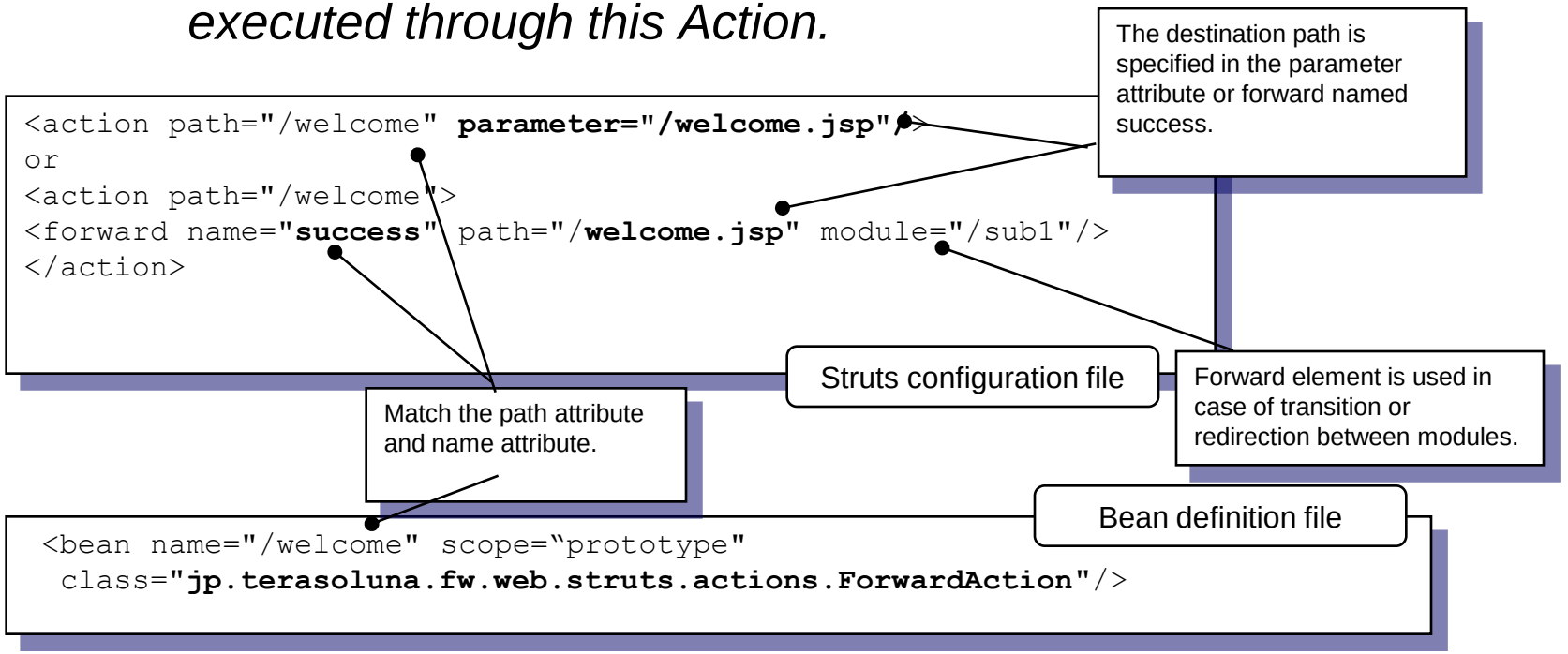
◆ ActionEx

- Base class of Action provided by framework.
- All Action classes provided by framework are derived from this class.
- It is recommend to inherit ActionEx for each new Action class in the application.
- Following functionalities are provided
  - Transaction token check functionality
    - » *Based on declaration in the Struts configuration file, provides checking and saving of transaction token.*
  - Message, error addition functionality
    - » Error/messages are added if they already exist in the session.

⑧ Action Extensions

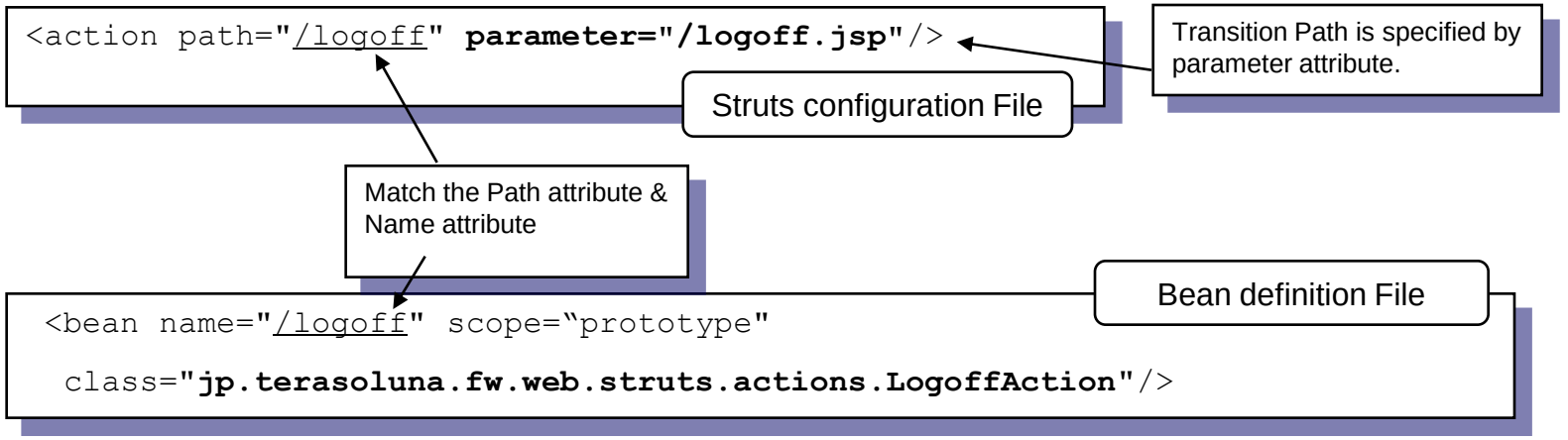
◆ ForwardAction

- This is the Action class that *only* forwards control to JSP and other physical/logical resources.
- Instead of direct access to JSP, *screen display must be executed through this Action.*



⑧ Action Extensions

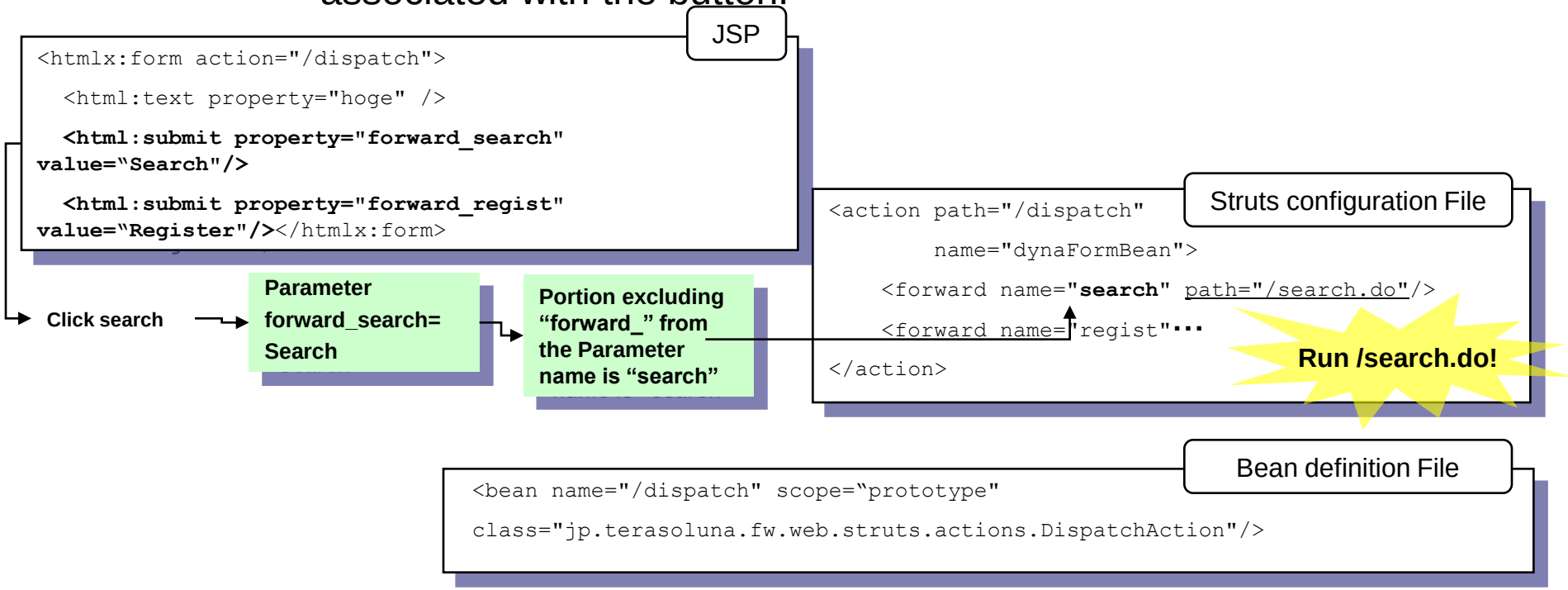
- LogoffAction
  - This is the Action Class *that forms the base* for the logoff process.
  - *Invalidates the* current session. This Class should be inherited if any other processing is required.



⑧ Action Extensions

DispatchAction

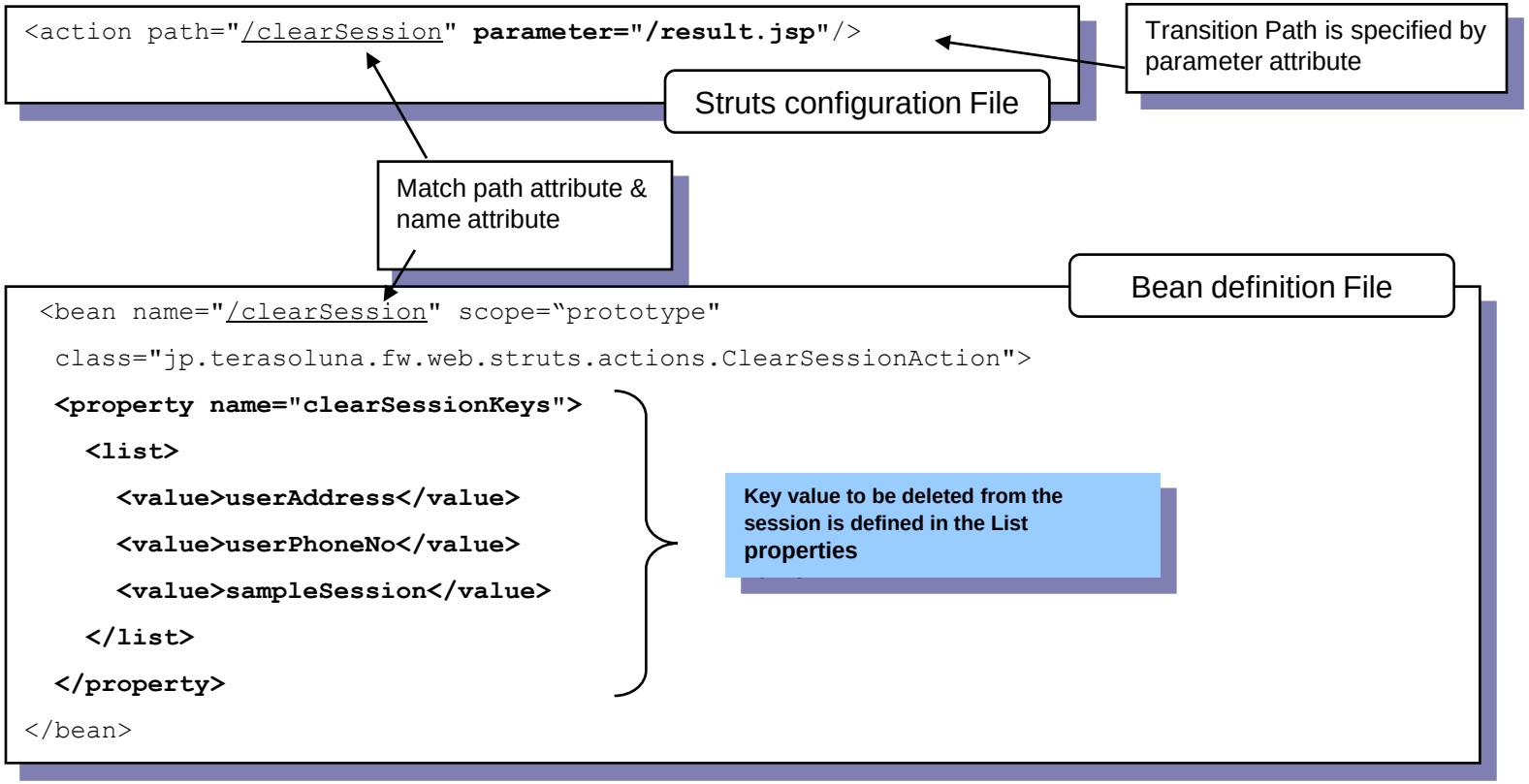
- When multiple buttons are available on the screen, this is the action to dispatch the submit for each button.
- On the server-side, the submit is forwarded to the action associated with the button.



⑧ Action Extensions

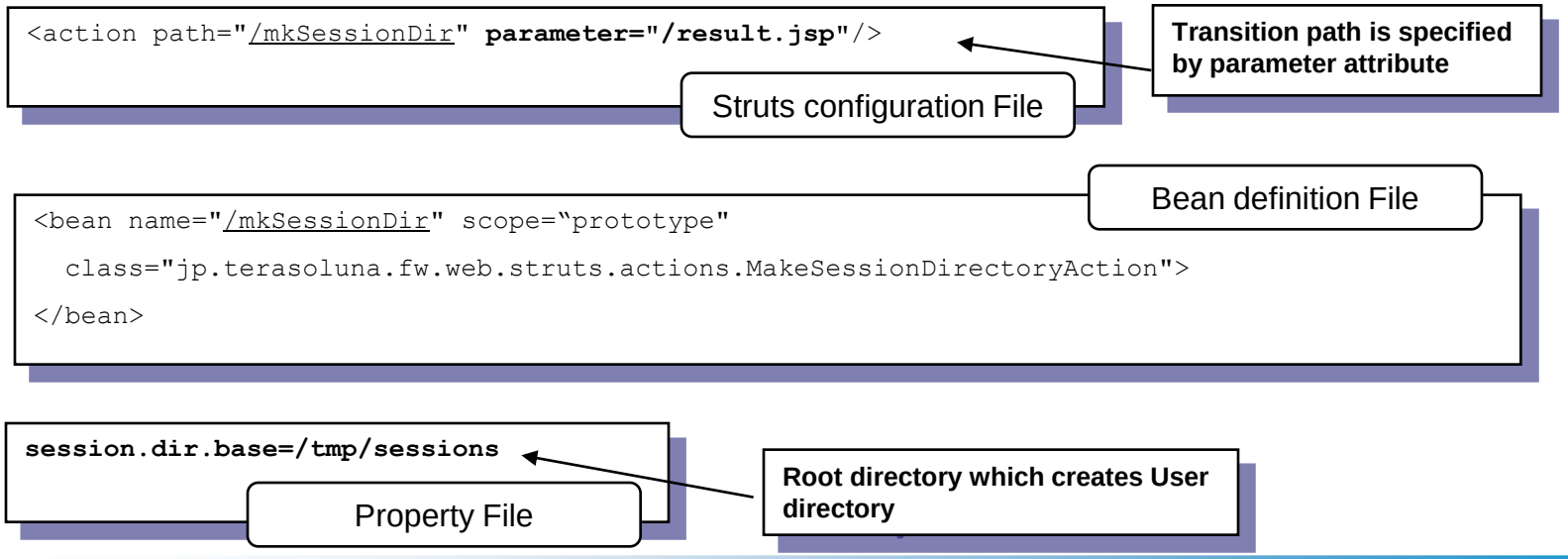
ClearSessionAction

- Removes the value of specified key from the session.



⑧ Action Extensions

- **MakeSessionDirectoryAction**
  - Action Class that creates a user-specific directory
  - It can be used as a temporary storage directory for reports.
  - Directory name is derived from the session ID
  - When HttpSessionListenerImpl is set, the user directory is automatically deleted at the time of log off.



⑧ Action Extensions

ReloadCodeListAction

- Updates the specified Code list.

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

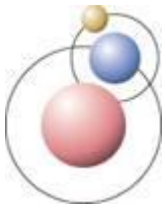
Transition Path is specified by Parameter attribute

Struts configuration File

Bean definition File

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

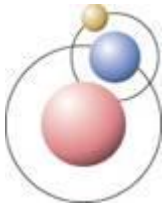
ReloadableCodeListLoader to be updated is specified



⑧ Action Extensions

- AbstractBLogicAction
  - Base Class of Action classes that execute business logic
  - The input/output process of business logic is delegated to AbstractBLogicMapper class. (as per settings in configuration file)
  - The Message Object generated in Service layer is converted into the Message Object of Struts
- BLogicAction
  - Action Class that executes business logic class *and* implements the BLogic interface.

※ Refer to next Chapter for Business Logic Execution details



⑨ System Exception Handling

System Exception Handling

- Provides a class to handle system exceptions using exception handling functionality of Struts
  - SystemExceptionHandler
    - Gets the message key from the thrown SystemException, *Fetches the message and sets it* in SystemException Class
    - When a SystemException is thrown in Business Logic, if this ExceptionHandler class is used, messages are fetched automatically.

```
<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>
```

ActionMessage is saved in the specified Message Key.

Screen is transferred to the Path specified by path attribute.

type is SystemException

className is ExceptionConfigEx

Set SystemExceptionHandler in handler

Specify module in set-priority, when the application is divided into modules.

Specify loglevel to set the level of log output

⑨ System Exception Handling

- DefaultExceptionHandler
  - It is possible to set the Log output level in this log handler
  - All Exceptions, except the SystemException thrown in the Business Logic, are handled.

```
<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>
```

ActionMessage is saved in the specified Message Key.

Screen is transferred to the Path specified by path attribute.

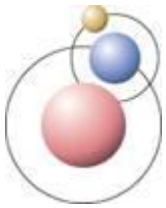
type is java.lang.Exception

className is ExceptionConfigEx

Set DefaultExceptionHandler in handler

Specify module in set-priority, when the application is divided into modules.

Specify loglevel to set the level of log output

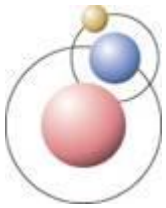


⑩ Business Logic Execution

Business Logic Execution

- Provides methods to execute Business Logic
  - For Business Logic Class, any of the following methods can be selected
    - Implementing BLogic Interface
    - Implement as a POJO
- Transfer data from Web layer ⇔ Service layer
  - The data transfer between layers can be automated based on the Configuration File
- Transaction Control
  - Use transaction control by SpringFramework
  - The developer can manage transactions declaratively without going into the details of the transaction API.

TERASOLUNA



⑩ Business Logic Execution

◆ Execution methods of Business Logic (1)

- *Use* of BLogic Interface
  - Implement the business logic as per BLogic Interface

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

  - Input *to the BLogic Interface*
    - » Implement as a POJO
    - » Define required properties as per screen/business logic specifications.
  - Return Value *of the BLogic Interface*
    - » Use the **BLogicResult class.**
    - » Result Object, Message information, Error information, Result String can be set.

TERASOLUNA

⑩ Business Logic Execution

- BLogic Interface Implementation example

```
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;
    }
}
```

Specify Generic type for input value class.

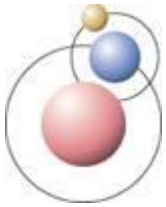
Input Value is obtained from Input Value Class

Output Object. Here, implemented in POJO. Map type can also be used.

Output object is set in return value

Message is set in return value.

The result of business logic processing is indicated by a logical String value.



⑩ Business Logic Execution

- **Configuration File**
  - Uses BLogicAction class

Struts configuration file

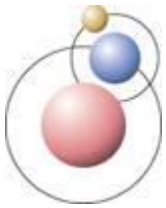
```
<action path="/add"
        name="addForm">
    <forward name="success" path="/add.jsp" />
</action>
```

Bean definition class

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

BLogic implemented class is injected (DI) in businessLogic properties of BLogicAction

TERASOLUNA

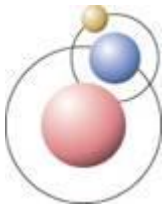


⑩ Business Logic Execution

Execution method of Business Logic(2)

- Implementing business logic using POJO
  - When portability (to use anywhere other than TERASOLUNA) of business logic is to be maintained, it is necessary to implement it in POJO.
  - Implement such that POJO does not depend on API of TERASOLUNA.
  - Input/Output Value of Business Logic
    - » Only the necessary values are to be specified in interface of business methods.
  - Create an Action class implementing AbstractBLogicAction that acts as a bridge to the Web layers (Struts,TERASOLUNA)

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



⑩ Business Logic Execution

- Implementation Example of AbstractBLogicAction and POJO

```
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;
    }
}
```

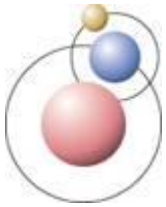
Specify Generic type for the argument of doExecuteBLogic

BusinessLogic implemented in POJO.
Set using DI Container
※ Here, getter/setter are omitted.

BusinessLogic is called. BusinessLogic is not written in Action class.

Return value of doExecuteBLogic is same as the return value of BLogic interface. Here, 'Map' is used as result object. Instead POJO can also be used.

TERASOLUNA



⑩ Business Logic Execution

- Implementation Example of AbstractBLogicAction and POJO

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

This business logic can be used in any application by implementing it so that it does not depend on Web layer, presentation layer, Struts, TERASOLUNA.

- Configuration File

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

Struts configuration file

Bean definition file

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

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

Business logic in POJO is injected (DI) in AddAction.

TERASOLUNA

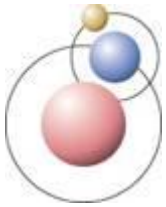
⑩ Business Logic Execution

Input/Output settings of Business Logic

- By describing input/output settings in configuration file, data transfer between Web layer ⇔Service layer can be automated.

- Format

Elements	No of elements	Attributes	Mandatory	Description
blogic-io	-	-	○	Root element.
blogic-params	1	bean-name	-	Configures the input parameter. Specify the class of input parameter in the bean-name attribute. (Not required when there is no input parameter)
set-property	n	property	○	Physical name in the Web layer for the input value.
		blogic-property	-	Physical name in the service layer of input value. It must be the name of an attribute on the bean name class.
		source	○	The source of the input value. request. session, form and application can be specified. The default value is request.
blogic-result	1	-	-	Configures the output value.
set-property	n	property	○	Physical name in Web layer of output value.
		blogic-property	-	Physical name in service layer of output value. It must be the name of an attribute on the result object.
		dest	○	The destination of output value. request, session, form can be specified. The default value is request.



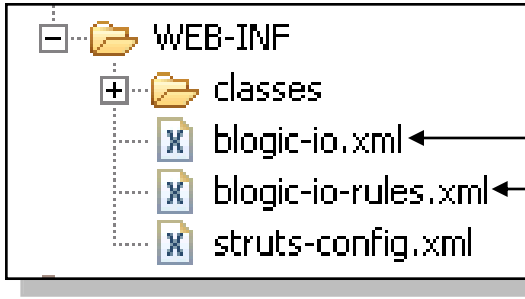
⑩ Business Logic Execution

- **Initialization of business logic input/output settings**
  - Initialize by using Struts plug-in functionality.
  - Settings Example

```
<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>
```

Struts configuration file

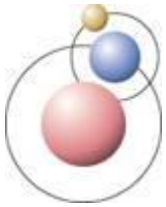
Class that executes the business logic input/output process. BLogicMapper is the default class provided by the framework.



Business logic input/output configuration file. File name is optional.

Rule file to parse the file containing business logic input/output configuration. Unless required, the file provided by framework can be used as it is.

TERASOLUNA

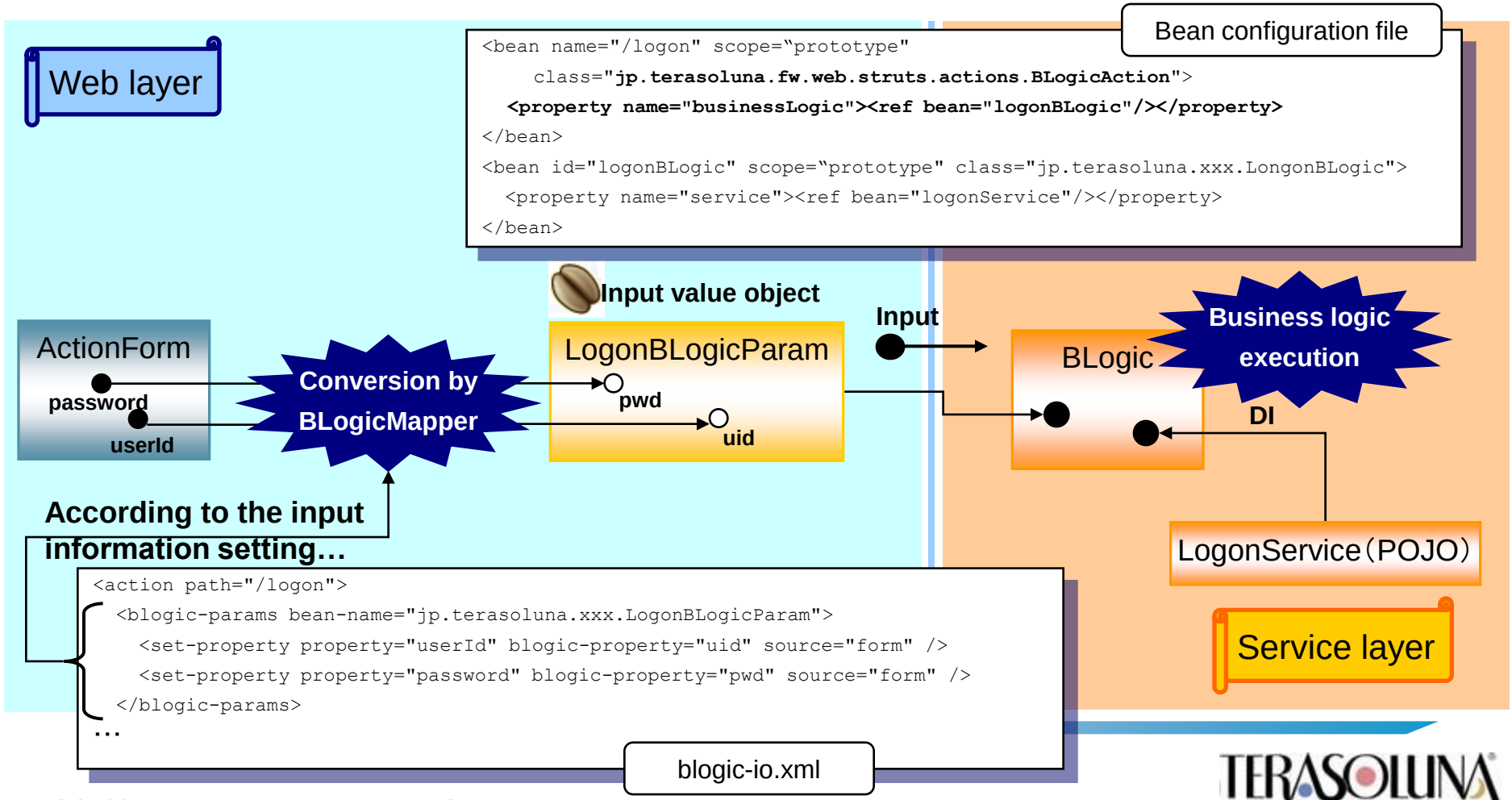


⑩ Business Logic Execution

- **Business logic input/output process**
  - Execute the process in the class that implements AbstractBLogicMapper.
  - Specify the implementation class in the Plug-in configuration file (Refer to the previous page)
  - A default class - BLogicMapper that can execute simple input/output process for Request, Session, and Action form, is provided.
  - The AbstractBLogicMapper implementation class does the conversion between input/output objects and the objects indicated by source/dest attribute in the business logic input/output file. (blogic-io.xml)
    - Input process method
      - `getValueFrom<String specified by source attribute>`
    - Output process method
      - `setValueTo<String specified by dest attribute>`
  - In default BLogicMapper, you can specify request, session, form and application in the source attributes, and you can specify request, session, form in the dest attributes. Respectively, input/output between HttpServletRequest, HttpSession, ActionForm, and ServletContext is possible.

⑩ Business Logic Execution

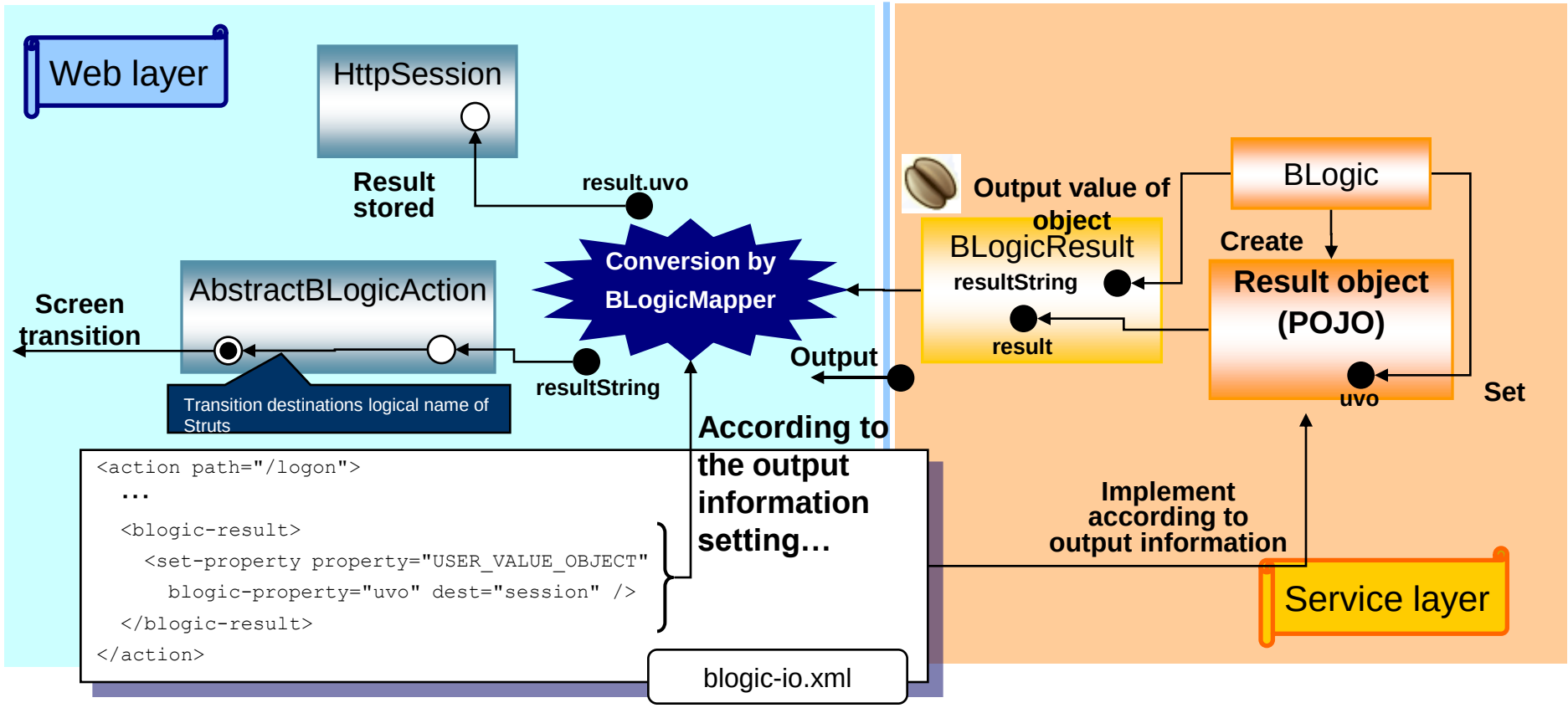
- Input setting example and image of Business Logic
  ※ Here, BLogic is used.

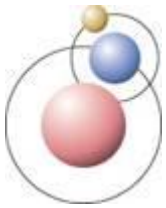


⑩ Business Logic Execution

- In case of output

※ Here, BLogic is used.





⑪ Transaction Control

Transaction Control

- Framework executes commit/rollback.
  - Transaction control is injected by AOP (developer need not implement any transaction code)
  - Transaction is started when service is started and it is committed when service is ended. Rollback is executed in case of exception.
  - Throw an exception when transaction needs to be rolled back due to errors such as validation error in input value.
- Transaction control considers service layer object as a boundary.
- 1 service is considered as 1 transaction.

⑪ Transaction Control

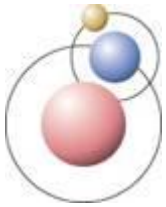
Example of Transaction Control Image (Bean definition file) 1/3

```
<?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">
```

Settings to use Bean definition file based on the schema.

Aop schema settings to be used in AOP settings and tx schema settings to be used in transaction settings, are set here

Specify the white-space separated schema location respective to each namespace



⑪ Transaction Control

Example of Transaction Control Image (Bean definition file) 2/3

```
<!--  Transaction manager for a single JDBC data source-->
<bean id="transactionManager" class="org .…….
DataSourceTransactionManager">
  <property name="dataSource" ref="dataSource"/>
</bean>

<!-- Transaction Settings -->
<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>
```

The class provided by Spring. Specifies which transaction control is executed by which method.

Tag provided by Spring. Settings to enable declarative transaction.

<tx:method/> element is used to specify what kind of transaction management is to be enable for which method.

In case of default settings of Spring, Roll-back is not executed even if custom exception is thrown.
As per the example, Rollback is enabled by specifying the class name of the exception in the rollback-for Attribute.

TERASOLUNA

⑪ Transaction Control

Example of Transaction Control Image (Bean definition file) 3/3

```
 <!- AOP settings-->
<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>
```

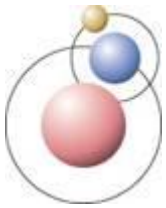
Tag provided by Spring.
AOP settings are done.

Specify the BeanID that applies the interceptor and attach the pointcutID

Connect the interceptor and pointcutID

```
<!- Definition of Business Object -->
<bean id="sumService"
  class="jp.terasoluna.sample2.service.impl.SumServiceImpl"/>
</bean>
```

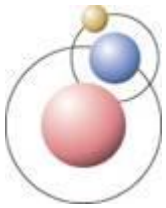
Business Logic programmer implements the settings in the red dotted line



⑫ Database Access

Database Access

- Provides the Data Access Object interface.
  - QueryDAO…DAO Interface used for query based database operations
  - UpdateDAO…DAO Interface used for update based database operations
  - StoredProcedureDAO…DAO Interface to execute stored procedures
  - QueryRowHandleDAO…DAO Interface used to process the results of query based database operations, one record at a time
- Hide the dependency to RDBMS or O/R mapping tools, in the DAO implementation class.
- Provides iBatis as the default implementation of DAO.
- Declarative transaction control by SpringAOP is used in the Service layer. So, the developer need not be aware of the transaction control of database connection.



⑫ Database Access

- **Implementation of Database access in Service layer.**
  - Inject the DAO instance From DI container to Business Logic

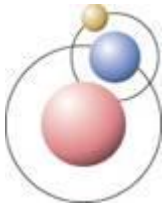
Business Logic

```
public class QueryBLogic {
    private QueryDAO dao = null; // Declare DAO as an interface.
    …dao getter/setter methods…….
.
    public Object dbAccess() {
        SelectUserArgBean bean = new SelectUserArgBean(); //POJO for substituting
parameter values in SQL placeholders
        …….
        SelectUserResult result =
            dao.queryForObject("select.user", bean, SelectUserResult.class);
        // first argument is the ID of settings to be executed (depends on the DAO
class)
    // second argument is a POJO that contains values to be set in SQL parameters.
Argument is null when not required.
        // third argument is the class of the return value.
       ……
```

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

Bean definition file

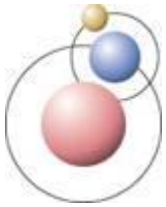
TERASOLUNA



⑫ Database Access

- **Default implementation of DAO that uses iBatis**
  - Provides the following classes where iBatis functionality is used
    - QueryDAOiBatisImpl
    - UpdateDAOiBatisImpl
    - StoredProcedureDAOiBatisImpl
    - QueryRowHandleDAOImpl

TERASOLUNA



⑫ Database Access

- Settings
  - Bean definition file… Sets the data source, sqlMapClient.

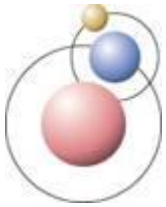
Bean Definition File

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

sqlMapClient

Path of iBatis configuration file

Settings of data source



⑫ Database Access

- Settings
  - Bean definition file (Continued)… Set DAO.

Bean definition file

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

DAO defined here is injected(DI) in business logic.

sqlMapClient is injected (DI) in DAO.
(Same in case of UpdateDAOiBatisImpl, StoredProcedureDAOiBatisImpl)

  - sqlMapConfig.xml settings
    - » A special definition is not required. Set only the path to sqlMap.xml.
    - » Match it to the path described in sqlMapClient settings of bean definition file.

sqlMapClient.xml

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

Settings of each data access are described in sqlMap.xml.

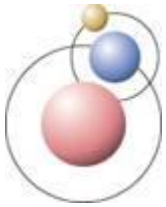
  - sqlMap.xml settings
    - » Provide settings for each data access to be executed.
    - » Create as per iBatis specifications.

sqlMap.xml

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

Set the class name that stores the result of data access.

TERASOLUNA

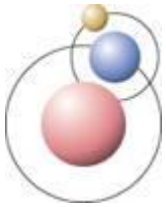


⑫ Database Access

- **Following functions are provided by using iBatis**
  - Dynamic SQL execution
    - In case of change in WHERE clause based on conditions:

```
<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>
```


TERASOLUNA

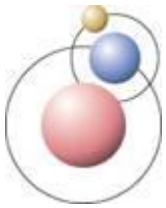


⑫ Database Access

- **Following functions are provided by using iBatis (Contd.)**
  - Dynamic SQL execution
    - Use iterate when number of “?” in IN clause or number of elements in OR clause cannot be determined.

```
1  <!-- prepend: Add AND or OR etc at the beginning of conditional
expression-->
2  <!- open: String to be added at the beginning of iteration-->
3  <!- conjunction: String to add AND or OR etc at the beginning of
each iteration -->
4  <iterate prepend=”AND” property=”userNameList”  open=”(” close=”)”
conjunction=”OR”>
5    username=#userNameList[]#
6  </iterate>
```


TERASOLUNA



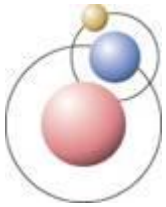
⑫ Database Access

- **Following functions are provided by using iBatis. (Contd.)**
  - Fetch specified count
    - Specify fetch start index and fetch count.
    - Can be used in list view with number of pages.

```
// dao is the field of QueryDAO interface type.
// Field definition of dao and setter/getter are separate and they
are omitted here.
List<UserBean> bean
        = dao.executeForObjectList(
             "getUser",null, 20, 10);
// or
UserBean[] bean
        = dao.executeForObjectArray(
             "getUser",null, UserBean.class, 20, 10);
```

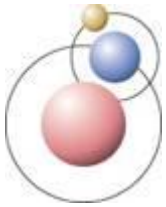
      - » The above example gets 10 user names from 21st user name onwards that matches with the specified SQL conditions.

    ※ Since this method loops over the resultset using java.sql.ResultSet#next() till it reaches the end of specified count, it may cause performance degradation. Do SQL tuning in case of significant performance degradation.



⑫ Database Access

- **Following functions are provided by using iBatis (Contd.)**
  - Data cache
    - Cache data can be flushed based on valid time for cache data and specified SQL execution timing.
      - When time is not specified, data is saved permanently
    - Result data is saved in static HashMap (Synchronized by Collections class)
      - A single storage area is created in VM. So, in cluster environment, it is difficult to use the data flash by executing specified SQL.
    - There are four types of cache
      - MEMORY: Weak（java.lang.ref.WeakReference）,Soft（java.lang.ref.SoftReference）and Strong (Independent Static class of iBatis) are set in child element. In case of Strong, existence of cache data continues unless it is explicitly deleted.
      - LRU: Removes the least recently used item
      - FIFO: First In First Out.
      - OSCACHE: Cache using the OSCache of OpenSymphony project.



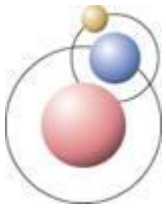
⑫ Database Access

- **Following functions are provided using iBatis (Contd.)**
  - Data cache functionality

[sql-config.xml File]

```
<!–Data is cached for 24 hours and updated by using insert, update, delete
<   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>
```


TERASOLUNA

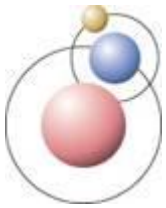


⑬ Screen Display (Custom Tags)

Custom Tags provided by terasoluna-thin

- **Authorization Check**
  - `<t:ifAuthorized>`
  - `<t:ifAuthorizedBlock>`
- **Server Blockage Check**
  - `<t:ifPreBlockade>`
- **Calendar Input**
  - `<t:inputCalendar>`
- **String Display**
  - `<t:write>`
- **Date Conversion**
  - `<t:date>`
- **Wareki Date Conversion**
  - `<t:jdate>`
- **Decimal Display**
  - `<t:decimal>`
- **Trim Functions**
  - `<t:rtrim>`
  - `<t:ltrim>`
  - `<t:trim>`
- **Substring Functions**
  - `<t:left>`
- **Code List Definition**
  - `<t:defineCodeList>`
- **Code List Count Output**
  - `<t:writeCodeCount>`
- **Specified Code List Value Display**
  - `<t:writeCodeValue>`

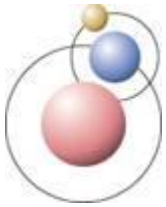
TERASOLUNA



⑬ Screen Display (Custom Tags)

Custom Tags provided by terasoluna-thin

- **Style Class Change**
  - `<ts:changeStyleClass>`
- **Message Display**
  - `<ts:errors>`
  - `<ts:messages>`
- **No Cache Form Tag**
  - `<ts:form>`
- **No Cache Link**
  - `<ts:link>`
- **Form Target Specification**
  - `<ts:submit>`
- **Message Popup**
  - `<ts:messagesPopup>`
  - `<ts:body>`
- **Error Message Check**
  - `<ts:ifErrors>`
  - `<ts:ifNotErrors>`
- **Client Check Extension**
  - `<ts:javascript>`
- **List Display Related**
  - Listing Method
    - `<logic:iterate>`
  - Page Link
    - `<ts:pageLinks>`

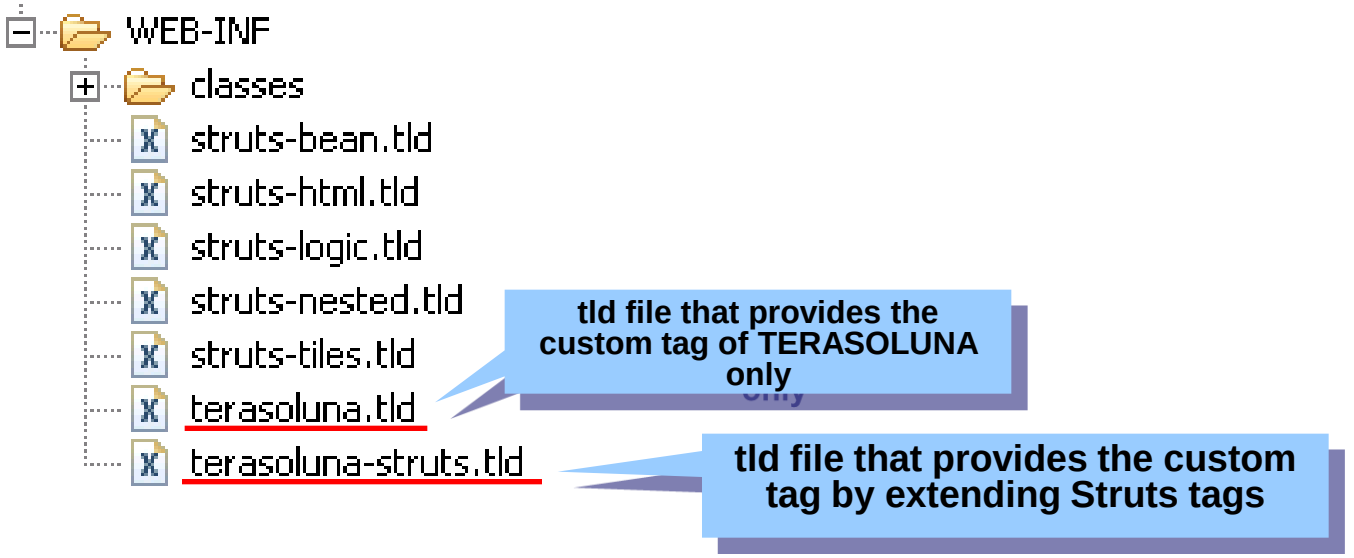


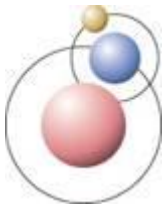
⑬ Screen Display (Custom Tags) ~ Custom Tag Setting

Tag Settings

- Custom Tags provided by Framework are classified as
  - Independent Custom Tags of TERASOLUNA
  - Tags extended from tags provided by Struts

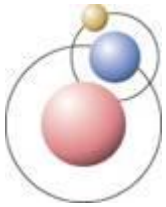
Since the Configuration files (tld) of these tags are separate files, create respective tld file to use these tags.





⑬ Screen Display (Custom Tags) ~ Authorization Check (1/3)

- **Outline**
  - Determines whether the currently logged on user has access permission to the specified path and accordingly switches between show / hide.
- **Description**
  - <t:ifAuthorized>
    - Evaluates the body of tag only when there is an access permission for the path specified by path attribute. AuthorizationController executes the access permission check.
  - <t:ifAuthorizedBlock>
    - Since this element controls the result of <t:ifAuthorized> element for each blockId, it links with <t:ifAuthorized> element using blockId, and determines whether to display it in the body.
      - By nesting this element, each access permissions can be controlled with flexibility. While nesting, it links the blockId attribute of parent with parentBlockId attribute of child, and determines whether to display it in the body.



⑬ Screen Display (Custom Tags) ~ Authorization Check (2/3)

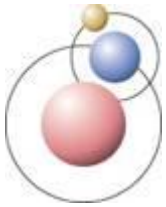
- Example of <t:ifAuthorized> element

```
<t:ifAuthorized path="/pathToSomewhere">
     Output if having access right to specified path

</t:ifAuthorized>
```

- Example of <t:ifAuthorizedBlock> element

```
<t:ifAthorizedBlock blockId="ABC" >
    Displayed only when
    IfAuthorizedBlockTag attached by blockId in the body is displayed.
    <t:ifAthorizedBlock blockId="EFG" parentBlockId="ABC" >
     Displayed only when IfAuthorizedTag
     attached by blockId in the body is displayed.
        <t:ifAthorized path="/sample1/test.do blockId="EFG" >
            It is output when the user has access rights for specified path.
        </t:ifAthorized>
    </t:ifAthorizedBlock>
</t:ifAthorizedBlock>
```

※Refer to the page (1) Authentication/Access Control for the settings of AuthorizationController



⑬ Screen Display (Custom Tags) ~ Authorization Check (3/3)

- List of attributes of <t:ifAuthorized> element

Attributes	Mandatory	Outline
path	○	Specifies targeted path.
blockId	-	blockId for association with <htmlx:IfAuthorizedBlock> element, which is parent of the specified tag.

- List of attributes of <t:ifAuthorizedBlock> element

Attributes	Mandatory	Outline
blockId	○	Specifies targeted blockId.
parentBlockId	-	blockId for association with <htmlx:IfAuthorizedBlock> element, which is parent of the specified tag.

⑬ Screen Display (Custom Tags) ~ Server Blockage Check

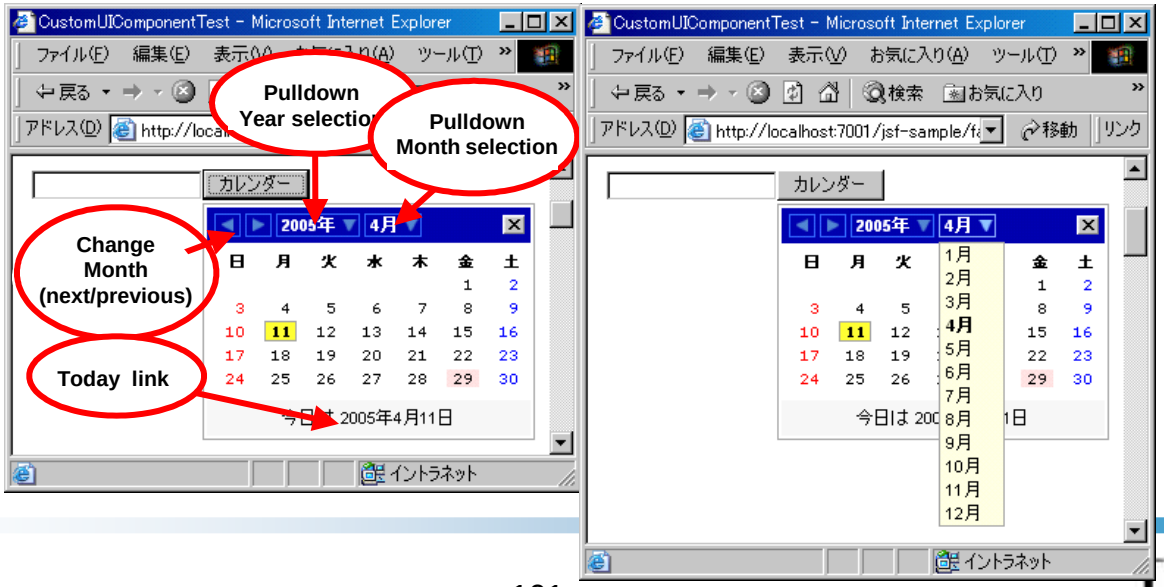
- **Outline**
  - Switches between show/hide by determining whether server is in pre-blockage state or blockage state.
- **Description**
  - <t:ifPreBlockade>
    - Body part of tag is displayed only when server is in blockage state or pre-blockage state. Server blockage check is executed by ServerBlockageController.
  - Example

```
<t:ifPreBlockade>
        ... // Items displayed only when server is in blockage state
or pre-blockage state.
</t:ifPreBlockade>
```

※ Refer to the page (1) Authentication/Access Control for the settings of AuthorizationController

⑬ Screen Display (Custom Tags) ~ Calendar Input (1/6)

- **Outline**
  - Provides Calendar Input functionality for specified text fields.
- **Description**
  - <t:inputCalendar>
    - Display the calendar screen using JavaScript , and enter the selected date in specified input field.



⑬ Screen Display (Custom Tags)
~ Calendar Input (2/6)

Functional description

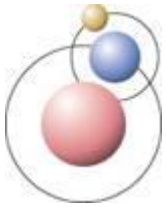
- Configuration file
  - Message resource file is used for internationalization.
  - Message resource file named “calendar.properties” must be created exactly under the class path.
- Holiday definition
  - In message resource file, holidays in the calendar can be specified as follows

```
calendar.holiday.1=0,1,1,New year
calendar.holiday.2=0,2,11,National Found
calendar.holiday.3=0,4,29,Greenery Day
calendar.holiday.4=2005,1,10,Coming-of
calendar.holiday.5=2005,3,20,Spring Equi
calendar.holiday.6=2005,9,19,Respect for
calendar.holiday.7=2005,9,23,Autumnal Eq
calendar.holiday.8=2005,10,10,National S
```

In message resource key must start with “calendar.holiday.” and after that, sequence number is added starting with “1”.

Parameters can be set as “Year”, ”Month”, ”Day” and ”Holiday description” with comma (,) as separator.

For holidays, that occur on the same date every year, “0” is specified for “Year”.



⑬ Screen Display (Custom Tags) ~ Calendar Input (3/6)

◆ Customize the string displayed on button

- The string on button to display the calendar can be changed by specifying the code in message resource file as given below.
- Set the message resource key, “calendar.button.string”. Default is “Calendar”.

```
calendar.button.string= Calendar
```

◆ Change prefix style

- The prefix of the style sheet used in calendar and the prefix of image file can be changed by specifying the code in message resource file as given below.
- Set the message resource key, “calendar.style.themeprefix”. Default is “BlueStyle”.

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

⑬ Screen Display (Custom Tags)
~ Calendar Input (4/6)

◆ Change the string displayed with current date

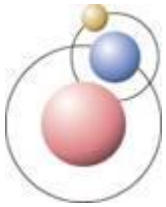
- The string assigned for the current date that is displayed at the bottom of the calendar can be changed by specifying the code in message resource file as given below.
- Set the message resource key, “calendar.today.string”. Default is “Today is”.

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

◆ Change the calendar image in save/store position

- The file location of the image to be used in the calendar input functionality can be changed by specifying the code in message resource file as given below.
- It should end with “/”. File location of the image can be changed. However, the image file name cannot be changed.
- Set the message resource key, “calendar.img.dir”. Default is “img/calendar/”.

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



⑬ Screen Display (Custom Tags) ~ Calendar Input (5/6)

Change the file location of style sheet

- The file location of the style sheet to be used in calendar input functionality can be changed by specifying the code in message resource file as given below. File location should end with “/”.
- File name of style sheet used by this functionality is “<Prefix> + InputCalendar.css”.
- Set the message resource key, “calendar.stylesheet.dir”. Default is “css/”.

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

Change the file location of external JavaScript file

- The file location of external JavaScript to be used in calendar input functionality can be changed by specifying the code in message resource file as given below. File location should end with “/”.
- File name of the JavaScript used by this functionality is “InputCalendar.js”.
- Set the message resource key, “calendar.javascript.dir”. Default is “js/”.

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


TERASOLUNA

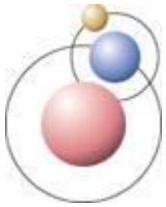
⑬ Screen Display (Custom Tags) ~ Calendar Input (6/6)

◆ Example

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

◆ Attributes list

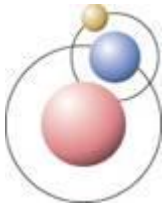
Attributes	Mandatory	Outline
for	O	Specifies the input field where the selected date is to be entered.
format	-	Specifies the format of calendar. Date format can be specified as “y(Year)”, “M(Month)”, “d(Day)”, and either of the delimiters ”/”, “-”, “.”, “Half width space” can be used. Only one type of delimiter character should be used. Multiple delimiters such as “yyyy/MM-dd” cannot be used.
formatKey	-	Specifies the key to get the calendar format from message resource.



⑬ Screen Display (Custom Tags) ~ Text Display (1/3)

- **Outline**
  - Converts the value of specified bean property and displays it.
- **Description**
  - <t:write>
    - Gets the specified bean property value and writes it to the current JspWriter. The following conversions are made in the attribute value.
      - Replace null or null character with “ ”
      - Replace half width space with “ ”
      - Replace line feed code with
      - Ignore line feed character
  - Example

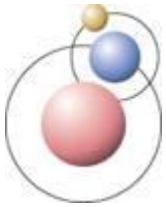
```
primaryField = " あいえうお¥n かきくけこ"
<t:write name="htmlxForm"
              property="primaryField" />
↓Output of above code
 あいうえお<br> かきくけこ
```



⑬ Screen Display (Custom Tags)
~ Text Display (2/3)

◆ Attributes List

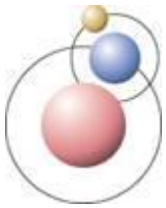
Attributes	Mandatory	Outline
filter	-	If this attribute is set to true, the rendered property value will be filtered for characters that are sensitive in HTML, and any such characters will be replaced by their entity equivalents. By default, filtering is executed. To disable this attribute, it is necessary to explicitly set false to this attribute.
replaceNullToNbsp	-	When this attribute is set to true and when the value of specified bean property is a empty string or null, is output. To disable this attribute, it is necessary to explicitly set false to this attribute.
replaceSpToNbsp	-	When this attribute is set to true and there is a space of 1Byte code in the value of specified bean property, the space is replaced with ' '. To disable this attribute, it is necessary to explicitly set false to this attribute.
replaceLFtoBR	-	When this attribute is set to true, the line feed code or carriage return of the value of the specified bean property is replaced to . To disable this attribute, it is necessary to explicitly set false to this attribute.
ignore	-	If this attribute is set to true, and the bean specified by the name and scope attributes does not exist, simply return without writing anything. Default value is false (A runtime exception to be thrown, consistent with the other tags in this tag library.).



⑬ Screen Display (Custom Tags) ~ Text Display (3/3)

◆ Attributes List

Attributes	Mandatory	Outline
name	○	Specifies the attribute name of the bean accessed by property in order to get the value specified by property (if specified). If property is not specified, the value of this bean is displayed.
property	-	Specifies the property name accessed on the bean specified by name. This value may be a simple, indexed, or a nested property expression. If not specified, the bean identified by name will itself be rendered. If the specified property returns null, no output will be rendered.
scope	-	Specifies the variable scope searched to retrieve the bean specified by name. If not specified, the default rules applied by PageContext.findAttribute() are applied.
fillColumn	-	Delimits at the number of characters specified by fillColoumn and assigns at the end. There is no discrimination between half width and full width characters, while counting the number of characters.
addBR	-	When this attribute is set to true, assigns to the end of property value. Default is false.

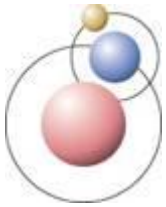


⑬ Screen Display (Custom Tags) ~ Date Conversion (1/3)

- **Outline**
  - Formats Date & time according to the specified format.
- **Description**
  - <t:date>
    - Formats and renders the string(pattern attribute) in the date/time format of java.text.SimpleDateFormat class. For details of date/time format, refer to the API documentation of java.text.SimpleDateFormat class.
  - **Example**

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

↓ Output of above code
2005/7/14 11:24 PM
```

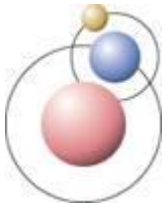

TERASOLUNA



⑬ Screen Display (Custom Tags) ~ Date Conversion (2/3)

Attributes List

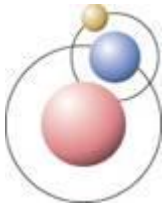
Attributes	Mandatory	Outline
id	-	Specifies when formatted string is to be set in scripting variable without outputting it to response. When the formatted string is to be set in scripting variable, HTML special characters do not escape regardless of filter attribute specifications.
filter	-	Specifies whether to escape the HTML special characters when formatted string is to be output. However, it is ignored when id attributes are specified.
ignore	-	Specifies whether to ignore, when the bean specified by name attribute could not be found. If false is specified, JspException is thrown when bean could not be found.
name	-	Bean name having string to be formatted in property. When property attribute is not specified, the instance specified by name attribute should be formatted. In such case, instance should be either in java.util.Date format or java.lang.String format (and in "yyyy/MM/dd hh:mm:ss" format). It is ignored when value attribute is specified.
property	-	Specifies the name of the property to be accessed on the bean specified by name attribute. It is ignored when value attribute is specified.
scope	-	Scope while searching the bean specified by name attribute.
value	-	String to be formatted. String should be in yyyy/MM/dd hh:mm:ss format. When value attribute is specified, the name attributes and property attributes are ignored.



⑬ Screen Display (Custom Tags) ~ Date Conversion（3/3）

◆ Attributes List

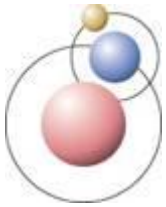
Attributes	Mandatory	Outline
pattern	-	Output format to be formated. The output format specified by pattern attribute is rendered in the subclass of DateFormatterTagBase class. For details, refer to the documents of subclass.
patternKey	-	Specifies the format pattern as a message resource key. When value is defined in pattern attribute, priority is given to the pattern attribute.



⑬ Screen Display (Custom Tags) ~ Wareki Date Conversion (1/3)

- **Outline**
  - Formats date-time data in Wareki (Japanese year).
- **Description**
  - <t:jdate>
    - When date-time data is to be formatted, converts it to the Wareki Gengo (Japanese name of era, e.g. “**昭和**”, “S”etc), Wareki year(instead of western calendar year “2002”, Heisei “14” etc), and Japanese notation of Day of the week (“**月曜日**”, “**月**” etc).
  - **Example**

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

↓Output of above code

平成17年07月14日(木曜日) 11時24分31秒
```



⑬ Screen Display (Custom Tags) ~ Wareki Date Conversion (2/3)

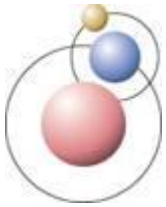
◆ Japanese Date Settings

- Gets the data of Japanese date from the property file through DateUtil class.
- Sets it in property file in the following format.

```
wareki.gengo.ID.name=Wareki Gengo(Japanese name of era)
wareki.gengo.ID.roman=Roman expression of Wareki Gengo
wareki.gengo.ID.startDate=Start date of Wareki Gengo (Year:yyyy/MM/dd format)
```

- Following is a common setting example

```
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
```



⑬ Screen Display (Custom Tags) ~ Wareki Date Conversion (3/3)

◆ Attributes List

Attributes	Mandatory	Outline
id	-	Specifies when formatted string is to be set in scripting variable without outputting it to response. When the formatted string is to be set in scripting variable, HTML special characters do not escape regardless of filter attribute specifications.
filter	-	If this attribute is set to true, the rendered property value will be filtered for characters that are sensitive in HTML. However, it is ignored when id attributes are specified.
ignore	-	Specifies whether to ignore, when the bean specified by name attribute could not be found. If false is specified, JspException is thrown when bean could not be found.
name	-	bean name having string to be formatted in property. When property attribute is not specified, the instance specified by name attribute should be formatted. In such case, instance should be either in java.util.Date format or java.lang.String format (and in "yyyy/MM/dd hh:mm:ss" format). It is ignored when value attribute is specified.
property	-	Specifies the name of the property to be accessed on the bean specified by name attribute. It is ignored when value attribute is specified
scope	-	Scope while searching the bean specified by name attribute.
value	-	String to be formatted. String should be in yyyy/MM/dd hh:mm:ss format. When value attributes are specified, the name attributes and property attributes are ignored.
pattern	-	Output format to be formated. The output format specified by pattern attribute is rendered to the subclass of DateFormatterTagBase class. For details refer documents of subclass.
format	-	Format of Date Time when value attribute is set. Default is “yyyy/MM/dd HH:mm:ss”

TERASOLUNA

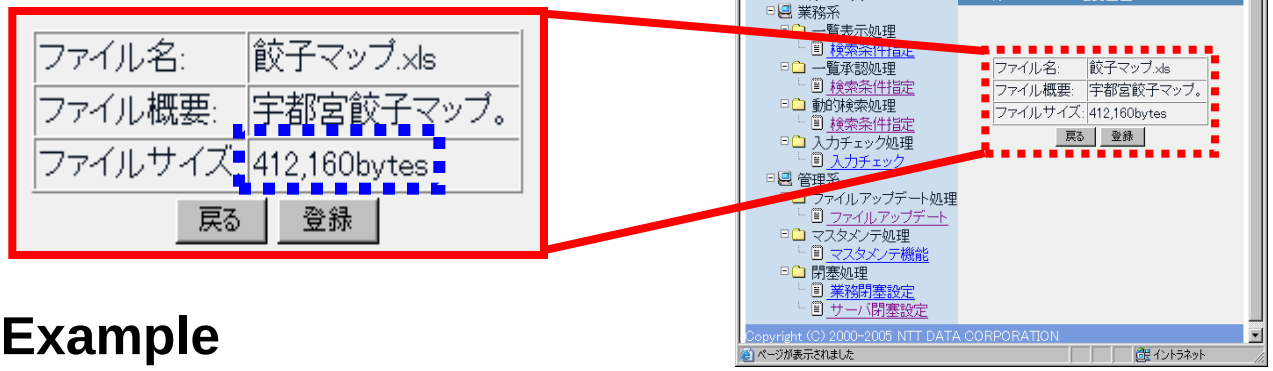
⑬ Screen Display (Custom Tags)
~ Decimal Display (1/3)

Outline

- Formats and outputs a symbol and decimal number or defines it as a scripting variable.

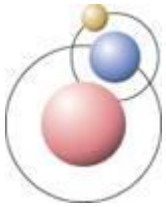
Description

- <t:decimal>



- **Example**

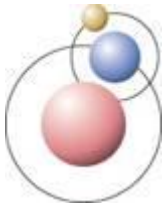
```
<td nowrap>File Size:</td>
<td nowrap>
  <t:decimal name="_fileForm" property="fileSize" pattern="###,###bytes"/>
</td>
```



⑬ Screen Display (Custom Tags) ~ Decimal Display (2/3)

Attributes List

Attributes	Mandatory	Outline
id	-	Specifies when formatted string is to be set in scripting variable without outputting it to response. When the formatted string is to be set in scripting variable, HTML special characters do not escape regardless of filter attribute specifications.
filter	-	Specifies whether to escape the HTML special characters when formatted string is to be output. However, it is ignored when id attributes are specified.
ignore	-	Specifies whether to ignore, when the bean specified by name attribute could not be found. If false is specified, JspException is thrown when bean could not be found.
name	-	bean name having string to be formatted in property. When property attribute is not specified, the instance specified by name attribute should be formatted. In such case, this instance should be in java.math.BigDecimal format or java.lang.String format (and can be rendered by the BigDecimal constructer after deleting the spaces on right side). It is ignored when value attribute is specified.
property	-	Specifies the name of the property to be accessed on the bean specified by name attribute. It is ignored when value attribute is specified.



⑬ Screen Display (Custom Tags) ~ Decimal Display (3/3)

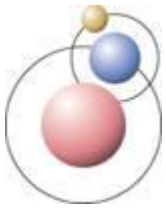
◆ Attributes List

Attributes	Mandatory	Outline
scope	-	Scope while searching the bean specified by name attribute.
value	-	String to be formatted. String can be rendered by the BigDecimal constructer after deleting the spaces on right side. When value attribute is specified, name attribute and property attribute are ignored.
pattern	○	Output pattern to be formatted. Output pattern specified by the pattern attribute is rendered as pattern of DecimalFormat class. Refer to the document of DecimalFormat class for more details.
scale	-	Indicates the number of digits after the decimal point after rounding operation. When n is specified, (n+1)th decimal place is rounded off. Round mode is specified by round attribute. When round attribute is not specified, default is 'round off'.
round	-	Round mode. It is valid when scale attribute is specified. ROUND_HALF_UP (round off), ROUND_FLOOR (round down), ROUND_CEILING (round up) can be set. By default ROUND_HALF_UP is executed. IllegalArgumentException is thrown when round is set with other that these 3 options.

⑬ Screen Display (Custom Tags) ~ Trim Function (1/2)

- **Outline**
  - Deletes spaces from the specified string.
- **Description**
  - <t:rtrim>
    - Deletes spaces from the right side of the string.
    - Example: “_string_” ⇒ delete ⇒ “_string”
  - <t:ltrim>
    - Deletes spaces from the left side of the string.
    - Example: “_string_“ ⇒ delete ⇒ “string_”
  - <t:trim>
    - Deletes spaces from both sides of the string.
    - Example: “_string_“ ⇒ delete ⇒ “string”
  - **Example**

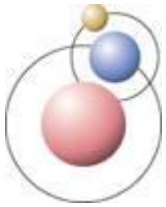
```
<t:rtrim name="form0001" property="field001" />
<t:ltrim name="form0001" property="field002" />
<t:trim  name="form0001" property="field003" />
```



⑬ Screen Display (Custom Tags) ~ Trim Function (2/2)

- **List of attributes of <t:rtrim>, <t:ltrim>, <t:trim> elements**

Attributes	Mandatory	Outline
id	-	It is specified when the formatted string is to be set to scripting variable and not to be output. When formatted string is set to scripting variable, HTML special characters are not escaped regardless of the filter attribute specification.
filter	-	Specifies whether to escape the HTML special character while displaying the formatted string. However, it is ignored when id attribute is specified.
ignore	-	Specifies whether to ignore, when the bean specified by name attribute could not be found. If false is specified, JspException is thrown when bean could not be found.
name	-	bean name having string to be formatted in property. When property attribute is not specified, the string representation (string returned by toString() method) of the instance specified by name attribute should be formatted. It is ignored when value attribute is specified.
property	-	Specifies the name of the property to be accessed on the bean specified by name attribute. It is ignored when value attribute is specified.
scope	-	Scope while searching the bean specified by name attribute
value	-	String to be formatted. When value attribute is specified, name attribute and property attribute are ignored.
replaceSpToNbsp	-	When this attribute is set to true and there is a space of 1Byte code in the value of specified bean property, the space is replaced with ' '. To disable this attribute, it is necessary to explicitly set false to this attribute. However, it is ignored when id attribute is specified.

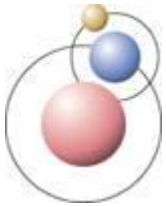


⑬ Screen Display (Custom Tags) ~ Substring Functionality (1/2)

- **Outline**
  - Extracts the specified number of characters from left side of the string.
- **Description**
  - <t:left>
    - substring() method of StringUtil class extracts the specified number of characters from the left side of string.
  - **Example**

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

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

↓Output of above code

"Java write"
```


TERASOLUNA



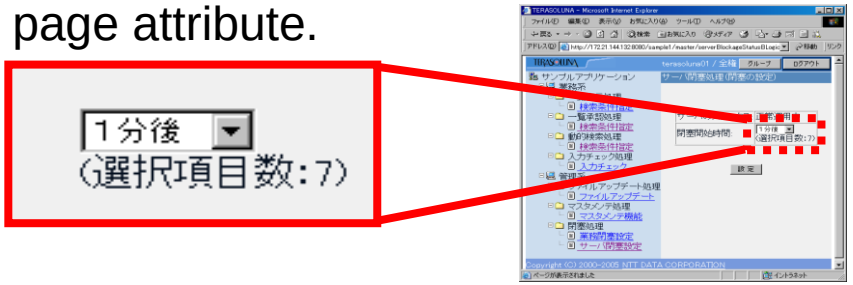
⑬ Screen Display (Custom Tags) ~ Substring Functionality (2/2)

Attributes List

Attributes	Mandatory	Outline
id	-	It is specified when the formatted string is to be set to scripting variable and not to be output. When formatted string is set to scripting variable, HTML special characters are not escaped regardless of the filter attribute specification.
filter	-	Specifies whether to escape the HTML special character while displaying the formatted string. However, it is ignored when id attribute is specified
ignore	-	Specifies whether to ignore, when the bean specified by name attribute could not be found. If false is specified, JspException is thrown when bean could not be found.
name	-	bean name having string to be formatted in property. When property attribute is not specified, the string representation (string returned by toString() method) of the instance specified by name attribute should be formatted. It is ignored when value attribute is specified.
property	-	Specifies the name of the property to be accessed on the bean specified by name attribute. It is ignored when value attribute is specified.
scope	-	Scope while searching the bean specified by name attribute.
value	-	String to be formatted. When value attribute is specified, name attribute and property attribute are ignored.
replaceSpToNbsp	-	When this attribute is set to true and there is a space of 1Byte code in the value of specified bean property, the space is replaced with ' '. To disable this attribute, it is necessary to explicitly set false to this attribute. However, it is ignored when id attribute is specified.

⑬ Screen display (Custom tags)
~ Define a code List (1/2)

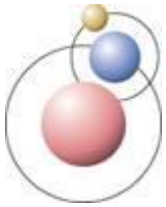
- **Outline**
  - To use a code list(already fetched) in jsp.
- **Description**
  - <t:defineCodeList>
    - Gets code list from CodeListLoader with the specified id and defines it as a bean of page attribute.



  - **Example**

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

※Refer to (3) Code List Functionality for setting of code list.



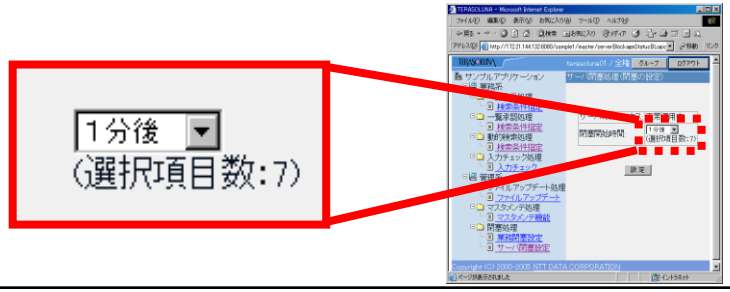
⑬ Screen Display (Custom tags) ~ Define a code List (2/2)

◆ Attributes List

Attributes	Mandatory	Outline
id	○	Code list is searched from this attribute. After tag declaration, code list can be referred in <logic:iterator>tag, <html:options> tag etc.

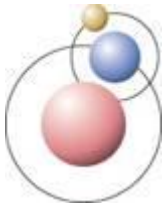
⑬ Screen Display (Custom tags)
~ Write count of code list (1/2)

- **Outline**
  - Writes the number of elements in a code list.
- **Description**
  - <t:writeCodeCount>
    - Gets the number of elements of code list from CodeListLoader with the specified id.
  - **Example**




```
<html:select property="server_blockage_time">
  <html:options collection="SQL_C0901" property="id" labelProperty="name" />
</html:select><br>
(Number of selected items:<t:writeCodeCount id="loader"/>)
```

※ Refer to (3) Code List Functionality for setting of code list.



⑬ Screen Display (Custom tags)
~ Write count of code list (2/2)

◆ Attributes List

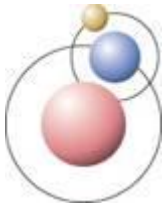
Attributes	Mandatory	Outline
id	○	Count of elements in the code list referred by this attribute is output. When code list is not found, 0 is returned.

⑬ Screen Display (Custom tags) ~ Display a code's name (1/2)

- **Outline**
  - Writes the display name of a specified code in the code list.
- **Description**
  - <t:writeCodeValue>
    - By specifying the code value and the id of CodeListLoader, the display name of a code is displayed on the screen.
  - **Example**


```
<tr>
  <th>Room type</th>
  <td><t:writeCodeValue codeList="roomTypeCodeList"
        key="roomType01"/></td>
</tr>
```

※ Refer to (3) Code list functionality for setting of code list.



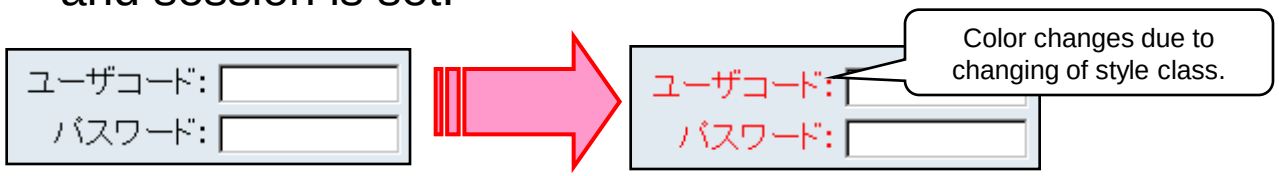
⑬ Screen Display (Custom Tags) ~ Display a a code's name (1/2)

◆ Attributes List

Attributes	Mandatory	Outline
codeList	○	BeanId of CodeListLoader instance that retains the CodeBean to be output.
key	-	Directly specifies the code value for acquiring the value from the acquired code list. When this value is omitted, the Bean name and property name for acquiring the code value should be specified by specifying the name and property attributes.
name	-	Bean name that retains the code value for acquiring the value from the acquired code list. Disabled when the key attribute is specified.
property	-	Property of Bean that retains the code value for acquiring the value from the acquired code list. Disabled when the key attribute is specified.
scope	-	Scope which contains the Bean that retains the code value for acquiring the value from the acquired code list.

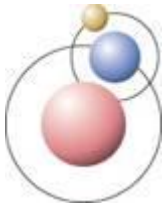
⑬ Screen Display (Custom Tags)
~ Change a style class (1/2)

- **Outline**
  - Changes the style sheet class (when error occurs)
- **Description**
  - <ts:changeStyleClass>
    - Changes the style sheet class depending on whether the error information corresponding to the field specified in the request and session is set.



  - **Example**

```
<td nowrap class='<ts:changeStyleClass name="userCode"
                                default="ItemLabel" error="ErrorItem"/>'>
  User Code:
</td>
```



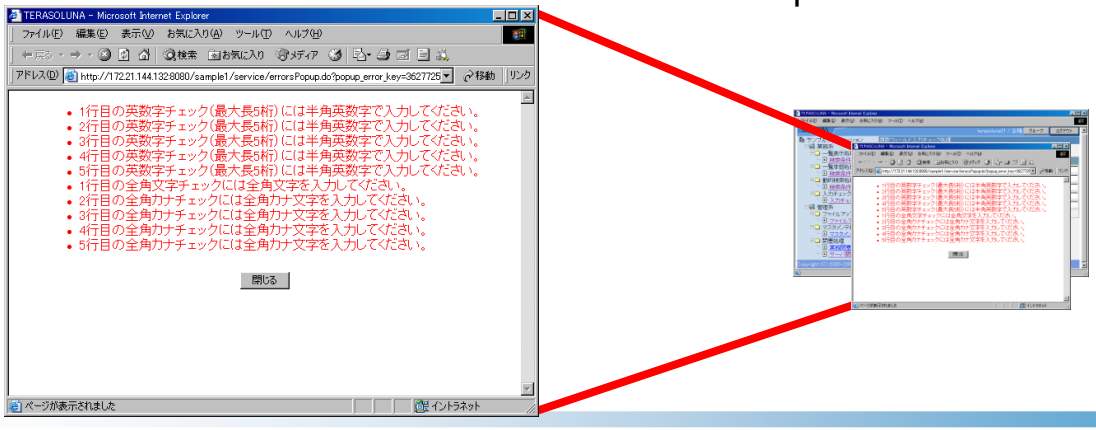
⑬ Screen Display (Custom Tags)
~ Change a style class (2/2)

◆ Attributes List

Attributes	Mandatory	Outline
name	○	Specifies the field name for which it needs to be determined whether error information is set.
default	○	Specifies the Style Sheet Class name when there is no error.
error	○	Specifies the Style Sheet Class name in case of an error.

⑬ Screen Display (Custom Tags)
~ Message Display (1/5)

- **Outline**
  - Displays Message or Error Message information.
- **Description**
  - <ts:errors>
    - Error Tag of Struts is extended.
      - Gets/displays the error information from session/request.
  - <ts:messages>
    - Message Tag of Struts is extended
      - Gets the information from session/request.



⑬ Screen Display (Custom Tags) ~ Message Display (2/5)

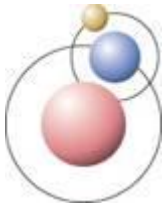
Example of <ts:errors>element

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

Example of <ts:messages> element

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

<ts:messages>elements differ from <ts:errors>elements, It only defines the bean in which the messages are stored. Hence need to write the code to display the messages.



⑬ Screen Display (Custom Tags)
~ Message Display (3/5)

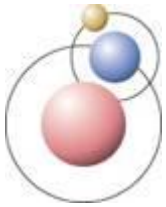
- **List of attributes of <ts:errors> element**
  - Same as <html:errors>
- **Caution in using <ts:errors>**
  - This tag cannot be used in pop up screens. Information from session is not deleted.

TERASOLUNA

⑬ Screen Display (Custom Tags) ~ Message Display (4/5)

List of attributes of <ts:messages> element

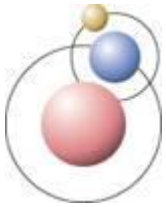
Attributes	Mandatory	Outline
id	○	Specifies bean name where message is to be stored.
bundle	-	Specifies message resource name. When not specified, it is the default message resource.
locale	-	Specifies the locale of output message. When not specified, default locale is used.
name	-	Separately specifies the message keys of action message to be displayed. When the value of 'message' attribute is 'true', Globals.MESSAGE_KEY is set. However, when it is not set, Globals.ERROR_KEY is set.
property	-	Specifies the Property name (form) to be displayed. When not specified, all Action Errors are displayed regardless of property name.
header	-	Specifies the header message key to be output before messages.
footer	-	Specifies the header message key to be output after messages.
message	-	When the value is specified to true, name attribute is set as Globals.MESSAGE_KEY.



⑬ Screen Display (Custom Tags) ~ Message Display (5/5)

- **Caution in using <ts:errors>**
  - As long as this tag is not used in pop-up screens the session information is not deleted.

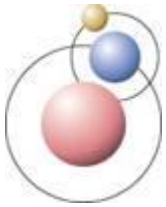
TERASOLUNA



⑬ Screen Display (Custom Tags) ~ No-cache form functionality (1/2)

- **Outline**
  - Adds random ID for no-cache to action URL.
- **Description**
  - <ts:form>
    - Extends <ts:form> element provided by Struts.
    - Adds random ID for no cache to action URL.
  - **Example**

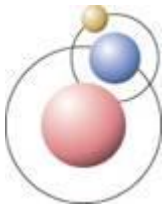
```
<table border="0">
 <ts:form action="/logonBLogic">
 <div align="center">

↓Output of above code

<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">
```



⑬ Screen Display (Custom Tags)
~ No-cache form functionality (2/2)

- **Attributes List**
  - Same as API of <html:form> element.

TERASOLUNA



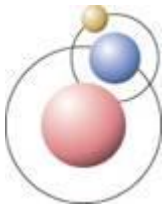
⑬ Screen Display (Custom Tags) ~ No Cache Link functionality

- **Outline**
  - Extends the <html:link> tag provided by Struts.
- **Description**
  - <ts:link>
    - Random ID for no cache is added to Action URL.
  - **Example**
    - Used same as <ts:link> element.

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

↓ Random ID is added while displaying.

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

  - **Attributes List**
    - Similar to API of <html:link> element

TERASOLUNA



⑬ Screen Display (Custom Tags)
~ Form Target Specification

- **Outline**
  - Specifies the form target.
- **Description**
  - <ts:submit>
    - Form target is specified by setting the target attribute value.
  - **Example**

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

  - **Attributes List**

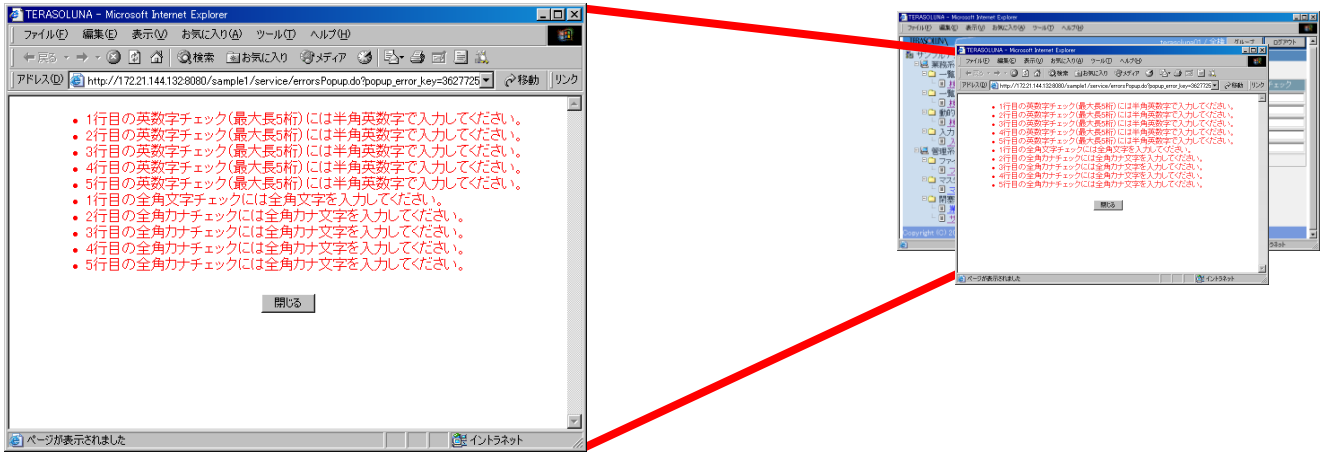
Attributes	Mandatory	Outline
target	-	Specifies target destination.

    - Other attributes are similar to that of <html:submit>.

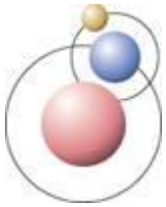
⑬ Screen Display (Custom Tags) ~ Message Popup (1/4)

- **Outline**
  - Displays the message or error message on Popup screen.
  - Adds the script, which is embedded in the PageContext with the key “ON_LOAD”, in onLoad event process.
- **Description**
  - <ts:messagesPopup>
  - <ts:body>
    - In the HTML <body> tag that is generated by this tag, the JavaScript function _onLoad_() is called as the script for onLoad event processing.
    - Since the definition of the JavaScript function _onLoad_() is generated by this tag, JavaScript of same name should not be specified in HTML.

⑬ Screen Display (Custom Tags)
~ Message Popup (2/4)



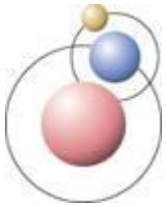

```
<ts:messagesPopup popup="/service/errorsPopup.do" title="TERASOLUNA"/>
<ts:body styleClass="MainPage">
<% String script="alert('Output');";
   pageContext.setAttribute("ON_LOAD", script); %>
<ts:body>
↓Output of above code
<body onLoad="__onLoad__()">
  <script type="text/javascript">
  <!--
   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



⑬ Screen Display (Custom Tags) ~ Message Popup (3/4)

List of attributes of <ts:messagesPopup> element

Attributes	Mandatory	Outline
popup	○	URL to be displayed on Popup Screen. It corresponds to the first argument of window.open() of JavaScript.
title	_	title of the popup screen, which displays the error.
param	_	Parameter string used while opening Popup screen in JavaScript.
paramType	_	Resource key to acquire the parameter string, which is used to open popup screen in JavaScript, from ApplicationResources file.
paramFunc	_	JavaScript function name, which acquires the parameter string, which is used to open popup screen in JavaScript.
windowId	_	JavaScript variable name, which retains opened popup screen.



⑬ Screen Display (Custom Tags) ~ Message Popup (4/4)

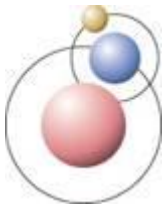
List of attributes of <ts:body> elements

Attributes	Mandatory	Outline
onload	-	Specifies JavaScript to be executed while displaying the screen.
onunload	-	Specifies JavaScript to be executed while unloading the screen.
styleClass	-	Specifies the class name of style sheet.
bgcolor	-	Specifies Background color.
background	-	Specifies the Image to be set at the background
text	-	Specifies Text character color.
link	-	Specifies Link color.
vlink	-	Specifies visited link color.
alink	-	Specifies active link color.

⑬ Screen Display (Custom Tags)
~ Error Message Check

- **Outline**
  - Determines whether the error information is set for the request or session, and accordingly switches between show / hide.
- **Description**
  - <ts:ifErrors>
    - Evaluates the tag body when there is an input check error or error information is set to the output parameter.
  - <ts:ifNotErrors>
    - Evaluates the tag body when there is no input check error and no error information is set to the output parameter.
  - **Example**

```
<ts:ifErrors>
    ... // Items to be displayed when there is an error
</ts:ifErrors>
<ts:ifNotErrors>
    ... // Items to be displayed when there is no error
</ts:ifNotErrors>
```

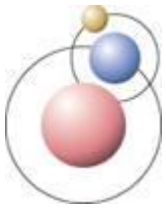


⑬ Screen Display (Custom Tags) ~ Client Check Extension

- **Outline**
  - Values to be used in client-side validation are written to javascript variables.
- **Description**
  - <ts:javascript>
    - The variables required by Japanese-specific validation provided by TERASOLUNA are output as javascript variables.
  - **Example**

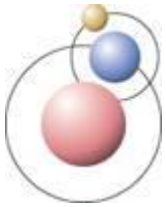
```
<meta http-equiv="Content-Type" content="text/html; charset=Windows-31J">
<title>Input check (Client) Test Screen</title>
<ts:javascript formName="/clientValidate"/>
</head>
<body>
```

  - **Attributes list**
    - Same as <html:javascript>



⑬ Screen Display (Custom Tags) ~ List Related Functionality (1/14)

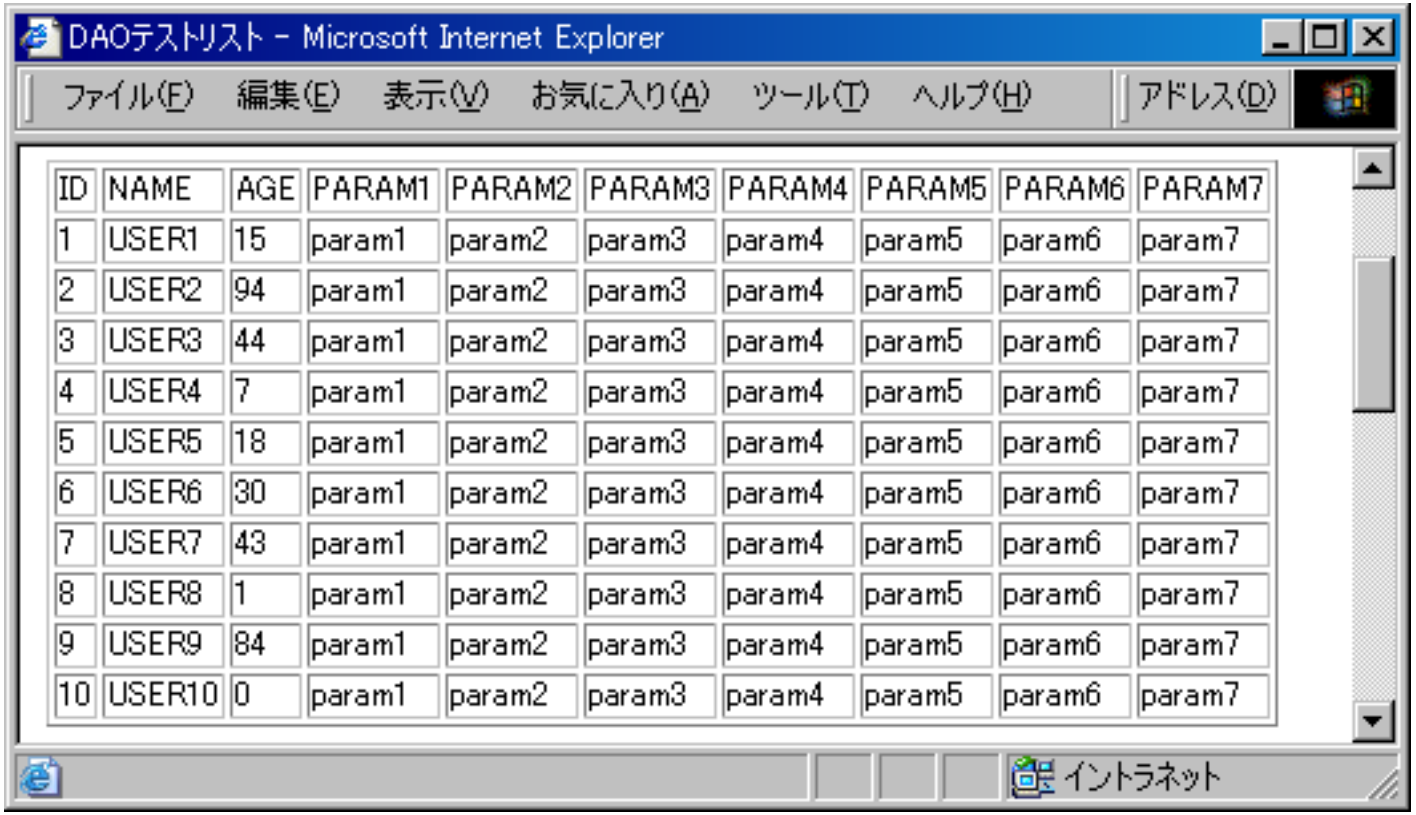
- **List methods**
  - Uses the list display functionality <logic:iterate> element of Struts.
  - Description
    - Gets the list data from Bean specified by the attribute, and loops over the list data.
    - The specified bean can be a Collection, ArrayList, Vector, Enumeration, Iterator, Map, HashMap, Hashtable, TreeMap or Array.
    - Specify the repeating items (<TR>~</TR> in HTML Table etc.) in enclosed format.



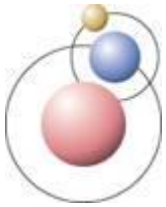
⑬ Screen Display (Custom Tags)
~ List Related Functionality (2/14)

Listing method

- Execution example of JSP based on listing functionality



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



⑬ Screen Display (Custom Tags)
~ List Related Functionality (3/14)

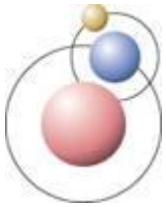
Listing method

- Implementation example of List Display functionality

```
<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>
```

Repeated

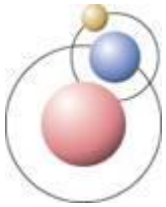
TERASOLUNA



⑬ Screen Display (Custom Tags) ~ List Related Functionality (4/14)

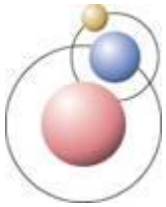
- List methods
  - List of <logic:iterate> element attribute

Attributes	Mandatory	Outline
id	○	Specifies the element bean that retains the 1st line data of List. Displays it in the body, by using the bean of specified name,.
type	-	Specifies the type of element Bean specified by id attribute.
name	-	Specifies Bean that retains collection. (Example: Action form name)
property	-	Specifies the property of Bean specified by name attribute.
scope	-	Specifies the scope to retrieve the Bean specified by name attribute.
collection	-	Specifies the collection in runtime expression.
length	-	Specifies the loop count of body.
offset	-	Specifies the start line of list information to be displayed.
indexId		Specifies the index name of the currently displayed line, which is to be used in body.



⑬ Screen Display (Custom Tags) ~ List Related Functionality (5/14)

- Page Link functionality
  - Functional Outline
    - Displays the page transition link of the list defined by <logic:iterate> element.
  - Description
    - <ts:pageLinks>
    - To use page link functionality, properties given below should be prepared in Action form.
      - Property to retain the rows
      - Property to retain the start index
      - Property to retain the total count of list information

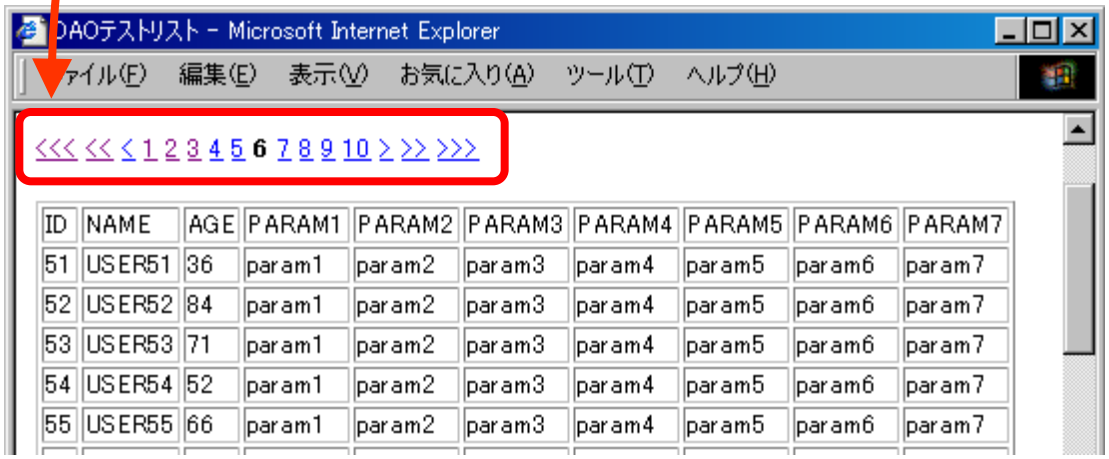


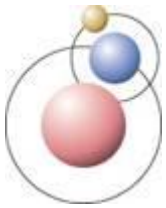
⑬ Screen Display (Custom Tags) ~ List Related Functionality (6/14)

Page Link Functionality

Example

```
<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>
```





⑬ Screen Display (Custom Tags)
~ List Related Functionality (7/14)

Page Link Functionality

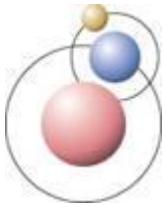
Attributes List (1/2)

Attributes	Mandatory	Outline
id	-	Saves the output destination of page link in page context, and not on the screen when string is specified by this attribute. This attribute is the key saved in page context.
action	△	Specifies the action path name that displays the list view. This attribute is mandatory when submit attribute is false.
name	-	Specifies the bean to retrieve rows, start index and total count of list information.
rowProperty	-	Specifies the property of rows.
indexProperty	-	Specifies the property of start index.
totalProperty	-	Specifies the property of total count.
scope	-	Specifies the scope to retrieve the Bean specified by name attribute.
submit	-	Set true to execute submit without link. By default it is false.
forward	-	This is used to dispatch by using DispatchAction of TERASOLUNA. When true is specified, Hidden tag of the value set by event attribute, is created. During 'submit', "forward_pageLinks" is set to value attribute of its Hidden tag. By default it is false.

Page Link Functionality

Attributes List (2/2)

Attributes	Mandatory	Outline
event	-	This attribute is valid when Forward attribute ~~it~~ is set to true and used to dispatch by using DispatchAction of TERASOLUNA. Hidden tag of the name specified by this attribute is created. By default, it is “event”.
resetIndex	-	When true is set, Hidden tags of startIndex and endIndex used to reset specified range are created.
currentPageIndex	-	Key to save current page count of corresponding list to page context. By default, it is "currentPageIndex”.
totalPageCount	-	It is the key to save total page numbers of corresponding list to page context. By default, it is "totalPageCount“.



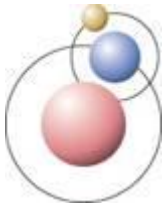
⑬ Screen Display (Custom Tags) ~ List Related Functionality (9/14)

Page Link Functionality

- Example of getting the list data from database
  - 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" />
```


TERASOLUNA



⑬ Screen Display (Custom Tags) ~ List Related Functionality (10/14)

Page Link Functionality

- Example of getting the list data from database
  - Struts configuration file

```
<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>
```

An array of Bean that has the list data

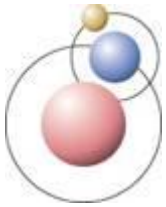
Rows displayed on each page. With default value

start Index. With default value.

Total count of list information

Calls the business logic to get the list data for each page.

Direct forwarding



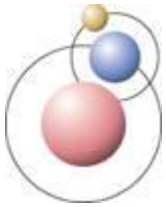
⑬ Screen Display (Custom Tags) ~ List Related Functionality (11/14)

Page Link Functionality

- Example of getting list data from database
  - Business Logic

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

//Conversion process of int
......

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);
```


TERASOLUNA

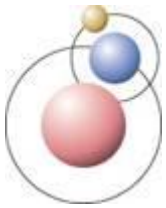


⑬ Screen Display (Custom Tags) ~ List Related Functionality (12/14)

Page Link Functionality

- Example of getting list data from Action form
  - 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" />
```



⑬ Screen Display (Custom Tags) ~ List Related Functionality (13/14)

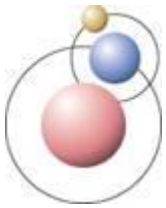
Page Link Functionality

- Example of getting list data from Action form
  - Struts configuration file

```
<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>
```

Business logic getting the total size of list is called initially.

Screen display action called by using page link functionality. It is necessary to specify Action form corresponding to name attribute.



⑬ Screen Display (Custom Tags) ~ List Related Functionality (14/14)

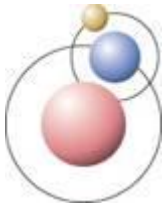
Page Link Functionality

- Example of Id, currentPageIndex and totalPageCount Attributes
  - 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" />
```

PageLink Tag displays the total count of link(<a> tag), current page count and list information stored in page context by using <bean:write> tag.

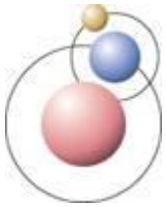
TERASOLUNA



⑭ Utility Functions

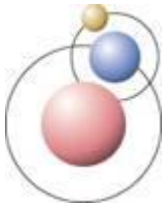
Utility Functionality

- Provides utility classes for commonly used functions in business development.
- terasoluna-commons
  - ClassUtil
  - DateUtil
  - FileUtil
  - HashUtil
  - PropertyUtil
  - StringUtil
  - JndiSupport
    - DefaultJndiSupport



⑭ Utility Functions

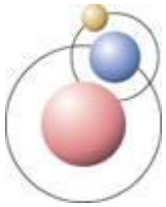
- **terasoluna-commons**
  - ClassUtil
    - create(String), create(String, Object[])
      - » Generates an instance from the fully qualified class name.
  - DateUtil
    - dateToWarekiString(String, Date)
      - » Converts date instance to the Wareki (Japanese year) specific format.
    - getSystemTime()
      - » Gets the system time.
    - getWarekiGengoName(Date)
      - » Gets the Wareki Gengo (Japanese name of era) for the specified date.
    - getWarekiGengoRoman(Date)
      - » Gets the Roman notation (abbreviated form) for Wareki Gengo of the specified date.
    - getWarekiYear(Date)
      - » Gets the Wareki Year of the specified date.



⑭ Utility Functions

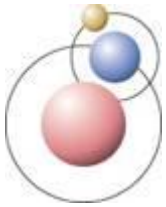
- **terasoluna-commons**
  - FileUtil
    - getSessionDirectory(String)
      - » Gets the directory for the specified session ID.
    - getSessionDirectoryName(String)
      - » Gets the directory name for the specified session ID.
    - makeSessionDirectory(String)
      - » Creates a directory for the specified session ID.
    - removeSessionDirectory(String)
      - » Deletes the directory for the specified session ID.
    - rmdirs(File)
      - » Deletes the specified directory.

TERASOLUNA



⑭ Utility Functions

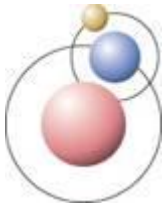
- **terasoluna-commons**
  - HashUtil
    - hash(String, String)
      - » Gets the hash value of a string by using the specified algorithm.
    - hashMD5(String)
      - » Gets the hash value of a string by using MD5 algorithm.
    - hashSHA1(String)
      - » Gets the hash value of a string by using SHA1 algorithm.



⑭ Utility Functions

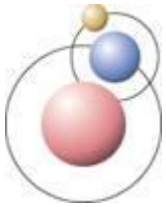
terasoluna-commons

- PropertyUtil
  - addPropertyFile(String)
    - » Reads the specified property file.
  - getPropertiesValues(Properties, Enumeration<String>)
    - » Gets the values of a list of keys from the specified Properties.
  - getPropertiesValues(String, String)
    - » Gets the property value from a property file for the specified key.
  - getProperty(String)
    - » Gets the property for a specified key.
  - getPropertyNames()
    - » Gets the list of all property keys.
  - getPropertyNames(String)
    - » Gets the list of property keys starting with the specified prefix.
  - loadProperties(String)
    - » Loads property object from the specified property file.



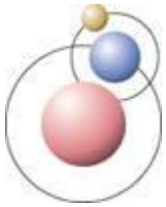
⑭ Utility Functions

- **terasoluna-commons**
  - StringUtil
    - capitalizeInitial(String)
      - » Capitalizes the initial character of the specified string.
    - getExtension(String)
      - » Gets the extension from the specified string.
    - hankakuToZenkaku(String)
      - » Converts half width string to full width string.
    - zenkakuToHankaku(String)
      - » Converts full width string to half width string.
    - parseCSV(String)
      - » Converts CSV format string to a string array.
    - toLikeCondition(String)
      - » Converts search condition string to a pattern string of LIKE predicate.
    - trim(String)
      - » Deletes white spaces on both sides of a string.



⑭ Utility Functions

- **terasoluna-commons**
  - JndiSupport
    - Class executing JNDI API methods.
    - Test can be executed by accessing the interface, even though AP server is not running.
    - lookup(String)
      - » Gets the object of specified name.
    - rebind(String, Object)
      - » Binds the name with object and overwrites the existing binding.
    - unbind(String)
      - » Unbinds the specified object.



⑭ Utility Functions

terasoluna-commons

- DefaultJndiSupport
  - JndiSupport implemented default class based on functionality provided by Spring.
  - Use after setting in Bean definition file.

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

Bean definition file

※ Specify dummy class implementing JndiSupport in another Bean definition file during test