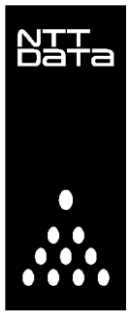


TERASOLUNA Server Framework for Java

Functional Guideline

Version 2.0.3.1

NTT DATA Corporation

Agenda

CA-01	TRANSACTION CONTROL
CB-01	DATABASE ACCESS
CC-01	JNDI ACCESS
CD-01	UTILITY FUNCTIONS
WA-01	LOGIN VALIDATION
WA-02	ACCESS CONTROL
WA-03	SERVER BLOCKAGE CHECK
WA-04	BUSINESS BLOCKAGE CHECK
WA-05	DENIAL OF DIRECT ACCESS TO EXTENSIONS
WB-01	USER INFORMATION RETENTION
WB-02	ACTIONFORM EXTENSIONS
WB-03	ACTIONFORM SWITCHING FUNCTIONALITY
WB-04	FORM PROPERTY RESET FUNCTIONALITY
WB-05	CODE LIST FUNCTION(S)
WC-01	EXCEPTION HANDLING
WD-01	SESSION DIRECTORY
WD-02	FILE UPLOAD FUNCTION
WD-03	FILE DOWNLOAD FUNCTION
WE-01	ACTION EXTENSIONS
WE-02	STANDARD DISPATCHER FUNCTION
WE-03	FORWARDING FUNCTION
WE-04	CODE LIST RELOAD FUNCTION
WE-05	CREATE SESSION DIRECTORY FUNCTION
WE-06	CLEAR SESSION FUNCTION
WE-07	LOGOFF FUNCTION
WF-01	INPUT VALIDATION EXTENSIONS
WG-01	MESSAGE RESOURCE EXTENSIONS
WH-01	BUSINESS LOGIC EXECUTION
WH-02	BUSINESS LOGIC INPUT/OUTPUT FUNCTION
WI-01	LIST DISPLAY FUNCTION
WJ-01 ~ WK-09	SCREEN DISPLAY

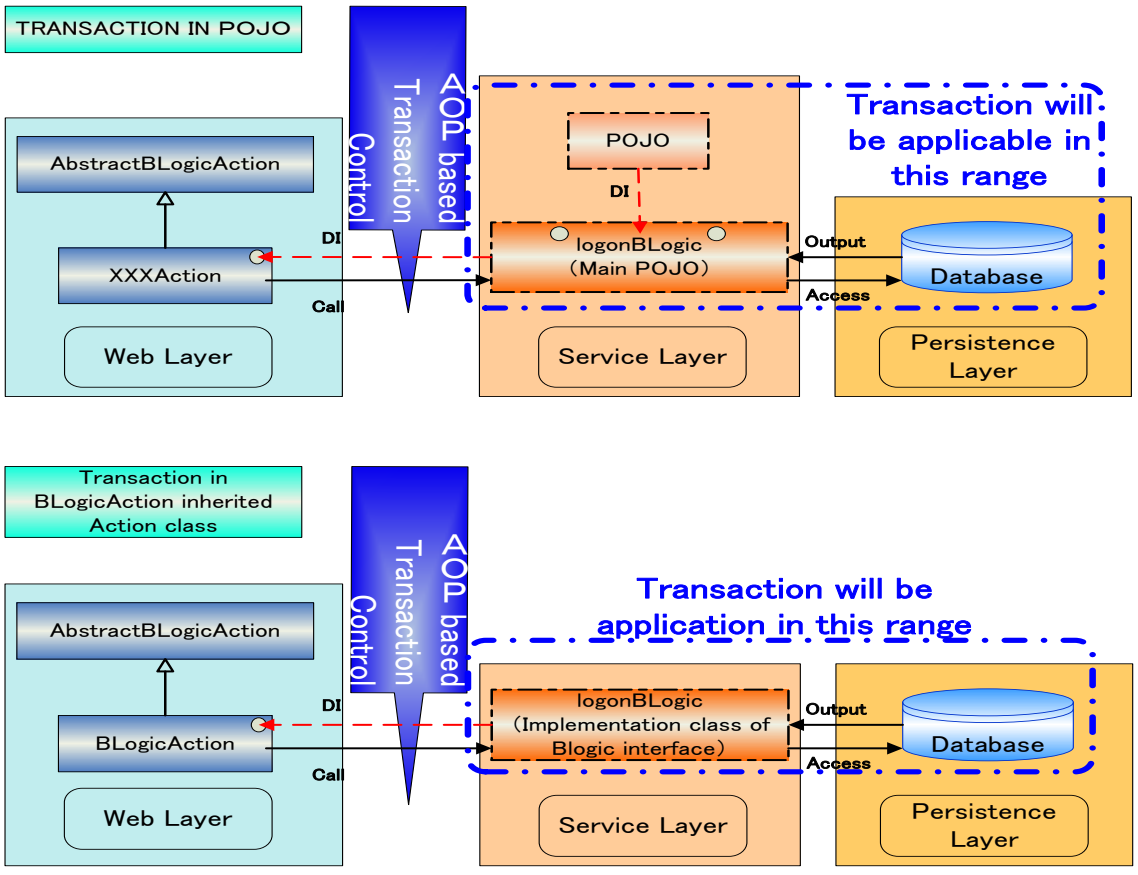
CA-01 Transaction Control

■ Overview

◆ Functional Overview

- Transaction control is provided using SpringAOP.
- The framework executes commit, rollback.
  - ➢ Because transaction control is implemented using AOP, the developer need not write code to implement transaction control.
  - ➢ Transaction is started automatically when the service starts, and committed when the service ends. In case of an exception, or by the call of a utility method, rollback is executed.

◆ Schematic Diagram



※ The above diagram is applicable in case of the Web based Terasoluna framework. In case of the rich-client based Terasoluna framework, the processing inside Action happens

in the Controller instead.

◆ Description

- Because transaction control is implemented using AOP, the developer need not write code to implement transaction control.
- Transaction control is executed within the boundary of the service layer object.
- The framework executes Commit/Rollback.
  - Transaction is started when the service starts and committed when service ends.
  - In case of an exception, or by the call of a utility method, rollback is executed.
- As a guideline, implement the business logic so that each transaction is one service.

■ Usage

◆ Coding Points

- Ways in which Transaction can be applied
  Transaction can be applied in following ways. However, TERASOLUNA Server Framework for Java, mainly uses “REQUIRED”. Please examine each of the following and modify as required.

Transaction	Overview
REQUIRED	Support the current transaction, and creates a new one if none exists. This is the default in TERASOLUNA Server Framework for Java.
SUPPORTS	Support the current transaction, execute without a transaction if none exists.
MANDATORY	Support the current transaction; throws an exception if none exists.
REQUIRES_NEW	Create a new transaction; suspend the current transaction if one exists.
NOT_SUPPORTED	Execute without a transaction, suspend the current transaction if one exists
NEVER	Execute without a transaction, throw an exception if a transaction exists.
NESTED	Execute within a nested transaction, if a current transaction exists; otherwise behave like REQUIRED.

In the Spring, it is also possible to define an isolation level, “timeout” value and

read-only transaction. Refer to the Spring API for its details.

- Commit and Rollback Settings

  Spring's Declarative Transaction Control is used in transaction control using the TERASOLUNA Server Framework for Java. Spring's declarative Transaction control automatically commits/rollbacks the transactions on the basis of settings in the bean definition file. The developer need not write code for commit/rollback. Spring's Declarative transaction control by default commits the transaction, and **rollback is executed only when an exception for which rollback is to be executed is thrown during execution**. Here, note that if an execption is not caught within the business logic or if an Error occurs, then the transaction gets rollbacked. If the exception is caught in the business logic, then the transaction is committed.
  If rollback is to be executed without throwing an exception, then use setRollbackOnly method of TransactionUtil class described later.

  Commit/Rollback settings can be modified depending upon the exception thrown by adding the following attributes to <tx:method/> element.
  Rollback-for ▪▪▪ Commit when this exception is thrown
  No-rollback-for ▪▪▪ Rollback when this exception is thrown
  The example bean definition on the next page shows the settings recommended in the TERASOLUNA Server Framework for Java (Web version). "java.lang.Exception" is set so that transaction will be rolled back when any type of exception is thrown. These settings can be customized as per the project requirements.

- Transaction settings using AOP

  For applying transaction settings using AOP, first of all, Advice (Transaction Interseptor) is to be defined and while executing business logic methods, the Advice needs to be weaved in using AOP.
  With the help of naming conventions, it is possible for multiple beans to share the same transaction information by specifying BeanID and method name, to which transaction is to be applied. Refer to SpringAPI for details regarding Transaction using AOP.
  DataSourceTransactionManager provided by Spring is used as Transaction manager. DataSourceTransactionManager is a Transaction manager executes transaction handling corresponding to a single datasource.

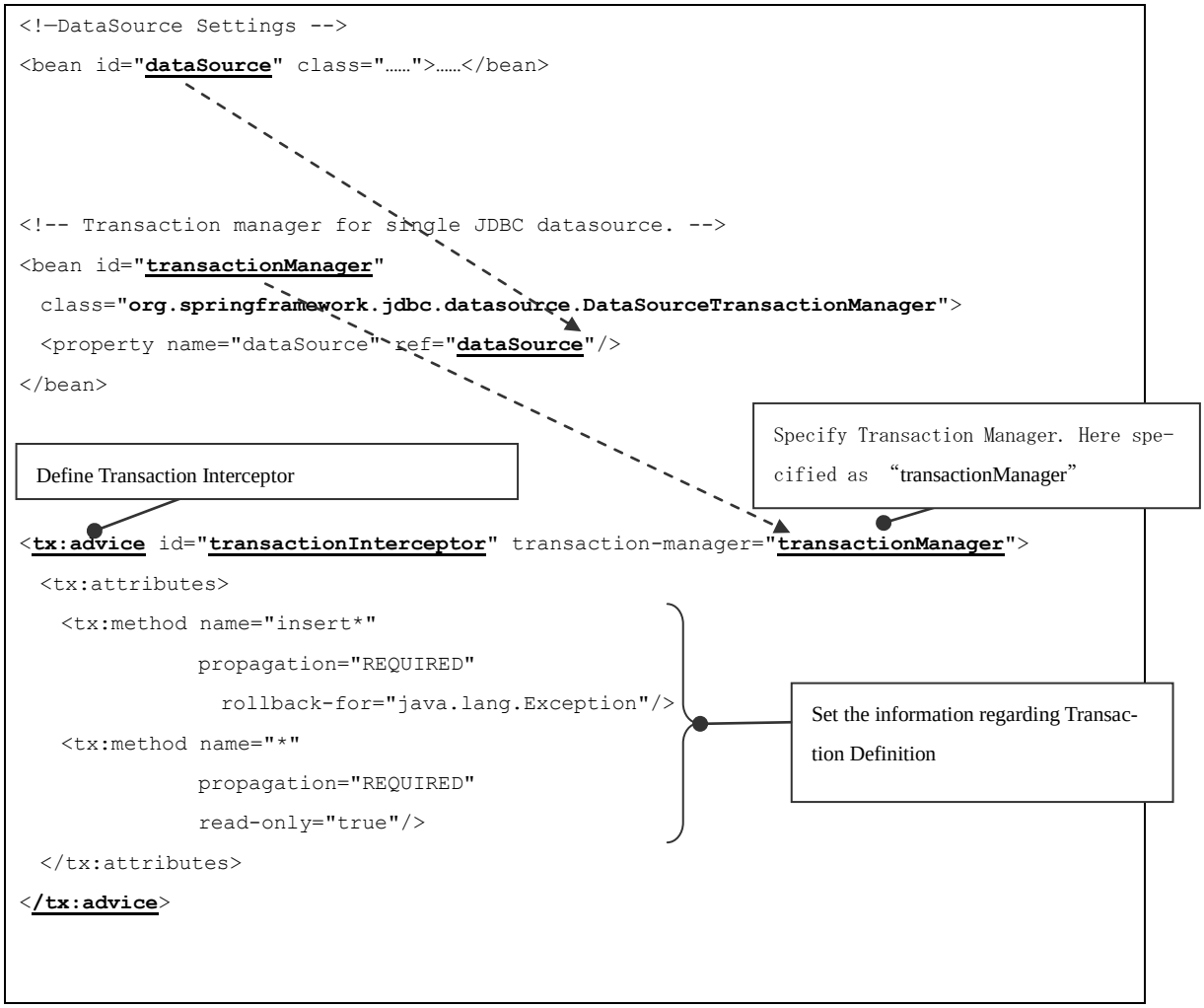
  Below shows an example of Bean Definition file.

  - An example bean definition file

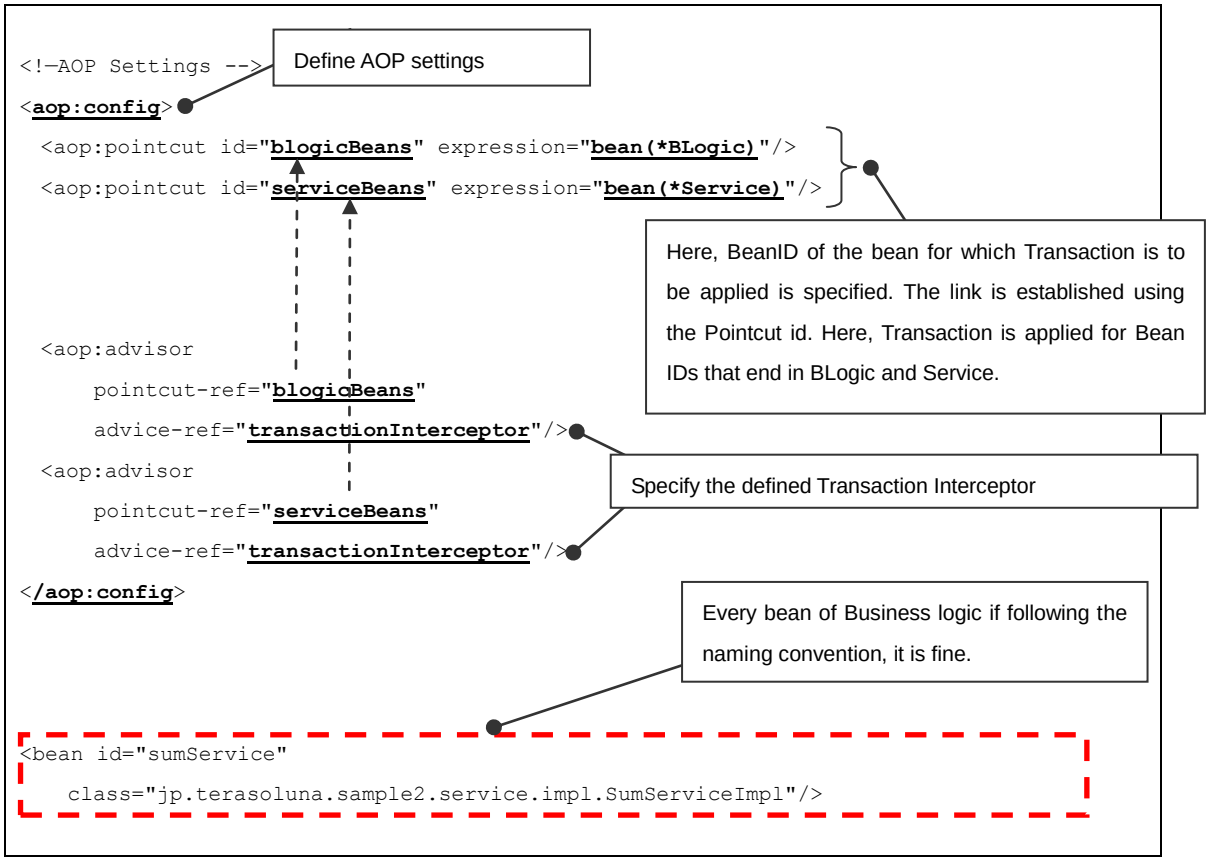
```
<?xml version="1.0" encoding="UTF-8" ?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:util="http://www.springframework.org/schema/util"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:tx="http://www.springframework.org/schema/tx"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

Define the schema

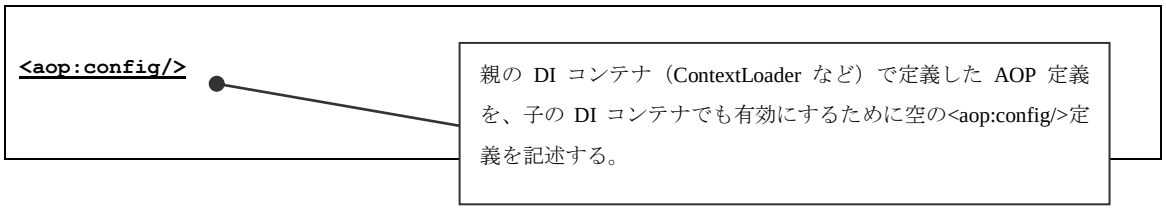
（continued）



（continued）



- The Advice defined in parent DI container (Bean Definition files that are read using contextLoader) are valid even in child DI container. This can be achieved by defining a empty <aop:config/> in the child DI container.
- Double weaving of Advice will occur if its defined in parent and child DI containers
- If multiple weaving of transaction interceptors occurs, it might create problems in setRollbackOnly method (described later).



- Implementation example of Rollback
  There are two cases for Rollback to occur. First case is when an exception is thrown at the time of execution. (In case of default settings of TERASOLUNA, Rollback occurs even in case of checked exception)
  Second case is to use TrasactionUtil class.
  Below is the implementation example of each case of Rollback.

  - Example of business logic in which exception is thrown.
    Here, exception is thrown explicitly. Even if exception is not thrown explicitly, framework will automatically rollback if the settings related to the exception that occurred are present.

```
public class RunTimeExceptionRollbackBLogicImpl implements BLogic{
   public BLogicResult execute(InputDTO inputDTO) {

       if(error in business logic){
           throw new RuntimeException();
       }
   }
}
```

実行時例外をスローする。

  - Example of business logic in which TransactionUtil class us used for Rollback
    There are cases when there is a need to inform about the abnormal state of transaction to the user without going to the error screen. In such case, just by calling setRollbackOnly method of TransactionUtil class in if clause etc on the basis of some condition rollback will be executed at the end of the Service.

```
import jp.terasoluna.fw.util.TransactionUtil;

public class NoExceptoinRollbackBLogicImpl implements BLogic {
   public BLogicResult execute(InputDTO inputDTO) {

       if(error in business logic){
          ///call setRollbackOnly method to execute rollback
          TransactionUtil.setRollbackOnly();
       }
       BLogicResult result = new BLogicResult();
       result.setResultString("failure");
       return result;
   }
}
```

Import TransactionUtil class

Call setRollbackOnly method

※ Also refer to notes regarding TransactionUtil#setRollbackOnly stated later

- Using multiple databases

  Transaction manager uses JtaTransactionManager (provided by Spring) when transaction control is to be executed across multiple DBs by using J2EE Server that supports JTA. JtaTransactionManager is a transaction manager that executes transaction, which may span over multiple resources and hence the transaction is acquired using JNDI. The datasource is set for individual DAO and not for the transaction manager. Refer to “CB-01 Database Access” for the details of DAO.

  Please refer to the SpringAPI for the details on JtaTransactionManager. Also for the application server JTA support, please contact the vendor of the application server.

- Example Bean definition file

```
<!—DataSource setting. -->
<bean id="dataSource" class="org.springframework.jndi.JndiObjectFactoryBean">
  <property name="jndiName"><value>DataSource1</value></property>
</bean>
<bean id="dataSource2" class="org.springframework.jndi.JndiObjectFactoryBean">
  <property name="jndiName"><value>DataSource2</value></property>
</bean>

<!—Transaction manager for multiple datasources. -->
<bean id="transactionManager"
    class"org.springframework.transaction.jta.JtaTransactionManager">
</bean>

(Setting of transaction is similar to that of transaction setting where au-
toproxy AOP is used)

<!—Definition of business object. Define it by wrapping in transaction proxy. -->
<bean id="sumService"
      class="jp.terasoluna.sample2.service.impl.SumServiceImpl">
  <property name="queryDAO"><ref bean="queryDAO"/></property>
</bean>

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

<!— DAO definition 2 -->
<bean id="queryDAO2" class="jp.terasoluna.fw.dao.ibatis.QueryDAOiBatisImpl">
  <property name="sqlMapClient"><ref local="sqlMapClient"/></property>
  <property name="dataSource"><ref bean="dataSource2"/></property>
</bean>
```

JtaTransactionManager is specified to transactionManager property.

Datasource used by DAO is specified.

- (Reference) <tx:method/> attribute list

Attribute	Required	Default	Explanation
name	○	-	Method name which will be target of trans action Example：'get*' , 'handle*' , 'on*Event'
propagation	-	REQUIRED	Settings related to propagation of Transact ion
isolation	-	DEFAULT	Isolation level of Transaction
timeout	-	-1	Transaction Timeout (Seconds) Default timeout time will be used if not sp ecified
read-only	-	false	Whether it is a readonly Transaction
rollback-for	-	-	Exception which will be target of Rollback Multiple exceptions can be specified. Com ma separated. Default is java.lang.RunTimeException and its derived classes
no-rollback-for	-	-	Exceptions which is removed from the tar get of rollback. Multiple exceptions can be specified. Comma separated.

◆ Additional points

None.

■ Reference

- “CB-01 Database Access”
- “WH-01 Business Logic Execution”

■ Example

- TERASOLUNA Tutorial　Rich Client Version
  - /webapps/WEB-INF/dataAccessContext-local.xml
  - /webapps/WEB-INF/applicationAOP.xml etc.

- Sample covering all TERASOLUNA (Web based) Functions
  - “UC02 Transaction Control”

- /webapps/transaction/*
- /webapps/WEB-INF/transaction/*
- jp.terasoluna.thin.functionsample.transaction.*

■ Remarks

- **Important points to be noted while using TransactionUtil#setRollbackOnly**
  1. When using this utility, its necessary that TransactionInterceptor is weaved into the target section
  2. In case when Transaction settings are nested at multiple levels, if setRollbackOnly is used in the inner level, it has to be used at the outer level also. In case above is not done, rollback will be performed but exception will also occur. Due to mistake in settings etc, if multiple transactionInterceptors are weaved corresponding to same location, exception will occur.
  3. If current utility is tested using Jnit, NoTransactionException might take place. This is because TransactionStatus cannot be read. Usually TransactionStatus is created by TransactionInterceptor
  4. For testing, temporary Transactionstatus needs to be created. Refer below mock-class to create a temporary TransactionStatus

```
import org.springframework.transaction.TransactionStatus;
import org.springframework.transaction.interceptor.DefaultTransactionAttribute;
import org.springframework.transaction.interceptor.TransactionAspectSupport;
import org.springframework.transaction.interceptor.TransactionAttribute;
import org.springframework.transaction.support.SimpleTransactionStatus;

public class MockTransactionAspectSupport extends TransactionAspectSupport {
   private TransactionStatus status = null;

   public MockTransactionAspectSupport() {
      TransactionAttribute txAttr = new DefaultTransactionAttribute();
      status = new SimpleTransactionStatus();
      String joinpointIdentification = null;
      this.prepareTransactionInfo(txAttr, joinpointIdentification, status);
   }

   public boolean isRollbackOnly() {
      return status.isRollbackOnly();
   }
}
```

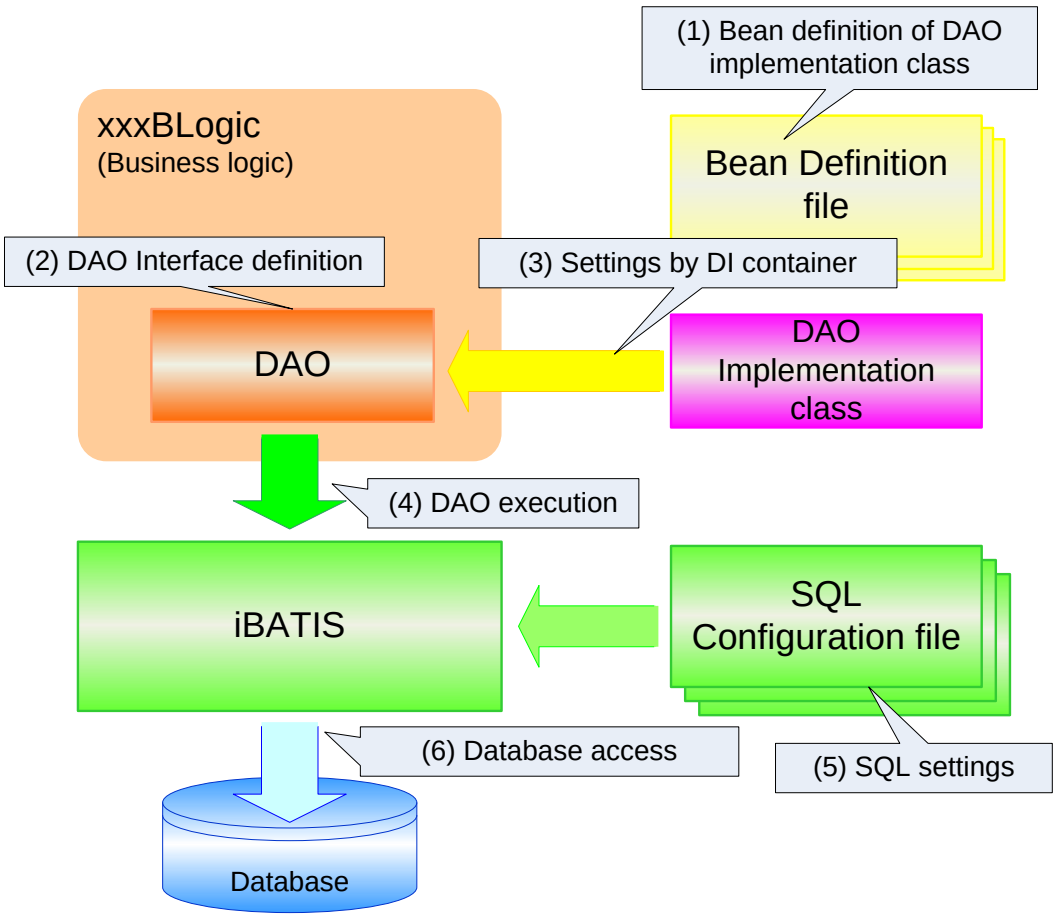
CB-01 Database Access

■ Overview

◆ Functional Overview

- Provides the DAO to simplify database access.
- Provides the DAO interfaces given below that remove the dependencies between the business logic and JDBC APIs, RDBMS products, and O/R Mapping Tools.
  - ➢ QueryDAO
    Used to search data.
  - ➢ UpdateDAO
    Used to insert / update / delete data.
  - ➢ StoredProcedureDAO
    Used to call stored procedures.
  - ➢ QueryRowHandleDAO
    Used when huge data is to be searched and each record to be processed one by one.
- Using Spring + iBatis, the following DAO implementation classes are provided as the default classes
  - ➢ QueryDAOiBatisImpl
    Used to search data for iBatis.
  - ➢ UpdateDAOiBatisImpl
    Used to insert / update / delete data for iBatis.
  - ➢ StoredProcedureDAOiBatisImpl
    Used to call stored procedures for iBatis.
  - ➢ QueryRowHandleDAOiBatisImpl
    Used when huge data is to be searched and each record to be processed one by one for iBatis.
- Declarative transactions are managed using AOP. So, the business logic programmer need not implement the transaction handling that involves opening/closing of connection, etc.
- Please refer to iBATIS Reference for details on the usage of iBATIS.
- All SQL statements are to be specified in the configuration files as per iBATIS specifications.

◆ Schematic diagram



◆ Description

(1) The DAO implementation class is defined in a bean definition file.

(2) Specify the DAO interface attribute in the business logic class and the setter method for the interface.

(3) Specify the DAO implementation class defined in (1), in the bean definition of the business logic in which database access is required. By doing this DI container will set the object of DAO implementation class in the DAO interface attribute in the business logic class.

(4) The iBATIS API is called through DAO implementation class set in business logic. Please refer to the Javadoc of the DAO implementation classes provided by TERASOLUNA Server Framework for Java.

(5) BATIS gets the SQL from iBATIS mapping file based on the SQLID specified in Business logic.

(6) The SQL accesses the database by using the data source that is set in the Bean definition file.

■ Usage

◆ Coding Points

- Bean definition of Data source

  Data source settings are defined in the application bean definition file.

  - ➢ Application Bean definition file
    - ✧ In case of JNDI, use JndiObjectFactoryBean.

      In case of Tomcat, note that (as per the settings), "java:comp/env/" should be attached at the beginning of data source name

```
<bean id="TerasolunaDataSource"
      class="org.springframework.jndi.JndiObjectFactoryBean">
  <property name="jndiName">
    <value>java:comp/env/TerasolunaDataSource</value>
  </property>
</bean>
```

    - ✧ In case when JNDI name is changed frequently

      By using <context:property-placeholder/> element of context schema, write the JNDI name in the property file.

```
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:util="http://www.springframework.org/schema/util"
      xmlns:context="http://www.springframework.org/schema/context"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://www.springframework.org/schema/util
http://www.springframework.org/schema/util/spring-util-2.5.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-2.5.xsd">

<!—properties related to JNDI -->
<context:property-placeholder location="WEB-INF/jndi.properties"/>
<bean id="TerasolunaDataSource"
      class="org.springframework.jndi.JndiObjectFactoryBean">
  <property name="jndiName">
    <value>${jndi.name}</value>
  </property>
</bean>
```

Define the Property-placeholder

Define the Context schema

Example of property file (jndi.properties)

```
jndi.name=java:comp/env/TerasolunaDataSource
```

✧ When JNDI is not used, use DriverManagerDataSource as given below. It is recommended that the settings for DB connection that will change as per the environment should be specified in property file. In such a case, use <context:property-placeholder/> element of context schema as shown below.

```
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:util="http://www.springframework.org/schema/util"
      xmlns:context="http://www.springframework.org/schema/context"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://www.springframework.org/schema/util
http://www.springframework.org/schema/util/spring-util-2.5.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-2.5.xsd">

  <!--properties related to JDBC -->
  <context:property-placeholder location="WEB-INF/jdbc.properties"/>

 <bean id="TerasolunaDataSource"
    class="org.springframework.jdbc.datasource.DriverManagerDataSource"
       destroy-method="close">
  <property name="driverClassName" value="${jdbc.driverClassName}"/>
  <property name="url" value="${jdbc.url}"/>
  <property name="username" value="${jdbc.username}"/>
  <property name="password" value="${jdbc.password}"/>
</bean>
```

Define property-placeholder

Define schema

Specify classpath

Define DriverManagerData-Source

✧ When multiple data sources are to be defined, define id attribute of bean element with different values.

➢ Example of Property File (jdbc.properties)

```
jdbc.driverClassName=oracle.jdbc.OracleDriver
jdbc.url=jdbc:oracle:thin:@192.168.0.100:1521:ORCL
jdbc.username=name
jdbc.password=password
```

- iBATIS mapping definition file

  This file defines for mapping for SQL statements used in business logic and the mapping of the SQL result to a JavaBean. A file should be created for every module. However, SQL ID should be unique in the application. Please refer to iBATIS reference for details of the SQL syntax.

  - Example of Select statement
    - Use <select> element while executing Select statement.
    - Specify the class to store the SQL result for resultClass attribute. In case of multiple results, array of specified class is returned.

```
<select id="getUser"
        resultClass="jp.terasoluna.·····.service.bean.UserBean">
   SELECT ID, NAME, AGE, BIRTH FROM USERLIST WHERE ID = #ID#
</select>
```

  - Execution of Insert, Update, Delete statements
    - Use <insert> element while executing Insert statement
    - Use <update> element while executing Update statement
    - Use <delete> element while executing Delete statement
    - In parameterClass attribute, specify the class that stores the data to be registered. Each value between "#" is replaced by the value of that property in the parameter class and the SQL is executed.

```
<insert id="insert_User"
  parameterClass="jp.terasoluna.·····service.bean.UserBean">
    INSERT INTO USERLIST (ID, NAME, AGE, BIRTH ) VALUES (
    #id#, #name#, #age#, #birth#)
</insert>
```

  - Execution of Procedure statement
    - Use <procedure> element while executing a procedure statement.
    - The basic usage is same as other elements, however, the number of attributes are lesser than the <select> element.
    - For setting values, getting result use the parameterMap attribute and <parameterMap> element.

```
<sqlMap namespace="user">
    <parameterMap id="UserBean" class="java.util.HashMap">
        <parameter property="inputId" jdbcType="NUMBER"
                javaType="java.lang.String" mode="IN"/>
        <parameter property="name" jdbcType="VARCHAR"
                javaType="java.lang.String" mode="OUT"/>
    </parameterMap>
    <procedure id="selectUserName" parameterMap="user.UserBean">
       {call SELECTUSERNAME(?,?)}
    </procedure>
```

- iBATIS configuration file

The iBATIS configuration file contains the iBATIS settings. However, in case of TERASOLUNA Server Framework for Java, only the location of iBATIS mapping definition file is specified. The data source and transaction management are implemented using the functionality that integrates it with Spring. So, these settings are not required in this file.

- It can contain multiple <sqlMap> elements. Hence in case when iBatis mapping file is divided, describe them with multiple <sqlMap> elements.
- By using useStatementNamespaces of <settings> attribute, it can be specified whether to specify SQLID with a fully-qualified name. When useStatementNamespace is true, SQLID is to be specified as [Namespace + "." + SQLID]. (For-example: namespace is "user", and SQLID is "getUser". Then SQLID is to be specified as "user. getUser")

- Bean definition of SqlMapClientFactoryBean

When iBATIS is used in Spring, the bean definition in iBATIS configuration file should be set to DAO using SqlMapClientFactoryBean. SqlMapClientFactoryBean manages the main class "SqlMapClient" used for iBATIS data access.
In the application, there is only one bean definition in the iBATIS configuration file.

- iBATIS configuration file
  - In "configLocation" property, path from the context root of iBATIS configuration file is specified.
  - In case of single database, set the Bean definition of data source to be used, in "dataSource" property.

```
<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>
```

✧ In case of multiple database, set only “configLocation” property without specifying “datasource” property,.

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

- Bean definition of DAO
  - DAO implementation class is basically defined in application Bean definition file. Transaction settings are to be set on the DAO also. Please refer to the framework manual of “CA-01 Transaction Control Function(s)” for transaction settings.
    The Bean definition of iBATIS configuration file should be set in "sqlMap-Client" property of DAO implementation class when defining Bean.

```
<bean id="sqlMapClient"
      class="org.springframework.orm.ibatis.SqlMapClientFactoryBean">
  <property name="configLocation" value="WEB-INF/sqlMapConfig.xml"/>
  <property name="dataSource" ref="TerasolunaDataSource"/>
</bean>
<bean id="queryDAO"
      class="jp.terasoluna.fw.dao.ibatis.QueryDAOiBatisImpl">
  <property name="sqlMapClient" ref="sqlMapClient"/>
</bean>
```

  - In case of multiple databases, the data source used in DAO implementation class should be specified not only in "sqlMapClient" property settings, but also in "dataSource" property.

```
<bean id="queryDAO"
      class="jp.terasoluna.fw.dao.ibatis.QueryDAOiBatisImpl">
  <property name="sqlMapClient" ref="sqlMapClient"/>
  <property name="dataSource" ref="TerasolunaDataSource"/>
</bean>
```

- Specification of return value of QueryDAOiBatisImpl method
  - In case the search result is a single record or an array of multiple records, and if the following method of QueryDAOiBatisImpl is to be used, the class of same type as that of return value should be passed as the argument. By doing so, the class cast error of business logic can be avoided (DAO throws IllegalClassTypeException).
    - executeForObject (String sqlID, Object bindParams, Class clazz)
    - executeForObjectArray(String sqlID, Object bindParams, Class clazz)
    - executeForObjectArray(String sqlID, Object bindParams, Class clazz, int beginIndex, int maxCount)

```
UserBean bean = dao.executeForObject("getUser", null, UserBean.class);
UserBean[] bean
        = dao.executeForArray("getUser", null, UserBean[].class);
```

  - In case the search result is List of multiple records, and if the following method of QueryDAOiBatisImpl is to be used, then unlike the case of array, the class of same type as that of return value should not be passed to the argument. Therefore, note that the type used as in case of array is not assured.
    - executeForObjectList(String sqlID, Object bindParams)
    - executeForObjectList(String sqlID, Object bindParams, int beginIndex, int maxCount)

```
List bean = dao.executeForList("getUser", null);
```

- Example to get the list data using QueryDAOiBatisImpl
  Setting and context example, when normal list data (1 page) is retrieved from the database by using QueryDAOiBatisImpl, is given below.

① Define DAO implementation class in Bean definition file as given below.
  - Bean definition file

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

② In the Bean definition of the business logic that executes data access, set the DAO implementation class defined in ①. Further, in Business logic, it is necessary to provide the attributes and their Setter for setting the DAO using DI container.
  - Bean definition file

```
<bean id="listBLogic" scope="prototype"
 class="jp.terasoluna.sample.service.blogic.ListBLogic">
  <property name="queryDAO" ref="queryDAO"/>
</bean>
```

- Business Logic

  Use executeForObjectArray(String sqlID, Object bindParams, Class clazz, int beginIndex, int maxCount) method of DAO set in Bean definition file. It is necessary to set "Start index" and "Total count" defined in ActionForm, in the argument of method.

  Please refer to “WI-01 List Display Function(s)” for usage of List Display.

```
private QueryDAO queryDAO = null;
    ......Setter is omitted
public UserBean[] getUserList(ListBean bean) {
    int startIndex = bean.getStartIndex();
    int row = bean.getRow();
    UserBean[] bean = queryDAO.executeForObjectArray(
            "getUserList", null, UserBean.class, startIndex, row);
    return bean;
}
```

List data is acquired from database by using DAO that is set.

To get a java.util.List type and not an array, use executeForObjectList(String sqlID, Object bindParams, int beginIndex, int maxCount) method.

```
private QueryDAO queryDAO = null;
    ......Setter is omitted
public List<UserBean> getUserList(ListBean bean) {
    int startIndex = bean.getStartIndex();
    int row = bean.getRow();
    List<UserBean> bean = queryDAO.executeForObjectList(
            "getUserList", null, startIndex, row);
    return bean;
}
```

List data is acquired from database by using DAO that is set.

- Example of data registration where UpdateDAOiBatisImpl is used

  The settings and source-code example where the data is registered in database by using UpdateDAOiBatisImpl, is described below. The definition / setting method of Bean definition file is same as that of QueryDAOiBatisImpl, hence it is not mentioned.

  - Business logic

    Use the method of DAO set in Bean definition file.

```
private UpdateDAO updateDAO = null;
    ......Setter is omitted
public void register(UserBean bean) {
    ......
    updateDAO.execute("insertUser", bean);
    ......
}
```

The data is registered in the database by using DAO that is set.

- Registration example of multiple data where UpdateDAOiBatisImpl is used (Online batch processing)

  The settings and source-code example where multiple data is registered in the database by using UpdateDAOiBatisImpl, is given below. The definition / setting method of Bean definition file is same as that of QueryDAOiBatisImpl, hence it may be omitted.

  Please refer to JavaDoc of UpdateDAOiBatisImpl for details.

  - Business logic

    Use executeBatch(List<SqlHolder>) method of DAO set in Bean definition file.

```
UserBean[] bean = map.get("userBeans");
List<SqlHolder> sqlHolders = new ArrayList<SqlHolder>();
for (int i = 0; i < bean.length; i++) {
    sqlHolders.add(new SqlHolder("insertUser", bean[i]);
}
updateDAO.executeBatch(sqlHolders);
......
```

Creates SqlHolder list, which is the update target, storing the sqlId, and objects as parameter.

  - Notes

    executeBatch uses batch execution function of iBATIS. executeBatch returns the number of records updated by SQL execution, as return value. However, since java.sql.PreparedStatement is used, the correct number of records may not be retrieved by the driver. When the driver that cannot get the correct number of records is used, batch update should not be used in case the count of updated records affects the transaction (e.g where error handling is executed when updated number of records is 0).

    (Reference documents)

    http://otndnld.oracle.co.jp/document/products/oracle10g/101/doc_v5/java.101/B13514-02.pdf

    Please refer to page number 450 in “Standard batch processing > Oracle implementation > updated row count”

- Example to get the data using StoredProcedureDAOiBatisImpl

  The following example describes the setting and coding for retrieving data from the database using StoredProcedureDAOiBatisImpl. The definition/setting method of Bean definition file is same as that of QueryDAOiBatisImpl, hence it is not mentioned.

  - Business logic

    Use DAO method set in Bean definition file.

```
    private StoredProcedureDAO spDAO = null;
        ......Setter is omitted
    public boolean register(UserBean bean) {
        ......
        Map<String, String> params = new HashMap<String, String>();
        params.put("inputId", bean.getId());
        spDAO.executeForObject("selectUserName", params);
        ......
    }
```

Procedure is executed by using DAO that is set.

- iBATIS mapping definition file
  - The settings for storing input/output of procedure are described by <parameterMap> element. jdbcType attribute should be specified. Please refer to iBATIS reference for more details on the settings.

```
<sqlMap namespace="user">
    <parameterMap id="UserBean" class="java.util.HashMap">
        <parameter property="inputId" jdbcType="NUMBER"
                javaType="java.lang.String" mode="IN"/>
        <parameter property="name" jdbcType="VARCHAR"
                javaType="java.lang.String" mode="OUT"/>
    </parameterMap>
    <procedure id="selectUserName" parameterMap="user.UserBean">
       {call SELECTUSERNAME(?,?)}
    </procedure>
```

- Procedure to be executed (Example showing usage of Oracle)

```
CREATE OR REPLACE PROCEDURE SELECTUSERNAME
(inputId IN NUMBER, name out VARCHAR2) IS
BEGIN
    SELECT name INTO name FROM userList WHERE id = inputId ;
END ;
```

- Example to get the data using QueryRowHandleDAOiBatisImpl

  The following example describes the setting and coding for retrieving data from the database using QueryRowHandleDAOiBatisImpl. The definition/setting method of Bean definition file is same as that of QueryDAOiBatisImpl, hence it is not mentioned.

  - Implementation of DataRowHandler

```
import jp.terasoluna.fw.dao.event.DataRowHandler;

public class SampleRowHandler implements DataRowHandler {
    public void handleRow(Object param) {
        if (param instanceof HogeData) {
            HogeData hogeData = (HogeData)param;
            // source-code that updates database
            // for each record
        }
    }
}
```

For each record handleRow method is called. Data of one record is passed to this method everytime.

If database is updated for each record, pass UpdateDAO to the DataRowHandler in advance.
In case of download, passing ServletOutputStream etc is good.

➢ Business Logic

```
    private QueryRowHandleDAO queryRowHandleDAO = null;
        ......Setter method is omitted

    public BLogicResult execute(BlogicParam params) {
        Parameter param = new Parameter();
        HogeDataRowHandler dataRowHandler = new HogeDataRowHandler();

        queryRowHandleDAO.executeWithRowHandler(
                "selectDataSql", param, dataRowHandler);

        BLogicResult result = new BLogicResult();
        result.setResultString("success");
        return result;
    }
```

Instance of DataRowHandler that is going to actually update the database for each record is passed

◆ Extension Points

None

■ Classes

	Class Name	Overview
1	jp.terasoluna.fw.dao.QueryDAO	DAO interface executing Query SQL
2	jp.terasoluna.fw.dao.UpdateDAO	DAO interface executing Update SQL
3	jp.terasoluna.fw.dao.StoredProcedureDAO	DAO interface executing StoredProcedure

4	jp.terasoluna.fw.dao.QueryRowHandleDAO	DAO interface that processes Query type SQL one record at a time.
5	jp.terasoluna.fw.dao.ibatis.QueryDAOiBatisImpl	Implementation class for iBATIS of QueryDAO interface
6	jp.terasoluna.fw.dao.ibatis.UpdateDAOiBatisImpl	Implementation class for iBATIS of UpdateDAO interface
7	jp.terasoluna.fw.dao.ibatis.StoredProcedureDAOiBatisImpl	Implementation class for iBATIS of StoredProcedureDAO interface
8	jp.terasoluna.fw.dao.ibatis.QueryRowHandleDAOiBatisImpl	Implementation class for iBATIS of QueryRowHandleDAO interface

■ Related Function(s)

- "CA-01 Transaction Control Function(s)"
- "WI-01 List Display Function(s)"

■ Example

- TERASOLUNA Server Framework for Java (Web based) Tutorial
  - 2.6 Database Access
  - 2.7 Register
  - List View, Registration Screen

- TERASOLUNA Server Framework for Java (Rich Client Version) Tutorial
  - 2.4 Database Access
  - /webapps/WEB-INF/dataAccessContext-local.xml
  - /webapps/WEB-INF/sql-map-config.xml
  - /sources/sqlMap.xml
  - jp. terasoluna.rich.tutorial.service.blogic.DBAccessBLogic.java etc.

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web based)
  - "UC01 Database Access"
    - /webapps/database/*
    - /webapps/WEB-INF/database/*
    - jp.terasoluna.thin.functionsample.database.*

- "Spring version Tutorial"
  - "2.6 Database Access"
  - "2.7 Registration"
  - "3.4 Database Access"

■ Remarks

- **Important Points to be noted while fetching large data**
  If a query to fetch a large size data is written using <statement>, <select>, <procedure> elements of iBATIS Mapping Definition file, there is need to specify appropriate value in fetchSize attribute.
  fetchSize contains the number of records to be fetched in one transmission between JDBC driver and Database. If it is not specified, default value depending of each JDBC driver will be used.
  For example in case of PostgreSQL JDBC driver (confirmed with postgresql-8.3-604.jdbc3.jar), all records searched will be fetched in one transmission. There should not be a problem in case of few thousand records. However in case of tens of thousands of records, there is a possibility of heap memory getting stressed.

CC-01 JNDI Access

■ Overview

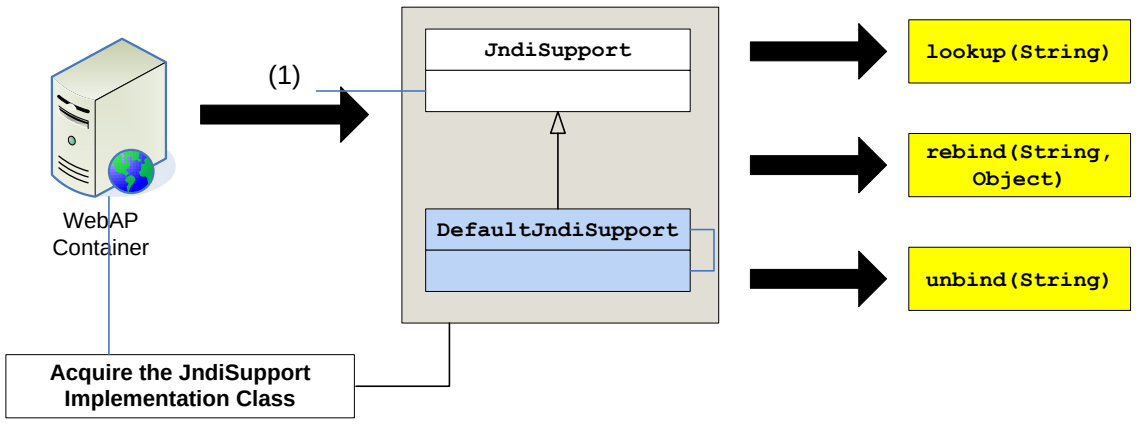
◆ Functional Overview

- Supports interface of JNDI related functions.
- This interface should be implemented while handling the JNDI resources of WebAP container.
- Interface and default implementation class DefaultJndiSupport is provided in TERASOLUNA Server Framework for Java.

◆ Utility methods

Method name	Overview
lookup(String name)	Gets the object of specified name
rebind(String name, Object obj)	Overwrites the existing binding after binding the name with the object.
unbind(String name)	Unbinds the specified object.

◆ Schematic Diagram



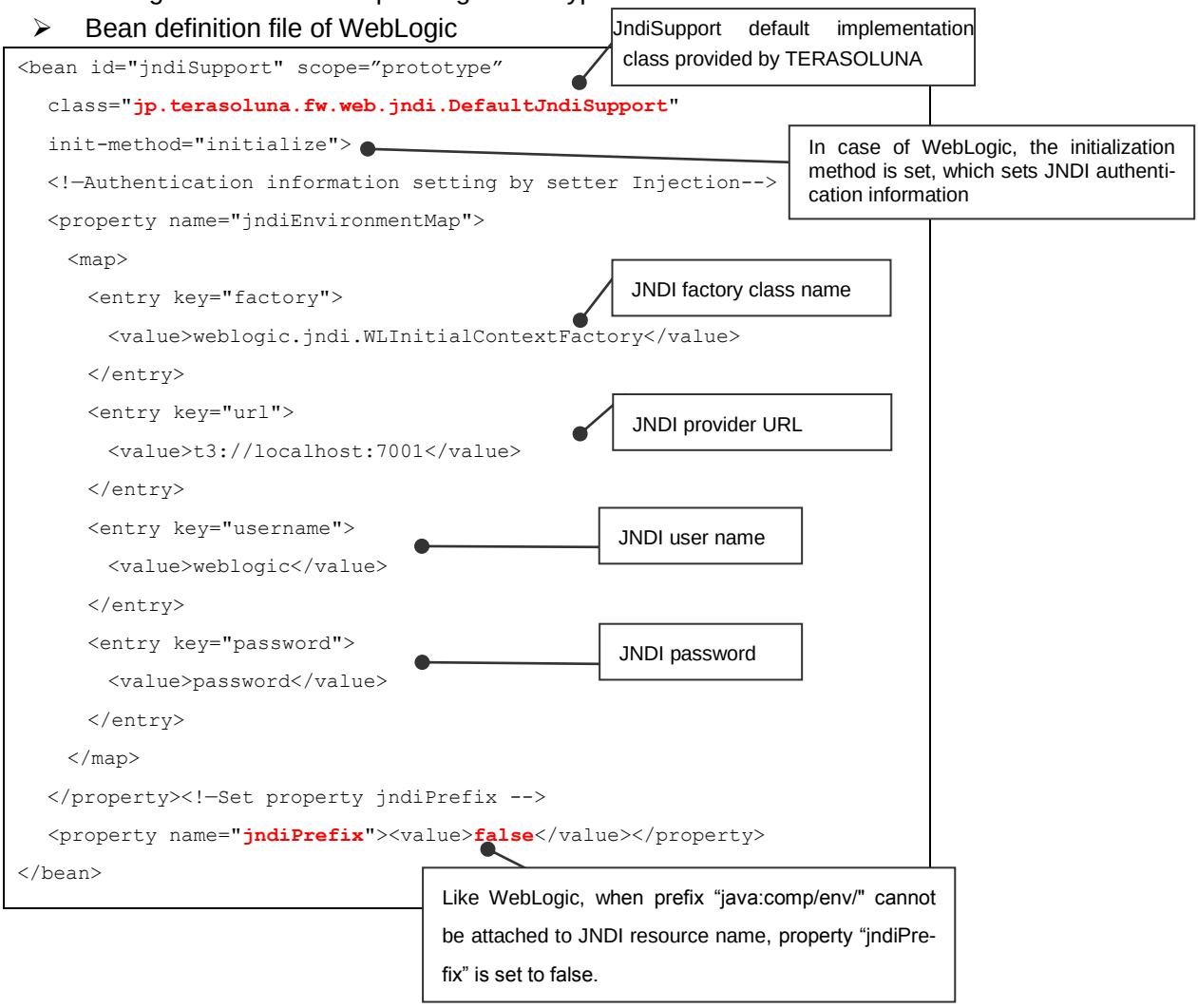
◆ Description

(1) JndiSupport implementation class is acquired from DI container and JNDI resource is used.

■ Usage

◆ Coding Points

- To authenticate using JNDI, set the required properties to Bean definition file as given below.
  The setting method differs depending on the type of JNDI server.
  - ➢ Bean definition file of WebLogic




```
<bean id="jndiSupport" scope="prototype"
  class="jp.terasoluna.fw.web.jndi.DefaultJndiSupport"
  init-method="initialize">
  <!—Authentication information setting by setter Injection-->
  <property name="jndiEnvironmentMap">
    <map>
      <entry key="factory">
        <value>weblogic.jndi.WLInitialContextFactory</value>
      </entry>
      <entry key="url">
        <value>t3://localhost:7001</value>
      </entry>
      <entry key="username">
        <value>weblogic</value>
      </entry>
      <entry key="password">
        <value>password</value>
      </entry>
    </map>
  </property><!—Set property jndiPrefix -->
  <property name="jndiPrefix"><value>false</value></property>
</bean>
```

- Tomcat Bean definition file

```
<bean id="jndiSupport" scope="prototype"
   class="jp.terasoluna.fw.web.jndi.DefaultJndiSupport" >
   <!—Settings of property jndiPrefix (Default value is false) -->
   <property name="jndiPrefix"><value>false</value></property>
</bean>
```

This attribute should be false in Tomcat.
Prefix "java:comp/env/" should be attached to JNDI resource name and read-only resources registered in the settings of Tomcat server become accessible if it is set to true. When it is necessary to access the resources registered in the setting of Tomcat server, specify "java:comp/env/" + JNDI name in Action implementation by keeping this setting as false.

- When JNDI resource is used in business logic, execute settings as given below.
  - Bean definition file

```
<bean id="jndiLogic" scope="prototype"
   class="jp.terasoluna.sample.JndiLogic">
   <property name="jndiSupport">
     <ref bean="jndiSupport" />
   </property>
</bean>
```

JNDI support class

  - Business logic implementation example

```
public class JndiLogic {
   private JndiSupport jndiSupport = null;

   public void setJndiSupport(jndiSupport) {
     this.jndiSupport = jndiSupport;
   }

   public Object jndiLookup(String name) {
     return jndiSupport.lookup(name);
   }
}
```

Gets the JndiSupport implementation class by setter injection.

In case of Weblogic, describe "JNDI resource name" only.

In case of Tomcat, while accessing the resources registered in the server setting, "java:comp/env/" and "JNDI resource name" are described. Since the resources retrieved in this case are read-only, rebind and unbind cannot be used.

■ Classes

	Class name	Overview
1	jp.terasoluna.fw.web.jndi.JndiSupport	It is the JNDI related utility interface.
2	jp.terasoluna.fw.web.jndi.DefaultJndiSupport	It is the JNDI related utility default implementation class provided by TERASOLUNA.
3	jp.terasoluna.fw.web.jndi.JndiException	It is used to indicate JNDI related errors

◆ Extension Points

Create the class that implements JndiSupport interface and set it in Bean definition file.

■ Related Function(s)

- None

■ Example

- Sample covering all functions of the TERASOLUNA framework (Web based)
  - "UC03 JNDI access"
    - /webapps/jndi/*
    - /webapps/WEB-INF/jndi/*
    - jp.terasoluna.thin.functionsample.jndi.*

■ Remarks

- None

CD-01 Utility Functions

■ Overview

◆ Functional Overview

- Provides the utility classes for frequently used functions in business development

◆ List of utility classes provided

Utility class	Overview
jp.terasoluna.fw.util.ClassUtil	Utility class to create an instance from a String.
jp.terasoluna.fw.util.DateUtil	Utility class for date / time / calendar operations.
jp.terasoluna.fw.util.FileUtil	Utility class for file operations.
jp.terasoluna.fw.util.HashUtil	Utility class to calculate the hash value.
jp.terasoluna.fw.util.PropertyUtil	Utility class to get properties from property file.
jp.terasoluna.fw.util.StringUtil	Utility class for string parsing/manipulation.

■ ClassUtil

◆ Overview

Utility class that creates an instance from a String.

◆ List of utility methods

Method Name	Overview
create(String className)	Creates an instance of the specified class name.
create(String className, Object[] constructorParameter)	Creates an instance of the specified class name. constructorParameter is passed as argument to the constructor.

◆ Usage

A Simple usage example is given below.

- Class where instance is to be created (With constructor)

```
package jp.terasoluna.xxx;
……
public class Messages {
  // Constructor.
  public Messages() {
  }
  // Constructor.
  public Messages(Error err) {
    this.add(err);
  }
  ……
}
```

- Example

```
// Generates an instance using a constructor Message(Error err).
Error err = new Error();
Object[] param = new Object[] { err };
Object obj = ClassUtil.create("jp.terasoluna.xxx.Messages", param);
```

Instance is created by using constructor Message (Error err)

■ DateUtil

◆ Overview

Utility class for Date / Time / Calendar operations

◆ List of utility methods

Method Name	Overview
getSystemTime()	Gets the system time.
dateToWarekiString(String format, java.util.Date date)	Converts java.util.Date instance as Wareki to the format specified in ‘format’.
getWarekiGengoName(Date date)	Gets WarekiGengo of the specified date.
getWarekiGengoRoman(Date date)	Gets roman notation (Abbreviated form) of WarekiGengo of the specified date.
getWarekiYear(Date date)	Gets WarekiYear of the specified date.

◆ Usage

While using Wareki-related methods, it is necessary to set Gengo name, roman notation of Gengo and start date of Gengo, in ApplicationResources.properties.

- Example of setting ApplicationResources.properties

```
wareki.gengo.0.name=Heisei
wareki.gengo.0.roman=H
wareki.gengo.0.startDate=1989/01/08
wareki.gengo.1.name=Showa
wareki.gengo.1.roman=S
wareki.gengo.1.startDate=1926/12/25
wareki.gengo.2.name=Taisho
……
```

Gengo name

Roman notation of Gengo

Start date of Gengo (Seireki: yyyy/MM/dd format)

Use only for association of 3 settings mentioned above. Any string can be set.

- Example

```
// Prepare arguments
df = new SimpleDateFormat("yyyy.MM.dd HH:mm:ss");
date = new Date(df.parse("1978.01.15 00:00:00").getTime());

// Convert to Wareki format (Result:wareki = " Showa 53, Jan. 15th
Sunday")
String wareki = DateUtil.dateToWarekiString("GGGGyyMMddEEEE", date);

// Get Gengo(Result:gengo = "Showa")
String gengo = DateUtil.getWarekiGengoName(date);

// Get Gengo (Roman notation)(Result : gengoR = "S")
String gengoR = DateUtil.getWarekiGengoRoman(date);

// Get Wareki year(Result : year = "53")
String year = DateUtil.getWarekiYear(date);
```

[G]: Pattern string indicating Gengo
Example:
(Pattern string with more than 4 characters)
Meisho, Taisho, Showa, Heisei
(Pattern string with less than 3 characters)
M, T, S, H

"y": Pattern string indicating year (Wareki)

"E": Pattern string indicating the day
Example:
(Pattern string with more than 4 characters)
Monday, Tuesday, Wednesday
(Pattern string with less than 3 characters)
月、火、水

■ FileUtil

◆ Outline

Utility class for file operations. In this utility class “Directory corresponding to session ID" is a directory whose name is equal to the hexadecimal hash value of session ID.
※ Note that rmdirs(File dir) is a method which is not related to session ID.

◆ Utility Methods List

Method Name	Outline
getSessionDirectoryName(String sessionId)	Gets the directory name for the specified session ID.
getSessionDirectory(String sessionId)	Gets the directory for the specified session ID from the base directory. When base directory is not set in property or when property value is a null character, the temp directory is used as base directory.
makeSessionDirectory(String sessionId)	Creates a directory corresponding to the specified session ID. Returns true when directory is created successfully.
removeSessionDirectory(String sessionId)	Deletes the directory corresponding to specified session ID. Returns true when directory is deleted successfully.
rmdirs(File dir)	Deletes the directory corresponding to specified session. Recursively deletes the directory even if there are files in a sub-directory.

◆ Usage

The base directory (to create session directories) is set in system setting property file (system.properties). temp directory (/temp) is used when the property is not set.

- Example of setting the System setting property file

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

Path of base directory is specified.

- Usage Example

```
// Get the directory name corresponding to specified session ID.
// (Result:dirname is the directory name corresponding to specified
session ID)
String dirName = FileUtil.getSessionDirectoryName("0123abc");

// Get the directory for the specified session ID
// (Result : In dir, the directory corresponding to
// session ID under /tmp/session is stored)
File dir = FileUtil.getSessionDirectory("0123abc");

// Create a directory for the specified session ID.
// (Result: Directory for the session ID under /temp/sessions is
created)
boolean makeResult = FileUtil.makeSessionDirectory("0123abc");

// Delete the directory for the specified session ID
// (Result: Directory for the session ID under /tmp/sessions is de-
leted)
boolean removeResult = FileUtil.removeSessionDirectory("0123abc");

// Delete the specified directory.
// (Result: Directory/rmtemp/rmdirs1 is deleted)
File dir = new File(/rmtemp/ + "rmdirs1");
boolean result = FileUtil.rmdirs(dir);
```

Directory corresponding to session ID = Hexadecimal hash value of “0123abc”

■ HashUtil

◆ Overview

Utility class to calculate the hash value.

◆ List of utility methods

Method Name	Overview
hash(String algorithm, String str)	Gets the hash value of string by the specified algorithm.
hashMD5(String str)	Gets the hash value of string by algorithm MD5.
hashSHA1(String str)	Gets the hash value of string by algorithm SHA1.

◆ Usage

A simple usage example is given below. Simple methods are not mentioned.

```
// Get the hash value.
// (Result:Value equal to
// MessageDigest.getInstance("MD5").digest("abc".getBytes())
// is stored in hashValue)
byte[] hashValue = HashUtil.hash(paramAlgorithm, paramStr);
```

■ PropertyUtil

◆ Overview

Utility class to get the property from a property file.

◆ List of utility methods

Method Name	Overview
addPropertyFile(String name)	Adds and loads the specified property file.
getProperty(String key)	Gets the property for the specified key.
getProperty(String key, String defaultValue)	Gets the property of specified key. The default value is returned when the property is not found.
getPropertyNames()	Gets the list of all property keys
getPropertyNames(String keyPrefix)	Gets the list of keys starting from the specified prefix.
getPropertiesValues(String propertyName, String keyPrefix)	Gets the value set, by specifying property file name and key string.
getPropertyNames(Properties localProps, String keyPrefix)	Gets the property, and gets the list of keys for the key prefix.
getPropertiesValues(Properties localProps,Enumeration<String> propertyNames)	Gets the values for properties in key list.
loadProperties(String propertyName)	loads property object from the specified property name.

◆ Usage

A simple usage example of each method is given below. The methods provided are classified into three types i.e. property loading, getting loaded property and handling of property object.

PropertyUtil loads the property from file "ApplicationResources.properties" on the class path. Moreover, other property files can be added and loaded by specifying a different file name in the property file.

※ When multiple properties for the same key exist in property file, the last value loaded becomes valid.

- Specification method to add property file

```
add.property.file.<Serial number from 1> = <Property file name to be added>
```

- Description example of property file (ApplicationResources.properties)

```
add.property.file.1=error.properties
add.property.file.2=xxxxx.properties
add.property.file.3=yyyyy.properties
```

- Contents of property file (errors.properties) added and loaded according to the description of Application.properties

```
error.message.01=Error message1
error.message.02=Error message2
```

Property file can be added and loaded by using PropertyUtil method as given below.

- Contents of property file (test.properties) added and loaded by using PropertyUtil method.

```
test.message.01=Test message1
test.message.02=Test message2
error.message.03=Error message3
```

- Loading property file

```
// Add and load property file. '.properties' may or may not be attached
PropertyUtil.addPropertyFile("test");
```

- Getting value of property

```
// Get property value of test.message.01 (Result:str = "Test message 1")
String str1 = PropertyUtil.getProperty("test.message.01");

// Get property value of error.message.04
// Get default message when there are no property settings.
// (Result:str2 = "Default")
String str2 = PropertyUtil.getProperty("error.message.04", "Default");

// Get the list of all keys of property.
// (Result: error.message.01~03, test.message.01~02 is stored in en1)
Enumeration en1 = PropertyUtil.getPropertyNames();

// Get list of keys of property starting with error.message
// (Result: error.message.01~03 is stored in en2)
Enumeration en2 = PropertyUtil.getPropertyNames("error.message");

// Get the property value of key starting with 'error' in error.properties.
// (Result: "Error message 1","Error message 2" are stored in set1)
Set set1 = PropertyUtil.getPropertiesValues("error", "error");
```

- Handling of Property Object

```
// Get Property object of test.properties.
// (Result:Property object when test.properties is loaded is stored)
Properties prop = PropertyUtil.loadProperties("test");

// Get the list of keys starting with 'error' in prop property object.
// (Result:error.message.03 is stored in en3)
Enumeration en3 = PropertyUtil.getPropertyNames(prop, "error");

// Get the set of property values of keys test.message.01 error.message.03
present in prop property object.
// (Result:"Test message1","Error message3" is 3 is stored in set2)
Enumeration en4 = new StringTokenizer("test.message.01 error.message.03");
Set set2 = PropertyUtil.getPropertiesValues(prop, en4);
```

■ StringUtil

◆ Overview

The utility class which performs character string operations

◆ List of utility methods

Method Name	Overview
isWhitespace(char c)	Determines if the specified string contains only white space.
rtrim(String str)	Deletes white space on the right side of the string. Returns null when argument is null.
ltrim(String str)	Deletes white space on the left side of the string. Returns null when argument is null.
trim(String str)	Deletes white space on the right and left side of the string. Returns null when argument is null.
toShortClassName(String longClassName)	Gets short class name (without qualified package) from a fully qualified class name.
getExtension(String name)	Gets extension from the specified string. When there is no extension, it returns a null string.
toHexString(byte[] byteArray, String delim)	Converts byte array to a hexadecimal string.
capitalizeInitial(String str)	Capitalizes initial character of the specified string.
parseCSV(String csvString)	Converts the string in CSV format to a string array. When a string starts and ends with comma, it is converted in such a way that the first or last element of the resultant array is a null string. In case of consecutive commas, it is converted as null string. When cvsString is null, it is converted to an array with 0 elements.
parseCSV(String csvString, String escapeString)	Converts the string in CSV format to a string array. When the string starts and ends with comma, it is converted in such a way that the first or last element of the resultant array is a null string. In case of consecutive commas, it is converted as null string. When cvsString is null, it is converted to an array of 0 number of elements. The comma next to the string set in an escape string is not identified as delimiter. The escape string after an escape string is not ident

	ified as escape character.
dump(Map<?, ?> map)	Acquires dump of argument map. When array is included in value object, executes toString() of each element object and outputs the string.
getArraysStr(Object[] arrayObj)	When value object to be dumped is of array format, it gets the values till all array elements are utilized.
hankakuToZenkaku(String value)	Converts a single byte string to a double byte string.
zenkakuToHankaku(String value)	Converts a double byte string to a single byte string.
filter(String str)	Executes HTML meta string conversion.
toLikeCondition(String condition)	Converts search condition string to LIKE predicate pattern string.
getByteLength(String value, String encoding)	Gets byte length of the specified string.

◆ Usage

A simple usage example of each method is given below. Methods that are easy to use are not mentioned.

```
// Get the short class name. (Result : shortClassName = "String")
String shortClassName = toShortClassName("java.lang.String");

// Get the extension.(Result:extension1 = ".txt", extenthin2 = ".xls")
String extention1 = getExtention("Test.txt");
String extention2 = getExtention("Test1.Test2.xls");

// Convert byte array to hexadecimal string.(Result : hexString =
"00/0A/64")
byte[] byteArray = {0, 10, 100};
String hexString = StringUtil.toHexString(byteArray, "/");

// Convert CSV to String array.
// (Result:strArray = {"bc", "def", "ghi", "jk,l"})
String[] strArray = parseCVS("abc,def,ghi,jk/,l", "/");
```

(Continued)

```
// Get dump of input/output map. (Result: dump = "String"
Map<String, String> map = new LinkedHashMap<String, String>();
map.put("1","Tokyo");
……
String dumpStr = dump(map);

// Get the repetitive values for tag till all the array elements are
utilized.
// (Result：arrayStr = "{1,2,3,4,5}")
String[] str = {"1", "2", "3", "4", "5"};
String arrayStr = StringUtil.getArraysStr(str);

// Converts single byte string to a double byte string. (Re-
sult:zenkaku = "ｱﾞ！Ａ８")
String zenkaku = StringUtil.hankakuToZenkaku("ｱﾞ!A8");

// Convert double byte string to single byte string. (Result:hankaku =
"A!ｱ")
String hankaku = StringUtil.zenkakuToHankaku("Ａ！ア");

// Execute HTML meta string conversion.
// (Result:meta = "&lt; &amp; &gt; &quot; ア")
String meta = StringUtil.filter("< & > ¥" ア");

// Convert to LIKE predicate pattern string.(Result:like = "~_a~%%")
String like = StringUtil.toLikeCondition("_a%");

// Get the byte length. (Result:i = 9)
int i = StringUtil.getByteLength("あああ", "UTF-8");
```

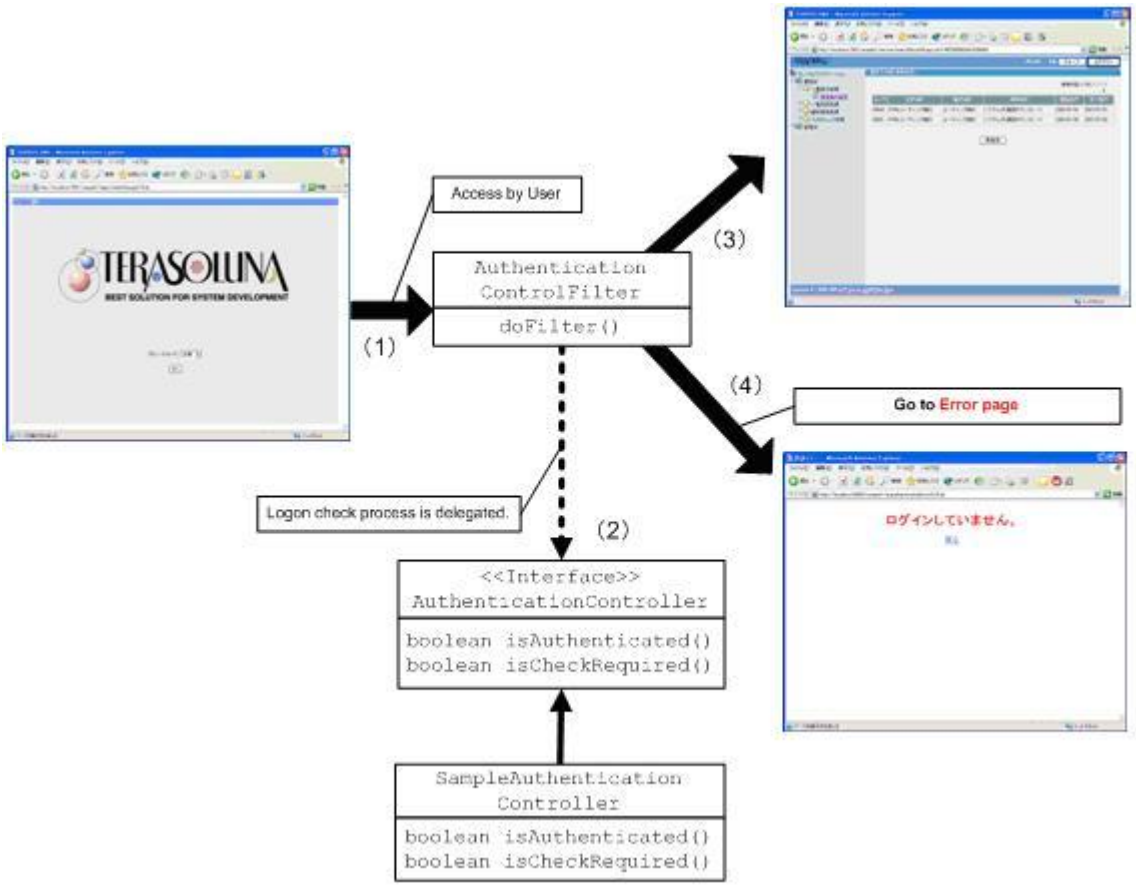
WA-01 Login Validation

■ Overview

◆ Functional Overview

- This function checks if the user has logged on. When the user has not logged on, access is blocked and the error page is displayed.

◆ Schematic Diagram



◆ Description

The following description assumes that AuthenticationControlFilter is set properly in the deployment descriptor (web.xml). For details, please refer to the Coding steps.

(1) Web application container delegates the user's request to AuthenticationControlFilter.

(2) AuthenticationControlFilter delegates the access privileges check to a class implementing the AuthenticationController interface as per the deployment descriptor (web.xml).
(3) When user is logged in, the next screen is displayed.
(4) When user is not logged in, the error page is displayed as per the settings in the deployment descriptor (web.xml).

■ Usage

◆ Coding steps

Login check is implemented using a servlet filter. The servlet filter obtains the authorization check implementation class from the DI container using a default id, (It is also possible to specify a custom bean id)
Both these examples are described below (i.e authorization check class having a default bean id, and authorization check class with a custom bean id)
Implementation classes in both the cases are similar, so only one of them is described
Further, the usage is same for “WA-02 Access Control”, “WA-03 Server Blockage Check” and “WA-04 Business Blockage Check”.

1. Configuration when a default value of id attribute is used

- Deployment Descriptor（web.xml）

```
<webapp>
  ……
  <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>/authenticatedError.jsp</location>
  </error-page>
  ……

</web-app>
```

The same name is used. Any name may be specified.

AuthenticationControlFilter is set.

Write request path for the filter.

UnauthenticatedException is set.

When not logged on, the display page is set.

- Bean Definition File

```
<beans>
……
  <bean id="authenticationController"
    class="jp.terasoluna.sample.SampleAuthenticationController"/>
……
</beans>
```

The default value of id attribute- authenticationController is used, for the AuthenticationControlFilter.

AuthenticationController imple-mentation class is set.

2. Setting method when value of id attribute is specified

- Deployment Descriptor（web.xml）

```
<web-app>
<filter>
……
  <filter-name>authenticationControlFilter</filter-name>
  <filter-class>
    jp.terasoluna.fw.web.thin.AuthenticationControlFilter
  </filter-class>
   <init-param>
     <param-name>controller</param-name>
     <param-value>
       sampleAuthenticationController
     </param-value>
   </init-param>
</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>/authenticatedError.jsp</location>
  </error-page>
  ……

</web-app>
```

AuthenticationControlFilter is set.

Controller is set as parameter name.

Any name can be specified, but it should be same at both places.

Write request path for the filter.

When not logged on, display page is set.

UnauthenticatedException is set

Any name can be specified, but it should be same at both places.

- bean definition file

```
<beans>
  ……
  <bean id="sampleAuthenticationController"
    class="jp.sample.SampleAuthenticationController"/>
  ……

</beans>
```

AuthenticationController implementation class is set.

※ Bean Definition of controller should be described in applicationConrext.xml.

3. Example of implementation class of AuthenticationController

```
package jp.terasoluna.sample;

import javax.servlet.ServletRequest;
import jp.terasoluna.fw.web.thin.AuthenticationController;

public class SampleAuthenticationController
       implements AuthenticationController {

  //Path where logged-on is not required.
  private String noCheckRequiredPath = "/sample/index.jsp";

  // Return true in case of access is allowed. Otherwise, return false.
  public boolean isAuthenticated(String pathInfo,
                                 ServletRequest req) {
    // Get session
    HttpSession session = ((HttpServletRequest) req).getSession();

    // Get UserValueObject from session
    SampleUserValueObject uvo = (SampleUserValueObject)
        session.getAttribute(UserValueObject.USER_VALUE_OBJECT_KEY);

    // For example, it is assumed that UserValueObject is stored in the ses-
sion only when user has looged in.
    // Thus return true when UserValueObject exists.
    if (uvo != null) {
      return true;

    return false;
  }

  // Return true, if log in check is required. Otherwise, return false.
  public boolean isCheckRequired(ServletRequest req) {
    if (noCheckRequiredPath.equals(RequestUtil.getPathInfo(req))) {
      return false;
    }
    return true;
  }
}
```

AuthenticationController interface is implemented.

Note that the argument pathInfo may have different values than RequestUtil.getPathInfo (req). Refer to "CD-04 Utility Functionality" for RequestUtil.

※ Note that to keep the above example simple, exception handling is not considered.

※ It is necessary to implement the AuthenticationController class in a thread safe manner. The above example does not contain any thread-safe specific code as it is not required.

◆ Additional Points

Implement AuthenticationController interface and to use Login Validation. Please refer to the example in Coding steps.

■ Classes

	Class Name	Outline
1	jp.terasoluna.fw.web.thin .AuthenticationControlFilter	Filter class that authenticates only login users.
2	jp.terasoluna.fw.web.thin. AuthenticationnController	Class called from AuthenticationControlFilter should implement this interface.
3	jp.terasoluna.fw.web.thin .UnauthenticatedException	Exception class to notify that the user has accessed without login

■ Related Functions

- ”CD-04 Utility Functions”

■ Example

- TERASOLUNA Server Framework for Java (Web based) Tutorial
  - 2.10 Access Restrictions
  - /webapps/WEB-INF/applicationContext.xml
  - /webapps/WEB-INF/web.xml
  - jp.terasoluna.thin.tutorial.web.SampleAuthController

- Sample covering all functions of TERASOLUNA Server Framework for Java (Web based)
  - ”UC04 Logged-on Check”
    - /webapps/authentication/*
    - /webapps/WEB-INF/authentication/*
    - jp.terasoluna.thin.functionsample.authentication.*

■ Remarks

- None

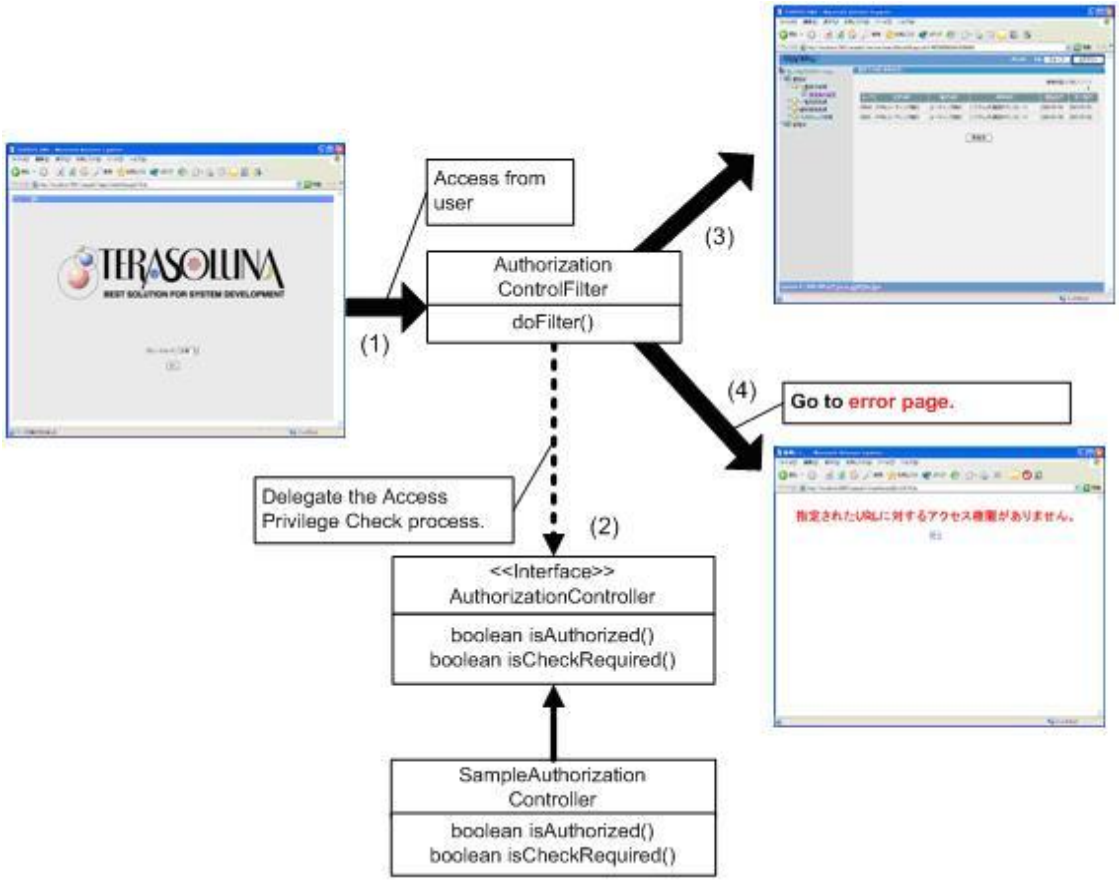
WA-02 Access Control

■ Overview

◆ Functional Overview

- Checks if the user has access privilege. If not, access is blocked and the error page is displayed.

◆ Schematic Diagram



◆ Description

The following description assumes that AuthorizationControlFilter is set correctly in the Deployment Descriptor (web.xml). For details please refer to Coding Points for the details.

(1) Web Application container receives request from user and delegates the process to AuthorizationControlFilter.
(2) AuthorizationControlFilter delegates access privilege check process to any AuthorizationController interface implementation class as per the settings in deployment descriptor (web.xml)
(3) When user has access privilege, the Next screen is displayed.
(4) When user does not have access privilege, an Error Page is displayed as per the settings in Deployment Descriptor (web.xml).

■ Usage

◆ Coding Points

Access Privilege Check Function is implemented using Servlet Filter. The filter gets the bean implementing the Access privilege check from the DI container. The id of the bean can be the default one, or it is also possible to specify a different id.
The below example describes both these cases (1) When the default value is used for id attribute, and (2) When the bean implementing access privilege check has an id other than the default value.

The implementation class for Access Priviledge Check is given only for one example, as it is the same in both the cases.

Further, Usage is same as in “WA-01 Login Check”, “WA-03 Server Blockage Check” and “WA-04 Business Blockage Check”.

1. Settings when default value of id attribute is used

- Deployment Descriptor (web.xml)

```
<web-app>
  ……
  <filter>
    <filter-name>authorizationControlFilter</filter-name>
    <filter-class>
      jp.terasoluna.fw.web.thin.AuthorizationControlFilter
    </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>/authorizedError.jsp</location>
  </error-page>
  ……

</web-app>
```

Any name may be specified, but it should be same in both places.

AuthorizationControlFilter is set.

Write the request path for starting the filter.

UnauthorizedException is set.

The page to be displayed when the user does not have access privilege

- Bean definition file

```
<beans>
……
  <bean id="authorizationController"
    class="jp.terasoluna.sample.SampleAuthorizationController"/>
……

</beans>
```

Default value of id attribute is “AuthorizationController” in case of AuthorizationControlFilter.

Implementation class of Authoriza-tionController.

2. Setting method when value of id attribute is different than the default value

- Deployment Descriptor (web.xml)

```
<web-app>
  ……
<filter>
  <filter-name>authorizationControlFilter</filter-name>
    <filter-class>
      jp.terasoluna.fw.web.thin.AuthorizationControlFilter
    </filter-class>
    <init-param>
      <param-name>controller</param-name>
      <param-value>
        sampleAuthorizationController
      </param-value>
    </init-param>
  </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>/authorizedError.jsp</location>
  </error-page>
  ……
</web-app>
```

AuthorizationControlFilter is set.

Controller is set as parameter name.

Any name may be specified, but it should be same in both places.

Write request path for starting the filter.

Specified with the same name. Any name may be specified.

UnauthorizedException is set.

The page to be displayed when there is no access permission

- bean definition file

```
<beans>
  ……
  <bean id="sampleAuthorizationController"
  class="jp.terasoluna.sample.SampleAuthorizationController"/>
  ……
</beans>
```

Implementation class of AuthorizationController is set.

※ bean definition of controller should always be described in applicationContext.xml.

3. Example of implementation class of AuthorizationController

```
package jp.terasoluna.sample;

import javax.servlet.ServletRequest;
import jp.terasoluna.fw.web.thin.AuthorizationController;

public class SampleAuthorizationController
       implements AuthorizationController {

  // Path where Access Privilege Check is requ
  private String checkRequiredPath = "/sample/admin.jsp";

  // Return true in case of access privilege. Otherwise, return false.
  public boolean isAuthorized(String pathInfo,
                              ServletRequest req) {
    // Get session
    HttpSession session = ((HttpServletRequest) req).getSession();

    // Get UserValueObject from session
    SampleUserValueObject uvo = (SampleUserValueObject)
      session.getAttribute(UserValueObject.USER_VALUE_OBJECT_KEY);

    // Does user has access privilege
    if (uvo.isAdmin()) {
      return true;
    }
    return false;
  }

  // Return true, if access privilege check is required. Otherwise, return
false.
  public boolean isCheckRequired(ServletRequest req) {
    if (checkRequiredPath.equals(RequestUtil.getPathInfo(req))) {
      return true;
    }
    return false;
  }
}
```

AuthorizationController interface is implemented.

Note that argument pathInfo may not have the same value as RequestUtil.getPathInfo(req). Please refer to "CD-04 utility function" for RequestUtil.

When permission information of user can be acquired by isAdmin method of SampleUserValueObject.

※ Note that to keep the example brief, exception handling is not mentioned.

※ It is necessary to implement AuthenticationController implementation class in a thread safe manner. However, this example does not include any thread-safe specific code

◆ Extension Points

Use Access Control by implementing the AuthorizationController interface. Please refer to the example in Coding Points.

■ Classes

	Class name	Overview
1	jp.terasoluna.fw.web.thin.AuthorizationControlFilter	Filter class authenticates only the users having access privilege.
2	jp.terasoluna.fw.web.thin.AuthorizationController	Class to be called from AuthenticationControlFilter should implement this interface.
3	jp.terasoluna.fw.web.thin.UnauthorizedException	Exception class to notify that the user has accessed without access privilege.

■ Related Function(s)

- “CD-04 Utility Functions”

■ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web based)
  - “UC05 Access Control”
    - /webapps/authorization/*
    - /webapps/WEB-INF/authorization/*
    - jp.terasoluna.thin.functionsample.authorization.*

- TERASOLUNA Server Framework for Java (Web based) tutorial
  - “2.10 Access Control”

■ Remarks

- None

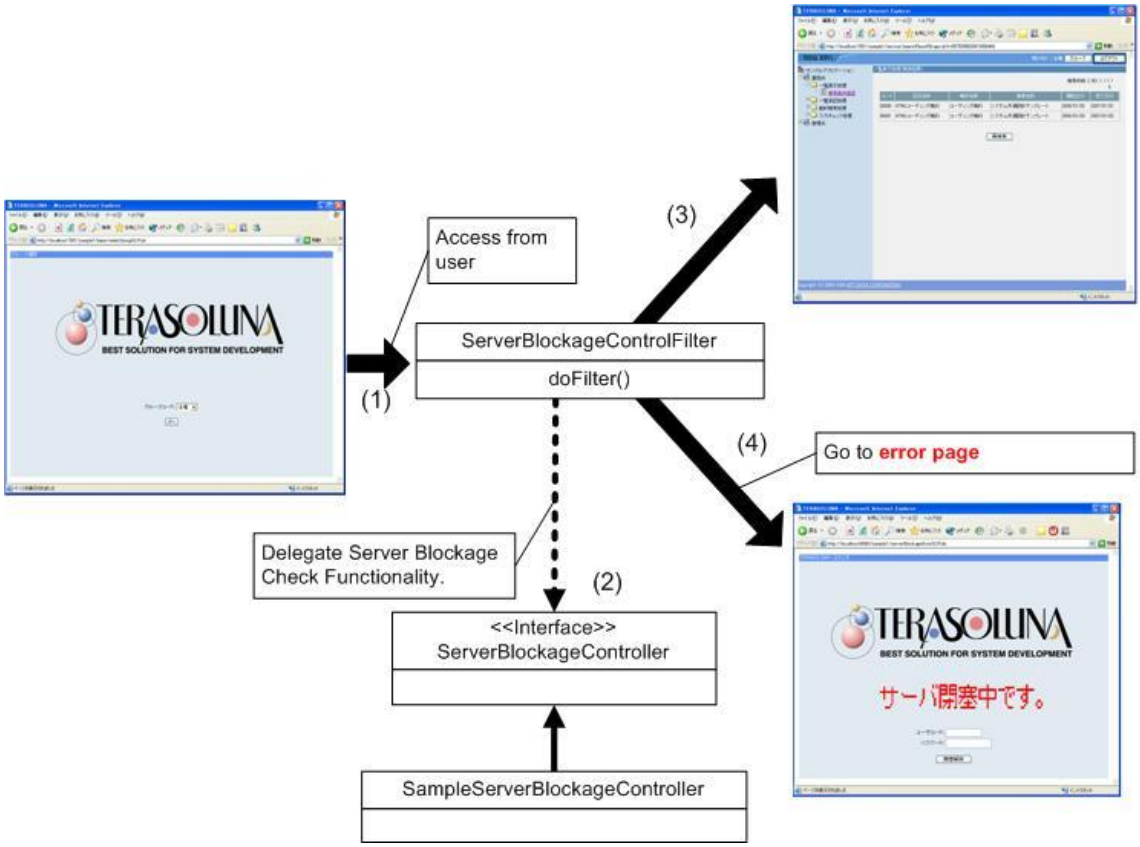
WA-03 Server Blockage Check

■ Overview

◆ Functional Overview

- For each access by the user, this function checks whether the server is in blocked state. Access is restricted and the error page is displayed if the server is in blocked state.

◆ Schematic Diagram



◆ Description

The following description assumes that the ServerBlockageControlFilter is properly set in the Deployment Descriptor (web.xml). Please refer to Coding steps for more details.

(1) Web Application container receives the request from user and delegates the process to ServerBlockageControlFilter.

(2) The ServerBlockage ControlFilter delegates the Server Blockage Check process to the class implementing the ServerBlockageController interface as specified in deployment descriptor (web.xml).
(3) The next screen is displayed when the server is not blocked.
(4) The error page is displayed as per settings in deployment descriptor (web.xml), if the server is blocked.

■ Usage

◆ Coding Points

The function to check if the server is blocked is implemented using a servlet filter. The bean implementing the ServerBlockage check is retrieved from the DI container. A default value is used for the bean's id, but it is also possible to use a value other than the default one.

Both these cases are described below (1) When the default value is used for the id attribute and (2) When a value other than the default value is used to get the bean implementing the ServerBlockage check.
The implementation of Server Blockage Check is described for only one example, as it is the same in both the cases.

Further, the usage is same for "WA-02 Access Permission Check", "WA-03 Server Blockage Check" and "WA-04 Business Blockage Check".

1. Setting method when default value of id attribute is used

- Deployment Descriptor (web.xml)

```
<web-app>
  ……
  <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>/serverBlockageError.jsp</location>
  </error-page>
  ……
</web-app>
```

Any name may be specified, but it should be same in both places.

ServerBlockageControlFilter is set

Write the request path for starting the filter

ServerBlockageException is set

The page to be displayed when the server is blocked

- Bean definition file

```
<beans>
  ……
  <bean id="serverBlockageController"
    class="jp.terasoluna.sample.SampleServerBlockageController"/>
  ……
</beans>
```

Default value of id attributes. serverBlockageControlFilter is set.

Implementation class of ServerBlockageController is set.

2. Setting method when value of id attribute is specified

- Deployment Descriptor (web.xml)

```
<web-app>
  ……
  <filter>
    <filter-name>serverBlockageControlFilter</filter-name>
    <filter-class>
      jp.terasoluna.fw.web.thin.ServerBlockageControlFilter
    </filter-class>
    <init-param>
      <param-name>controller</param-name>
      <param-value>
        sampleServerBlockageController
      </param-value>
    </init-param>
  </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>/serverBlockageError.jsp</location>
  </error-page>
  ……
</web-app>
```

ServerBlockageControlFilter is set

Controller is set as parameter name

Any name may be specified, but it should be same in both places

Write the request path for starting the filter

ServerBlockageException is set

Any name may be specified, but it should be same in both places

The page to be displayed when server is blocked

- Bean definition file

```
<beans>
  ……
  <bean id="sampleServerBlockageController"
    class="jp.terasoluna.sample.SampleServerBlockageController"/>
  ……

</beans>
```

Implementation class of ServerBlockageController is set

3. Example of implementation class of ServerBlockageController

```
package jp.terasoluna.sample;

import java.util.Date;
import javax.servlet.ServletRequest;
import jp.terasoluna.fw.web.thin.ServerBlockageController;

public class SampleServerBlockageController
       implements ServerBlockageController {

  // Open state
  public static final int OPEN = 0;
  // Pre blockage state
  public static final int PRE_BLOCKADED = 1;
  // Blockage state
  public static final int BLOCKADED = 2;

  // Paths allowed even when the server is blocked.
  private String[] alwaysOpenPaths
    = {"/admin/index.jsp", "/admin/action.do"};

  // variable indicating if the server is blocked.
  private int state = OPEN;

  // Time after which server should be blocked.
  private Date blockadingDate = null;
```

Implementation class of Server-BlockageController

※Continued

```
  // Return true if server is in blocked state, otherwise, return false
  public boolean isBlockaded(String pathInfo) {
    // Do not perform check for paths, for which the server blockage check
    // should be skipped.
    for (int i = 0; I < alwaysOpenPaths.length; ++i) {
      if (alwaysOpenPaths[i].equals(pathInfo)) {
        return false;
      }
    }
    // Check the server state
    if (state == BLOCKADED) {
      return true;
    }
    // Server is not blocked, if the blocking time is not specified.
  if (blockadingDate == null) {
      return false;
    }
    // Block if the blocking time has elapsed
    synchronized (this) {
      if (blockadingDate != null) {
        Date now = new Date();
        // Transit to blockage state if time to transit to blockage state has
already elapsed
        if (blockadingDate.before(now)) {
          state = BLOCKADED;
        }
      }
    }
    if (state == BLOCKADED) {
      return true;
    }
    return false;
}
```

Transfer in the format of jp.co.nttdata.terasoluna.fw.web.RequestUtil.getPathInfo(req).

In this example, since the server state is controlled by this instance, it is locked to avoid inconsistency.

※Continued

```
  public boolean isBlockaded() {
    // Since path information is not used, delegate it to the method which is
overloaded
    return isBlockaded(null);
  }

  // Determine Pre-blockage state.
  public boolean isPreBlockaded() {
    if (state == BLOCKADED || state == PRE_BLOCKADED) {
      return true;
    }
    return false;
  }

  // Block the server.
  public void blockade() {
    synchronized (this) {
      state = BLOCKADED;
    }
  }

  // Open server
  public void open() {
    synchronized (this) {
      state = OPEN;
      blockadingDate = null;
    }
  }
  // Set the server to Pre-blockage state
  public void preBlockade() {
    synchronized (this) {
      state = PRE_BLOCKADED;
    }
  }

  // Set server to Pre-blockage state and block for the specified time.
  public void preBlockade(Date time) {
    synchronized (this) {
      state = PRE_BLOCKADED;
      blockadingDate = time;
    }
  }
}
```

The methods hereonwards are not called from the filter, but are necessary in Server Blockage. Generally, these are called from management screen and Block and Open is executed.

In this example, since the server state is controlled by this instance, it is locked to avoid inconsistency.

※ Note that to keep the example brief, exception handling is not considered.

Extension Points

- Use Server Blockage Check function by implementing ServerBlockageController interface. Please refer to the example of Coding steps.

■ Classes

	Class Name	Overview
1	jp.terasoluna.fw.web.thin.ServerBlockageControlFilter	Filter class that prevents access when the server is blocked.
2	jp.terasoluna.fw.web.thin.ServerBlockageController	Class to be called from ServerBlockageControlFilter should implement this interface.
3	jp.terasoluna.fw.web.thin.ServerBlockageException	Exception class to indicate that the server is blocked during an access.

■ Related Function(s)

- None

■ Example

- Sample covering all functions on the TERASOLUNA Server Framework for Java (Web based)
  - "UC06 Server Blockage Check"
    - /webapps/serverblockage/*
    - /webapps/WEB-INF/serverblockage/*
    - jp.terasoluna.thin.functionsample.serverblockage.*

■ Remarks

- None

WA-04 Business Blockage Check

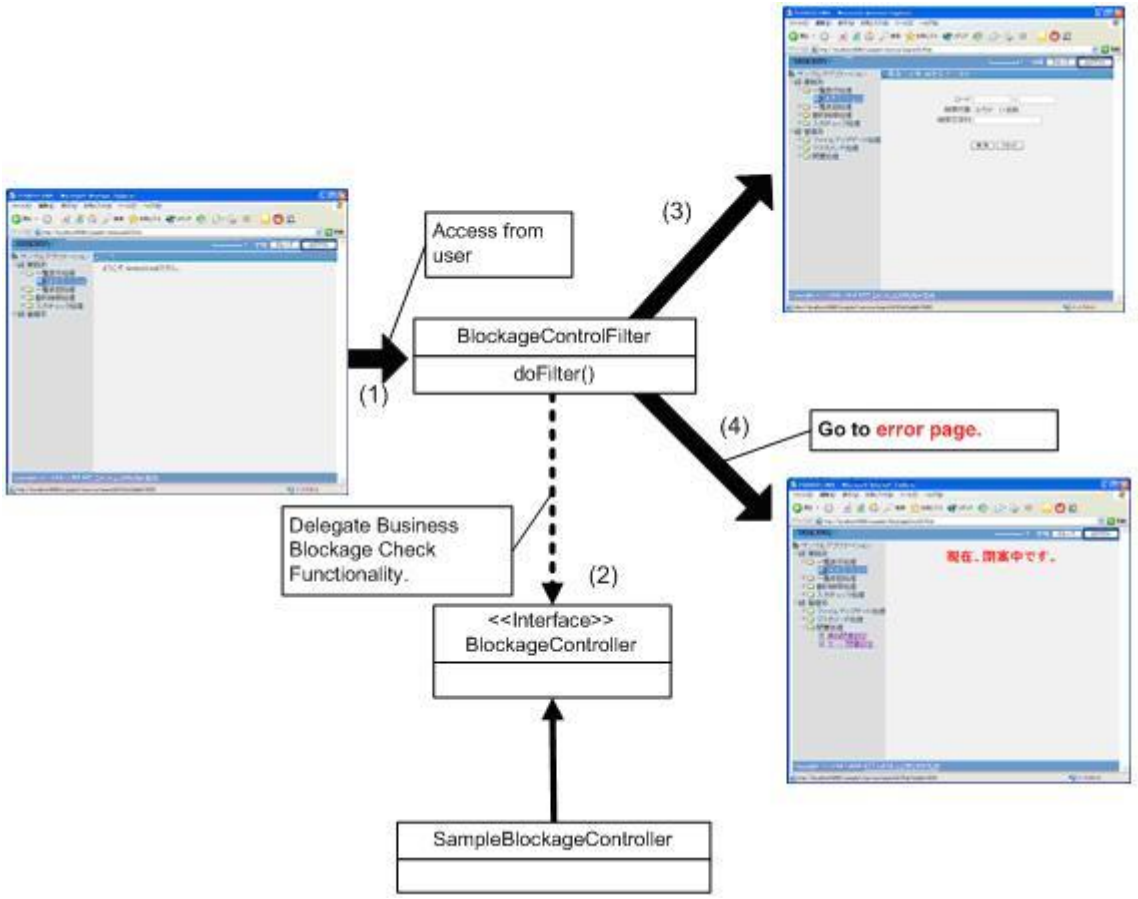
■ Overview

◆ Functional overview

- For each access by the user, this function checks if the server is in business blocked state. Access is restricted and an error page is displayed, if the server is in business blocked state.

※ While using this function, it should be set so that it is possible to identify the business from the request path.

◆ Schematic diagram



◆ Description

The following description assumes that BlockageControlFilter is properly set in the deployment descriptor (web.xml). Pleas refer to the Coding points for more details.

(1) Web Application container receives the request from user and delegates the process to BlockageControlFilter.
(2) The BlockageControlFilter delegates the Business Blockage Check process to any class implementing the BlockageController interface as per the setting in the deployment descriptor (web.xml).
(3) The next screen is displayed when the business is not blocked,
(4) The error page is displayed as per settings in deployment descriptor (web.xml), if the business is blocked.

■ Usage

◆ Coding points

The business blockage check function is implemented using servlet filter. Get the bean implementing the business blockage check from the DI container. A default value is used as the bean's id, but a value other than the default value can also be used.
Both cases are described below; (1) A default value is used for id attribute and (2) A value other than default is specified for the id attribute defining the bean which implements the business blockage check

The implemention of business blockage check is described for only one example, as it is the same for both the cases.

Further, usage is same for "WA-01 Login Check", "WA-02 Access control" and "WA-03 Server Blockage Check".

1. Setting method when default value of id attribute is used.

- Deployment Descriptor(web.xml)

```
<web-app>
  ……
  <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>/blockageError.jsp</location>
  </error-page>
  ……
</web-app>
```

Any name may be specified, but it should be same at both places

BlockageControlFilter is set

Write request path for starting the filter

BlockageException is set

Page to be displayed if the business is blocked

- Bean definition file

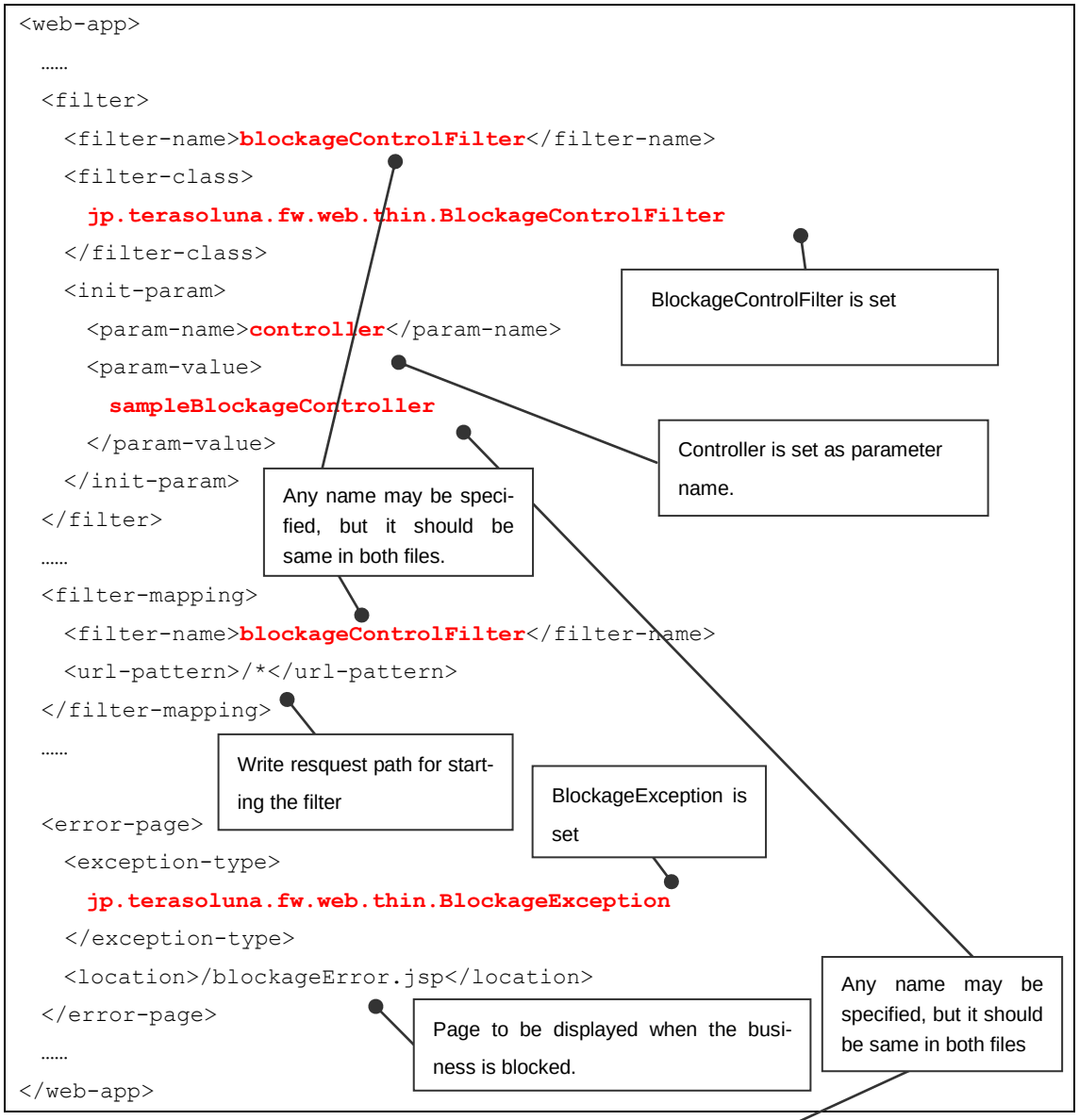
```
<beans>
……
  <bean id="blockageController"
class="jp.terasoluna.sample.SampleBlockageController"/>
  ……
</beans>
```

Default value 'blockageControlFilter' of id attributes is set.

Implementation class of BlockageController is set

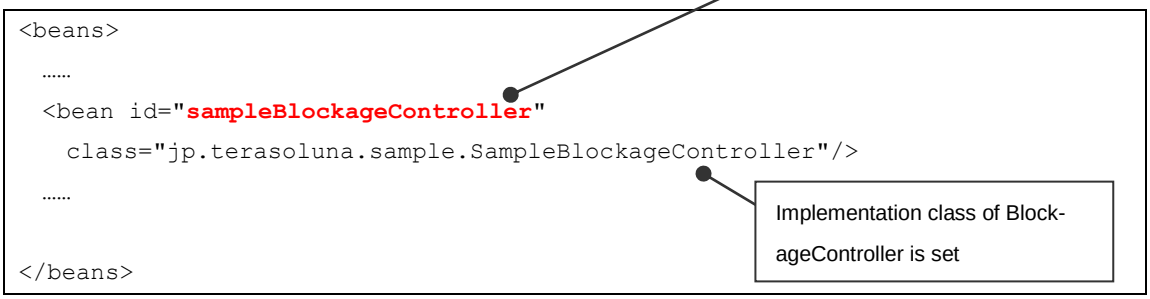
2. Setting method when value of id attribute is specified

- Deployment Descriptor (web.xml)




```
<web-app>
  ……
  <filter>
    <filter-name>blockageControlFilter</filter-name>
    <filter-class>
      jp.terasoluna.fw.web.thin.BlockageControlFilter
    </filter-class>
    <init-param>
      <param-name>controller</param-name>
      <param-value>
        sampleBlockageController
      </param-value>
    </init-param>
  </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>/blockageError.jsp</location>
  </error-page>
  ……
</web-app>
```

- bean definition file




```
<beans>
  ……
  <bean id="sampleBlockageController"
    class="jp.terasoluna.sample.SampleBlockageController"/>
  ……

</beans>
```

3. Example of Implementation Class of BlockageController

```
package jp.terasoluna.sample;

import javax.servlet.ServletRequest;
import jp.terasoluna.fw.web.thin.BlockageController;

public class SampleBlockageController
       implements BlockageController {

  // Path for which check is not performed
  private String noCheckPath = "/sample/index.jsp";

  // Path with business blockage
  private HashSet<String> blockadedPaths = new HashSet<String>();


  // Return true in case of business blockage state, otherwise return false
  public boolean is Blockaded(String path,
                              ServletRequest req) {
    synchronized (blockadedPaths) {
    if (blockadedPaths.contains(path)) {
        return true;
      }
      return false;
    }
  }

  // Determine whether the check for business blockage is necessary or not,
  // If it is necessary, return true, otherwise return false
  public boolean isCheckRequired(ServletRequest request) {
  //Determine whether it is a request of path where check is not necessary
    if (noCheckPath.equals(RequestUtil.getPathInfo(req))) {
      return false;
    }
    return true;
  }
```

BlockageController interface is implemented

Note that it is not always same as RequestUtil.getPathInfo(req)

Use 'req' to perform user-wise check.

Implement in a thread safe manner. blockadedPaths is synchronized so that it is not modified by other threads.

※Continued

The methods here onwards are not called from the filter, but are necessary in Business Blockage

```
 public boolean isBlockaded(String path) {
  // Since request information is not used, delegate it to overloaded method
   isBockaded(path, null);
 }

 public void blockade(String path) {
// Since request information is not used, delegate it to overloaded method
   blockade(path, null);
 }

 // Block the path that calls specified business
 public void blockade(String path,
                      ServletRequest req) {
   synchronized (blockadedPaths) {
     blockadedPaths.add(path);
   }
 }

 public void open(String path) {
 // Since request information is not used, delegate it to overloaded method
   open(path, null);
 }

 // Release the path that calls the specified business
 public void open(String path,
                  ServletRequest req) {
   synchronized (blockadedPaths) {
     blockadedPaths.remove(path);
   }
 }
}
```

It is assumed that business blockage filter will block path-wise. So, pass the path to be blocked.

Use 'req' to perform user-wise check.

Implement in a thread safe manner. Blockaded-Paths is synchronized so that it is not modified by other threads.

Implement in a thread safe manner. Blockaded-Paths is synchronized so that it is not modified by other threads.

※ Note that exception occurrence is not considered in this implementation example, since it is in brief.

◆ Extension Points

Use Business Blockage Check function(s) by implementing the BlockageController interface. Please refer to the example of Coding steps.

■ Classes

	Class name	Overview
1	jp.terasoluna.fw.web.thin.BlockageControlFilter	Filter class that prevents access to business which is blocked,
2	jp.terasoluna.fw.web.thin.BlockageController	Class to be called from BlockageControlFilter should implement this interface.
3	jp.terasoluna.fw.web.thin.BlockageException	Exception class to indicate that the business is blocked when user has accessed

■ Related function(s)

- None

■ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web based)
  - "UC07 Business Blockage check"
    - /webapps/blockage/*
    - /webapps/WEB-INF/blockage/*
    - jp.terasoluna.thin.functionsample.blockage.*

■ Remarks

- None

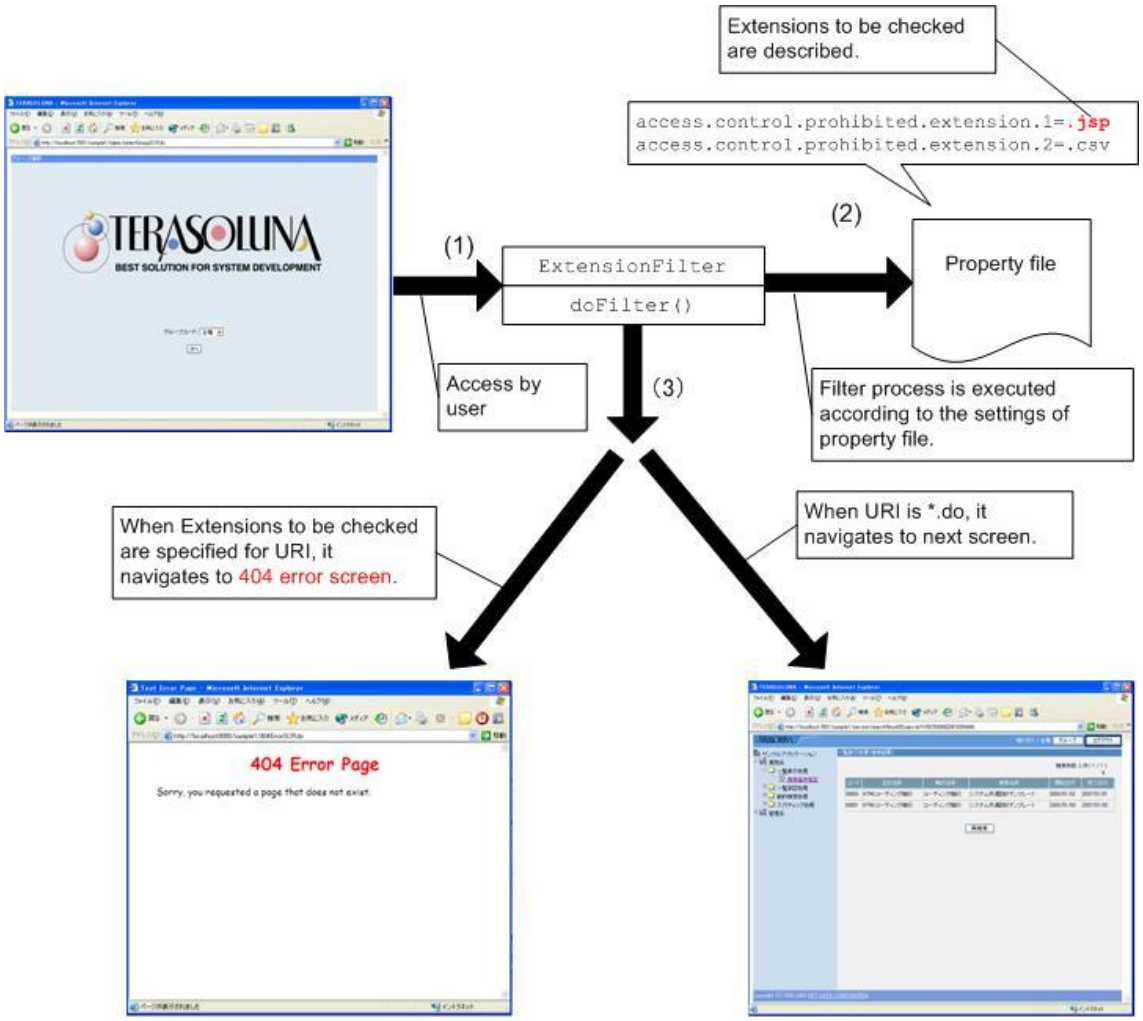
WA-05 Denial of direct access to extensions

■ Overview

◆ Functional Overview

It prevents browsers from directly accessing contents (files) with specified extensions.

◆ Schematic Diagram



◆ Description

The following description assumes that ExtensionFilter is set properly in the deployment descriptor (web.xml). Please refer to the Coding for details of setting method.

(1) Web Application container receives request from user and delegates the process to ExtensionFilter.
(2) Determine whether the extension of the resource requested is restricted as per the settings in property files.
(3) When the extension is prohibited, 404 error page is displayed.

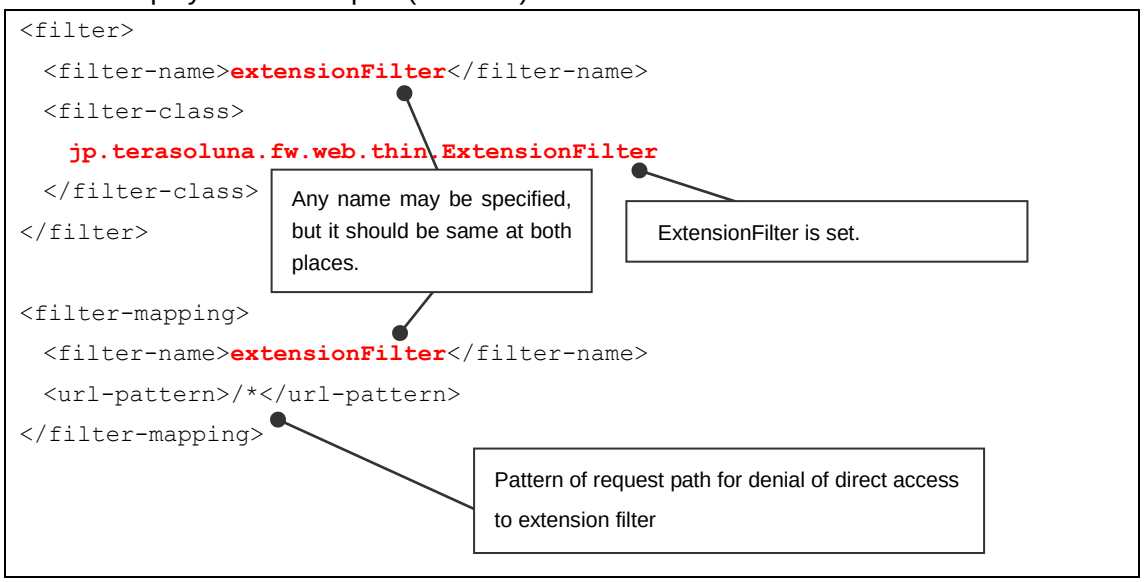
■ Usage

◆ Coding Points

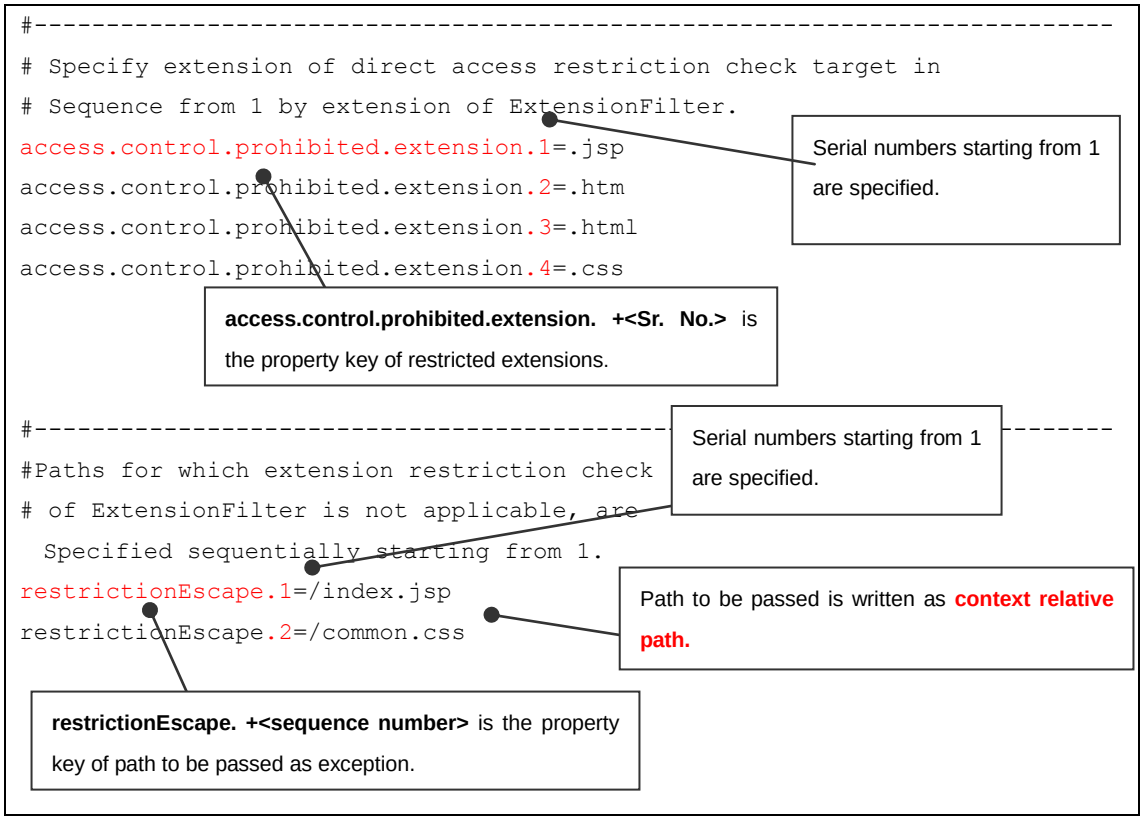
To use this filter, configure it as a filter in deployment descriptor (web.xml), and write a list of extensions prohibited for direct access in the system property file (system.properties). Similarly, in the system property file (system.properties), prepare the exception list i.e the list of files (with paths) for which direct access should be allowed even when the extensions are prohibited.
Please refer to the following example

Further, please refer to “CD-01 Utility Functions” for handling of system property file (system.properties).

- Deployment Descriptor (web.xml)



- System setting property file (system.properties)



※ For <sequence number>, start numbering sequentially from “1”.
When some number is missing, subsequent numbers become invalid.

◆ Extension Points

None

■ Classes

	Class Name	Overview
1	jp.terasoluna.fw.web.thin.ExtensionFilter	Filter class that denies direct access to specified extension.

■ Related Function(s)

- “CD-01 Utility Functions”

■ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web based)
  “UC08 Denial of direct access to extensions”
  - /webapps/extension/*
  - /webapps/WEB-INF/extension/*
  - jp.terasoluna.thin.functionsample.extension.*

■ Remarks

- None

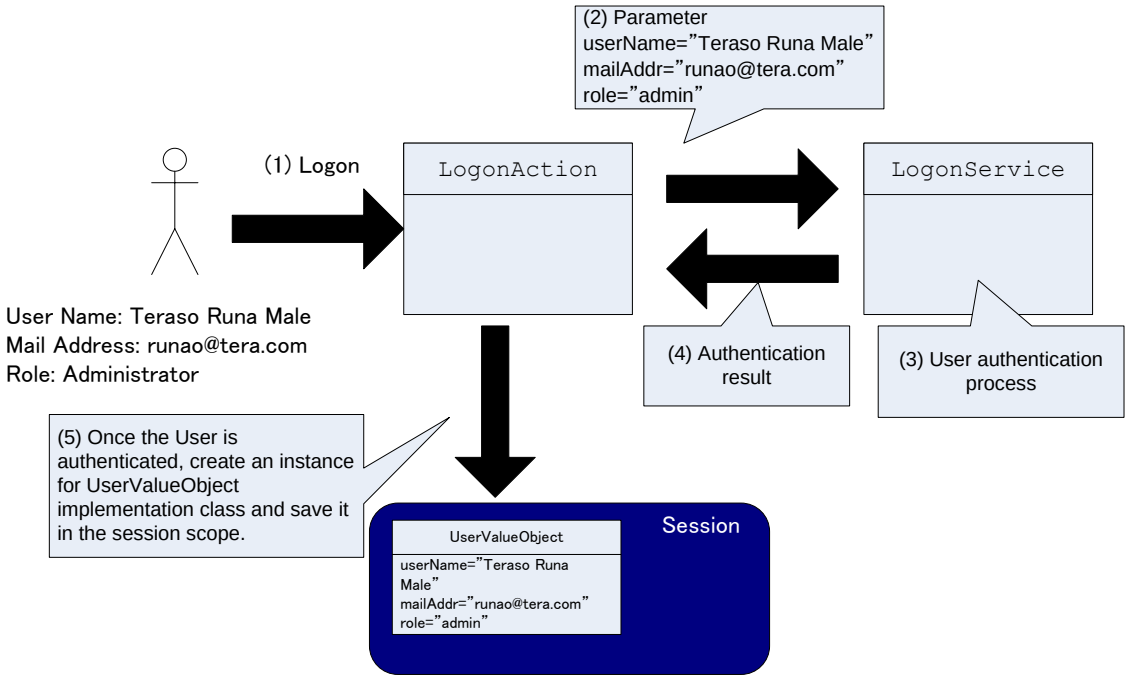
WB-01 User Information Retention

■ Overview

◆ Functional Overview

- This function retains the information of Logged-on user and provides a value object to store it in the session.
- This function passes Logged-on user information to Business Logic in such a way that user information can be used in the Business Logic.
- As User Value Object (UVO) is system-specific, it is designed so that it can be easily interchanged.

◆ Schematic Diagram



◆ Description

(1) An unauthenticated user accesses from the Login screen.
(2) The Action class (optional), for login authentication, delegates the authentication process to the business logic (optional) passing it the required parameters..
(3) Business logic executes the authentication check as per application specifications.
(4) Business Logic returns authentication process result to Action class.
(5) Action class creates an instance of the UserValueObject class, sets the required

information and saves it in the session, if the user is authenticated.

■ Usage

◆ Coding Points

The User Information Retention Class is created by inheriting from the abstract class provided in the TERASOLUNA Server Framework for Java (Web Version). Instance of User Information Retention Class can be created from the implemention class name specified in property file using the utility method,

- Property file

```
user.value.object=jp.terasoluna.sample.xxxx.SampleUVO
```

UserValueObject implementation class name is set considering user.value.object as key.

- UserValueObject Implementation Class

```
public class SampleUVO extends UserValueObject {
    /**User Name*/
    private String userName = null;
    /**User Role*/
    private String userRole = null;
    /**Set User Name.*/
    public void setUserName(String userName) {
        this.userName = userName;
    }
    /**Get User Name.*/
    public String getUserName() {
        return this.userName
    }
    ......
}
```

Required attributes and access method are defined.

- Action Class that executes User authentication

```
public class AuthenticationAction
       extends AbstractBLogicAction<AuthenticationParams> {
   public BLogicResult doExecuteBLogic(AuthenticationParams params)
       throws Exception {
       //Call Business Logic
       if (authenticated) {
           SampleUVO uvo = (SampleUVO) UserValueObject.createUserValueObject();
           BLogicResult result = new BLogicResult();
           result.setResultObject(uvo);
           ……
       }
   }
}
```

- Please refer to ”WH-01 Business Logic Execution”, ”WH-02 Business Logic Input/Output” for Business Logic implementation and reflecting Business Logic Result in session.

◆ Extension Points

None.

■ Classes

	Class Name	Overview
1	jp.terasoluna.fw.web.UserValueObject	Abstract class inherited by User Information Retention Class.Create implemention class instance from class name specified in property file.

■ Related Function(s)

- “WA-01 Login Check”
- “WA-02 Access Control”
- “WH-01 Business Logic Execution”
- “WH-02 Business Logic Input/Output Function(s)”

■ Example

- Sample covering all functions in the TERASOLUNA Server Framework for Java (Web based)
  - “UC09 User Information Retention”

- /webapps/uvo/*
- /webapps/WEB-INF/uvo/*
- jp.terasoluna.thin.functionsample.common.FunctionUVO.java

- TERASOLUNA Server Framework for Java (Web based) tutorial
  - Login screen

■ Remarks

None.

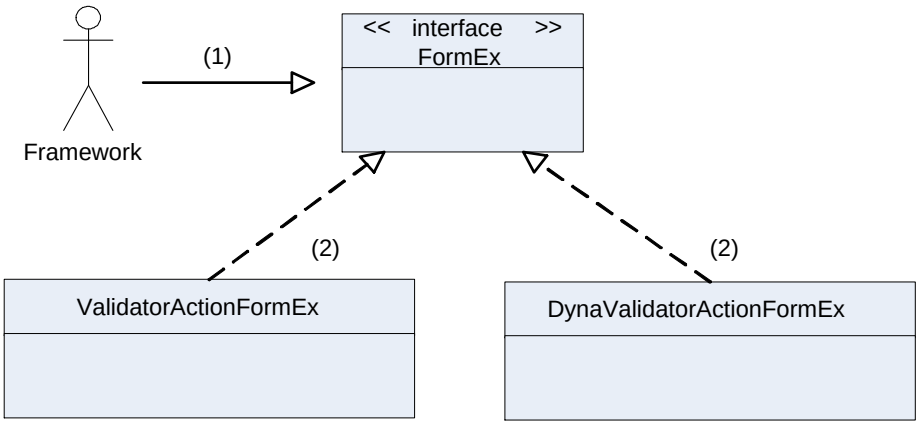
WB-02 ActionForm Extensions

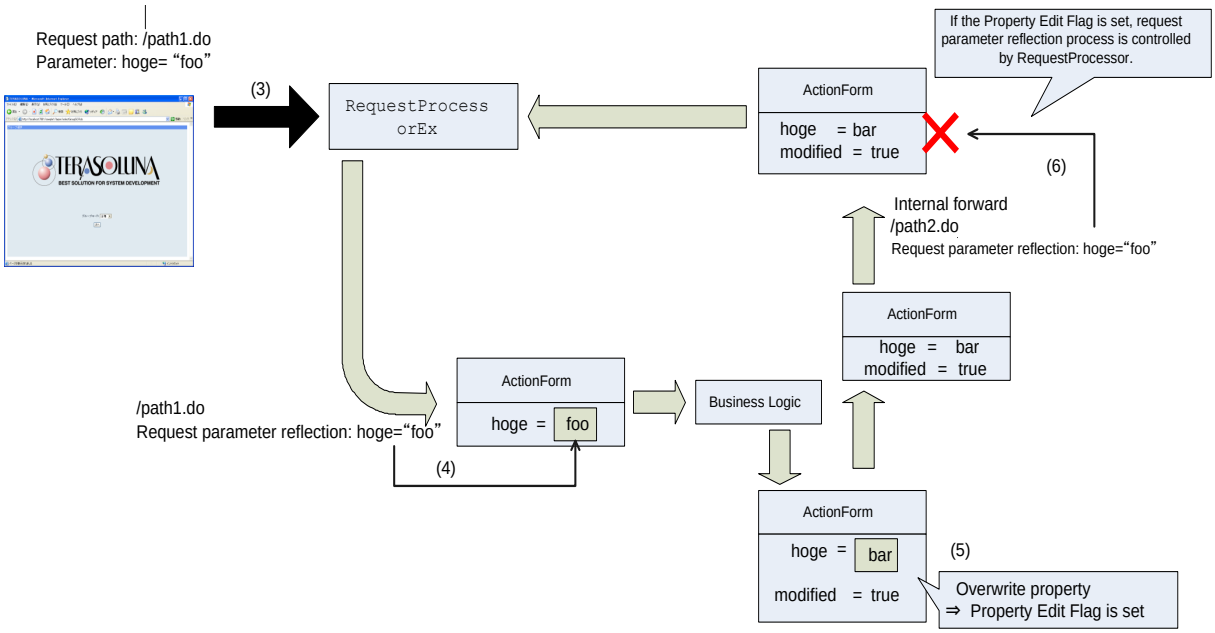
■ Overview

◆ Functional Overview

- The framework provides a common interface and abstract base class so that framework can commonly handle dynamic and static Action forms
- Provides methods for typical handling of properties.
- Provides a function to stop the overwriting request parameters at the time of internal forwarding (used when properties of ActionForm change after executing BusinessLogic)
- Provides a function that ensures that ActionForm with "_" at the beginning of formname are unique in a session.
  Please refer to "WB-03 ActionForm Switch Functionality"
- Provides function to auto initialize properties from configuration file.
  Please refer to "WB-04 Form Properties Reset Functionality"
- Provides typical input check rules by extending rules of commons-validator.
  Please refer to "WF-01 Extended Input Check"

◆ Schematic Diagram





◆ Description

ActionForm function(s) provided by the Framework.

(1) By using FormEx, (a common interface to ActionForm), framework can execute ActionForm operations in a common manner irrespective of whether the implemented class is a dynamic ActionForm or a static ActionForm.

(2) FormEx implementation classes provided are DynaValidatorActionFormEx, (dynamic ActionForm) and ValidatorActionFormEx, (static ActionForm).

Request parameter overwrite function(s)

※ This function assumes that RequestProcessorEx settings are configured in Struts configuration file.

(3) User sends request from screen to server.

(4) The request parameters are copied to the relevant ActionForm by processPopulate method of RequestProcessor.

(5) A property edit flag is set, when a property in ActionForm set in (4) is modified by business logic.

(6) When internal forwarding process is executed after Business logic termination, processPopulate of Requestprocessor is called again. It will normally overwrite the property changed in (5) from the request parameter. However, since the property edit flag is set, the processPopulate method is not executed.

■ Usage

◆ Coding Points

In case of DynaValidatorActionFormEx, Specify form name, properties in Struts configuration file.

- Struts configuration file (struts-config.xml)

```
<form-bean name="logonForm"
  type="jp.terasoluna.fw.web.struts.form.DynaValidatorActionFormEx" >
  <form-property name="userId" type="java.lang.String"/>
</form-bean>
```

Setting method is similar to DynaActionForm

While using ValidatorActionFormEx, create inherited implementation class.

- Example of ValidatorActionFormEx implementation class

```
public class SampleForm extends ValidatorActionFormEx {
  private String userName = null;
  public void setUserName(String userName) {
    this.userName = userName;
  }
  public String getUserName() {
    return this.userName;
  }
}
```

DynaValidatorActionFormEx and ValidatorActionFormEx are implemented so that exception is not thrown when accessing array/list with index value outside the range.

- Array/List Operation Method

```
 DynaValidatorActionFormEx formEx = (FormEx) form;
 String[ ] stringArray = new String[] {
   "a", "b", "c"
 };
 formEx.set("hoge", stringArray);  // Set array of 3 elements in hoge
property
  formEx.get("hoge", 5);  // Normally an exception is thrown when ele-
                           ments are out of array range,
                          //null is returned
 formEx.set("hoge", 3, "c");  // Normally an exception is thrown when
elements are out of array range,
                         // number of elements of array is increased
                         // to 4 and "c" is inserted as 4th element.
```

◆ Extension Points

None

■ Classes

	Class Name	Overview
1	jp.terasoluna.fw.web.struts.form.FormEx	Common interface of ActionForm accessed by framework.
2	jp.terasoluna.fw.web.struts.form.DynaValidatorActionFormEx	Dynamic FormEx implementation class. Class implementation is not required.
3	jp.terasoluna.fw.web.struts.form.ValidatorActionFormEx	Static FormEx implemention class. ActionForm is implemented by inheriting this class.
4	jp.terasoluna.fw.web.struts.form.ActionFormUtil	Class with utility methods for FormEx operations.

■ Related Function(s)

- “WB-03 ActionForm Switching Function(s)”
- “WB-04 Form Properties Reset Function(s)”
- “WF-01 Input Validation extensions”

■ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web based)
  - “UC10 ActionForm Extension”
    - /webapps/formex/*
    - /webapps/WEB-INF/formex/*
    - jp.terasoluna.thin.functionsample.formex.*

- TERASOLUNA Server Framework for Java (Web based) tutorial
  - 2.4 Logon
  - 2.5 List Display
  - /webapps/WEB-INF/struts-config.xml

■ Remarks

None

WB-03 ActionForm Switching Functionality

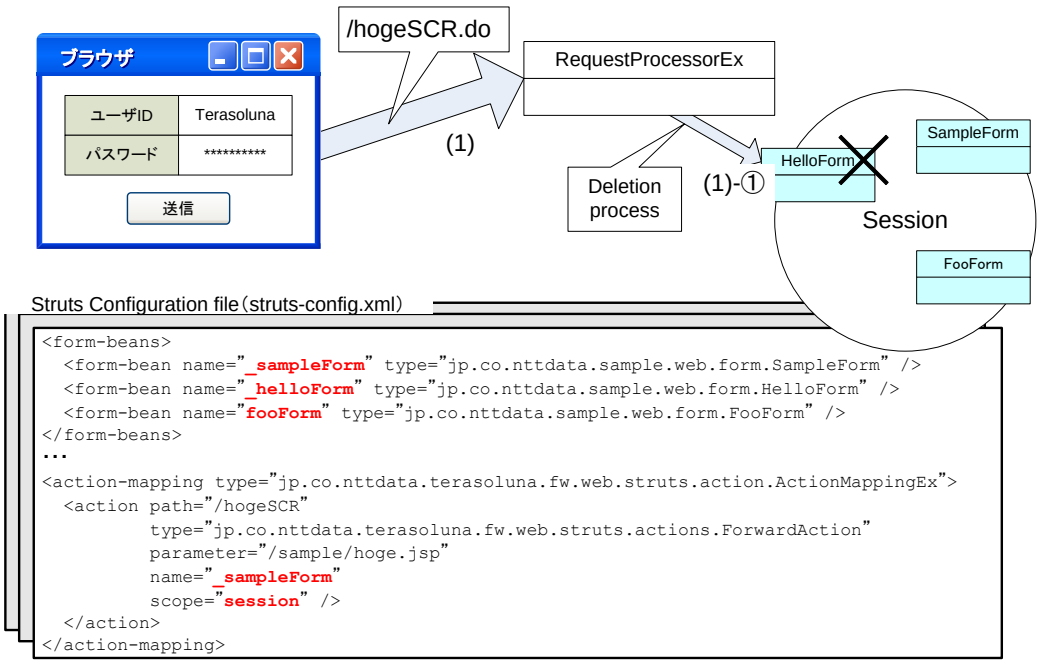
■ Overview

◆ Functional Overview

- Using RequestProcessorEx provided by TERASOLUNA Server Framework for Java (Web version) and by attaching "_" at the beginning of ActionForm name, single ActionForm with "_" can be maintained in the session. Using this function, an ActionForm can be defined for each use case, and the same ActionForm can be shared between similar use cases. It is possible to configure, so that the ActionForm is automatically deleted when control is forwarded to a different use case.
- This function cannot be used in case a single user operates multiple use cases at the same time in the application.

◆ Schematic Diagram

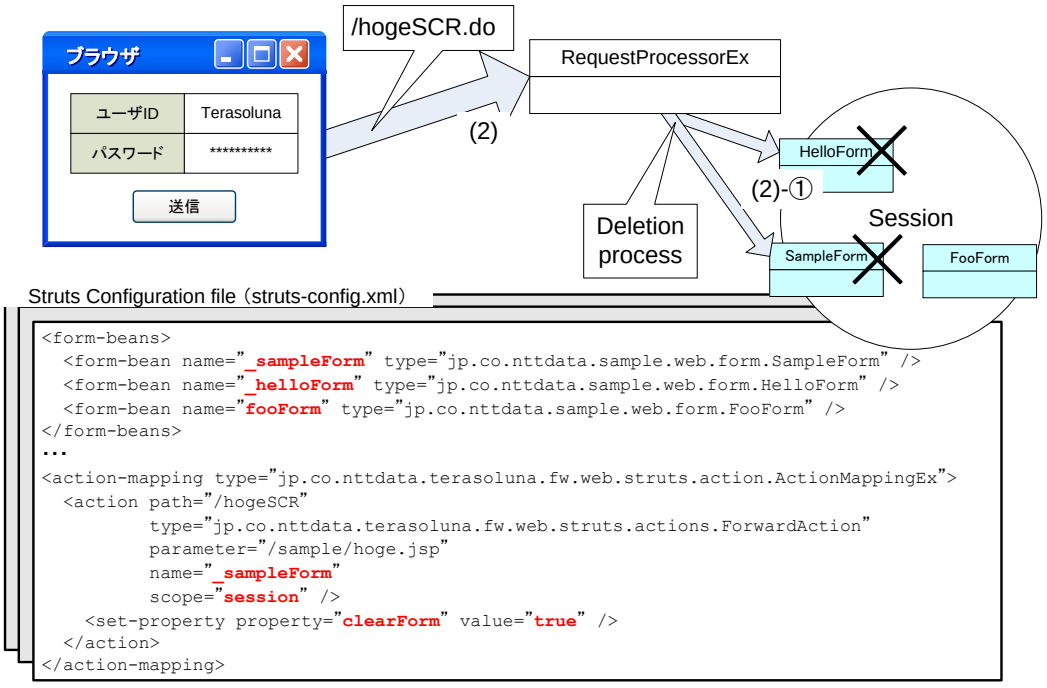
If the path associated with the ActionForm having the logical name starting with “_”, is accessed...



Struts Configuration file（struts-config.xml）

```
<form-beans>
  <form-bean name="_sampleForm" type="jp.co.nttdata.sample.web.form.SampleForm" />
  <form-bean name="_helloForm" type="jp.co.nttdata.sample.web.form.HelloForm" />
  <form-bean name="fooForm" type="jp.co.nttdata.sample.web.form.FooForm" />
</form-beans>
...
<action-mapping type="jp.co.nttdata.terasoluna.fw.web.struts.action.ActionMappingEx">
  <action path="/hogeSCR"
          type="jp.co.nttdata.terasoluna.fw.web.struts.actions.ForwardAction"
          parameter="/sample/hoge.jsp"
          name="_sampleForm"
          scope="session" />
  </action>
</action-mapping>
```

If the path associated with ActionForm having the logical name starting with “_” and having “clearForm property” set with “true”, is accessed...



Struts Configuration file（struts-config.xml）

```
<form-beans>
  <form-bean name="_sampleForm" type="jp.co.nttdata.sample.web.form.SampleForm" />
  <form-bean name="_helloForm" type="jp.co.nttdata.sample.web.form.HelloForm" />
  <form-bean name="fooForm" type="jp.co.nttdata.sample.web.form.FooForm" />
</form-beans>
...
<action-mapping type="jp.co.nttdata.terasoluna.fw.web.struts.action.ActionMappingEx">
  <action path="/hogeSCR"
          type="jp.co.nttdata.terasoluna.fw.web.struts.actions.ForwardAction"
          parameter="/sample/hoge.jsp"
          name="_sampleForm"
          scope="session" />
    <set-property property="clearForm" value="true" />
  </action>
</action-mapping>
```

◆ Description

(1) When client sends a request with a path associated with a session-scoped ActionForm having logical name starting with "_".
   - ➢ All other ActionForms, (except the one associated with the current request) that have a logical name starting with "_" are deleted from the session by RequestProcessorEx.

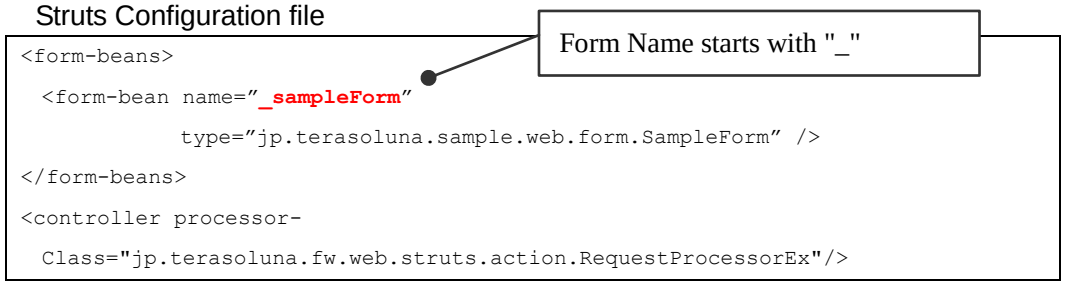
(2) When the client sends a request to a path associated with a session-scoped ActionForm having logical name starting with "_", and if "true" is set in clearForm property.
   - ➢ All ActionForm having logical name starting with "_" are deleted from the session by RequestProcessorEx.

■ Usage

◆ Coding Points

Set RequestProcessorEx. Attach "_" in the beginning of ActionForm name.

- Struts Configuration file




```
<form-beans>
  <form-bean name="_sampleForm"
             type="jp.terasoluna.sample.web.form.SampleForm" />
</form-beans>
<controller processor-
  Class="jp.terasoluna.fw.web.struts.action.RequestProcessorEx"/>
```

ActionForm is used in the session scope.

- Struts Configuration file

```
<action-mappings
  type=" jp.terasoluna.fw.web.struts.action.ActionMappingEx">
  <action
    path="/hogeSCR"
    type="jp.terasoluna.fw.web.struts.actions.ForwardAction"
    parameter="/sample/hoge.jsp"
    name="_sampleForm"
    scope="session" />
</action-mappings>
```

By setting clearForm property, all ActionForms having name starting with "_" in the session can be deleted.

- Struts Configuration file

```
<action-mappings
  type=" jp.terasoluna.fw.web.struts.action.ActionMappingEx">
  <action
    path="/hogeSCR"
    type=" jp.terasoluna.fw.web.struts.actions.ForwardAction"
    parameter="/sample/hoge.jsp"
    name="_sampleForm"
    scope="session" />
    <set-property property="clearForm" value="true" />
</action-mappings>
```

◆ Extension Points

None.

■ Classes

	Class Name	Overview
1	jp.terasoluna.fw.web.struts.action.RequestProcessorEx	Execute switching process of ActionForm

■ Related Function(s)

- "WB-02 ActionForm Extensions"

■ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web based)
  - "UC11 ActionForm toggle"
    - /webapps/formtrans/*
    - /webapps/WEB-INF/formtrans/*

■ Remarks

None.

WB-04 Form Property Reset Functionality

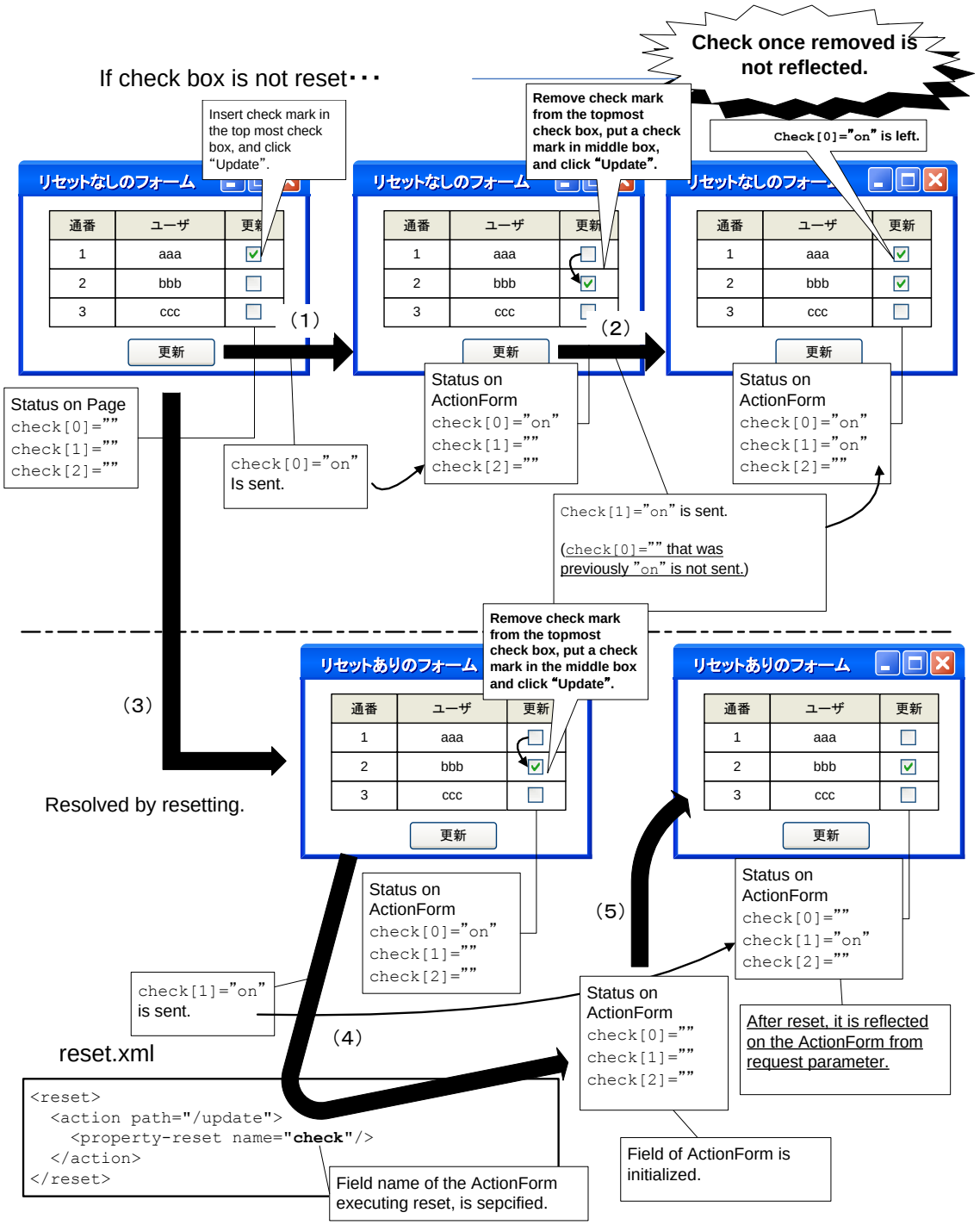
■ Overview

◆ Functional Overview

- In Struts, the reset method in ActionForm is called before updating the ActionForm with the request parameters. The “Form Property Reset functionality” overrides the reset method in ActionForm, and provides a mechanism to delegate the reset process to a class that implements the Resetter interface.
- The TERASOLUNA Server Framework for Java (Web version) provides ResetterImpl class as the default implementation class of Resetter interface. As per settings in configuration file, this class provides a mechanism to automatically initialize specified fields of ActionForm before updating them from request parameters.
- When a ListView has check boxes extending over several pages, it is necessary to initialize only the property of the check box for the currently displayed page. ResetterImpl class provides a mechanism to initialize only a specified range of properties in Array and List type.
- Reset method is provided with the aim to initialize properties that store the values of check box, select box, radio button etc fields of a session-scoped ActionForm. For other fields, the property value should be initialized by setting reset functionality to the actionpath of BLogicAction only. If reset function is set to the actionpath other than BLogicAction, due to extended actionform functionality, reset processing might take place at unexpected timings or might not occur at all. If reset functionality is to be executed reliably, then reset processing should be done using business logic without using this functionality.

◆ Schematic Diagram

Following is an example of reset of HTML check box using ResseterImpl



◆ Description

The below description assumes that the ActionForm is in session scope.

(1) [When check box is not reset]
Put a check mark in one of the check boxes and click Submit. While sending the request, only the fields with check mark are sent.
(check[0]=”on” is set)

(2) The ActionForm is updated for the location where a check mark was put in (1). Remove the check mark from the checked fields, and instead put a check mark in the non-checked fields and submit, "on" will get set on both check[0] and check[1].

(3) [When check box is reset]
Configure the reset function of ActionForm in Reset Definition File (reset.xml) and Struts Configuration File (struts-config.xml), and click on submit.

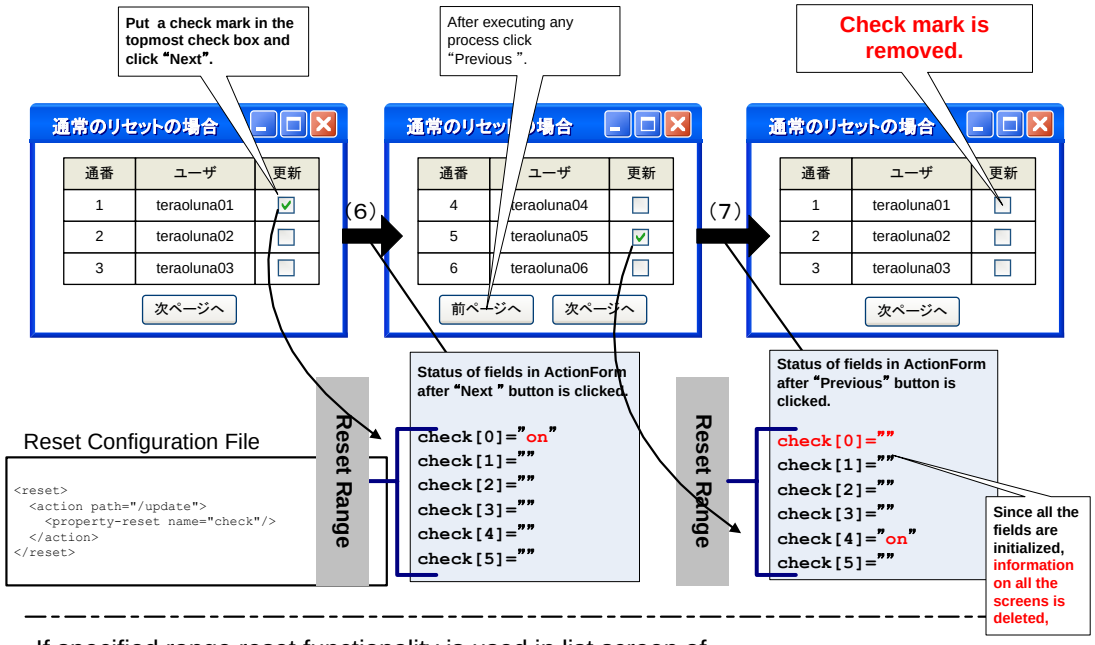
(4) In the Struts configuration file (struts-config.xml), configure the action path name similar to the action path name used during form submit, and configure the ActionForm field to be reset in the Form Reset Definition File (reset.xml). Only specified fields of ActionForm will be reset by this configuration. Here, check that the array is initialized.

(5) After reset, the values in request parameters will be copied to the ActionForm. ActionForm is also updated as per the status of the previous screen only (and not previous to previous screens). Thereby, the old values of check[0]=”on” can be cleared.

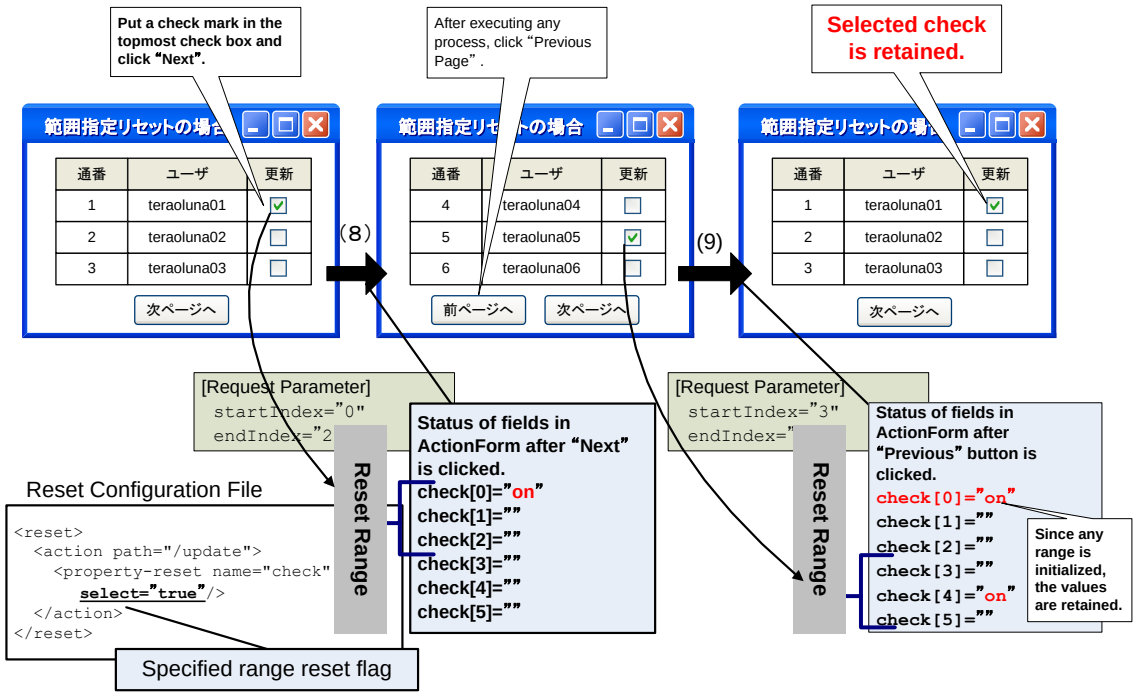
◆ Schematic Diagram

The following example describes reset of a specified range using ResetterImpl class.

If common reset process is executed in list screen of multiple pages・・・



If specified range reset functionality is used in list screen of multiple pages・・・



◆ Description

The below description assumes that ActionForm is session-scope.

(1) [In case of reset (Specified Range Reset Function-Disable)]
Enable the reset function of ActionForm, and submit.

(2) When check boxes extend over multiple screens (pages), if the specified field of ActionForm is reset using the usual reset configuration, then the check marks are removed from check boxes on all the pages.

(3) [Incase of reset (Specified Range Reset Function-Enabled)]
Enable the reset function of ActionForm, and enable the settings of specified range reset function of the specified field, and submit.

(4) Only the specified fields (List or array) of ActionForm in the specified range will be reset. By this, instead of removing the check marks from the check boxes on all the pages only the check marks on a specified page can be reset.

■ Usage

◆ Coding Points

The reset configuration file is initialized using plug-in functionality of Struts.

- Struts Configuration File (struts-config.xml)

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

Set plug-in class provided by framework.

Class that executes the reset process.

Configuration file of reset process. Associate the action path and property to be reset.

In Reset Configuration File, associate the Action Path and property to be reset.

- Reset Configuration File (reset.xml)

```
<reset>
  <action path="/resetAction">
    <property-reset name="field1" />
  </action>

  <action path="/selectResetAction">
    <property-reset name="field2" select="true"/>
  </action>

</reset>
```

Action path for which reset process is to be executed.

Property name of the action form to be initialized. Based on specifications of JXPathIndexedBeanWrapper, nested properties can be described.

While executing the specified range reset, set the select attribute to true.

When Specified Reset process is executed, the index, startIndex and endIndex within the range to be reset, are sent as request parameters from the screen to the server.

- Example of JSP that executes Specified Range Reset

```
 . . .
<ts:form action="/selectResetAction.do">
  . . .
  <html:hidden property="startIndex" value="0" />
  <html:hidden property="endIndex" value="3" />
  . . .
</ts:form>
 . . .
```

Only the 0 to 3 elements from List and Array are initialized.

While using <ts:pageLink> tag, startIndex and endIndex can be automatically output to JSP by using the Tag function. Please refer to "List View Function(s)", for details of <ts:pageLink>,.

◆ Extension Points

When contents of reset process are to be changed, create a class that implements Resetter interface, and write the class name in the configuration file.
Following is an example of a Resetter, which initializes value1 and value2 fields of ActionForm to “true”, when path is "~reset.do".

- Example of Resetter Implementation Class

```
public class MyResetter implements Resetter {

    public void reset(FormEx form, ActionMapping mapping,
              HttpServletRequest request) {
        String path = request.getRequestURI();
        if (path != null && path.endsWith("reset.do")) {
            form.set("value1", "true");
            form.set("value2", "true");
        }
    }
}
```

Resetter interface is implemented and Reset Process is described in Reset Method.

- Example of Struts Configuration File (struts-config.xml)

```
<plug-in
    className="jp.terasoluna.fw.web.struts.plugins.ResetterPlugIn">
    <set-property
        property="resources"
        value="/WEB-INF/reset.xml"/>
    <set-property
        property="resetter"
        value="jp.terasoluna.sample.MyResetter"/>
    ......
```

Created Resetter Implementation Class is described.

■ Classes

	Class Name	Overview
1	jp.terasoluna.fw.web.struts.plugins.ResetterPlugin	Plug-in that reads the Reset Configuration File and initializes the Reset Implementation Class.
2	jp.terasoluna.fw.web.struts.reset.Resetter	Interface that executes Reset Process.
3	jp.terasoluna.fw.web.struts.reset.ResetterImpl	Default Implementation Class of Resetter interface. Considering Configuration file as the base, the Reset Process is automatically executed.
4	jp.terasoluna.fw.web.struts.reset.ResetterResources	Class that stores the contents read from reset configuration file.
5	jp.terasoluna.fw.web.struts.reset.ActionReset	Class that stores the settings of Action path of Reset Configuration File.
6	jp.terasoluna.fw.web.struts.reset.FieldReset	Class that stores the settings of the fields of Reset Configuration File.

■ Related Function(s)

- “WB-02 ActionForm Extensions”

■ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web based)
  - “UC12 Form Property Reset”
    - /webapps/reset/*
    - /webapps/WEB-INF/reset/*
    - jp.terasoluna.thin.functionsample.reset.*

■ Remarks

None.

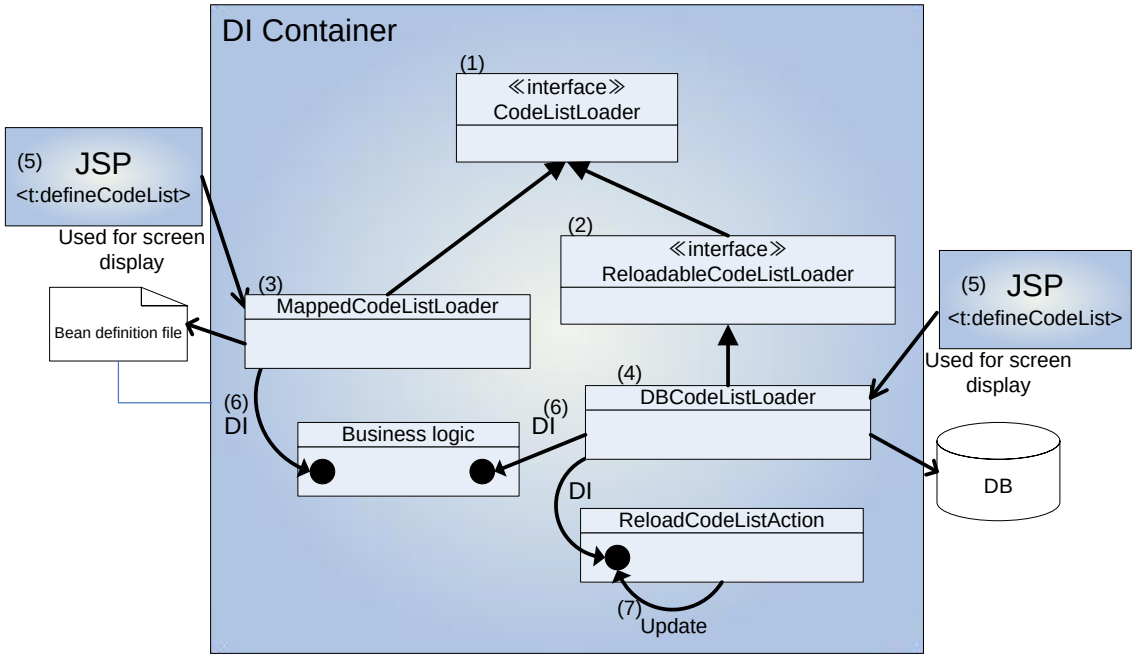
WB-05 Code List Function(s)

■ Outline

◆ Functional Outline

- In the data used by application, pairs of specific name and value not updated (or, updation frequency is minimum) during the execution of application is called as code list.
- It is the function that reads the code list from Bean definition file or from database.
- The code list that is read from database can be updated.

◆ Schematic Diagram



◆ Description

(1) Interface CodeListLoader that is implemented by the class loading the code list, is provided.
(2) Interface ReloadableCodeListLoader that is implemented by code list loader class that can be updated, is provided.
(3) Loader class, MappedCodeListLoader that loads the settings of code list from Bean definition file, is provided.

(4) DBCodeListLoader class that loads/updates the settings of code list from DB, is provided.
(5) DefineCodeListTag class is provided to display the information of code list on the screen.
※ Refer to “WJ-01~WK-06 Screen Display Function(s)”.
(6) Interface of CodeListLoader implementation class is set in the business logic instance by using DI container function, when the information of code list is to be referred from business logic.
(7) ReloadableCodeListLoader instance to be updated is set in ReloadCodeListAction class and doExecute() method is executed, when the code list is to be updated.

■ Usage Method

◆ Coding Points

[Initialization of code list using Bean definition file]

The code list Information is directly described in Bean definition file, when the code list information is to be read from Bean definition file. However, the Bean definition file describing the information should be the file that is read by ContextLoaderListener. (It should be defined in the Bean definition file that is loaded after plug-in).

- Deployment descriptor（web.xml）

```
<listener>
  <listener-class>
    org.springframework.web.context.ContextLoaderListener
   </listener-class>
</listener>

<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/applicationContext.xml</param-value>
</context-param>
```

Bean definition file where the code list read by ContextLoaderListener is to be defined

- Bean definition file

```
<bean id="sampleCodeList"
 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>
```

MappedCodeListLoader is specified to class attribute and load should be specified by init-method.

The code list information is set in codeList-Map variable with Map type. Key name of code list is set to entry element and value is set to value element.

[Initialization of the code list that uses the database]

SQL statement that acquires the code list information is set in Bean definition file, when the code list information is to be read from database.

SQL statement that acquires the code list can be set by any method, however as a result of acquisition from database, key values and values of code list are automatically set in 1st column and 2nd column respectively. They are set in Bean definition file that is read by ContextLoaderListener same as the code list read from Bean definition file.

- Bean definition file

```
       <bean id="sampleDBCodeList"
 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</value>
 </property>
</bean>
```

DBCodeListLoader is specified to class attribute and load should be specified by init-method.

Data source from which the code list information is to be read is specified to dataSource property. SQL statement that acquires the code list is set to sql. In this example, KEY is the code list id and VALUE is code list value.

[Use code list in screen display]

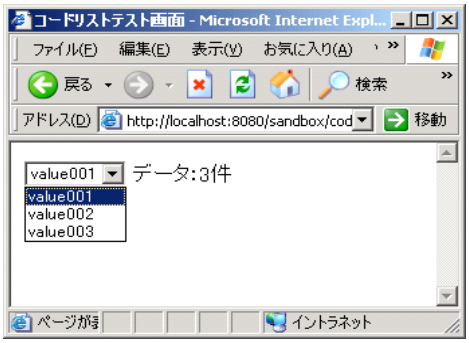
Using the code list in screen display is the intended purpose. Usage example of code list is given below.

- JSP

```
        <%@ taglib prefix="t"  uri="/WEB-INF/terasoluna.tld"%>

<html><head><body>
<html:form action="/codeList">
<t:defineCodeList id="sampleCodeList"/>
<html:select property="codeListId">
  <html:options collection="sampleCodeList" property="id" labelProper-
    ty="name"/>
</html:select>
Data : <t:defineCodeCount id="sampleCodeLIst" />count
</html:form></body></html>
```

<defineCodeList> tag defines code list in page scope. BeanId of CodeListLoader is specified to id and its value is considered as variable name on page.

<defineCodeCount> tag that displays the code list count. Code list count is directly displayed on screen.



[Refer the code list by Business logic]

Business logic implementation class should contain the reference to codeListLoader interface, when the code list is to be referred by Business logic. It is recommended to use DI function of Spring framework for MappedCodeListLoader and DBCodeListLoader that is provided by TERASOLUNA Server Framework for Java (Web version).

- Bean definition file

```
<bean id="sampleCodeList"
 class="jp.terasoluna.fw.web.codelist.MappedCodeListLoader"
  ……
</bean>
<bean id="sampleBusinessLogic"
  class="jp.terasoluna.sample.SampleBusinessLogic">
  <property name="codeListLoader">
    <ref bean="sampleCodeList"/>
  </property>
</bean>
```

CodeListLoader instance is set to Business logic implementation class.

- Business logic implementation class

```
        public class SampleBusinessLogic {
  private CodeListLoader codeListLoader = null;
  public void setCodeListLoader(CodeListLoader codeListLoader) {
    this.codeListLoader = codeListLoader;
  }

  public void someBusiness() {
    CodeBean[] beans = codeListLoader.getCodeBeans();
    ……
  }
}
```

Attribute and access method of CodeListLoader type is defined.

The code list information is stored in CodeBean class. CodeBean array can be acquired by getCodeBeans() method.

※ The code list information is read-only in business logic. As a general rule, the code list is the information that cannot be modified during execution of application. Hence, the code list contents should not be changed in the business logic.

[Update Code list]

DBCodeListLoader, that implements the updatable code list interface ReloadableCodeListLoader, can update the values from database through action class. Defining ReloadCodeListAction is enough to execute the update process.

When Codelist is to be updated access should be denied to the user by setting the Application to blockage state etc. In addition, the code list status is managed for each DI container. Hence, update process is essential in the environment where multiple JVM are operated such as cluster environment.

- Bean definition file

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

ReloadCodeListAction is specified to action class.

◆ Extension Points

Independent code list loader class can be defined by implementing the CodeListLoader interface.

- CodeListLoader.java

```
        public interface CodeListLoader {

   void load();

   CodeBean[] getCodeBeans();

}
```

The code list information is initialized by load() method. At the time of initialization, application should call this load() method and should initialize the code list information.

getCodeBeans() method returns the array of CodeBean class that implements the code list. As a general rule, the code list is the information that cannot be modified during execution of application. Hence, the reference in the class that uses this method, is not changed.

Updatable code list implements ReloadableCodeListLoader interface.

- ReloadableCodeListLoader.java

```
public interface ReloadableCodeListLoader extends CodeListLoader {

   void reload();

}
```

reload() method updates the code list information. The code list can also be updated through ReloadCodeListAction, if ReloadableCodeListLoader interface is implemented.

■ Classes

	Class Name	Outline
1	jp.terasoluna.fw.web.codelist.CodeListLoader	Interface implementing the class that reads code list
2	jp.terasoluna.fw.web.codelist.ReloadableCodeListLoader	Interface implementing the class that reads updatable code list
3	jp.terasoluna.fw.web.codelist.MappedCodeListLoader	CodeListLoader implementation class that reads the code list on the basis of Bean definition settings.
4	jp.terasoluna.fw.web.codelist.DBCodeListLoader	ReloadableCodeListLoader implementation class that reads code list from database.
5	jp.terasoluna.fw.web.codelist.DBCodeListQuery	Class used by DBCodeListLoader class to access the database.
6	jp.terasoluna.fw.web.codelist.CodeBean	Class implementing one record from the code list information.
7	jp.terasoluna.fw.web.taglib.DefineCodeListTag	Tag to define the code list in page variable.
8	jp.terasoluna.fw.web.taglib.DefineCodeCountTag	Tag to display the code list count on the page.
9	jp.terasoluna.fw.web.struts.actions.ReloadCodeListAction	Action class that executes the update process of ReloadableCodeListLoader.

■ Related Functionalities

- “WE-04 Code List Reload Function”
- “WJ-10 Code List Definition Function”
- “WJ-11 Codelist Count Function”
- “WJ-12 Specified Codelist Value Display Function”

■ Example

- Sample covering all TERASOLUNA Server Framework for Java (Web version) Functions
  - “UC13 code list”
    - /webapps/codelist/*
    - /webapps/WEB-INF/codelist/*
    - jp.terasoluna.thin.functionsample.codelist.*

■ Remarks

None.

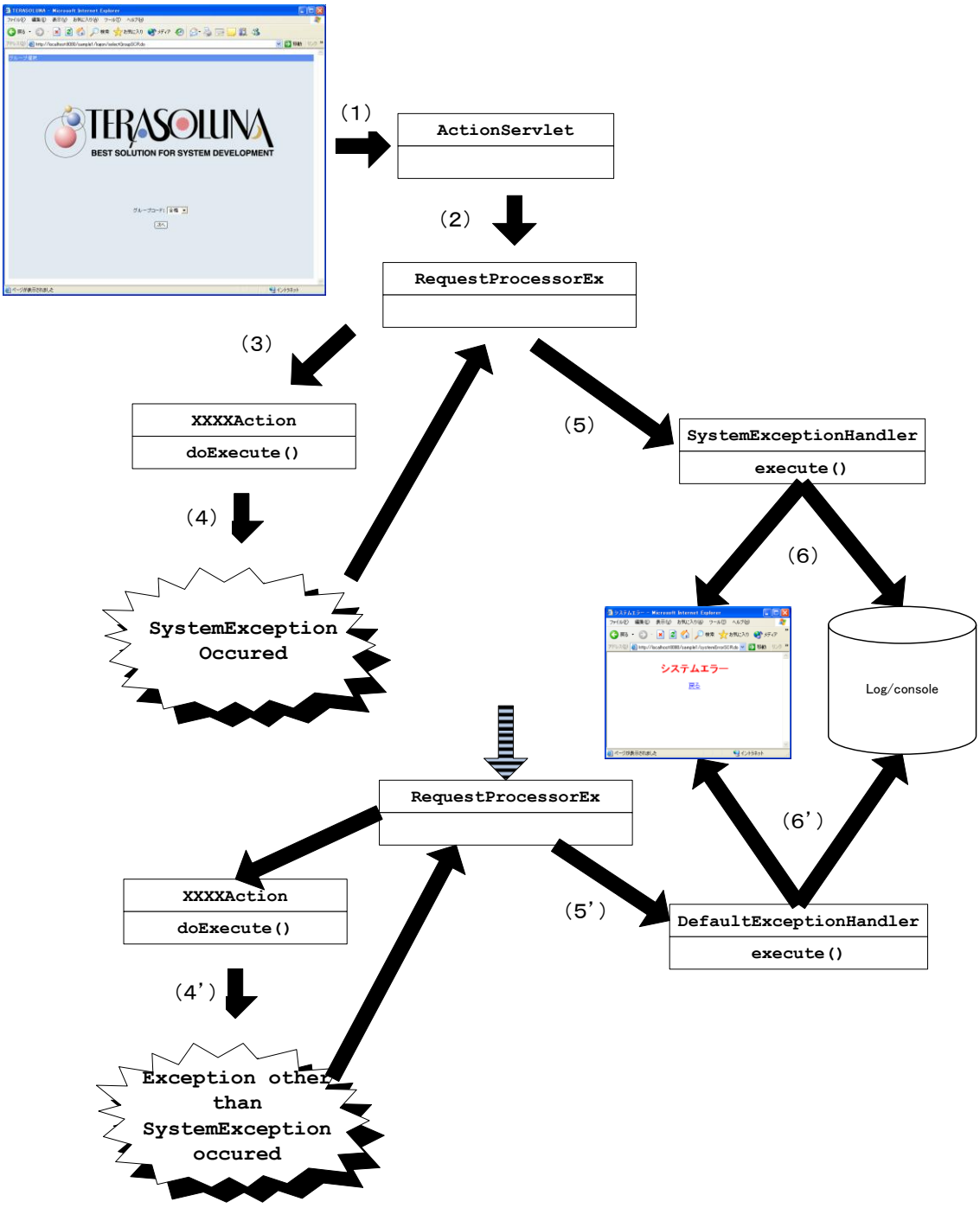
WC-01 Exception Handling

■ Overview

◆ Functional Overview

- Provides common exception handling functionality to handle exceptions in an action (and business logic).
- SystemExceptionHandler:
  - Exception handler specially for SystemException
  - If replacement string is present, it can be replaced in an error message
  - Write the exception stack trace to log
- DefaultExceptionHandler
  - General purpose exception handler
  - Write the exception stack trace to log

◆ Schematic Diagram



◆ Description

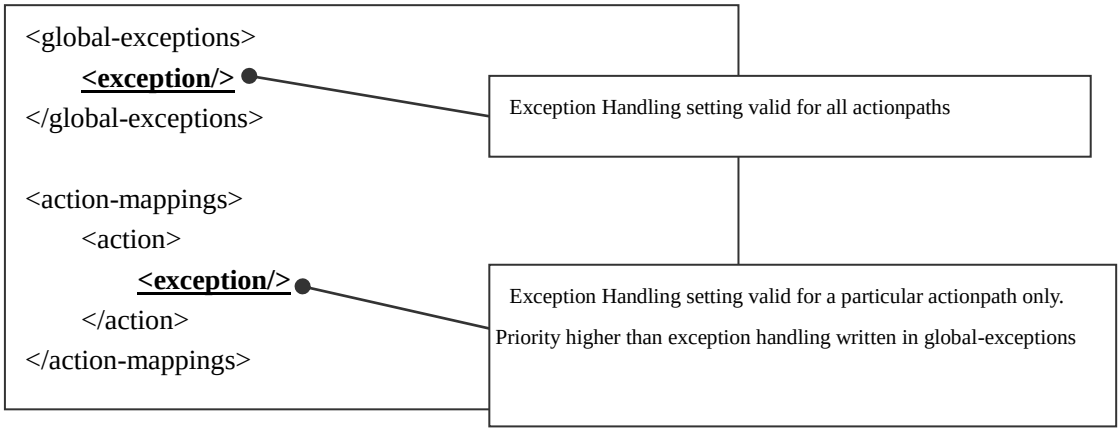
The following description assumes that SystemExceptionHandler and DefaultExceptionHandler are configured properly in Struts configuration file (struts-config.xml). Please refer to Coding steps for the details of configuration.

(1) Request is received from the user.
(2) RequestProcessorEx is called from the ActionServlet.
(3) The ActionEx#doExecute() method (corresponding to the request's action path) is executed by RequestProcessorEx.
(4) System exception occurs in the action.
(5) Exception is caught and the class handling system exception (SystemExceptionHandler#execute() method) is called.
(6) In the SystemExceptionHandler#execute() method, the error message is retrieved as per the message key stored for the thrown system exception, and the stack trace is written to log. Further, it is assumed that the system exception is also used in JSP, which will be displayed, and PageContext.EXCEPTION is stored in the request as a key, and then the control is forwarded (Here, it is forwarded to the System error screen) to the action path specified in the path attribute of <exception> element in Struts configuration file (struts-config.xml).
(7) Exception occurs (other than SystemException and its inherited classes)
(8) Exception is caught and the class handling system exception (DefaultExceptionHandler#execute() method) is called。
(9) In the DefaultExceptionHandler#execute() method, the error message is retrieved as per the message key stored for the thrown system exception, and the stack trace is written to log. Further, it is assumed that the system exception is also used in JSP, which will be displayed, and PageContext.EXCEPTION is stored in the request as a key, and then the control is forwarded (Here, it is forwarded to the System error screen) to the action path specified in the path attribute of <exception> element in Struts configuration file (struts-config.xml)

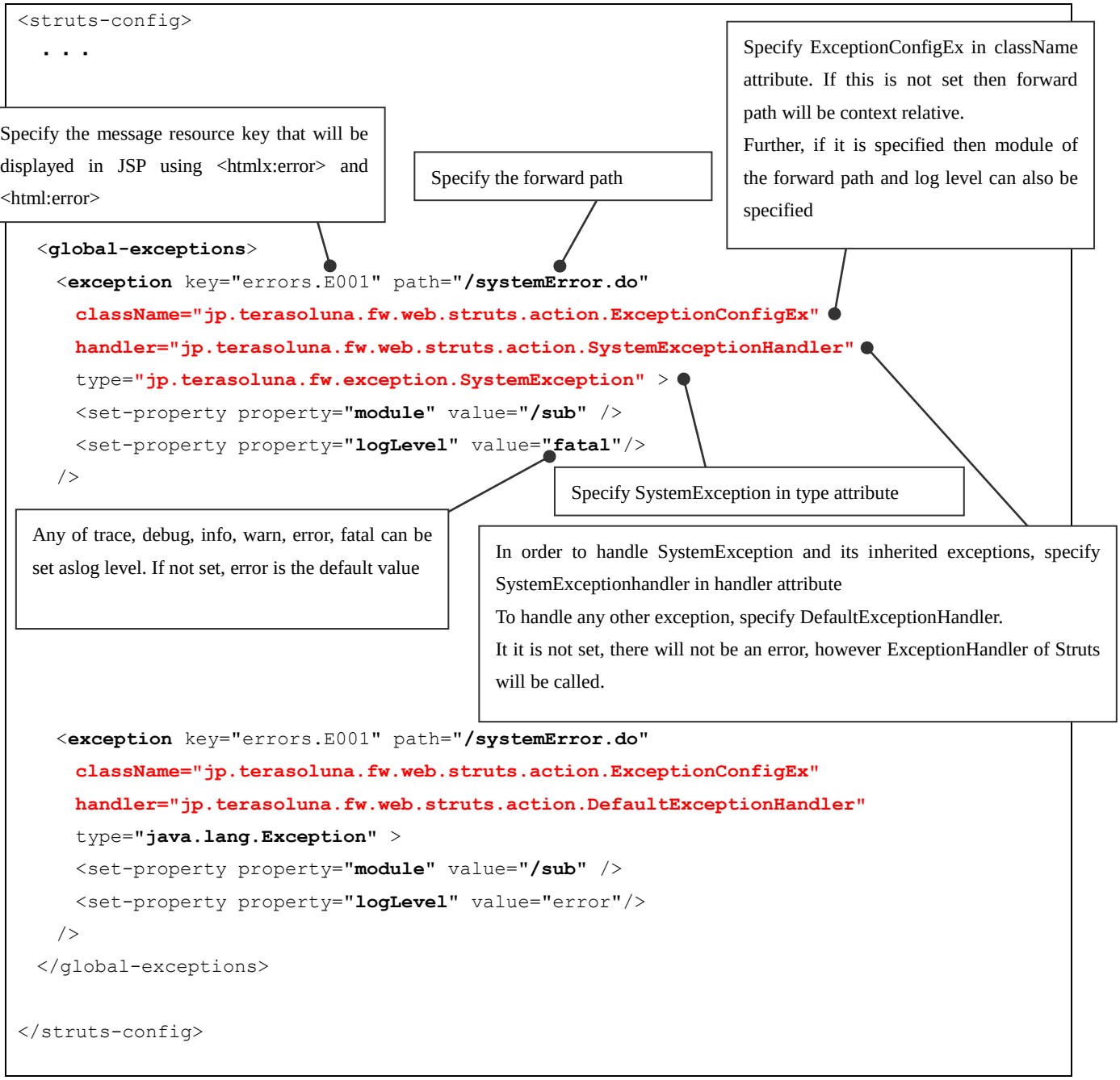
■ Usage

◆ Coding Points

Writing log when a system exception occurs and the error page to be displayed is specified in the <global-exceptions> element or <exception> element inside the <action> element of Struts configuration file (struts-config.xml).

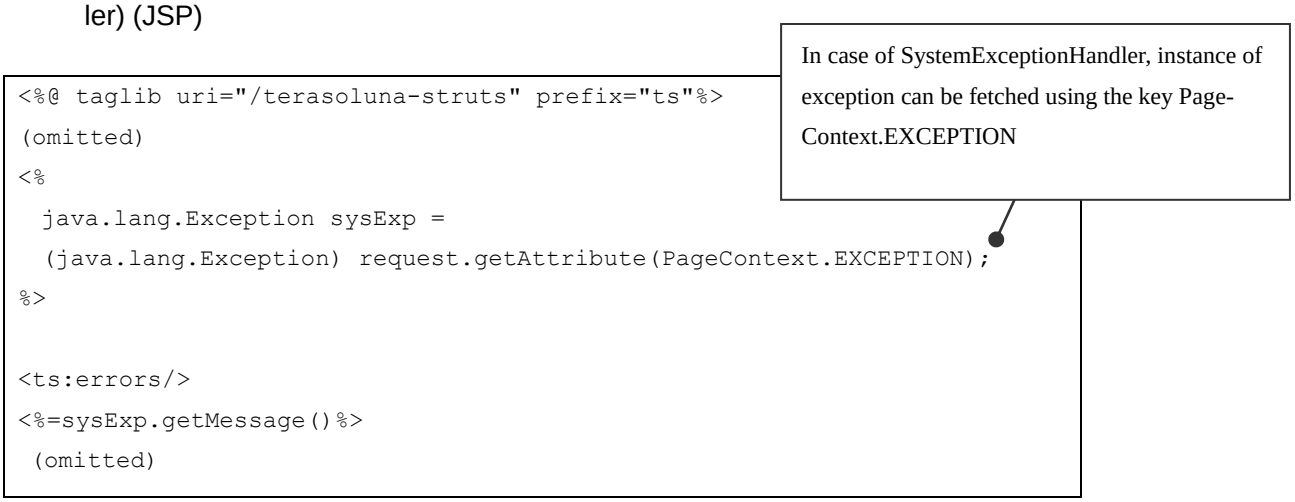


- global-exceptions element of Struts configuration file (struts-config.xml)



- action element of Struts configuration file (struts-config.xml)

- Example of Error screen (when exception is handled using SystemExceptionHandler) (JSP)



- Example of Error screen (when handling exception using DefaultExceptionHandler) (JSP)

```
<%@ taglib uri="/terasoluna-struts" prefix="ts"%>
(omitted)
<%
  java.lang.Exception exp =
  (java.lang.Exception) request.getAttribute(Globals.EXCEPTION_KEY);
%>

<ts:errors/>
<%=exp.getMessage()%>
(omitted)
```

In case of DefaultExceptionHandler, instance of exception can be fetched using the key Globals.EXCEPTION_KEY

◆ Extension Points

- None

■ Classes

	Class Name	Overview
1	jp.terasoluna.fw.web.struts.action.SystemExceptionHandler	Handler class that handles system exception.
2	jp.terasoluna.fw.web.struts.action.DefaultExceptionHandler	Handler class that handles exceptions other than system exception.
2	jp.terasoluna.fw.exception.SystemException	System exception class that is generated by framework.

■ Related Function(s)

- None

■ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web based)

- "UC14 Exception Handling"
  - /webapps/exception/*
  - /webapps/WEB-INF/exception/*
  - jp.terasoluna.thin.functionsample.exception.*

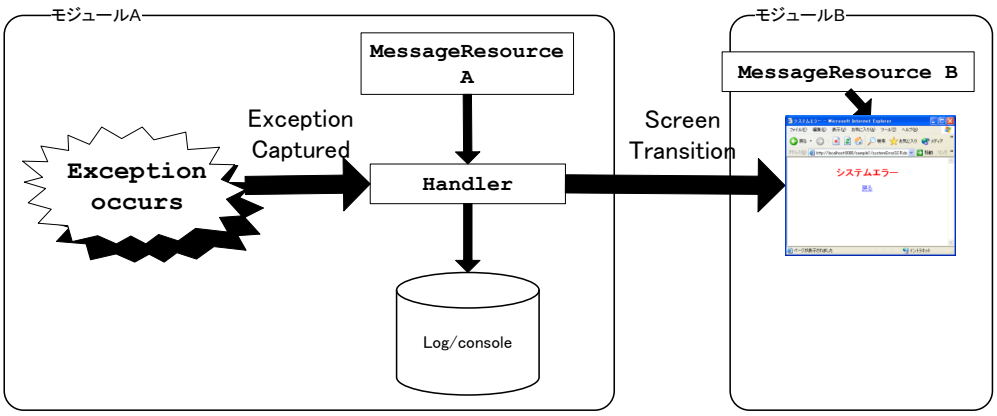
- TERASOLUNA Server Framework for Java (Web version) tutorial
  - 2.9 Exception Handling
  - /webapps/WEB-INF/struts-config.xml

■ Remarks

- **Regarding MessageResource used in Exception Handling**

  In a case where an exception occurs in a submodule which is captured using aexception handler and display an error screen, if the error screen is in separate module, then the message resource reference for log display and error screen display can be different.

  In this case, message resource used for log display will be of the module in which exception occurred. However, the message displayed on the screen will be the one present in message resource of the module in which error screen is placed.



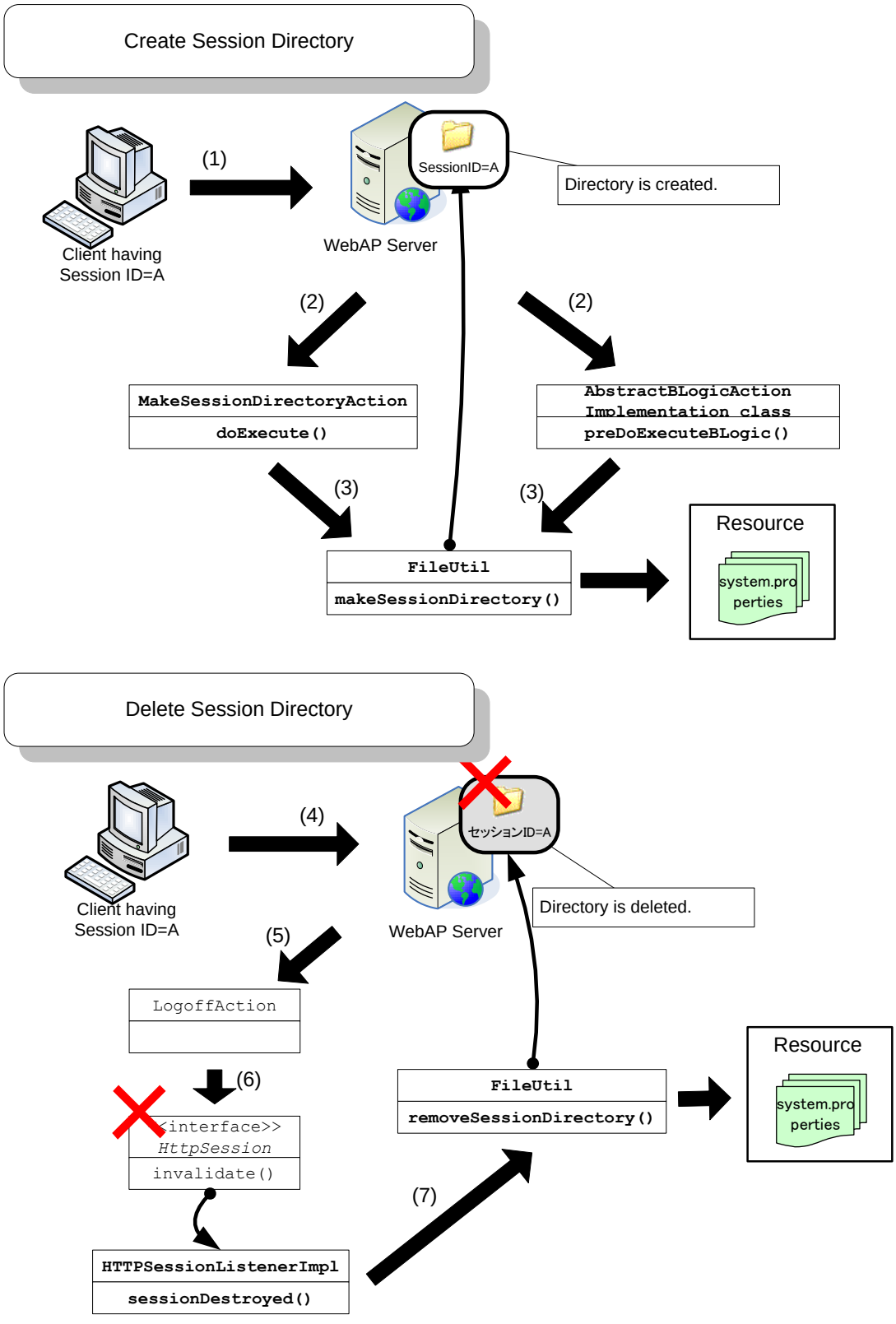
WD-01 Session Directory

■ Outline

◆ Functional Overview

- Creates a temporary directory (Hereafter, referred as the session directory) to store PDF files etc for every logon user on the server side. The temporary directory is deleted during logoff.
- Creates a session directory and destroys it during logoff.
- Following 2 methods can be used to create a session directory.
  - ➢ Usage of MakeSessionDirectoryAction
    - ✧ Please refer to “WE-05 Create Session Directory”.
  - ➢ Directly creating the session directory from the ActionClass
- Following method can be used to destroy the session directory.
  - ➢ Usage of LogoffAction Class
    - ✧ Please refer to ”WE-07 Logoff Function(s)”.

◆ Schematic diagram



◆ Description

(1) Request is received from the user during logon.
(2) Action to create session directory is executed and FileUtil#makeSessionDirectory() method is called. The Action class should use the MakeSessionDirectoryAction provided by TERASOLUNA Server Framework for Java (Web version) and the AbstractBLogicAction class implemented by the business developer.
(3) In FileUtil, Session directory is created as per session ID.
(4) Request is received from the user during logoff.
(5) LogoffAction (the Action class for Logoff) is executed.
(6) The callback method HttpSessionListenerImpl#sessionDestroyed() is called when session is invalidated by LogoffAction.
(7) Session directory is deleted by sessionDestroyed() method.

■ Usage

◆ Coding points

1. Property setting

The session directory for storing the files is set in the System properties configuration file (system properties).

➢ System properties Configuration File (system.properties)

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

Session directory is stored under /tmp/sessions

2. Creation of Session Directory

The following two methods can be used to create a session directory.

(a) Usage of MakeSessionDirectoryAction

Specify the Session directory creation class provided by TERASOLUNA in the action mapping.

➢ Struts Configuration file（struts-config.xml）

```
<action path="/makeSessionDir"
  scope="session"
  parameter="/foo.jsp">;
</action>
```

Any name may be specified, but it should be same in both files.

➢ Bean definition file

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

Session directory creation class provided by TERASOLUNA.

(b) Directly creating the session directory from the ActionClass

Before executing business logic, session directory is created in AbstractBLogicAction#preDoExecuteBLogic()

➢ AbstractBLogicAction implementation class

```
        package  jp.terasoluna.sample1.actions;
public class SampleAction extends AbstractBLogicAction {


  public void preDoExecuteBLogic(HttpServletRequest req,
      HttpServletResponse res,
     P params)
     throws Exception {
    ……
    HttpSession session = (HttpSession) request.getSession();
    boolean result = FileUtil.makeSessionDirectory(session.getId());
    ……
  }
}
```

AbstractBLogicAction is inherited.

Method called before executing Business Logic.

After getting the session, FileUtil#makeSessionDirectory(String sessionId) is called with the session's id.

Set the action mapping in the struts configuration file and Bean definition file.

➢ Struts configuration file（struts-config.xml）

```
        <action path="/makeSessionDir"
   name="_sampleForm"
    scope="session">
  <forward name="success" path="/logonSCR.do"/>
  <forward name="failure" path="/errRedirect.do"/>
</action>
```

Any name may be specified, but it should be same in both files.

➢ Bean definition file

```
        <bean name="/makeSessionDir" scope="prototype"
   class="jp.terasoluna.sample1.actions.SampleAction">
  <property name="sampleBLogic">
    <ref bean="SampleBLogic"></ref>
  </property>
</bean>
<bean id="SampleBLogic"
  class="jp.terasoluna.sample1.blogic.SampleBLogic">
</bean>
```

Specify the achat implements logic to create the session directory implemented

3. Session directory deletion

Specify the class HttpSessionListenerImpl provided by TERASOLUNA Server Framework for Java (Web version) for session life cycle event management in the deployment descriptor.

➢ Deployment Descriptor（web.xml）

```
<web-app>
……
<listener>
  <listener-class>
    jp.terasoluna.fw.web.HttpSessionListenerImpl
  </listener-class>
</listener>
……
</web-app>
```

Specify the listener class for session creation and invalidation.

The action mapping definition for logoff is set in Struts configuration file and Bean definition file.

Please refer to ”WE-08 Logoff function(s)” of Logoff Action for the details.

➢ Struts Configuration file（struts-config.xml）

```
<action path="/logoff"
  scope="session"
  parameter="/foo.jsp">;
</action>
```

During logoff, session is invalidated and session directory is deleted.

➢ Bean definition file

```
<bean name="/logoff" scope=”prototype”
   class="jp.terasoluna.fw.web.struts.actions.LogoffAction">
</bean>
```

■ Classes

	Class Name	Overview
1	jp.terasoluna.fw.util.FileUtil	A utility class to create delete and get a file or directory. Please refer to ”CD-01 Utility functions” for the details.
2	jp.terasoluna.fw.web.struts.actions.MakeSessionDirectoryAction	Creates a session directory for every logon user. Please refer to “WE- 06 Create Session Directory” for the details.
3	jp.terasoluna.fw.web.struts.actions.LogoffAction	An ActionClass that is executed during logoff. Invalidates the session as part of logoff. Please refer to “WE-08 Logoff” for the details.
4	jp.terasoluna.fw.web.HttpSessionListenerImpl	The class implementing HttpSessionListener interface that specifies the logic during creation/invalidation of session.
5	javax.servlet.http.HttpSessionListener	An interface with methods for creation and invalidation of session.

◆ Extension Points

- The Session directory creation process can be customized by extending AbstractBLogicAction. Please see “Directly creating the session directory from the ActionClass” in the Coding Points.
- The Session directory deletion can be extended by specifying a custom class implementing HttpSessionListener interface in deployment descriptor (without using the HttpSessionListenerImpl class provided by TERASOLUNA Server Framework for Java (Web version))

■ Related Function(s)

- “CD-01 Utility Functions”
- “WE-05 Create Session Directory Function(s)”
- “WE-07 Logoff”

■ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - “UC15 Session Directory”
    - /webapps/sessiondir/*
    - /webapps/WEB-INF/sessiondir/*
    - jp.terasoluna.thin.functionsample.sessiondir.*

■ Remarks

- None.

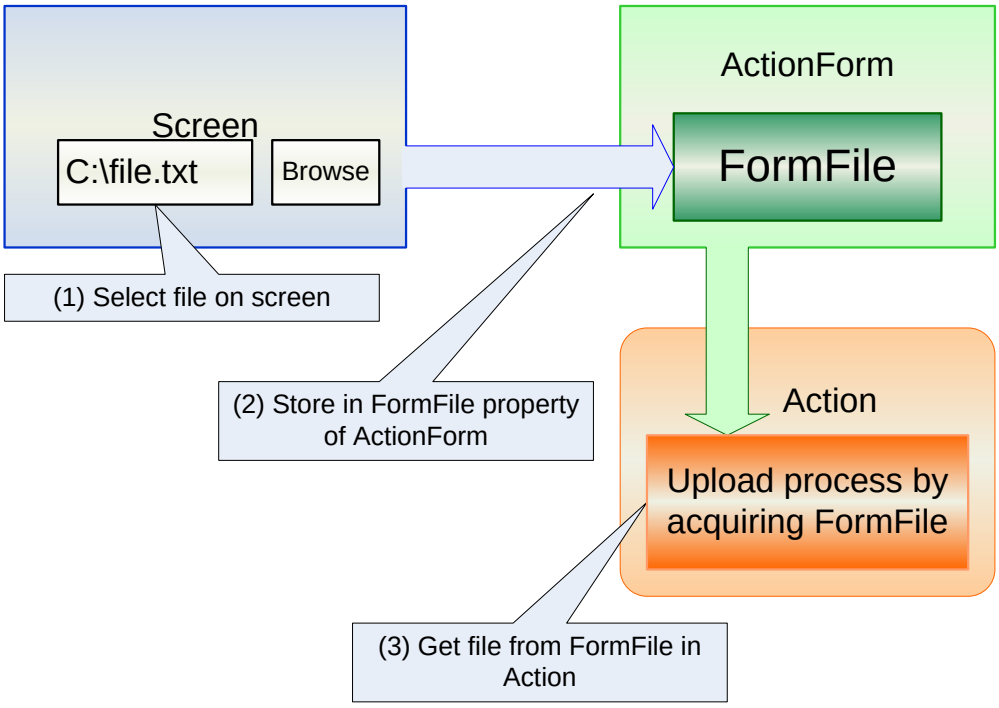
WD-02 File Upload Function

■ Overview

◆ Functional Overview

- Provides function to upload a file from a Web browser.
  - ➢ Function provided by Jakarta Commons FileUpload and Struts FormFile is used as it is.

◆ Schematic Diagram



◆ Description

(1) Create an fle Input field using <html:file> and select the file to be uploaded.
(2) The selected file is uploaded and stored in FormFile properties of ActionForm.
(3) The action retrieves the stored FormFile, executing the file writing and database updating logic.

■ Usage

◆ Coding Points

- Struts configuration file
  - ➢ <controller> element
    - ✧ bufferSize attribute
      Specify the size of buffer used in memory while reading the file.
      By default, It is 4096 bytes.
    - ✧ maxFileSize attribute
      Specify the maximum size of the file that can be uploaded. “K”, “M” and “G” can be used as units. Example: 100M→100 megabytes is the maximum size of file that can be uploaded.
    - ✧ memFileSize attribute
      Struts allows to either save the contents of uploaded file in memory as a byte array, or to save them in hard disk as a temporary file. The threshold value of memory storage and hard disk storage is specified in the memFileSize attribute. Similar to maxFileSize, “K”, “M” and “G” can be used as units.
    - ✧ tempDir attribute
      Specify the name of directory when an uploaded file is saved as a temporary file.
- JSP
  - ➢ <ts:form> element
    - ✧ enctype attribute
      Specify as Multipart/form-data as shown below.

```
<ts:form action="/fileup" enctype="multipart/form-data">
```

  - ➢ <html:file> element
    - ✧ property attribute
      Specify the FormFile property on the ActionForm.

```
<html:file name="dynaFormBean" property="fileup" />
```

- ActionForm
  - ➢ Struts configuration file (When DynaValidatorActionFormEx is used)
    - ✧ Specify FormFile in the type attribute of <form-property> element.

```
<form-property name="fileup"
               type="org.apache.struts.upload.FormFile"/>
```

  - ➢ ActionForm class (When ValidatorActionFormEx is used)
    - ✧ Create getter/setter methods for FormFile.

```
        private FormFile fileup = null;
public FormFile getFileup() {
    return fileup;
}
public void setFileup(FormFile fileup) {
    this.fileup = fileup;
}
```

- Action
  - In the action, file is written using the FormFile specified in Struts. Please study the following example and customize the file writing logic as necessary.

```
        // Get the uploaded file
FormFile fileup = (FormFile) dynaForm.get("fileup");

// Get the input stream by using getInputStream method
InputStream is = fileup.getInputStream();

// Use buffered input stream
BufferedInputStream inBuffer = new BufferedInputStream(is);

// Open file output stream at file destination.  FileOutput-
 Stream fos = new FileOutputStream
                    ("c:/tmp/" + fileup.getFileName());
// Use buffered output stream
BufferedOutputStream outBuffer = new BufferedOutputStream(fos);

// Execute read/write till there is no input data
int data = 0;
while ((data = inBuffer.read()) != -1) {
    outBuffer.write(data);
}

// flush
outBuffer.flush();
inBuffer.close();
outBuffer.close();
// delete the data uploaded in temporary area
fileup.destroy();
```

  - Setting of output location

    The above example writes the file to “c:/tmp/”. However, it is recommended to specify the output location in the property file. The creation/deletion of directory can be easily done using TERASOLUNA Server Framework for Java (Web version) “WD-01 Session Directory Function”, if the file is written only for one-time.

◆ Extension Points

None.

■ Related Function(s)

- “WD-03 File Download Function”

■ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - “WD-02 File Upload Function”
    - /webapps/upload/*
    - /webapps/WEB-INF/upload/*
    - jp.terasoluna.thin.functionsample.upload.*

■ Remarks

None.

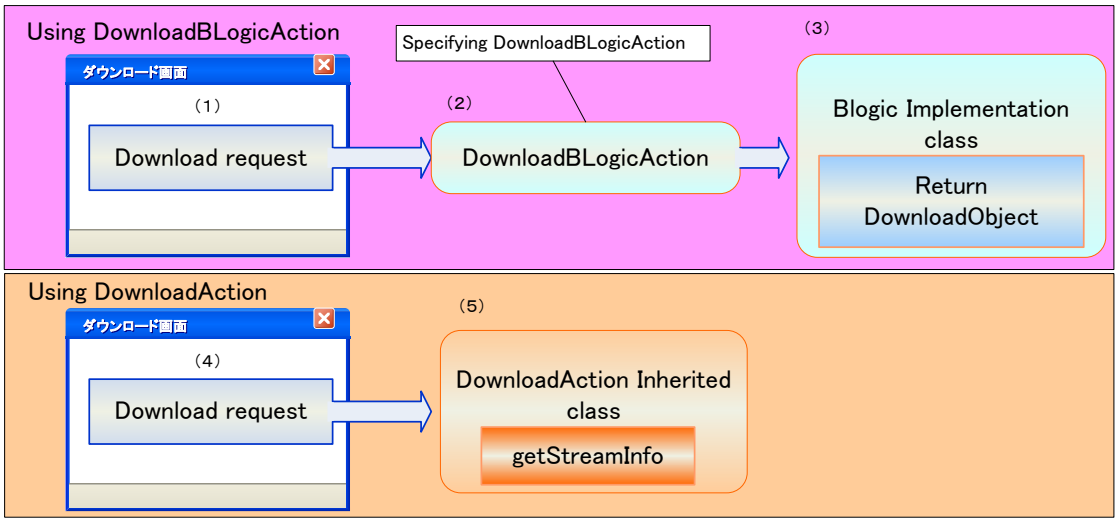
WD-03 File Download Function

■ Overview

◆ Functional Overview

- Provides the functionality to download a file to the client machine.
  - File download process can be implemented by using DownloadBLogicAction class created based on TERASOLUNA Server Framework for Java (Web version) or by creating a class inheriting from DownloadAction in Struts.
  - There is not much difference between the two implementation ways. However, if business logic needs to be consistent with BLogic interface, then DownloadBLogicAction should be used.

◆ Schematic Diagram



◆ Description

(1) File download is requested from the screen or another other class.
(2) File download process is executed by calling the getStreamInfo method of a class that inherits from DownloadAction.

■ Usage

◆ Coding Points

- Implementing file download using DownloadBLogicAction
  - Implementation regarding default filename encoding is done only for IE. In Firefox etc, there can be garbage characters in the filename. Please refer to Extention Point ”Ways to avoid garbage characters in filename related to cross-browser” for further details
    - ✧ Specifying DownloadBLogicACtion in Bean definition file.

```
<bean name="/downloadBLogic"
        class="jp.terasoluna.fw.web.struts.actions.DownloadBLogicAction"
        scope="singleton">
    <property name="businessLogic" ref="downloadBLogic"/>
</bean>
<bean id="downloadBLogic"
      class="jp.sample.project.blogic.DownloadBLogic" scope="singleton"/>
```

  - Implementing BLogic interface
    In BLogic interface implementation class, set the subclass of AbstractDownloadObject according to the type of download file, in BLogicResult.
    Below is the example that uses DownloadFile.
    In the example, DownloadFile is directly set into BLogicResult. However, setting it in the DTO and then setting the DTO in BLogicResult also works fine.

```
public class DownloadBLogic implements BLogic {

    public BLogicResult execute(DownloadInput param) {
        BLogicResult result = new BLogicResult();

        File file = new File("filepath¥¥download.txt");
        DownloadFile downloadFile = new DownloadFile(file);
        result.setResultObject(downloadFile);

        return result;
    }
}
```

- Implementation of DownloadAction#getStreamInfo()
  - Return the StreamInfo object
    - ✧ getStreamInfo () should return the StreamInfo object. StreamInfo is an interface and DownloadAction provided FileStreamInfo class that imple-

ments the StreamInfo interface.

- Usage of FileStreamInfo

  File content type to be downloaded and File object are necessary as shown below. Example of using FileStreamInfo.

```
        /**
 * Implementation example of getStreamInfo()
 */
protected StreamInfo getStreamInfo(ActionMapping mapping,
    ActionForm form, HttpServletRequest request,
    HttpServletResponse response) throws Exception {

    // get the file name
    String fileName = new String(
        bean.getFileName().getBytes(encoding), "ISO-8859-1");

    // get the content type
    String contentType = bean.getFileContentType();

    // set the response header
    response.setHeader("Content-disposition",
        "attachment; filename=" + fileName);

    // create file to download
    File file = new File("c:/tmp/" + fileName);

    // return the FileStreamInfo object
    return new FileStreamInfo(contentType, file);
}
```

- Prevent garbled characters in file name in case of fixed environment
  - ✧ In case the browser or OS to be used by the end-user is going to be fixed, then garbage characters in filename can be avoided in the following way. Specify appropriate character code like “Windows-31J” in getBytes() argument. Please refer to Extention Point ”Ways to avoid garbage characters in filename related to cross-browser” for further details

```
        String fileName = = new String(
    bean.getFileName().getBytes(encoding), "ISO-8859-1");
```

- Content-disposition header
  - ✧ It is necessary to set Content-disposition header. This example sets the header in the getStreamInfo() method.

    The “attachment” setting determines whether the download file is to be opened in the browser. File can be opened in the browser by specifying it

as “inline”.

```
response.setHeader("Content-
disposition",
"attachment; filename=" + fileName);
```

◆ Extension Points

- Ways to avoid garbage characters in filename related to cross-browser

  Due to differences of IE and Firefox etc, filename set in Content-disposition header gets corrupted. The cause is, in case of IE, filename represented in UTF-8 is encoded into x-www-form-url. While Firefox follows the RFC2047.

  For cross-browser compatibility, there is a need to encode the filename for each browser in User-Agent of Http header

  Below is the example showing settings of IE and Firefox

  - Downloading the file using DownloadBLogicAction

    Implement DownloadFileNameEncoder and perform the settings in jp.terasoluna.fw.web.struts.actions.FileDownloadUtil#setEncoder method at the time of starting the application.

    In the example, if the browser is found to be Firefox, encoding is done using commons-codec of Jakarta project

```
public class MyEncoder implements DownloadFileNameEncoder {

    public String encode(String original, HttpServletRequest request,
            HttpServletResponse response) {
        String userAgent = request.getHeader("User-Agent");
        // In case of IE
        if (StringUtils.contains(userAgent, "MSIE")) {
            return encodeForIE(original);
        // In case of Firefox
        } else if (StringUtils.contains(userAgent, "Gecko")) {
            return encodeForGecko(original);
        }
        return encodeForIE(original);
    }

    protected String encodeForGecko(String original) {
        try {
            return new BCodec().encode(original);
        } catch (EncoderException e) {
            return original;
        }
    }
```

```
    protected String encodeForIE(String original) {
        try {
            return URLEncoder.encode(original,
                    AbstractDownloadObject.DEFAULT_CHARSET);
        } catch (UnsupportedEncodingException e) {
            return original;
        }
    }
}
```

- Downloading file by inheriting DownloadAction
  Implementation is same as in case of DownloadBLogicAction. However it is to be done in the getStreamInfo() method

■ Related Function(s)

- "WD-02 File Upload Function"

■ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - "WD-03 File Download Function"
    - /webapps/download/*
    - /webapps/WEB-INF/download/*
    - jp.terasoluna.thin.functionsample.download.*

■ Remarks

None.

WE-01 Action Extensions

■ Overview

◆ Functional Overview

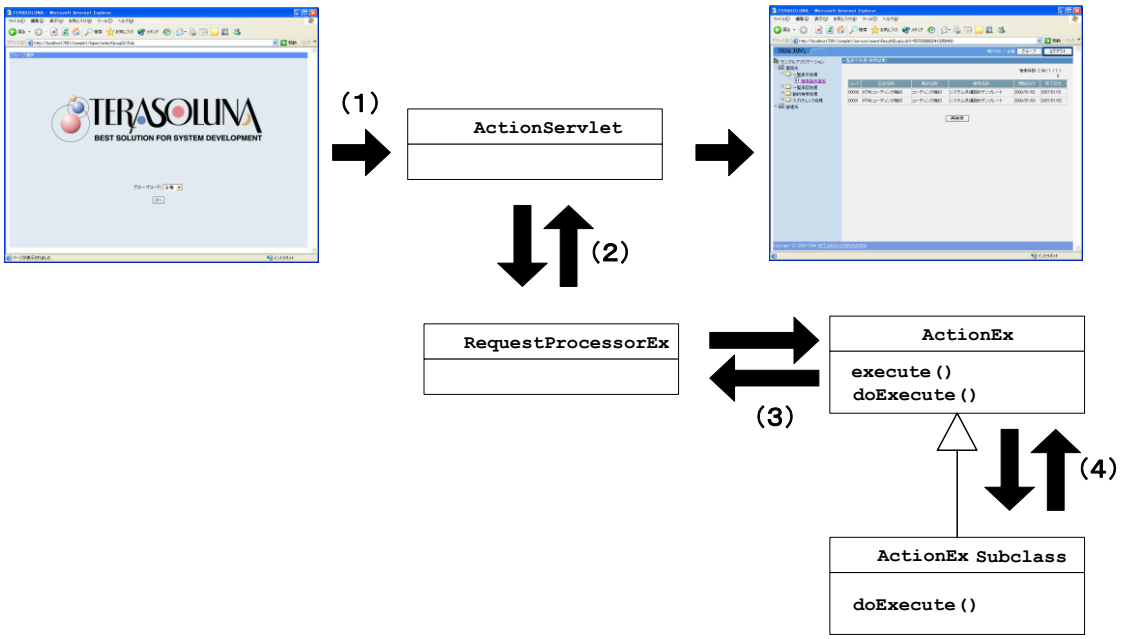
- Provides the base class of all action classes in the TERASOLUNA framework.
- Provides the following functions
  - ➢ Executes the transaction token check.
  - ➢ After execution of business logic, saves error and message information in session scope.

※ It is recommended to inherit ActionEx for each new Action in the business application.

※ Transaction token check is a function to prevents duplicate submits (that might happen due to use of the browser's "Back" button or by double clicking on the Submit button).

※ As a basic rule, it is recommended to specify the Action class in DI container and use it with scope="prototype".

◆ Schematic Diagram



◆ Description

(1) Request is sent to server from the browser.
(2) RequestProcessorEx is called from ActionServlet.
(3) ActionEx#execute() is executed by RequestProcessorEx#processActionPerform().
(4) “Transaction token check”, “flag indicating if the extended ActionForm is modified” are set to false, and then the execute() method in ActionEx subclass is called.

■ Usage

◆ Coding point

- Regarding the scope of Bean definition of Action class.
  - Specify scope=”prototype” or scope=”singleton”.
    - In case of singleton, Action class implementation need to be multithread-safe. While in case of prototype thread-safe implementation is not required. However, in case of prototype, everytime a new instance is created and hence performance might be compromised.
- Transaction token check
  - Configure as follows for the transaction token check.
    - In the Bean definition of action class, “true” should be specified for “token-Check” in <property> tag.
    - In the <forward> tag of struts-config.xml (possible in <global-forwards> also), specify the path when token error occurs having "txtoken-error" name.
  - When ActionEx is used, token is automatically saved by saveToken(). However, if it is not saved, do the following settings.
    - In Bean definition of action class, “false” should be specified for "saveToken" of <property> tag.

➢ Struts Configuration File (struts-config.xml)

```
<struts-config>
  ......
  <action-mappings type="jp.terasoluna.fk.web.struts.action.ActionMappingEx">
  <!—action setting -->
  <action path="/logoff"
            name="logonSampleForm"
            scope="session"
            parameter="/logoffForward.do">
    <forward name="txtoken-error" module="/sub" path="/doubleRegistError.do" />
  </action>
  <action path="/logoffForward" parameter="/logoff.jsp" />
  </action-mappings>
  ......
</struts-config>
```

Extension action mapping is specified

When failed in transaction token check, navigates to logical forward name "txtoken-error".

Any name may be specified, but it should be same in both files.

➢ Bean Definition file

```
<bean name="/logoff" scope="prototype"
    class="jp.terasoluna.fw.web.struts.actions.LogoffAction">
  <property name="tokenCheck">
    <value>true</value>
  </property>
  <property name="saveToken">
    <value>false</value>
  </property>
</bean>
<bean name="/logoffForward" scope="prototype"
    class="jp.terasoluna.fw.web.struts.actions.ForwardAction" />
```

Transaction token check is executed by specifying true in tokenCheck property. Default value is false.

Transaction token cannot be saved by specifying false in saveToken property. Default value is true.

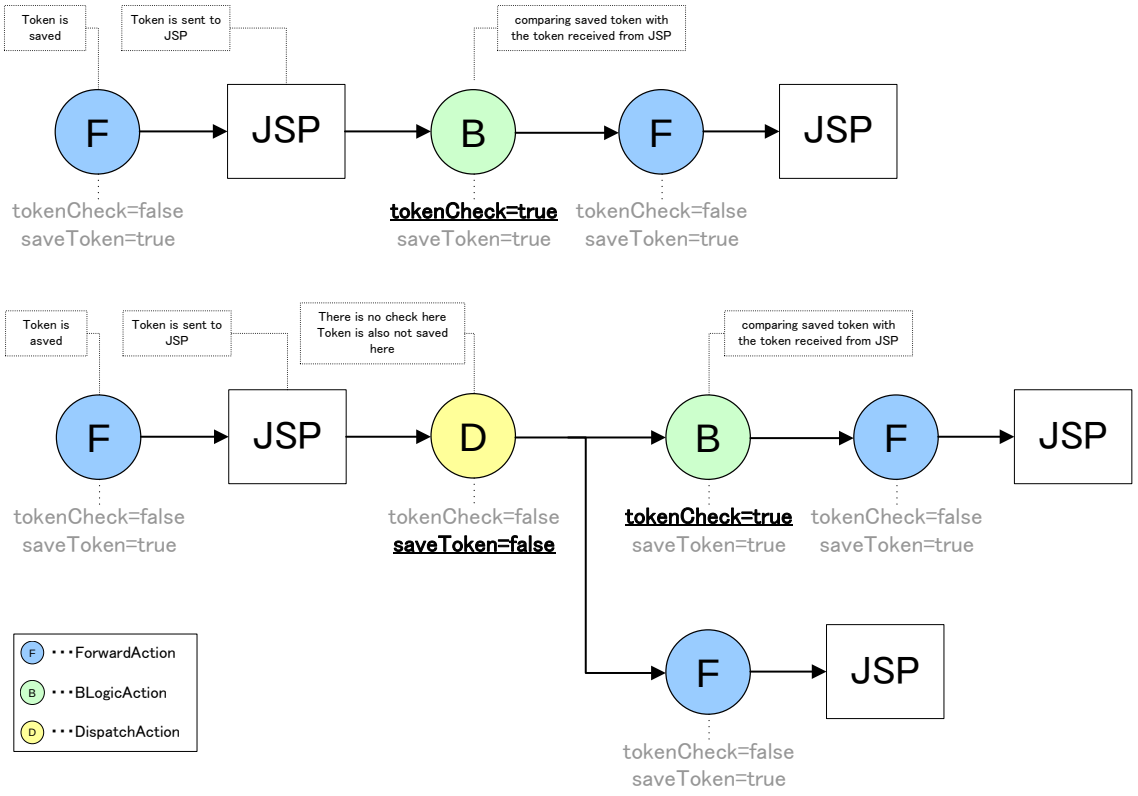
※ Here, since the token is saved in subsequent action (/logoffForward), it is set to false beforehand.

- Example of settings for Transaction Token check

  Below diagram shows 2 examples of settings for Transaction Token check.

  First example is a pattern in which business logic for which token check is required is directly called and next screen is displayed. In this case, by setting the tokenCheck property in the Bean definition of BLogicAction to true, token saved in the source screen is checked before executing the business logic.

  Second example is the pattern in which the destination is determined on the basis of button pressed or radio button selection etc on the screen. In this case, token check is performed at respective BLogicAction and not at DispatchAction level. The point to note here is that, in the Bean definition of DispatchAction, saveToken should be set to false to prevent saving of the Token.



■ Classes

	Class name	Overview
1	jp.terasoluna.fw.web.struts.actions.ActionEx	Extension action class provided by TERASOLUNA framework.
2	jp.terasoluna.fw.web.struts.action.ActionMappingEx	Extension Action Mapping Class.

3	jp.terasoluna.fw.web.struts.actions.AbstractBLogicAction	Abstract action class that executes business logic. It copies input/output data between business logic and the action form/ session. Please refer to “WH-01Business Logic Execution” for the details.
4	jp.terasoluna.fw.web.struts.actions.DispatchAction	Action that decides the dispatch location as per a value of a request parameter. Please refer to “WE-02 Standard Dispatcher Function(s)” for the details.
5	jp.terasoluna.fw.web.struts.actions.ForwardAction	Action that forwards as per the path in struts configuration. Please refer to “WE-03 Forwarding Function(s)” for the details.
6	jp.terasoluna.fw.web.struts.actions.ReloadCodeListAction	Action that reloads code list stored in ServletContext. Please refer to “WE-04Code List Reload Function(s)” for the details.
7	jp.terasoluna.fw.web.struts.actions.MakeSessionDirectoryAction	Action that creates user specific directories. Please refer to “WE-05 Create Session Directory” for the details.
8	jp.terasoluna.fw.web.struts.actions.ClearSessionAction	Action class that deletes the key defined in Bean definition file from session. Refer to “WE-06 Session Clear Function(s)” for details.
9	jp.terasoluna.fw.web.struts.actions.LogoffAction	Action that invalidates session at the time of logoff. Please refer to “WE-07 Logoff Function(s)” for the details.

◆ Extension Points

While implementing action class newly by TERASOLUNA Server Framework for Java (Web version) extension and business, class that inherits ActionEx should be created.

■ Related Function(s)

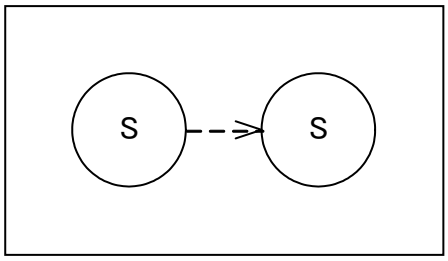
- “WE-02 Standard Dispatcher function(s)”
- “WE-03 Forwarding Function(s)”
- “WE-04 Code List Reload Function(s)”
- “WE-05 Create Session Directory function(s)”
- “WE-06 Clear Session Function(s)”
- “WE-07 Logoff Function(s)”
- “WH-01 Business Logic Execution”

■ Example

- Sample covering all functions of TERASOLUNA Server Framework for Java (Web version)
  - “UC16 Action Extension”
    - /webapps/actionex/*
    - /webapps/WEB-INF/actionex/*
    - jp.terasoluna.thin.functionsample.actionex.*

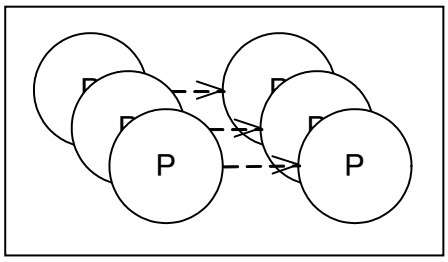
■ Remarks

- While injecting prototype Bean into the singleton Bean, below mentioned poin ts need to be taken care of:
  - Scope can be written separately for each bean in Bean definition file. H owever, when DI container combines such beans, the behavior can be d ifferent from expectation. For example, since a singleton bean is just a s ingle object, if a prototype bean is injected into it, then prototype bean will effectively be a singleton bean.
  - A singleton bean referring another singleton bean



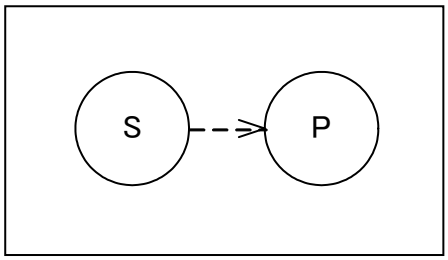
When a singleton bean is injected into another singleton bean, the combinati on of the two is also a single object across the system.

- A prototype Bean referring another prototype bean



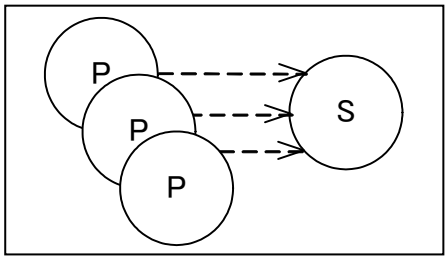
When a prototype bean is injected into another prototype bean, the combinati on of the two is also gets instantiated every time

- A singleton Bean referring a prototype bean



When a prototype bean is injected into a singleton bean, since only a single objectexists for the singleton bean and hence if there is no injection into the prototype bean, then the prototype bean is also a single object. This prototyp e bean also needs to be thread-safe.

- When prototype bean refers a singleton bean.



In this case multiple objects are referring a single object.

In the above case, the scope of the first object that will be fetched by DI co ntainer should be considered. In case of TERASOLUNA Server Framework f or Java (Web version), in the beginning, Action class will be fetched from DI container for each request (for almost all AP Servers, 1 request=1 thread).

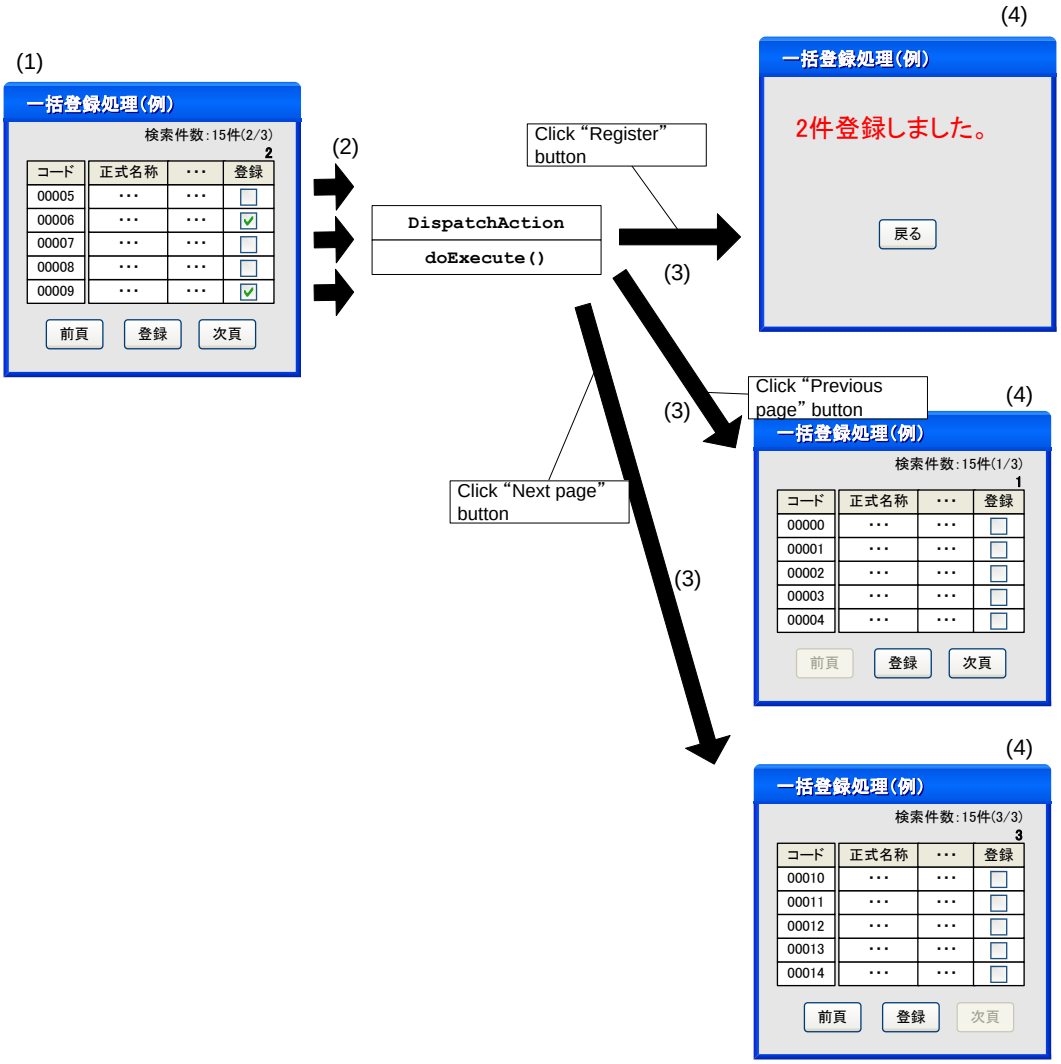
WE-02 Standard Dispatcher Function

■ Overview

◆ Functional Overview

- TERASOLUNA Server Framework for Java (Web version) provides the following 2 dispatcher functionalities.
  - The action to be called for each “Submit” button can be specified in JSP, Struts configuration file (struts-config.xml) and Bean definition file (it is not necessary to create a separate action).
  - Further, even for screens having a single submit button, the action to be called can be selected by setting the request parameter.

◆ Schematic Diagram



◆ Description

(1) Specify Multiple “Submit” buttons in one JSP.
(2) Specify ”forward_XXXX” value in the request parameter and then send it.
(3) Get the logical forward name from the request parameter value that is sent.
(4) Navigate to the next screen as per the logical forward name.

■ Usage

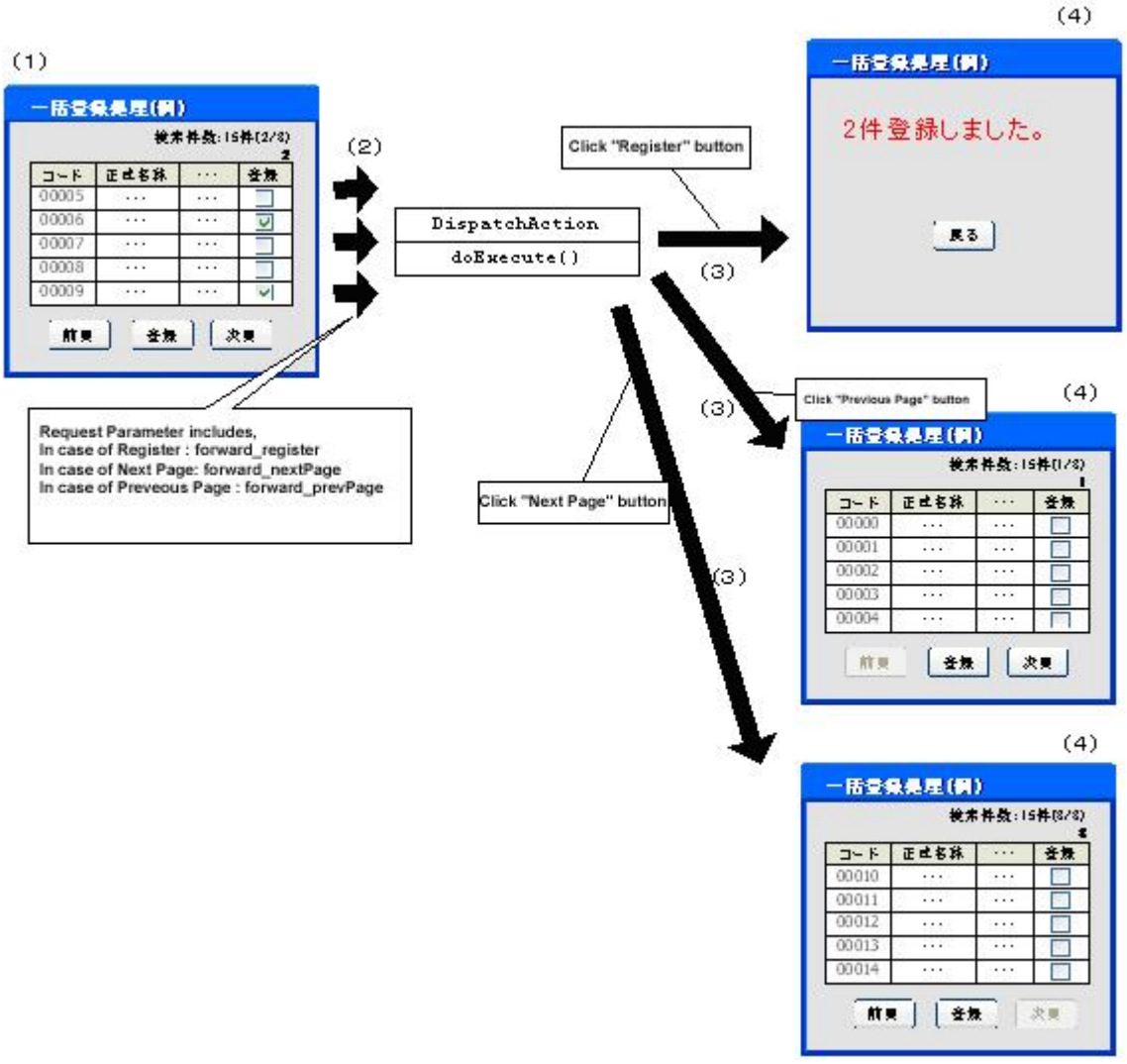
◆ Coding Points

- Standard Dispatcher Function(s) can be used in the following 2 ways.

(1) The method to dispatch using multiple “Submit” buttons.

While using this function, it is necessary to edit Struts configuration file (struts-config.xml), the Bean definition file, and JSP.

When the “Register” button is clicked, the request parameter contains the forward_regist key and the next screen is selected using regist as the logical forward name.



➢ Struts configuration file (struts-config.xml)

```
        <struts-config>
  ……
  <action-mappings type="jp.terasoluna.fw.web.struts.action.ActionMappingEx">
    ……
    <action path="/fileUploadConfirmDSP" name="_fileForm">
      <forward name="back" path="/initFileUploadAction.do"/>
      <forward name="default" path="/fileUploadConfirmSCR.do"/>
      <forward name="regist" path="/fileRegistBLogic.do"/>
    </action>
  <action-mappings>
  ……
</struts-config>
```

Action path

Action forward is defined for each value where "forward_", sent by request parameter, is omitted.

➢ Bean definition file

```
        <bean name="/sample/selectDispatch" scope="prototype" class=
  "jp.terasoluna.fw.web.struts.actions.DispatchAction">
</bean>
```

DispatchAction is specified.

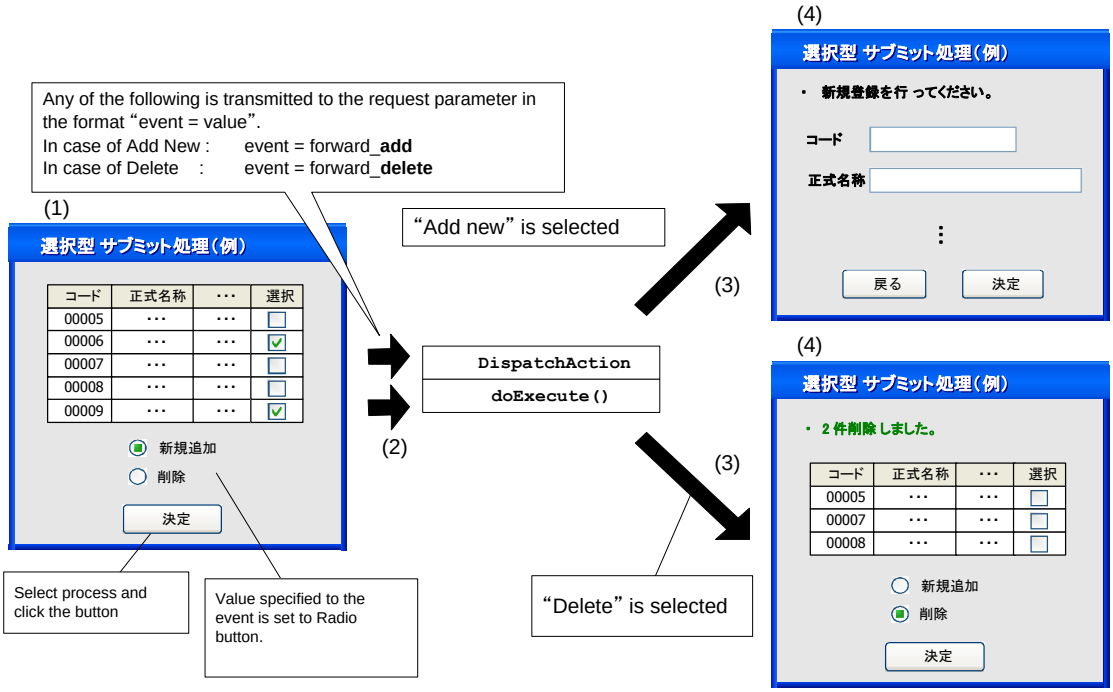
➢ JSP file

```
        <html:html>
  ……
  <html:form action="/fileUploadConfirmDSP">
     ……
    <html:submit property="forward_back" value=" return "/>
    <html:submit property="forward_regist" value=" register"/>
   ……
  </html:form>
   ……
</html:html>
```

Action path defined in Struts configuration file is specified.

Request parameter to be sent in "forward_XXXX" format is specified.
XXXX is used as logical forward name by DispatchAction#doExecute().

(2) Example of setting the navigation page by clicking radio button is shown below. Here, it is assumed that “event” field on the form is already used and the request parameter key, which indicates navigation page, is changed from “event” to “dispatchName”.



➢ Struts configuration file (struts-config.xml)

```
        <struts-config>
  ......
  <action-mappings type=" jp.terasoluna.fw.web.struts.action.ActionMappingEx">
    ...
    <action path="/fileUploadConfirmDSP" name="_fileForm">
      <forward name="back" path="/initFileUploadAction.do"/>
      <forward name="default" path="/fileUploadConfirmSCR.do"/>
      <forward name="regist" path="/fileRegistBLogic.do"/>
    </action>
  <action-mappings>
  ......
</struts-config>
```

Action path

Action forward is defined for each value where “forward_”, sent by request parameter, is omitted.

➢ Bean definition file

```
        <bean name="/sample/selectDispatch" scope="prototype"
   class="jp.terasoluna.fw.web.struts.actions.DispatchAction">
  <property name="event">
    <value>dispatchName</value>
  </property>
</bean>
```

DispatchAction is specified.

Request parameter key, which specifies dispatch destination, is changed from event to dispatchName.

➢ JSP file

```
        <html:html>
 ......
  <html:form action="/fileUploadConfirmDSP">
    ......
      <html:radio property="dispatchName" value="forward_back"/> Return
      <html:radio property="dispatchName" value="forward_regist"/> Register
  ......
      <html:submit property="decide" value=" decide ">
  </html:form>
  ......
</html:html>
```

Action path defined in Struts configuration file is specified.

Request parameter key defined in Struts configuration file is specified.

Request parameter value is defined as "forward_" + Logical forward name.

- Regarding priority of forward path

  Priority of the forward string is decided as below:

  1. Request parameter key values with "forward_" removed
  2. Request parameter key values with "forward_" removed
  3. "default" fixed（in case of request parameters for which event="XXXX"、"forward_XXXX" type of values couldnot be fetched etc）

  If the result of above is "#input", then forward with value "input" element of "struts-config.xml" will become the target destination. If the output of above is not "#input", then destination is determined on the basis of contents of forward element of "struts-config.xml"

◆ Extension Points

To determine the page to be displayed when the "Cancel" button is clicked, override cancelled() method by inheriting this class.

In the below example, the page to be displayed when "Cancel" is clicked is determined from the logical name "cancelled".

Here, in /fileUploadConfirmDSP.do action process, it navigates to /backward.do on clicking the "Cancel" button.

➢ DispatchAction Inherited Class (DispatchActionEx)

DispatchAction is inherited.

```
public class DispatchActionEx extends DispatchAction {
    ……
    protected ActionForward cancelled(ActionMapping mapping,
                                      ActionForm form,
                                      HttpServletRequest request,
                                      HttpServletResponse response) {
        // Get forward destination at the time of cancel from action mapping.
        ActionForward forward = mapping.findForward("cancelled");
        return forward;
    }
    ……
}
```

The navigation page defined as "cancelled" is searched from action forward definition in Struts configuration file, and ActionForward instance is returned.

➢ Struts configuration file (struts-config.xml)

```
<action-mappings
    type="jp.terasoluna.fw.web.struts.action.ActionMappingEx">
  ……
  <action path="/fileUploadConfirmDSP" name="_fileForm">
    <forward name="back" path="/initFileUploadAction.do"/>
    <forward name="default" path="/fileUploadConfirmSCR.do"/>
    <forward name="regist" path="/fileRegistBLogic.do"/>
    <forward name="cancelled" path="/backward.do"/>
  </action>
<action-mappings>
```

Navigation page on clicking “Cancel” button is specified in action forward definition “cancelled”.

➢ Bean definition file

```
        <bean name="/fileUploadConfirmDSP" scope=”prototype”
    class="jp.terasoluna.sample.action.DispatchActionEx">
</bean>
```

Class inheriting the DispatchAction created above is specified.

■ Classes

	Class Name	Overview
1	jp.terasoluna.fw.web.struts.actions.DispatchAction	This is a dispatcher class used to dispatch to the next navigation path. The logical forward name for the navigation path is determined from value of the request parameter - event or forward_XXX.

■ Related Function(s)

- None

■ Example

- Sample covering all functions in the TERASOLUNA framework (web based)
  - "UC17 Standard Dispatcher"
    - /webapps/dispatch/*
    - /webapps/WEB-INF/dispatch/*
    - jp.terasoluna.thin.functionsample.dispatch.*

■ Remarks

- When request parameter containing forward information is not sent by the browser
  In some browsers, if Enter key is pressed on a text field of HTML form, then the form gets submitted without setting the name and value attribute values in request parameters.
  In above case, destination defined as "default" will be selected as target destination.
  If "default" forward is not defined in struts-config.xml then empty screen might get displayed. Hence, default forward should be defined in all cases.

- When Struts-config.xml doesn't contain the forward name
  When struts configuration file contains any of the below:
  1. Mistake in writing forward name
  2. Falsification of request parameters

  etc there is a possibility that a forward name that is not defined in struts configuration file is sent in the request parameters. (for example, event=forward_**notdefined or** forward_**notdefined**=yyyy is sent in request parameters) then default forward will be used.

- Struts Configuration file（struts-config.xml）

```
<action-mappings
    type="jp.terasoluna.fw.web.struts.action.ActionMappingEx">
  ……
  <action path="/registerDSP" name="_registerForm">
    <forward name="regist" path="/registerBLogic.do"/>
    <forward name="default" path="/backward.do"/>
  </action>
<action-mappings>
```

Since a forward named **notdefined** is not present, default forward will be used.

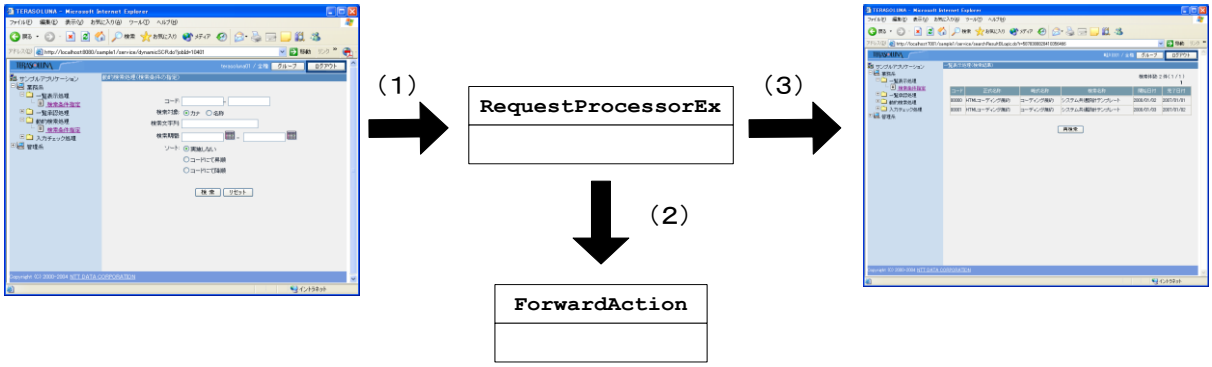
WE-03 Forwarding Function

■ Overview

◆ Functional Overview

- As a principle, in the TERASOLUNA Server Framework for Java (Web version), the browser is denied direct access to URI with the extension "jsp". Therefore, the framework provides an action to display JSP and write the access log.
  ※ Please refer to “WA-05 Denial of Direct Access to Extensions” for the details of denial of direct access to JSP by specifying the address.
- Similar to the ForwardAction provided by Struts, the forward destination can be specified to parameter attribute of <action> element or by using <forward> element in struts-config.xml. In case of redirect or forward between modules, the forward destination should be specified by using <forward> element.

◆ Schematic Diagram



◆ Description

(1) Request is sent from the browser.

(2) RequestProcessorEx calls ForwardAction. ForwardAction does not execute any business logic. It writes the log, calls the base class ActionEx method and returns the ActionForward for the path specified in the parameter attribute of <action> element in Struts configuration file (struts-config.xml). When the parameter attribute is not set, ActionForward of logical forward name ‘success’ is returned. Please refer to “WE-01 Action Extensions” for details on ActionEx.

(3) Forward the path of ActionForward. When forward destination is not set, SC_NOT_FOUND (404) error is returned.

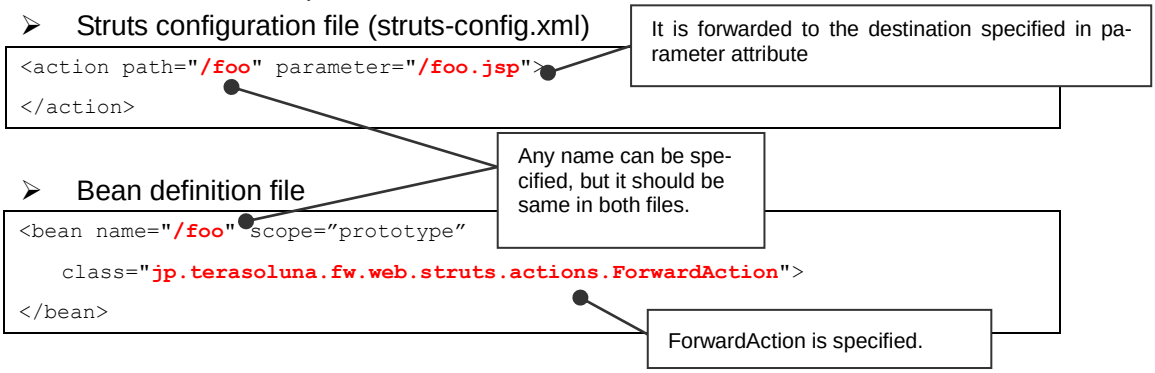
■ Usage

◆ Coding Points

- Forward function can be implemented by configuring Struts configuration file (struts-config.xml) and Bean definition file.

- When forward destination is to be specified by parameter attribute of <action> element, set the context relative path from the module as below. (in case when contextRelative is false)

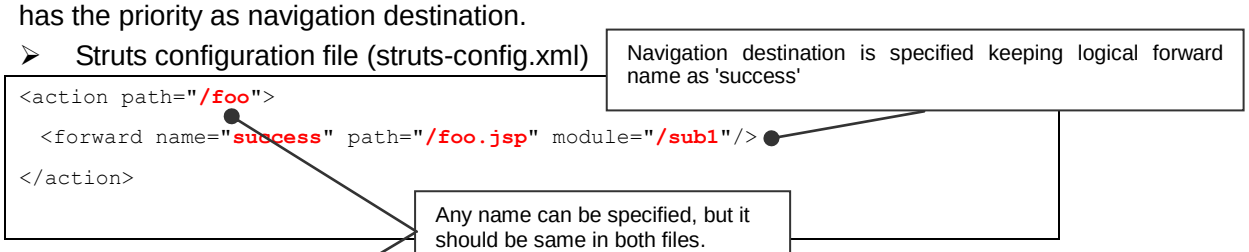
  ➢ Struts configuration file (struts-config.xml)



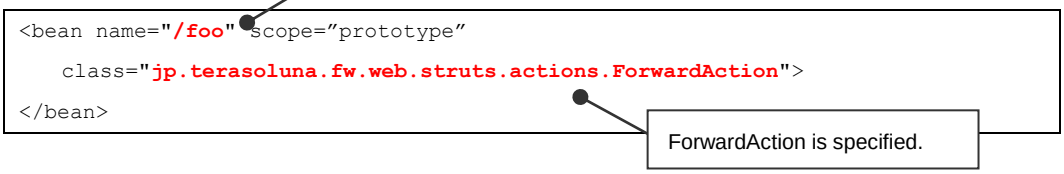
  ➢ Bean definition file

- When forward destination is to be specified by <forward> element, configure as below. Set navigation or redirect to the path extending a module that cannot be specified in context relative path that is set in above mentioned parameter element. When parameter attribute of <action> element is set, note that parameter attribute has the priority as navigation destination.

  ➢ Struts configuration file (struts-config.xml)



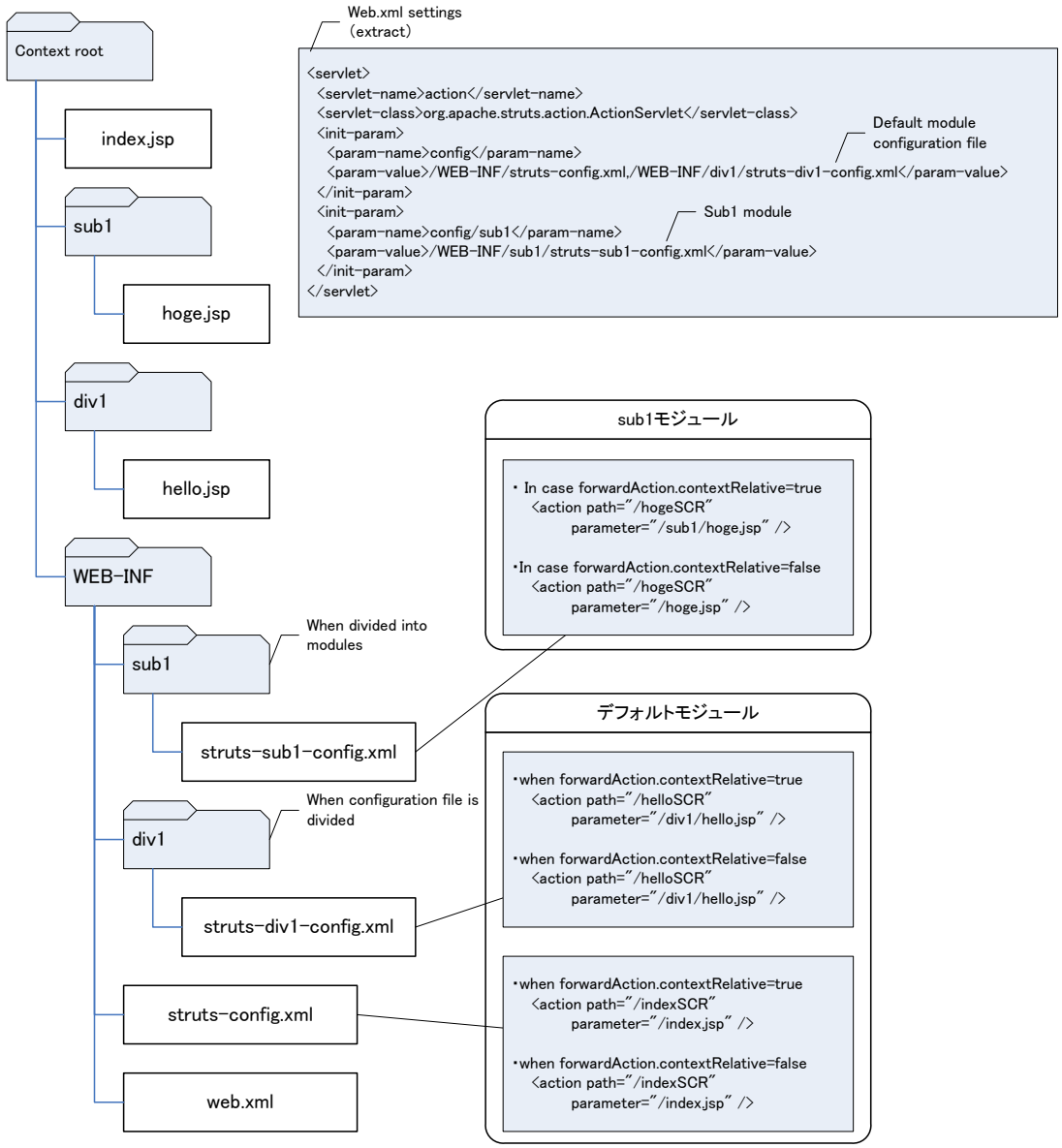
  ➢ Bean Definition File



- If parameter attribute of <action> element is used, property file can also be used to specify the path.
  If fowardAction.contextRelative is true, specify absolute path from context root. If it's false, specify relative path from the module. If this option is not set, the false is the default value.

- system.properties file

```
# contextRelative settings for ForwardAction
#  true  : Struts standard
#  false : Terasoluna mode
forwardAction.contextRelative=false
```



■ Classes

	Class Name	Overview
1	jp.terasoluna.fw.web.struts.actions.ForwardAction	Action class that forwards to specified destination.
2	jp.terasoluna.fw.web.struts.actions.ActionEx	The process executed by ActionEx#execute() is inherited by ForwardAction.

◆ Extension Points

None

■ Related Function(s)

- “WA-05 Denial of Direct Access To Extensions”
- “WE-01 Action Extensions”

■ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - “UC18 Forward”
    - /webapps/forward/*
    - /webapps/WEB-INF/forward/*
    - jp.terasoluna.thin.functionsample.actionex.web.action.ActionExAction.java etc.

- TERASOLUNA Server Framework for Java (Web version) tutorial
  - /webapps/WEB-INF/moduleContext.xml

■ Remarks

- None

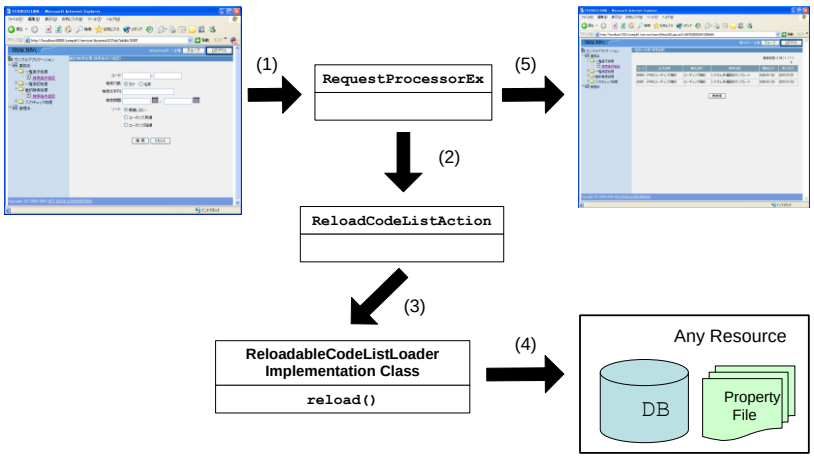
WE-04 Code List Reload Function

■ Overview

◆ Functional Overview

- Reloads code list data when ReloadCodeListAction is executed. Code list data can be refreshed by using this function.
- Code list data can be retrieved from DB or property file.
  However, in case of a web application, property files inside WAR/EAR files cannot be modified, and so reloading of code lists from property list is not possible.
  Please refer to “WB-05 Code list Function” for the ReloadableCodeListLoader interface implementation class.

◆ Schematic Diagram



◆ Description

(1) Request is sent from the browser.

(2) RequestProcessorEx calls ReloadCodeListAction. After completing the process in (3), ReloadCodeListAction writes log, calls the super class ActionEx method, and returns the ActionForward for the path specified in the parameter attribute of the <action> element in Struts configuration file (struts-config.xml)
Please refer to “WE-01 Action Extensions” for the details of ActionEx.

(3) ReloadCodeListAction class gets the ReloadableCodeListLoader implementation

class set in the codeListLoader property from the Spring configuration file.

(4) The reload() method of ReloadableCodeListLoader class got in (3) is called and code list is reloaded.

(5) Forward to ActionForward path. SC_NOT_FOUND (404) error is returned, if the forward path is not set.

■ Usage

◆ Coding Points

- Implementation of code list data reload function requires setting in Struts configuration file (struts-config.xml) and Bean definition file.
  - ➢ Struts Configuration File (struts-config.xml)

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

Specified with same name. Any name may be specified.

Specify forward path in the parameter attribute.

  - ➢ Bean Definition File

Specify ReloadCodeListAction.

```
<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>
  <property name="sql">
     <value>SELECT KEY, LABEL FROM CODE_LIST</value>
  </property>
</bean>
```

Specify the codelist to be updated by ReloadableCodeListLoader.

ReloadableCodeListLoader implementation class provided by TERASOLUNA

Specify Datasource from which code list information is to be read in the dataSource property. Set SQL statement to get the code list in sql. In this example, KEY is the id of code list and LABEL is the value of code list.

■ Classes

	Class Name	Overview
1	jp.terasoluna.fw.web.struts.actions.ReloadCodeListAction	It is the action class that forwards control to the next destination.
2	jp.terasoluna.fw.web.struts.actions.ActionEx	ReloadCodeListAction uses the processing in the super class ActionEx#execute().
3	jp.terasoluna.fw.web.codelist.ReloadableCodeListLoader	It is the interface that executes reloading of code lists. Please refer “WB-05 Code List Function” for the details.
4	jp.terasoluna.fw.web.codelist.DBCodeListLoader	It is the ReloadableCodeListLoader implementation class that executes initialization of code list information from the database. Please refer to “WB-05 Code List Function” for the details.

◆ Extension Points

Code list reload function can be customization by extending the ReloadableCode-ListLoader interface implementation class shown in the struts configuration file (struts-config.xml) of the Coding points mentioned above. Please refer to “WB-05 Code List Function” for the ReloadableCodeListLoader interface implementation class.

■ Related Functionalities

- “WB-05 Code List Function(s)”
- “WE-01 Action Extensions”

■ Example

- Sample covering all functions of TERASOLUNA Server Framework for Java (Web based)
  - “UC13 Code List”
    - /webapps/codelist/*
    - /webapps/WEB-INF/codelist/*
    - jp.terasoluna.thin.functionsample.codelist.*

■ Remarks

- None

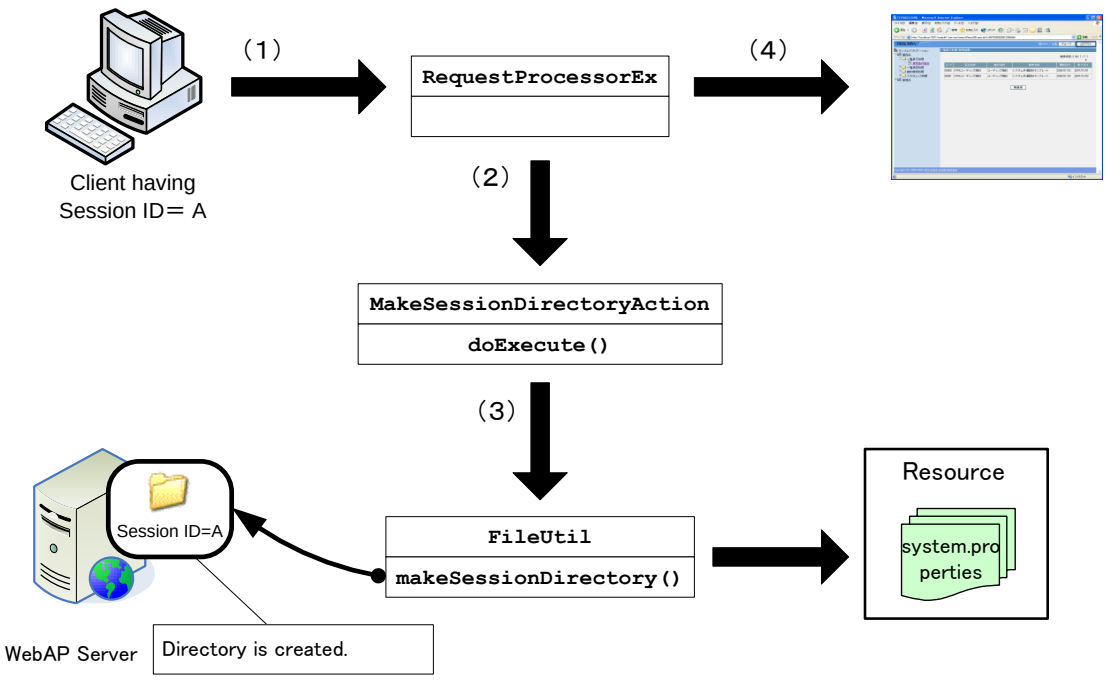
WE-05 Create Session Directory Function

■ Overview

◆ Functional Overview

- Creates a temporary directory (hereafter referred to as the session directory) to store PDF files etc, for every logon user on the server side.
- When the session is invalidated, callback for session listener is executed and session directory is deleted. Please refer to “WD-01 Session Directory Function” for the details.

◆ Schematic Diagram



◆ Description

(1) Request is sent by the user during logon.

(2) RequestProcessorEx starts MakeSessionDirectoryAction. After executing the process given in (3), MakeSessionDirectoryAction class writes log, calls the super class ActionEx method and returns the ActionForward as per the path specified in the parameter attribute of the <action> element in Struts configuration file (struts-config.xml).
Please refer to “WE-01 Action Extensions” for the ActionEx details.

(3) Create Session Directory.

(4) Executes FileUtil#makeSessionDirectory() process.
Please Refer to “CD-01 Utility Functions” for the details on FileUtil.

(5) Forward to path of ActionForward. SC_NOT_FOUND (404) error is returned when the forward destination is not set.

■ Usage

◆ Coding Points

- Set the base directory (under which the session directories are created) in the system properties configuration file (System.properties).
  - ➢ System configuration property file (system.properties)

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

Session directories are created under /tmp/sessions.

- Set the class provided by TERASOLUNA Server Framework for Java (Web version) to create the session directory in the action mapping.
  - ➢ Struts configuration file (struts-config.xml)

```
<action path="/makeSessionDir"
  scope="session"
  parameter="/foo.jsp">;
</action>
```

Any name may be specified, but it should be same in both files.

  - ➢ Bean definition file

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

■ Classes

	Class Name	Overview
1	jp.terasoluna.fw.util.FileUtil	A utility class to create, delete or get a file/directory. Please refer to “CD-01 Utility Functions” for the details.
2	jp.terasoluna.fw.web.struts.actions.MakeSessionDirectoryAction	Creates a session directory for every logon user.
3	jp.terasoluna.fw.web.struts.actions.ActionEx	ActionEx#execute() is overriden by MakeSessionDirectoryAction.

◆ Extension Points

None

■ Related Functionaly

- “CD-01 Utility Functions”
- “WD-01 Session Directory Function(s)”

■ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - “UC15 Session Directory”
    - /webapps/sessiondir/*
    - /webapps/WEB-INF/sessiondir/*
    - jp.terasoluna.thin.functionsample.sessiondir.*

■ Remarks

- None

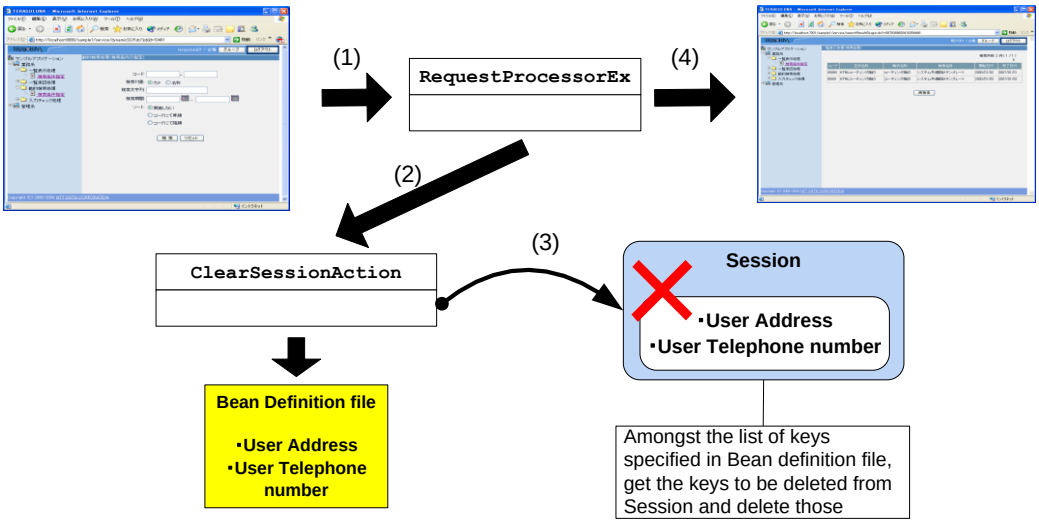
WE-06 Clear Session Function

■ Overview

◆ Functional Overview

- Deletes the specified properties from session.
  - ➢ Gets the list of keys specified in Bean Definition file, and clears the values in the session for those keys.
- Please Refer to “WE-07 Logoff Function” on clearing the complete current session.

◆ Schematic Diagram



◆ Description

(1) Request is sent by the browser.

(2) RequestProcessorEx starts ClearSessionAction. After executing the process given in (3), ClearSessionAction Class writes the log, calls the base class ActionEx’s method and returns the ActionForward as per the path specified in the parameter attribute of the <action> element in Struts configuration file (struts-config.xml) Please refer to “WE-01 Action Extensions” for the details of ActionEx.

(3) Using the list of keys specified in Bean Definition file, clear the values in session for those keys

(4) Forward to the path for ActionForward. When the forward destination is not set, SC_NOT_FOUND (404) error is returned.

■ Usage

◆ Coding Points

- Set the ClearSessionAction provided by TERASOLUNA Server Framework for Java (Web version) in the action mapping.
  - ➢ Struts configuration file (struts-config.xml)

```
<action path="/clearSession"
  scope="session"
  parameter="/foo.jsp">;
</action>
```

Any name may be specified, but it should be same in both files.

- The keys to be deleted from session are set in the clearSessionKeys properties of the Bean Definition file.
  - ➢ Bean definition file

```
<bean name="/clearSession" scope="prototype"
  class="jp.terasoluna.fw.web.struts.actions.ClearSessionAction">
  <property name="clearSessionKeys">
    <list>
      <value>userAddress</value>
      <value>userPhoneNo</value>
      <value>sampleSession</value>
    </list>
  </property>
</bean>
```

Values of keys to be deleted from session are defined by list properties.

■ Classes

	Class Name	Overview
1	jp.terasoluna.fw.web.struts.actions.ClearSessionAction	Deletes specified properties from session.
2	jp.terasoluna.fw.web.struts.actions.ActionEx	ActionEx#execute() is overridden by ClearSessionAction.

◆ Extension points

None.

■ Related Function(s)

- “WE-01 Action Extensions”
- “WE-07 Logoff Function”

■ Example

- Sample covering all functions on TERASOLUNA Server Framework for Java (Web version)
  - “UC19 Session clear”
    - /webapps/clearsession/*
    - /webapps/WEB-INF/clearsession/*
    - jp.terasoluna.thin.functionsample.clearsession.*

■ Remarks

- None

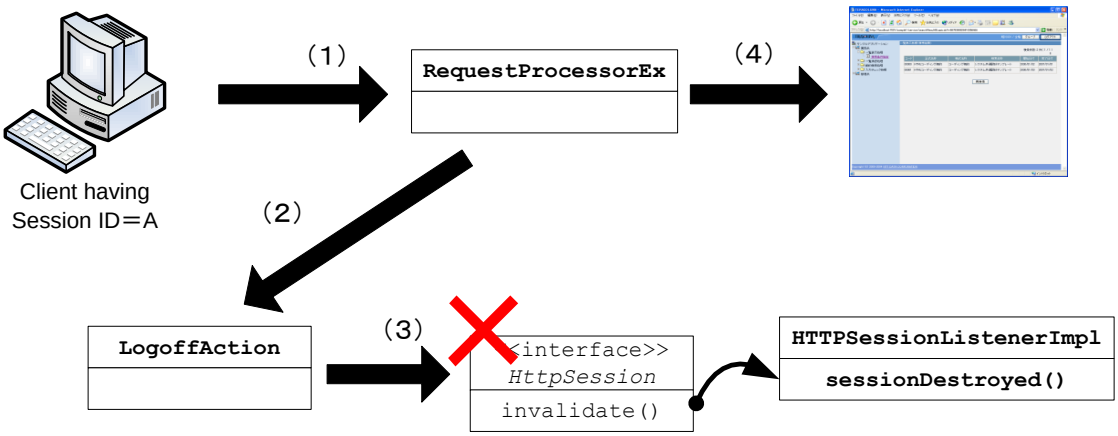
WE-07 Logoff Function

■ Overview

◆ Functional Overview

- A temporary directory is created (hereafter, referred to the Session Directory) for every logon user to store the PDF files etc. on the server side.
- When session is invalidated, callback of session listener is executed and session directory is deleted. Please refer to “WD-01 Session Directory Function(s)” for the details.

■ Schematic Diagram



◆ Description

(1) Request is sent by the user during logoff.

(2) RequestProcessorEx starts LogoffAction. After executing the process given in (3), LogoffAction class writes log, executes the base class ActionEx method and then returns the ActionForward as per the path specified in the parameter attribute of the <action> element in Struts configuration file (struts-config.xml)
Please refer to “WE-01 Action Extensions” for the ActionEx details.

(3) Execute invalidation process of session.
When session is invalidated, callback of HttpSessionListene-rImpl#sessionDestoroyed() method is executed.

(4) Forward to ActionForward path. When the forward destination is not set,

SC_NOT_FOUND (404) error is returned.

■ Usage

◆ Coding points

- Set LogoffAction provided by TERASOLUNA in the action mapping definition.
  - ➢ Struts Configuration File (struts-config.xml)

```
<action path="/logoff"
    scope="session"
    parameter="/foo.jsp">;
</action>
```

Any name may be specified, but it should be same in both files.

  - ➢ Bean Definition File

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

Logoff class provided by TERASOLUNA.

■ Classes

	Class Name	Overview
1	jp.terasoluna.fw.web.struts.actions.LogoffAction	The action class that executes Logoff process.
2	jp.terasoluna.fw.web.struts.actions.ActionEx	ActionEx#execute () is overridden by LogoffAction.
3	jp.terasoluna.fw.web.HttpSessionListenerImpl	A class that implements the HttpSessionListener interface and deletes/invalidates the session.

◆ Extension points

None

■ Related Function(s)

- “WD-01 Session Directory Function(s)”
- “WE-01 Action Extensions”

■ Example

- Sample covering all functions in the TERASOLUNA Server Framework for Java (Web version)
  - "UC00 Common Functions"
    - /webapps/*
    - /webapps/WEB-INF/*
    - jp.terasoluna.thin.functionsample.common.*

- TERASOLUNA Server Framework for Java (Web version) tutorial
  - 2.4 Logon
  - /webapps/WEB-INF/moduleContext.xml

■ Remarks

- None

WF-01 Input Validation extensions

■ Overview

◆ Functional Overview

- Typical input validation rules required in an application are provided using commons-validator.

◆ List of Rules Provided

The input validation rules provided by TERASOLUNA Server Framework for Java (Web version) are given below.
※ The rules provided by Struts are not given.

Rule Name	Overview	Server side	Client side
alphaNumericString	Single byte alphanumeric string check	○	○
hankakuKanaString	Single byte Kana string check	○	○
hankakuString	Single byte string check	○	○
zenkakuString	Double byte string check	○	○
zenkakuKanaString	Double byte Kana string check	○	○
capAlphaNumericString	Capital alpha numeric character string check	○	○
Number	Numeric 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	Multi field correlation check	○	×

■ Usage

◆ Coding points

Method to initialize Input validation extension is shown below. Please refer to subsequent sections for usage of each rule.

- Struts configuration file

```
<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
  <set-property
    property="pathnames"
    value="/WEB-INF/validator-rules.xml,/WEB-INF/validator-rules-
ex.xml,/WEB-INF/validation.xml"/>
  <set-property
    property="stopOnFirstError" value="false"/>
</plug-in>
```

Rule file of Extension Input Check Functionality. The file provided by TERASOLUNA Server Framework for Java (Web version) is used without modifying.

■ Input Validation Rule Description

◆ alphaNumericString

Checks that characters entered are only Single byte alpha numeric characters.

- Example of input validation configuration file (validation.xml)

```
……
  <form name="/validate">
    <field property="userName"
      depends="alphaNumericString">
      <arg key="label.userName" position="0" />
    </field>
  </form>
……
```

"Enter single byte alpha numeric characters in errors.alphaNumericString= {0}" is the default error message.

◆ hankakuKanaString

Checks that characters entered are only Single byte Kana characters.

- Example of input validation configuration file (validation.xml)

```
.....
  <form name="/validate">
    <field property="userName"
      depends="hankakuKanaString">
      <arg key="label.userName" position="0" />
    </field>
  </form>
.....
```

"Enter single byte Kana characters in errors.alphaNumericString= {0}" is the default error message.

Following characters fall under Single byte kana characters.

➢ ｱｲｳｴｵｧｨｩｪｫｶｷｸｹｺｻｼｽｾｿﾀﾁﾂｯﾃﾄﾅﾆﾇﾈﾉﾊﾋﾌﾍﾎﾏﾐﾑﾒﾓﾔﾕﾖｬｭｮﾗﾘﾙﾚﾛﾜﾝﾞﾟｰ･、。「」

Characters corresponding to Single byte Kana characters can be changed by registering in property file.

- Property file

```
validation.hankaku.kana.list=ｱｲｳｴｵｧｨｩｪｫｶｷｸｹｺｻｼｽｾｿﾀﾁﾂｯﾃﾄﾅﾆﾇﾈﾉﾊﾋﾌﾍﾎﾏﾐﾑﾒﾓﾔﾕﾖｬｭｮﾗﾘﾙﾚﾛﾜﾝﾞﾟ
```

◆ hankakuString

Checks that characters entered are only Single byte characters.

- Example of input validation configuration file (validation.xml)

```
.....
  <form name="/validate">
    <field property="userName"
      depends="hankakuString">
      <arg key="label.userName" position="0" />
    </field>
  </form>
.....
```

"Enter single byte characters in errors.hankakuString = {0}" is the default error message.

Determines whether Single byte characters excluding 「＼¢ £§¨―°±´¶×÷」, correspond to code from '¥u0000' to '¥u00ff' of UNICODE or hankakuKanaString rule. Please refer to the description of hankakuKanaString rule for the details.

◆ zenkakuString

Checks that characters entered are only Double byte characters.

- Example of input validation configuration file (validation.xml)

```
.....
  <form name="/validate">
    <field property="userName"
      depends="zenkakuString">
      <arg key="label.userName" position="0" />
    </field>
  </form>
.....
```

"Enter full width characters in errors.zenkakuString = {0}" is the default error message.

Double byte characters are determined by applyling the reverse rule of the hankakuString rule. Please refer to the description of hankakuString rule for details.

◆ zenkakuKanaString

Checks whether the characters entered are only double byte Kana characters.

- Example of input validation configuration file (validation.xml)

```
.....
  <form name="/validate">
    <field property="userName"
      depends="zenkakuKanaString">
      <arg key="label.userName" position="0" />
    </field>
  </form>
.....
```

"Enter double byte Kana characters in errors.zenkakuKanaString = {0}" is the default error message.

By default, following characters fall under Double byte Kana characters.

- アイウヴエオァィゥェォカキクケコヵヶガギグゲゴサシスセソザジズゼゾタチツテトダヂヅデドナニヌネノハヒフヘホバビブベボパピプペポマミムメモヤユヨャュョラリルレロワヮヰヱヲッンー

Characters corresponding to double byte Kana characters can be configured in property file.

- Property file

```
validation.zenkaku.kana.list=アイウエオカキクケコサシスセソ……
```

◆ capAlphaNumericString

Checks whether the characters entered are only Single byte capital alphanumeric characters.

- Example of input validation configuration file (validation.xml)

```
.....
  <form name="/validate">
    <field property="userName"
      depends="capAlphaNumericString">
      <arg key="label.userName" position="0" />
    </field>
  </form>
.....
```

"Enter English capital letters or numeric characters in errors.capAlphaNumericString = {0}" is the default error message.

◆ number

Checks whether the entered character string can be changed to numeric value.

- <var> element that should be set in input validation configuration file (validation.xml)

var-name	var-value	Mandatory	Overview
integerLength	Integer digits	false	Specifies number of integer digits. When isAccordedInteger is not specified, validates that length is less than or equal to specified digits. Validation is not done for non-numeric values.
scale	Fraction digits	false	Specifies number of fraction digits. When isAccordedScale is not specified, validates that length is less than or equal to the required digits. Validation is not done for non-numeric values.
isAccordedInteger	Integer length check	false	If true, check that number of digits in the integer matches exactly. If omitted (or other than true), checks that number of digits is less than or equal to the required length
isAccordedScale	decimal length check	false	If true, checks that number of decimals match exactly. If omitted (or other than true), checks that number of decimals is less than or equal to the required decimals

- Example of input validation configuration file (validation.xml)

```
.....
  <form name="/validate">
    <field property="userName"
      depends="number">
      <arg key="label.userName" position="0" />
      <arg key="${var:integerLength}" position="1" resource="false"/>
      <arg key="${var:scale}" position="2" resource="false"/>
        <var>
          <var-name>integerLength</var-name>
          <var-value>3</var-value>
        </var>
        <var>
          <var-name>scale</var-name>
          <var-value>2</var-value>
        </var>
        <var>
          <var-name>isAccordedInteger</var-name>
          <var-value>false</var-value>
        </var>
        <var>
          <var-name>isAccordedScale</var-name>
          <var-value>true</var-value>
        </var>
    </field>
  </form>
.....
```

"Enter integer upto {1} digit and decimal upto {2} digits in errors.number= {0}" is the default error message.

In this setting example, numeric value check is executed according to "ineger upto 3 digits, decimal upto 2 places" rule.

◆ numericString

Checks that the characters entered are only numbers.

- Example of input validation configuration file (validation.xml)

```
.....
  <form name="/validate">
    <field property="userName"
      depends="numericString">
      <arg key="label.userName" position="0" />
    </field>
  </form>
.....
```

"Enter only numeric value in errors.numericString = {0}" is the default error message.

◆ prohibited

Checks whether prohibited characters are included in the entered character string.

- <var> element that should be set in input validation configuration file (validation.xml)

var-name	var-value	Mandatory	Outline
chars	Input prohibited charactors	false	If any of the entered characters correspond to prohibited characters, an error occurs. When omitted, check is not executed.

- Example of input validation configuration file (validation.xml)

```
.....
  <form name="/validate">
    <field property="userName"
      depends="prohibited">
      <arg key="label.userName" position="0" />
      <arg key="${var:chars}" position="1" resource="false" />
      <var>
        <var-name>chars</var-name>
        <var-value>!"#$%&'()</var-value>
      </var>
    </field>
  </form>
.....
```

"Prohibited character {1} is included in errors.prohibited={0}" is the default error message.

Input check error occurs when "!"#$%&'()" are included in the entered string.

- Remarks

  Client-side validation is not supported in this rule.

◆ stringLength

Checks the digits in entered string.

- <var> element that should be set in input validation configuration file (validation.xml)

var-name	var-value	Mandatory	Overview
stringLength	Digits in entered string	false	An error occurs when the digits in entered string do not match with the specified number of digits. When omitted, check is not executed.

- Example of input validation configuration file (validation.xml)

```
.....
  <form name="/validate">
    <field property="userName"
      depends="stringLength">
      <arg key="label.userName" position="0" />
      <arg key="${var:stringLength}" position="1" resource="false" />
      <var>
        <var-name>stringLength</var-name>
        <var-value>5</var-value>
      </var>
    </field>
  </form>
.....
```

"Set errors.stringLength= {0} to {1} character" is the default message.

Check that entered string consists of 5 digits.

◆ byteLength

Checks the number of bytes of entered string.

- <var> element that should be set in input validation configuration file (validation.xml)

var-name	var-value	Mandatory	Overview
byteLength	Entered character string bytes	false	An error occurs when the number of bytes in the entered string does not match with the specified number. When omitted, check is not executed.
encoding	Character code at the time of bytes conversion.	false	Character code used in conversion of entered character string to byte array. When omitted, VM default character code is used.

- Example of input validation configuration file (validation.xml)

```
.....
  <form name="/validate">
    <field property="userName"
      depends="byteLength">
      <arg key="label.userName" position="0" />
      <arg key="5" position="1" resource="false" />
      <var>
        <var-name>byteLength</var-name>
        <var-value>10</var-value>
      </var>
      <var>
        <var-name>encoding</var-name>
        <var-value>Windows-31J</var-value>
      </var>
    </field>
  </form>
.....
```

"Set errors.byteLength ={0} to {1} character" is the default message.

Checks whether the entered string is of 10 bytes.

- Remarks

  Client-side validation is not supported in this rule.

◆ byteRange

Checks whether the entered string is within the specified byte range.

- <var> element that should be set in input validation configuration file (validation.xml)

var-name	var-value	Mandatory	Outline
maxByte	Maximum number of bytes	false	Maximum number of bytes for validating the length of entered character string. When omitted, int type maximum value is used.
minByte	Minimum number of bytes	false	Minimum number of bytes for validating the length of entered character string. When omitted, 0 is used.
encoding	Character code used during numer of bytes conversion	false	Character code used in conversion of entered string to byte array. When omitted, VM default character code is used.

- Example of input validation configuration file (validation.xml)

```
.....
  <form name="/validate">
    <field property="userName"
      depends="byteRange">
      <arg key="label.userName" position="0" />
      <arg key="${var:minByte}" position="1" resource="false" />
      <arg key="${var:maxByte}" position="2" resource="false" />
      <var>
        <var-name>maxByte</var-name>
        <var-value>16</var-value>
      </var>
      <var>
        <var-name>minByte</var-name>
        <var-value>8</var-value>
      </var>
      <var>
        <var-name>encoding</var-name>
        <var-value>Windows-31J</var-value>
      </var>
    </field>
  </form>
.....
```

"Enter in the range from {1} to {2} in {0}" is the default message.

Checks that the entered string is within 8 to 16 bytes.

- Remarks

  Client-side validation is not supported in this rule.

◆ dateRange

Checks whether the entered string can be converted to the specified date format, and is in the specified date range.

- <var> element that should be set in input validation configuration file (validation.xml)

var-name	var-value	Mandatory	Outline
startDate	Start date	false	Date, which is the threshold value at the starting of date range. It should conform to date format specified by datePattern or datePatternStrict.
endDate	End date	false	Date, which is the threshold value at the end of date range. It should conform to date format specified by datePattern or datePatternStrict.
datePattern	Date pattern	false	String showing date pattern. It follows the Date format rule.
datePatternStrict	Date pattern	false	Whether the date pattern is checked strictly. In case the date pattern is yyyy/MM/dd, 2001/1/1 is an error. When datePattern is specified, the format specified by datePattern has higher priority.

- Example of input validation configuration file (validation.xml)

```
.....
  <form name="/validate">
    <field property="userName"
      depends="dateRange">
      <arg key="label.userName" position="0" />
      <arg key="${var:startDate}" position="1" resource="false" />
      <arg key="${var:endDate}" position="2" resource="false" />
      <var>
        <var-name>startDate</var-name>
        <var-value>2000/1/1</var-value>
      </var>
      <var>
        <var-name>endDate</var-name>
        <var-value>2010/12/31</var-value>
      </var>
      <var>
        <var-name>datePattern</var-name>
        <var-value>yyyy/MM/dd</var-value>
      </var>
    </field>
  </form>
.....
```

"Enter in the range from {1} to {2} in errors.dateRange= {0}" is the default message.

Checks that the entered date is in the 2000/1/1~2010/12/31 range.

◆ multiFieldCheck

Executes multi-field correlation check e.g where the entered string depends on value of other property of ActionForm. Business developer creates the class that executes multi-field check by implementing MultiFieldValidator interface provided by the framework.

- <var> element that should be set in input validation configuration file (validation.xml)

var-name	var-value	Mandatory	Overview
fields	Other property names required for validation.	false	In case of multiple fields, specify the field names delimited by comma.
multiFieldValidator	MultiFieldValidator implementation class name	false	Name of MultiFieldValidator implementation class that executes multifield correlation check.

- Example of input validation configuration file (validation.xml)

```
<form name="/validate">
  <field property="value" depends="multiField">
    <msg key="errors.multiField"
         name="multiField"/>
    <arg key="label.value" position="0" />
    <arg key="label.value1" position="1" />
    <arg key="label.value2" position="2" />
    <var>
      <var-name>fields</var-name>
      <var-value>value1,value2</var-value>
    </var>
    <var>
      <var-name>multiFieldValidator</var-name>
      <var-value>sample.SampleMultiFieldValidator</var-value>
    </var>
  </field>
</form>
```

Default error message does not exist in this rule, hence message should be set.

Other property names required for validation.

Name of the class that implements multi-field check

- Example of MultiFieldValidator implementation class

```
public class SampleMultiFieldValidator implements MultiFieldValidator {
  public boolean validate(String value, String[] fields) {

    int value0 = Integer.parseInt(value);
    int value1 = Integer.parseInt(fields[0]);
    int value2 = Integer.parseInt(fields[1]);
    return (value1 <= value0 && value2 >= value0);
  }
}
```

Value to be validated and value of other fields required for validation are passed to first and third arguments respectively

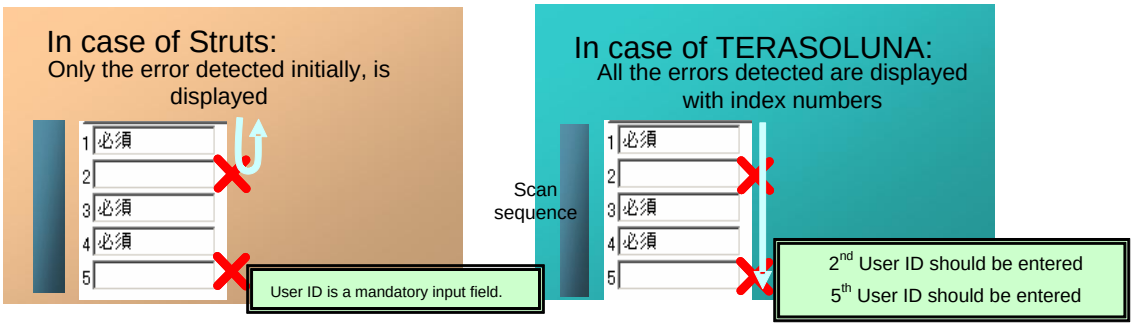
Validation result is Boolean type. In case of error, false is returned.

- Remarks

  Client-side validation is not supported in this rule.

◆ Input validation of Array / Collection properties

Struts and commons-validator supports the validation of repetitive items (arrays/collections). Since validation ends when an error occurs, the framework extends the validation process so that all the elements are checked even though an error occurs during the input validation of repetetive items.

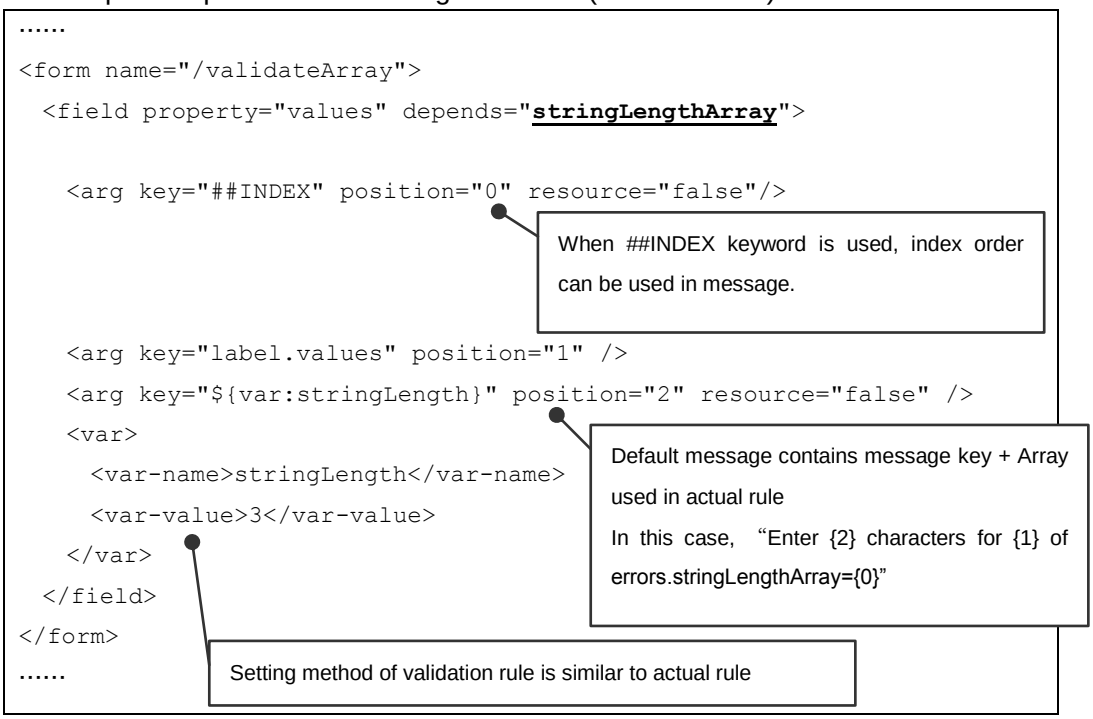


Rules for checking Array / Collection property are as follows.

Rules				
requiredArray	requiredIfArray	validWhenArray	minLengthArray	maxLengthArray
maskArray	byteArray	shortArray	integerArray	longArray
floatArray	doubleArray	dateArray	intRangeArray	doubleRangeArray
floatRangeArray	creditCardArray	emailArray	urlArray	alphaNumericStringArray
hankakuKanaStringArray	hankakuStringArray	zenkakuStringArray	zenkakuKanaStringArray	capAlphaNumericStringArray
numberArray	numericStringArray	prohibitedArray	stringLengthArray	dateRangeArray
byteLengthArray	byteRangeArray			

※ The validation logic and configuration for each rule is similar to the corresponding rule with "Array". Please refer to Struts documentation for the details on each rule.

- Example of input validation configuration file (validation.xml)



※ It is possible that property name specified in property attribute describes the nested property name according to the JXPathIndexedBeanWrapperImpl specifications.

◆ Important Points

- In order to check a string that contains only half width space.
  Except required check, in the validation rules provided by Struts as well as TERASOLUNA Server Framework for Java (Web version), if a string that comprises of half-width space only is passed as input value, it will not result in an error. To produce an error, either the rule should be used in combination with required check or a new rule should be added.

- In case of Date Input
  In date check, there is a datePattern as well as datePatternStrict to specify the pattern of date input. However, in this check a close check of specified pattern is not executed. In reality, there are cases in which a value which is wrong as a date will also pass the check. This is because parse method of SimpleDateFormat is used in this method. (This is registered as a bug in Java Bug Database. Bug ID: 5055568). In order to closely check a specified pattern use regular expressions

- In case of numericString, alphaNumericString, capAlphaNumeric check methods
  In above check methods, LF Code 0x0a is not detected. While 0x0d(CR), 0x0d0a(CRLF) can be detected. Due to this, if a LF character is included in input

parameter then it will pass the validation and an unintended value will be passed to the business logic which might affect the output of business logic execution. Usually all browsers confirming to RFC, encode newline code to 0x0d0a(CRLF). However, care is required since some ill-intended attacker might can send request data containing 0x0a(LF).

Further this also occurs if the validateNumericString(), validateAlphaNumericString(), validateCapAlphaNumericString() methods of FieldChecksEx are directly called.

For a closer check, either regular expression should be used or check should be performed in business logic.

If regular expression is used, the the examples of patterns are as below:

- numericString check → ” ^ [0-9]*(?!¥n)$”
- alphaNumericString check → ” ^ [0-9a-zA-Z]*(?!¥n)$”
- capAlphaNumericString check → ” ^ [0-9A-Z]*(?!¥n)$”

■ Example

- Sample covering all functions on the TERASOLUNA framework (Web based)
  - “UC20 Extension Input validation”
    - /webapps/validation/*
    - /webapps/WEB-INF/validation/*
    - jp.terasoluna.thin.functionsample.validation.*

- TERASOLUNA Tutorial framework (Web based)
  - “2.8.1 Single item check”
  - “2.8.2 Correlation check”
  - /webapps/WEB-INF/validation.xml

■ Remarks

- None

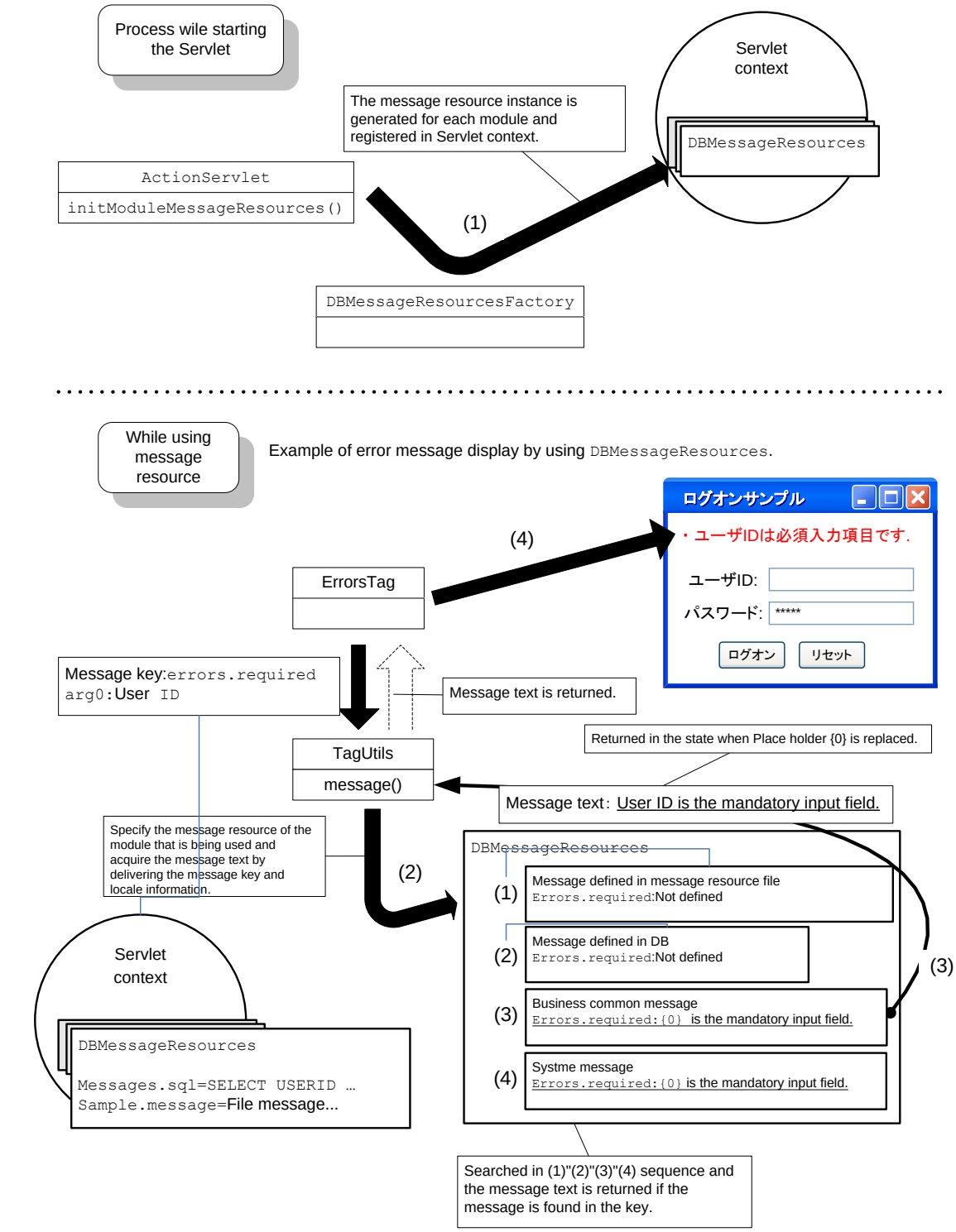
WG-01 Message Resource Extensions

■ Overview

◆ Functional Overview

- Extends the message resource of Struts (which requires a separate file for each module) and provides a mechanism to specify default messages for common business messages, system errors (that are not module-specific), using TERASO-LUNA Server Framework for Java (Web version).
- Unlike Struts, a priority can be specified for the messages

◆ Schematic Diagram



◆ Description

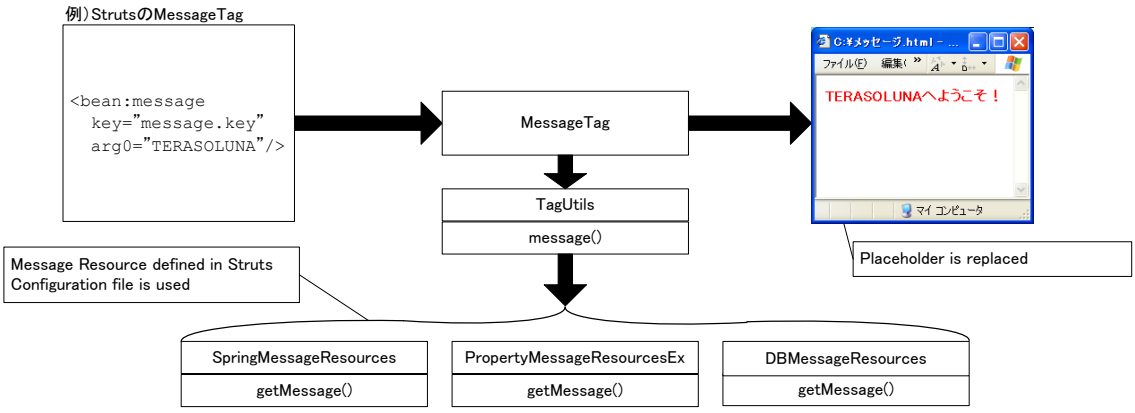
(1) While starting the Servlet, an instance of message resource is created by using the message resource factory class and stored in the servlet context. (Several message resources can be stored in 1 module.)
(2) When error occurs in business logic, a message can be displayed by fetching the message from the message resource using TagUtil.
(3) Priority is considered for the message resources and the message from the message resource with highest priority is used.
(4) The obtained message is displayed on the screen.

■ Usage

◆ Coding Points

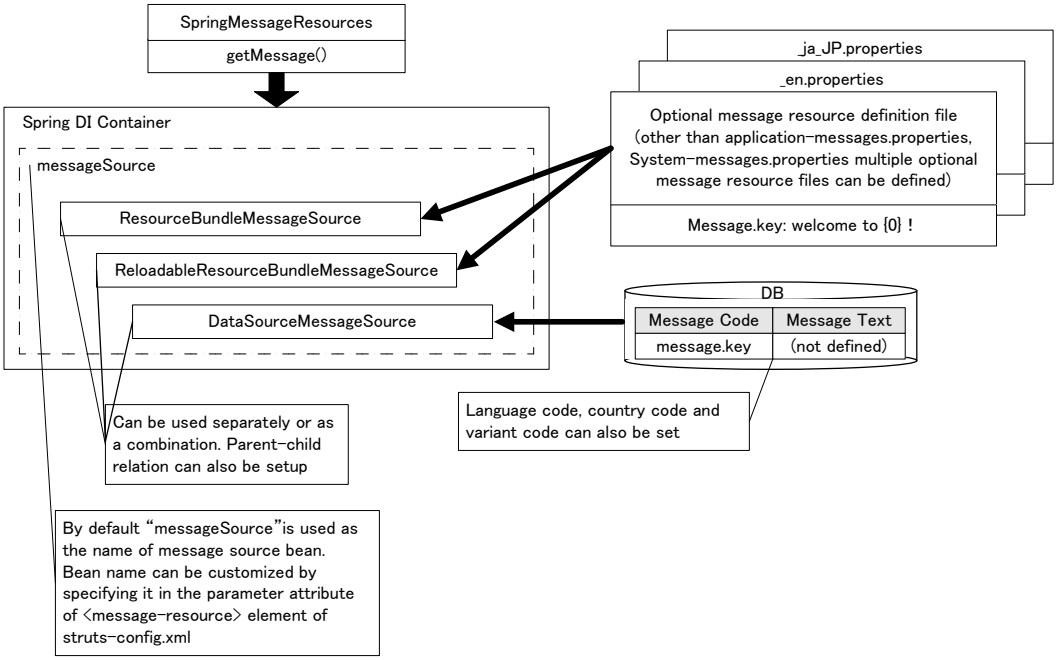
The two types of message resources provided by message resource extensions are as given below.

- Spring Message Resource
- When defined by using the messageSource of Spring
- Extended property message resource
  When defined on the basis of property file
- DB message resource
  When defined in DB



Priority of message resource definition (In case of Spring message resource)

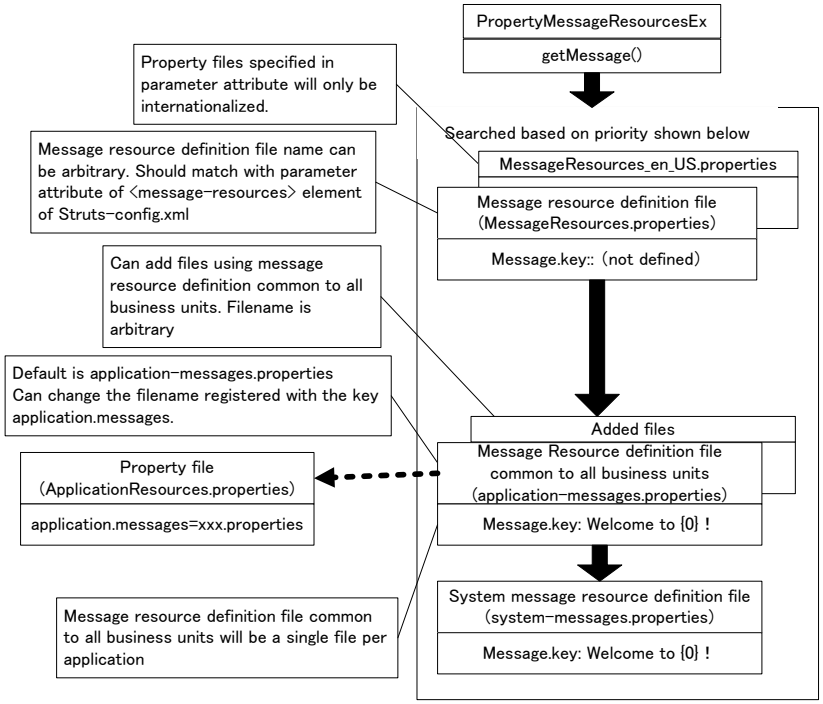
It corresponds to the settings of MessageSource.



- Priority of message resource definition (In case of Extended property message resource)

  Message resource can be defined at following locations. The priority is as follows.

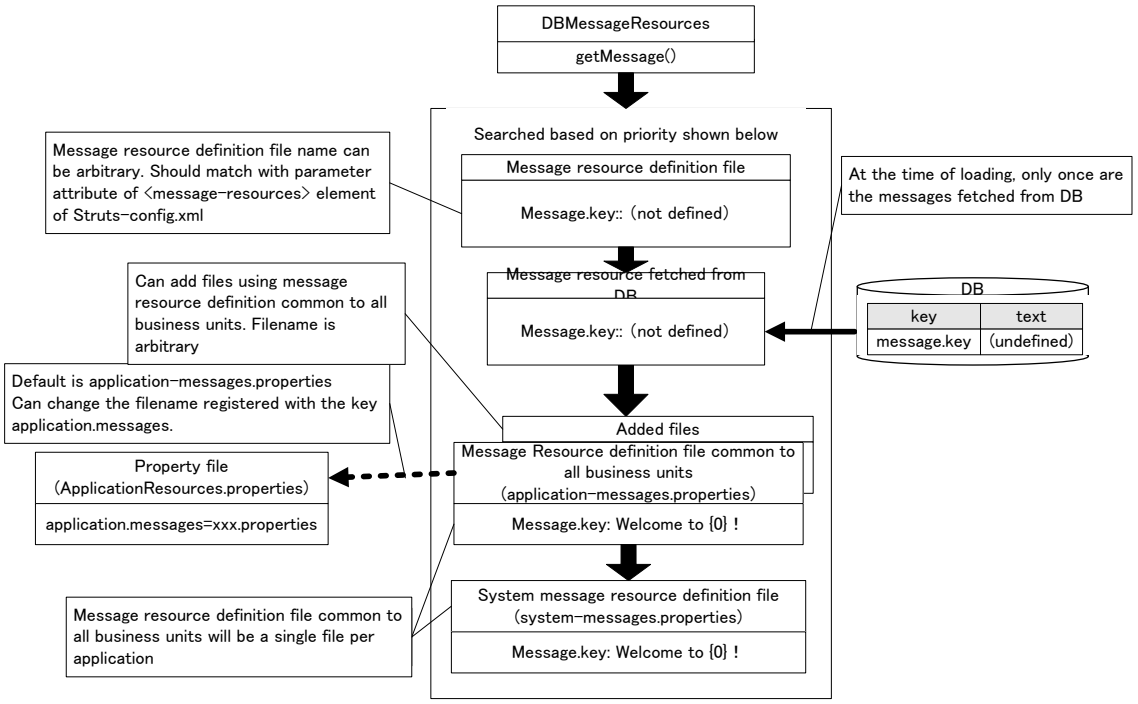
1. Struts standard Message resource definition file (MessageResources.properteis)
2. Business common message resource definition file (application-messages.properties)
3. System message resource definition file (system-messages.properties)



- Priority of message resource definition (In case of DB message resource)

  Message resource can be defined at following locations. The priority is as follows.

  1. Struts standard Message resource definition file (MessageResources.properteis)
  2. DB message resource definition
  3. Business common message resource definition file (application-messages.properties)
  4. System message resource definition file (system-messages.properties)



- Message resource definition file of Struts standards (MessageResources.properties)
  The message resource of property file format is used for Struts standard message resource definition.

  Any name other than **ApplicationResources.properties** may be assigned to the message resource definition file. PropertyUtil (“CD-01 Utility functions”) uses this property file name
  Generally, MessageResources.properties is used in the file name of message resource definition file.

  When extended property message resource (1) is used, standard message resource definition file of Struts can be defined as global.
  However, the keys that are not in this message resource are searched from the business common message resource definition file (application-messages.properties) and system message resource definition file (system-messages.properties), and note that these keys are not internationalized.

- DB message resource definition
  The message resource can be acquired from DB while using DB message resource.

  Class implementing MessageResourcesDAO interface is used for accessing DB. The implementation class MessageResourcesDaoImpl is provided, and the settings using the Spring framework are as follows:
  The following three points should be set for using MessageResourcesDaoImpl.
  1. Data source settings
  2. Definition of SQL that obtains the data
  3. Declaration of using MessageResourcesDaoImpl

  Among them 1 is written in dbMessageResources.xml and, 2 and 3 are written in system setting property file (system.properties).

  ➢ Bean definition file (dbMessageResources.xml)

```
<bean id="dataSource"
      class="org.springframework.jndi.JndiObjectFactoryBean">
  <property name="jndiName">
    <value>jdbc/sample</value>
  </property>
</bean>
```

JndiObjectFactoryBean is specified.

JNDI name is specified.

  dbMessageResources.xml is deployed at the class path location.
  In addition, dbMessageResources.xml is the Bean definition file. Separate Bean

definition file is used to control the business logic where only the data source definition is described.
File name dbMessageResources.xml is fixed.

- System setting property file (system.properties)

```
messages.sql=SELECT MESSAGE_KEY, MESSAGE_VALUE FROM MESSAGES
```

SQL that acquires message resource is specified.

```
messages.dao=jp.terasoluna.fw.web.struts.action.MessageResourcesDaoImpl
```

Class implementing MessageResourcesDAO is specified with Fully qualified name

In MessageResourcesDaoImpl, while defining the message resource in DB, the SQL statement that acquires two columns is defined. At that time, value in first column is treated as message key and value in second column is treated as message text. There are no specific restrictions for DB column name and table name.

- Business common message resource definition file (application-messages.properties)
  In Business common message resource definition file, the message resource shared by all modules classified by using Module classification function of Struts, is defined.

  By default, the application-messages.properties is used for the file name, but if any other name is specified in the system setting property file (system.properties), the file is used with application.messages as a key.

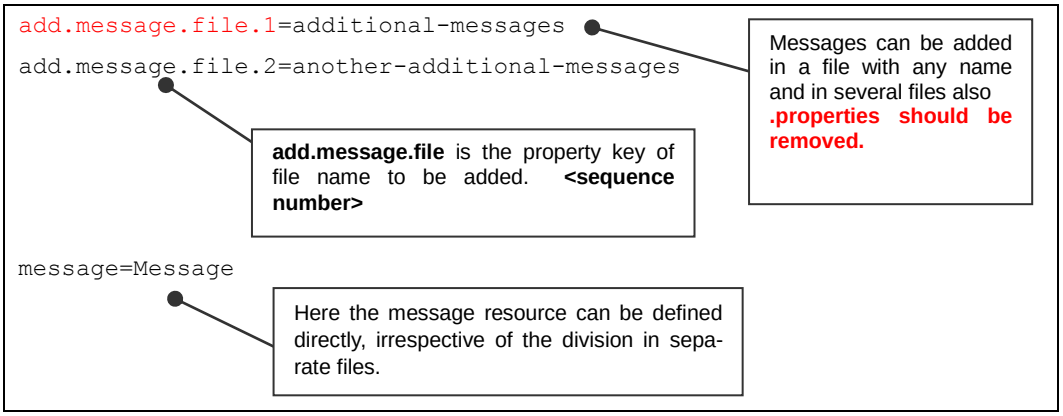
  - System setting property file (system.properties)

```
application.messages=another-application-messages
```

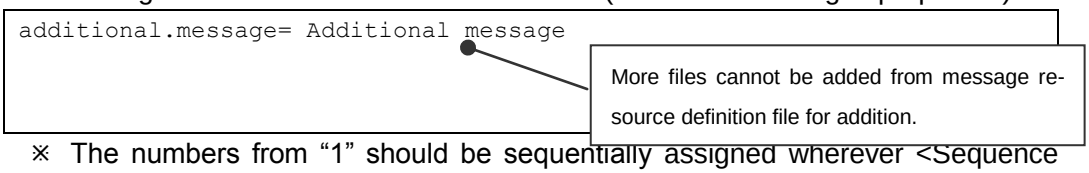
.properties should be removed.

In addition, business common message resource definition file can be divided in separate files for adding the messages. Settings in such case are given below.

  - Business common message resource definition file (application-messages.properties)



➢ Message resource definition file for addition (additional-messages.properties)



※ The numbers from “1” should be sequentially assigned wherever <Sequence number> is mentioned.
If any number is missing, all the numbers after that number become invalid.

- System message resource definition file (system-messages.properties)
  In the System message resource definition file (system-messages.properties), the message resource shared by all modules (as per module function of Struts) is defined. System message resource is defined as a default value for the system error message that is output by the framework.
  As a general rule, the business developer need not modify it.

  To change the value of system error message based on business requirements, system message resource definition file (system-messages.properties) need not be modified. It should be registered with the same key in business common message resource definition file (application-messages.properties) etc.

  Also, unlike business common message resource definition file (application-messages.properties), the name of the file system-messages.properties is fixed.

- Setting method of the message resource provided by Extended message resource
  Two types of message resource are provided in extended message resource function(s). Their respective setting method is given below.

1. In case of extended property message resource

➢ Struts configuration file (struts-config.xml)

```
<struts-config>
  ……
  <message-resources
  factory="jp.terasoluna.fw.web.struts.action.PropertyMessageResourcesExFactory"
    parameter="MessageResources" />
  ……
</struts-config>
```

PropertyMessageResourcesExFactory is specified.

Message resource definition file of Struts standards is specified. Any name can be specified. .properties may not be attached.

2. In case of DBmessage resource

➢ Struts configuration file (struts-config.xml)

```
<struts-config>
  ……

  <message-resources
    factory="jp.terasoluna.fw.web.struts.action.DBMessageResourcesFactory"
    parameter="MessageResources" />
  ……
</struts-config>
```

DBMessageResourcesFactory is specified.

Message resource definition file of Struts standards is specified. Any name can be specified. .properties may not be attached.

➢ System setting property file (system.properties)

SQL is defined that acquires the stored result where message key and message text from DB is stored by using messages.sql key.

```
messages.sql=SELECT MESSAGE_KEY, MESSAGE_VALUE FROM MESSAGES
messages.dao=jp.terasoluna.fw.web.struts.action.MessageResourcesDaoImpl
```

Implementation class that acquires the message resource from DB by using messages.dao key is specified.

※ MessageResourcesDaoImpl is being provided for implementation.

◆ Extension points

- MessageResourcesDAO interface is provided to get message resources from DB, and encapsulating the DB access In Spring, to access DB without dependence, this interface should be implemented and used instead of the MessageResourcesDaoImpl provided by TERASOLUNA Server Framework for Java (Web version). However, the “Data source setting” method used for accessing the data source should be determined in implementation class.

■ Classes

	Class Name	Overview
1	jp.terasoluna.fw.web.struts.action.PropertyMessageResourcesEx	Message resource class that inherits the PropertyMessageResources of Struts
2	jp.terasoluna.fw.web.struts.action. PropertyMessageResourcesExFactory	Factory class of PropertyMessageResourcesEx. In the configuration portion of message resource of Struts configuration file (struts-config.xml), factory class is specified and not message resource class.
3	jp.terasoluna.fw.web.struts.action.DBMessageResources	Message resource class that can get the messages common in the module set in DB.
4	jp.terasoluna.fw.web.struts.action.DBMessageResourcesFactory	Factory class of DBMessageResources. In the configuration portion of message resource of Struts configuration file (struts-config.xml), factory class is specified and not message resource class.
5	jp.terasoluna.fw.web.struts.action.MessageResourcesDAO	Interface for accessing DB by DBMessageResources.
6	jp.terasoluna.fw.web.struts.action.MessageResourcesDaoImpl	Implementation classs for accessing DB by DBMessageResources that implements MessageResourcesDAO
7	jp.terasoluna.fw.web.struts.action. GlobalMessageResources	Class that retains the business common / system message resource.

■ Related Function(s)

- “CD-01 Utility Functions”

■ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - "UC21 Message resource extensions"
    - /webapps/messageex/*
    - /webapps/WEB-INF/messageex/*
    - jp.terasoluna.thin.functionsample.messageex.*
    - /webapps/messageex2/*
    - /webapps/WEB-INF/messageex2/*
    - jp.terasoluna.thin.functionsample.messageex2.*

- TERASOLUNA Server Framework for Java (Web version) tutorial
  - webapps/WEB-INF/struts-config.xml

■ Remarks

- None

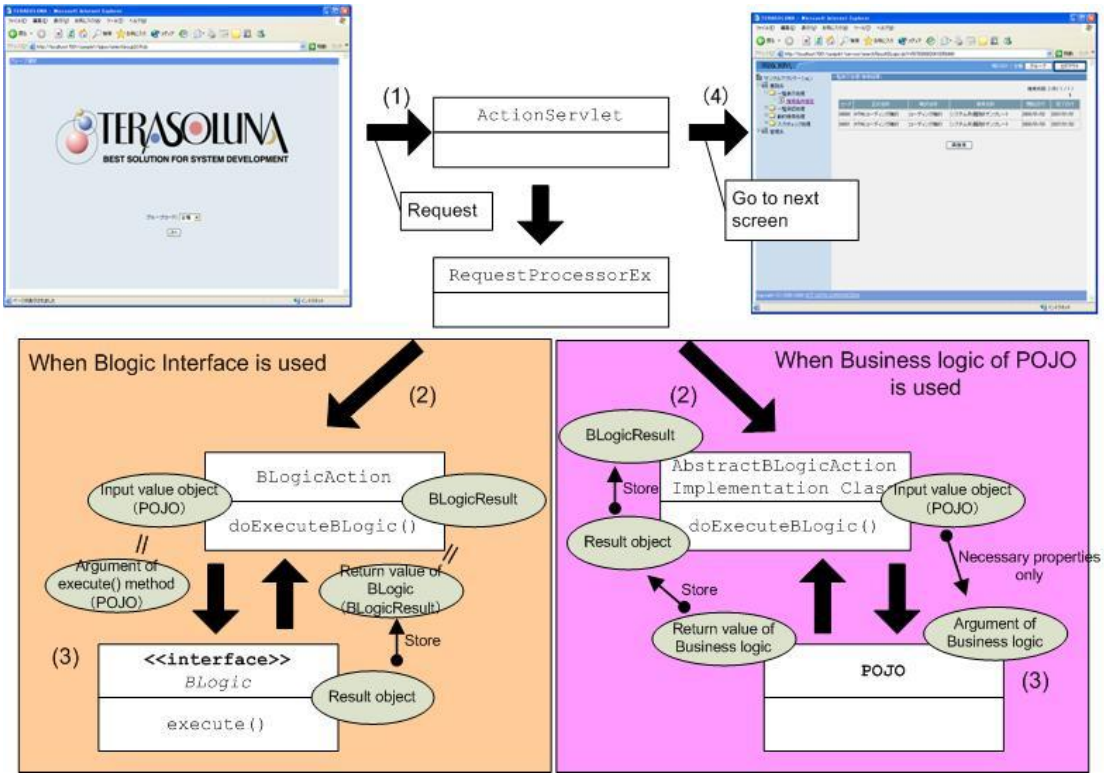
WH-01 Business Logic Execution

■ Overview

◆ Functional Overview

- Provides methods to execute business logic. The Business logic class can either be a BLogic Interface Implementation Class or a POJO. The steps to implement by business developer, for each of these cases is as follows:.
  - ➢ Use BLogic Interface Implementation Class
    - ✧ BLogic Interface Execution Class (called from BLogicAction class)
  - ➢ Use POJO
    - ✧ AbstractBLogicAction Implementation Class
    - ✧ Business logic class of POJO (called from AbstractBLogicAction Implementation Class)
- Convert message information (BLogicMessages) set in BLogicResult, returned from business logic execution, to ActionMessages instance for Struts, and set it in Web layer.

◆ Schematic Diagram



◆ Description

(1) The Web application server receives the request sent from the browser.

(2) Action class associated with action path is called. While using Business Logic Execution function, specify BLogicAction provided by TERASOLUNA Server Framework for Java (Web version), or AbstractBLogicAction Implementation Class implemented by business developer, in the action class.

(3) Execute business logic.

- While using BLogic interface, load BLogic implementation object that is set in businessLogic properties (specified by business developer) to the Bean definition of BLogicAction, and execute execute() method.
- While using business logic of POJO, execute directly the business logic implemented in POJO by doExecuteBLogic() method of AbstractBLogicAction implementation class.

The character strings, result objects and messages indicating the business logic execution result are stored in BLogicResult, which is the return value.

(4) After the objects and messages set in BLogicResult are reflected in Web layer, the next screen to be forwarded is determined from the character string indicating business logic execution result.

■ Usage

◆ Coding points

- Use BLogic Interface

  Implement BLogic interface and implement the business logic. Implement the input parameters of BLogic interface in POJO and define necessary properties according to Screen specifications and Business logic specifications. If there is no input value for the Business logic, it can be null. Further, BLogicResult is used as output value.

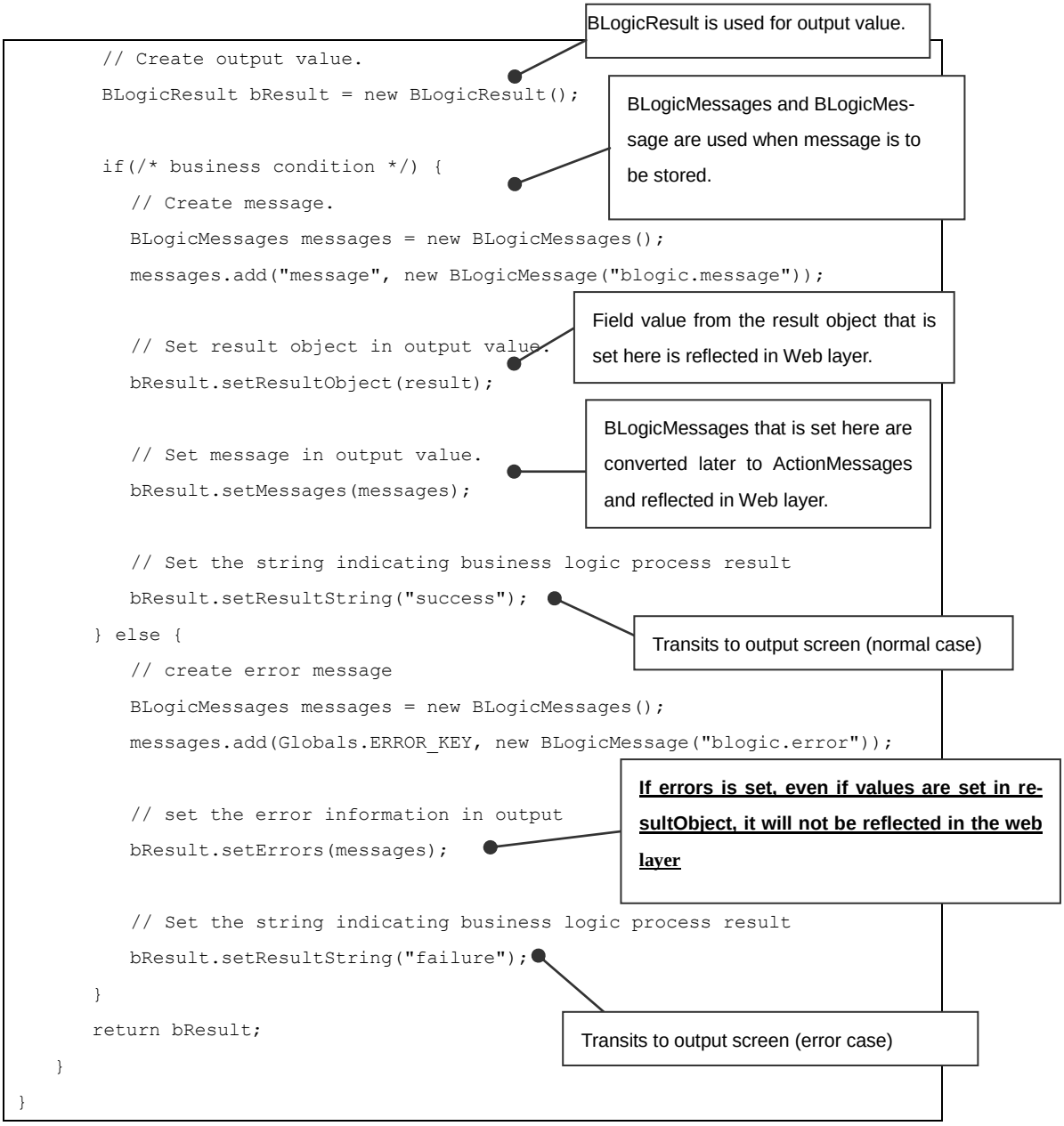
  - Implementation example of BLogic Interface

```
public class AddBLogic implements BLogic<AddBLogicParam> {
    public BLogicResult execute(AddBLogicParam params) {
        // Get value from input value class.
        int value1 = params.getValue1();
        int value2 = params.getValue2();

        // Result object.
        AddBLogicResult result = new AddBLogicResult();
        result.setResult(value1 + value2);
```

Generic type is specified for Input value class. When there is no input value, Generic is not specified.

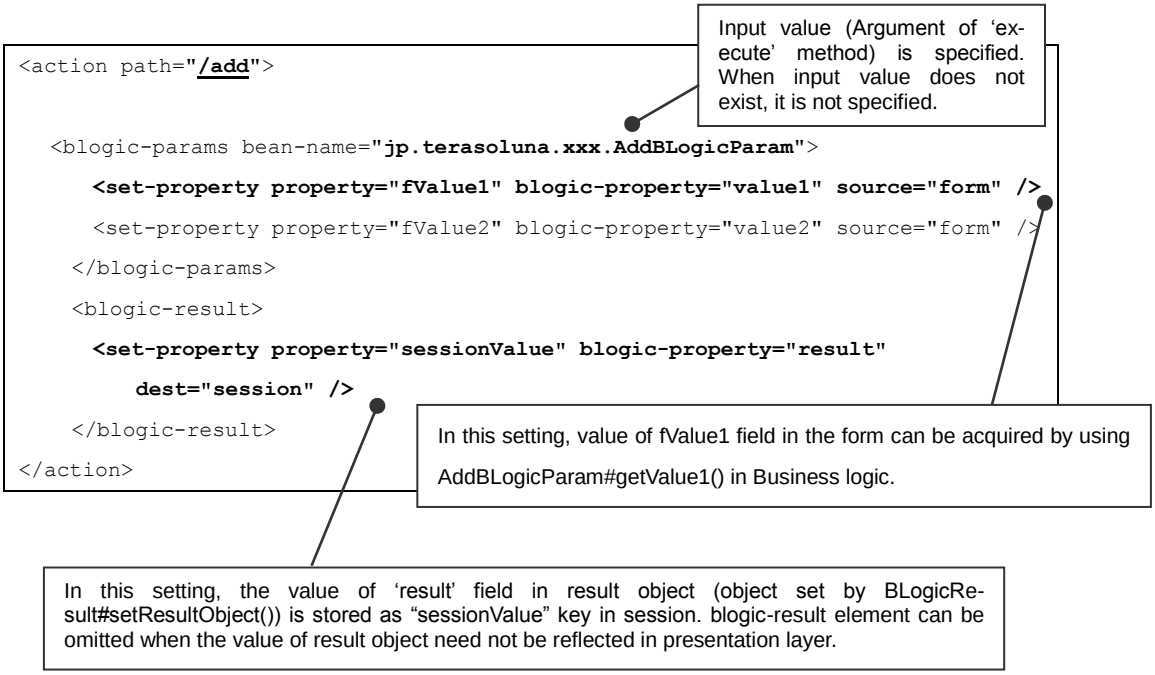
Here, it is implemented in POJO. Map type can also be used.

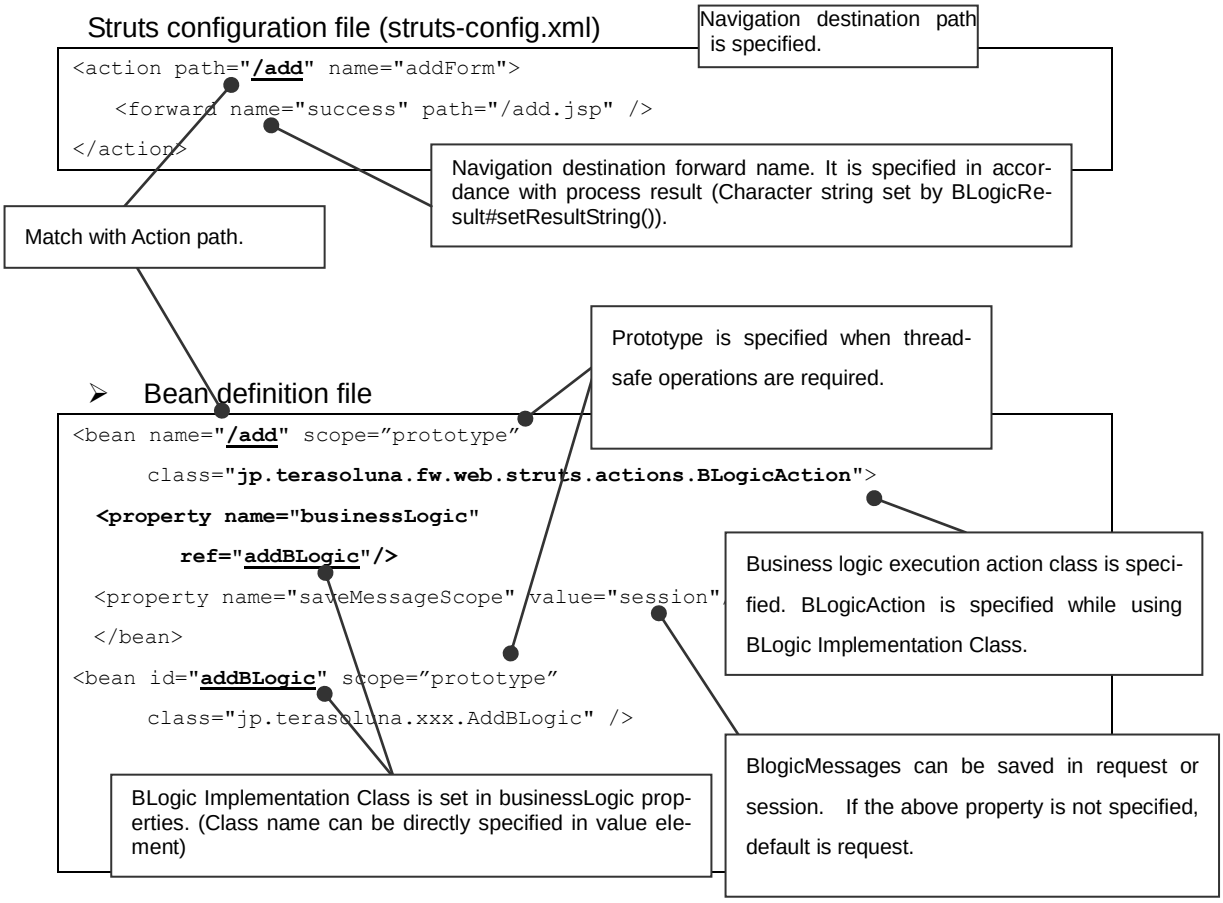
(Continued……)




```
        // Create output value.
        BLogicResult bResult = new BLogicResult();

        if(/* business condition */) {
           // Create message.
           BLogicMessages messages = new BLogicMessages();
           messages.add("message", new BLogicMessage("blogic.message"));

           // Set result object in output value.
           bResult.setResultObject(result);

           // Set message in output value.
           bResult.setMessages(messages);

           // Set the string indicating business logic process result
           bResult.setResultString("success");
        } else {
           // create error message
           BLogicMessages messages = new BLogicMessages();
           messages.add(Globals.ERROR_KEY, new BLogicMessage("blogic.error"));

           // set the error information in output
           bResult.setErrors(messages);

           // Set the string indicating business logic process result
           bResult.setResultString("failure");
        }
        return bResult;
    }
}
```

- Business logic input/output information definition file (blogic-io.xml)
  Please refer to “WH-02 Business Logic Input/Output Function” for the description or functional details of individual element.



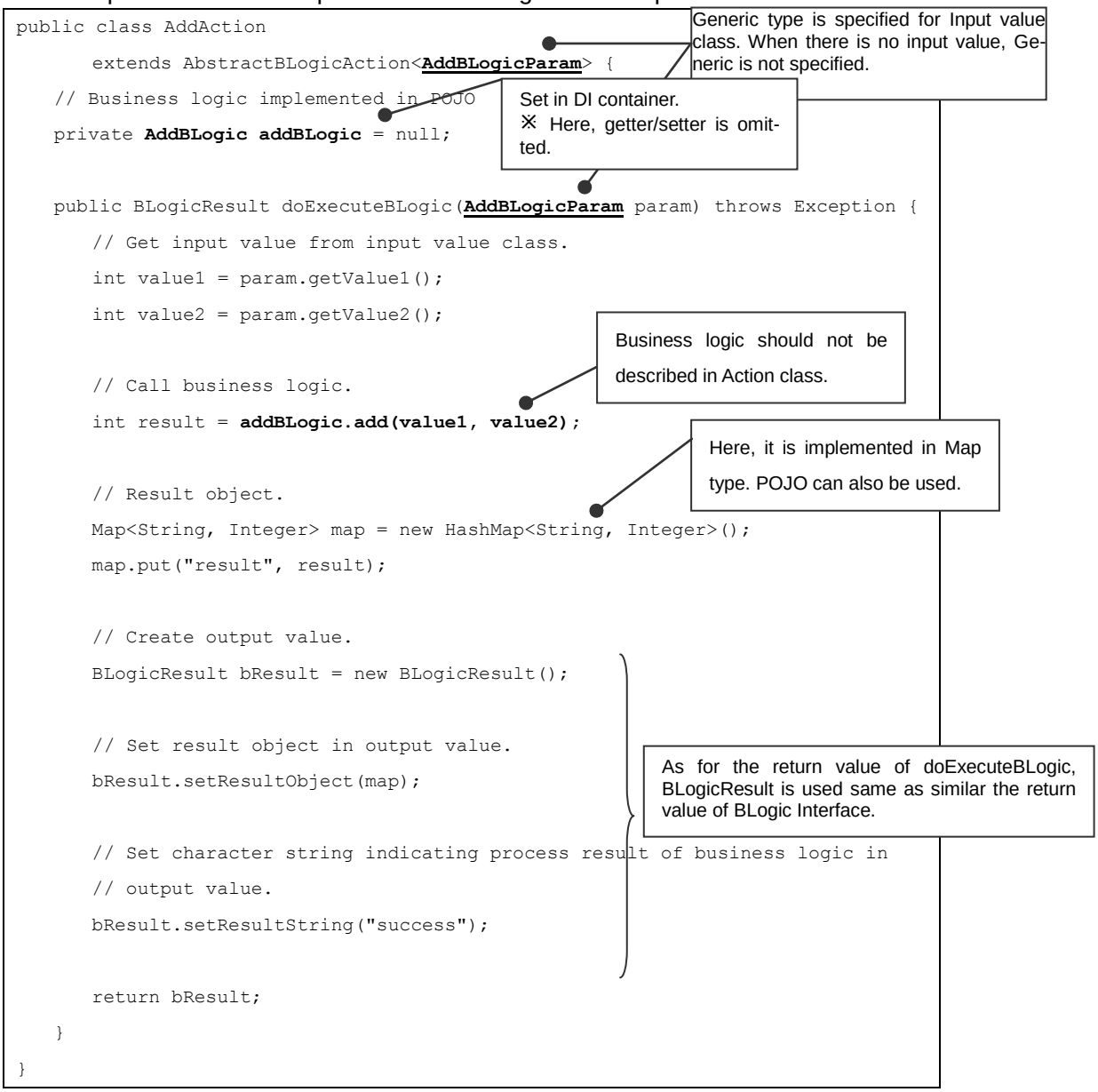


※ "Module name + Action path" should be set in 'name' attribute while executing module partitioning.

- Use Business logic class of POJO

  When business logic portability is required, the business logic class needs to be implemented as a POJO. As far as possible, implement the POJO in such a way that it will not depend on the TERASOLUNA Server Framework for Java (Web version) API and specify only the values necessary in the interface of business method. Create Action class, where AbstractBLogicAction is implemented, for bridging with Web layer (Struts and TERASOLUNA Server Framework for Java (Web version)).

  - Implementation example of AbstractBLogicAction Implementation class




```
public class AddAction
       extends AbstractBLogicAction<AddBLogicParam> {
   // Business logic implemented in POJO
   private AddBLogic addBLogic = null;

   public BLogicResult doExecuteBLogic(AddBLogicParam param) throws Exception {
       // Get input value from input value class.
       int value1 = param.getValue1();
       int value2 = param.getValue2();

       // Call business logic.
       int result = addBLogic.add(value1, value2);

       // Result object.
       Map<String, Integer> map = new HashMap<String, Integer>();
       map.put("result", result);

       // Create output value.
       BLogicResult bResult = new BLogicResult();

       // Set result object in output value.
       bResult.setResultObject(map);

       // Set character string indicating process result of business logic in
       // output value.
       bResult.setResultString("success");

       return bResult;
   }
}
```

- Implementation example of POJO

  Create interface for POJO to implement interface base.

```
public interface AddBLogic {
   int add(int value1, int value2);
}
```

  Create POJO where the created interface is implemented.

```
public class AddBLogicImpl implements AddBLogic
   public int add(int value1, int value2) {
       return value1 + value2;
   }
}
```

This business logic is the class which can be used in any application, by implementing in such a way that it will not depend on Web layer, presentation layer, Struts and TERASOLUNA.

  Business logic should be implemented in such a way that it will not depend on the TERASOLUNA Server Framework for Java (Web version) API. But when accessing the database, DAO class of TERASOLUNA Server Framework for Java (Web version) should be used in the business logic. Please refer to the "CB-01 Database Access" for the details of database access.

- Business Logic Input/Output Information Definition File (blogic-io.xml)

  This is similar to the case when BLogic Interface is used.

- Struts configuration file (struts-config.xml)

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

- Bean definition file

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

Other settings are similar to the case when BLogic Interface is used.

AbstractBLogicAction Implementation Class is set.

Business logic of POJO is set in AddBLogic properties of AddAction. (Class name can be directly specified in value element)

- Transaction Control

  Please refer to "CA-01 Transaction Control" for the transaction control method.

◆ Extension points

None.

■ Classes

	Class Name	Overview
1	jp.terasoluna.fw.web.struts.actions.AbstractBLogicAction	Abstract class implementing functions that are common to Action classes that execute Business logic.
2	jp.terasoluna.fw.web.struts.actions.BLogicAction	Class to execute business logic. Concrete class for AbstractBLogicActio.
3	jp.terasoluna.fw.service.thin.BLogic	Interface to be implemented by Action class that executes Business logic.
4	jp.terasoluna.fw.service.thin.BLogicResult	Class to store result from business logic.
5	jp.terasoluna.fw.service.thin.BLogicMessage	Message information class.
6	jp.terasoluna.fw.service.thin.BLogicMessages	Message information list class. It stores BLogicMessage instance.

■ Related Function(s)

- “CA-01 Transaction Control”
- “CB-01 Database Access”
- “WH-02 Business Logic Input/Output Function(s)”

■ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - "UC22 Business Logic Execution"
    - /webapps/blogic/*
    - /webapps/WEB-INF/blogic/*
    - jp.terasoluna.thin.functionsample.blogic.*

- TERASOLUNA Server Framework for Java (Web version) Tutorial
  - "2.4 Logon"
  - /webapps/WEB-INF/moduleContext.xml

■ Remarks

- None.

WH-02 Business Logic Input/Output Function

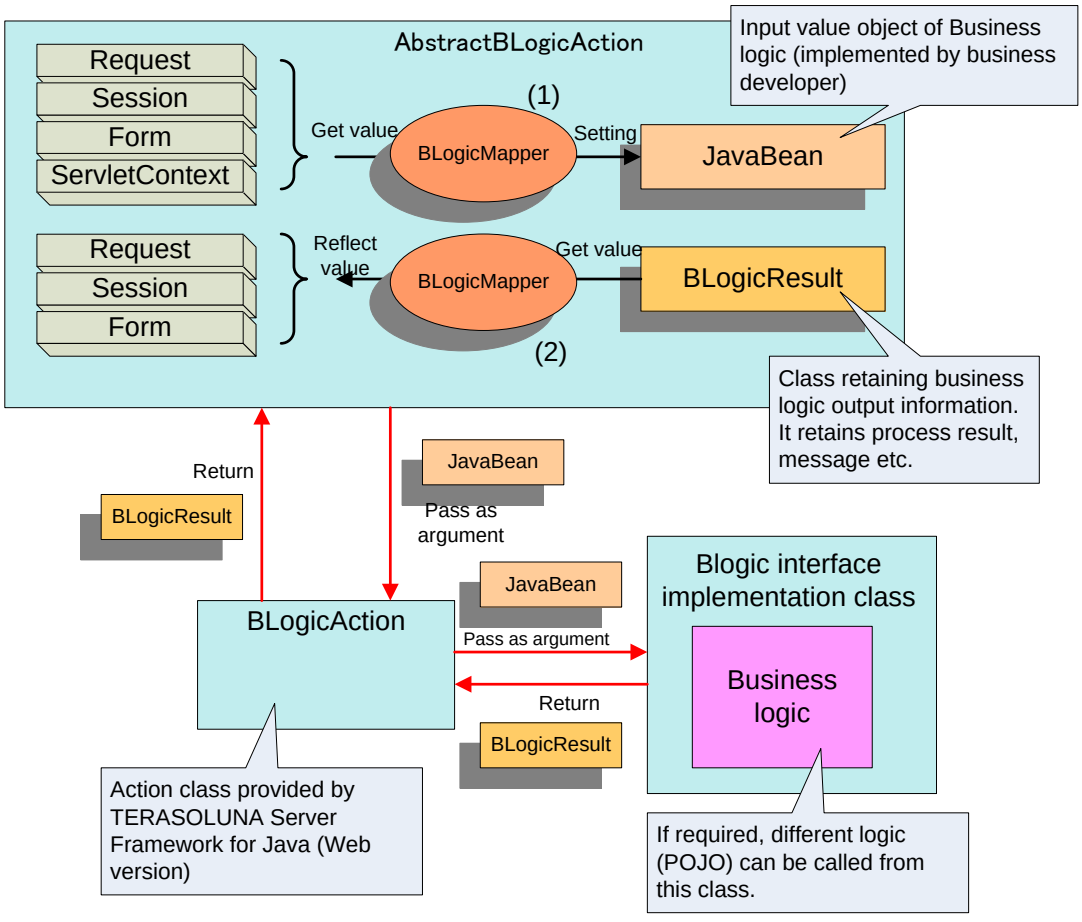
■ Overview

◆ Functional Overview

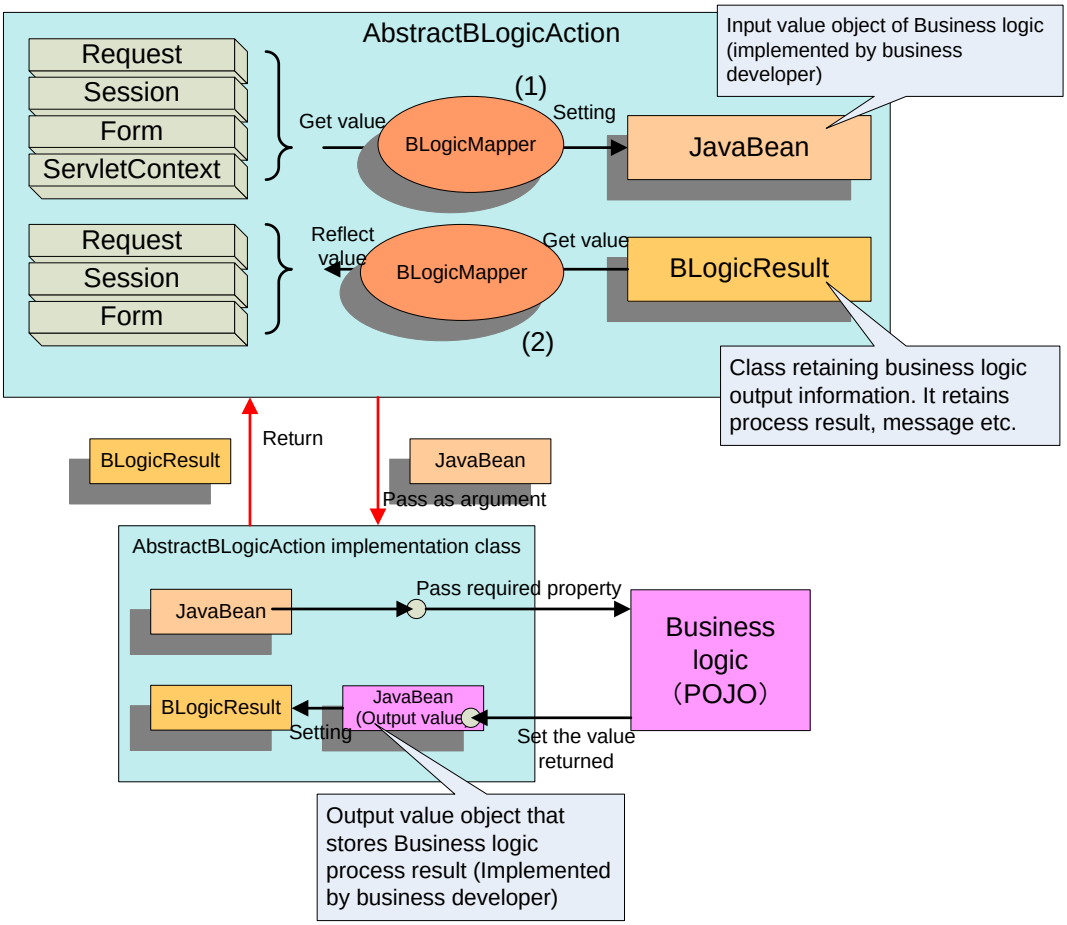
- Automates the conversion of data between presentation layer and service layer by defining the input/output setting in a configuration file.
- Uses plug-in function(s) of Struts in order to initialize the Business Logic input/output values.
- By default, it inputs the data to Business logic from Request, Session, ActionForm and Servlet Context, and outputs the same to Request, Session and ActionForm from Business logic.

◆ **Schematic Diagram**

- Flow of Input/Output information when BLogicAction is used



- Flow of Input/Output information when AbstractBLogicAction Implementation Class is used



◆ Description

- Initialization
  - The Initialization of input/output information definition is done from a plugin while executing the servlet. The Business Logic Input/Output Information Definition File (blogic-io.xml) is read and the settings are saved in ServletContext. The settings for the current action path are obtained fom the servlet context and used in the input/output settings of Business Logic.
- Input/Output Setting Process

(1) If Business Logic Execution Action is called, from the presentation layer BLogicMapper prepares the required input values to Business logic based on input information setting of blogic-io.xml. It stores the input values in a value object with the name specified in input information setting, and passes these as argument to the Business Logic Execution Action (BLogicAction or AbstractBLogicAction implementation class). If Business Logic has no input value, the bean-name attribute of blogic-io.xml is omitted and the Business Logic is executed with input value ob-

ject as ‘null’.

(2) After executing Business Logic, the stored BLogicResult is returned. BLogicMapper obtains the value from BLogicResult based on output information setting of blogic-io.xml and reflects the execution results in presentation layer. When output value need not be reflected in presentation layer, blogic-result element can be omitted.

Please refer to “WH-01 Business Logic Execution” for the details of Business Logic execution.

■ Usage

◆ Coding Points

The following points should be implemented for Business Logic Input/Output Function(s).

1. Setting of object of presentation layer
2. Creation of Business Logic Input/Output Information Definition File
3. Process to obtain input information within the Business Logic.
4. Process to reflect output information within the Business Logic.

The following example uses the userid and password fields in ActionForm (DynaValidatorActionFormEx) to fetch the user information from DB..

1. The UserValueObject is created and stored in the session, if user information exists in DB. Success is indicated by returing ‘success’ in resultString of BLogicResult (Field indicating execution result of Business Logic).
2. The error message is stored in BLogicResult, if user information does not exist in DB. Failure is indicated by returing ‘failure’ in resultString of BLogicResult.

- Business Logic Input/Output Information Definition File (blogic-io.xml)
  - At the top of blogic-io.xml, DTD to be used as validation rule while parsing from XML to object in Digester is described as follows.
    The description rules are rigid and changes are not necessary unless it is extended.

```
<?xml version="1.0" encoding="Windows-31J" ?>
<!DOCTYPE blogic-io [
  <!ELEMENT blogic-io     (action*)>
  <!ELEMENT action        (blogic-params?,blogic-result?)>
  <!ATTLIST action        path           CDATA #REQUIRED>
  <!ELEMENT blogic-params  (set-property*)>
  <!ATTLIST blogic-params bean-name      CDATA #IMPLIED>
  <!ELEMENT blogic-result  (set-property*)>
  <!ELEMENT set-property  EMPTY>


  <!ATTLIST set-property  property       CDATA   #REQUIRED>
  <!ATTLIST set-property  blogic-property CDATA   #IMPLIED>
  <!ATTLIST set-property  source         CDATA   #IMPLIED>
  <!ATTLIST set-property  dest           CDATA   #IMPLIED>
]>
```

Action element is set for every action path, in which Business Logic Execution Functionality

Setting of business logic input information. Input value object of Business Logic is specified in bean-name attribute. 'bean-name' is omitted in case of no input value. 'set-property' element describes number of input items only.

Setting of business logic output information. 'set-property' element describes number of output items only. blogic-result element is omitted when output value is not reflected.

  - Separate set-property element is set for Business Logic Input/Output value.
    property attribute …Specifies the field name from where the value is acquired.
    blogic-property attribute …Specifies the key of input/output information used in Business Logic. If it is omitted, process is executed with the value similar to that of property attribute.
    source attribute …Specifies the source of getting value. (Required in blogic-params element)
    dest attribute …Specifies destination of value. (Required in blogic-result element.)
    ※ Output to ServletContext is not the default target.
  - In case the source is
    ActionForm :「form」、
    Session :「session」、
    Request :「request」、
    Servlet Context :「application」
    should be specified
    When "request" is specified only values of request attributes can be fetched (getAttribute) and values of request parameters (getParameter) can not be fetched.

    In case the destination is
    ActionForm :「form」、
    Session :「session」、
    Request :「request」、
    should be specified

Example of blogic-io.xml text is shown below.

```
<blogic-io>
  <action path="/logon" >
    <blogic-params bean-name="jp.terasoluna.xxx.LogonBLogicParam">
      <set-property property="userId"   blogic-property="uid" source="form" />
      <set-property property="password" blogic-property="pwd" source="form" />
    </blogic-params>
    <blogic-result>
      <set-property property="USER_VALUE_OBJECT"  blogic-property="uvo" dest="session" />
    </blogic-result>
  </action>
......
<blogic-io>
```

Input value object of Business Logic is specified. Omit bean-name in case of no input value.

Matched with Action path.

The userId field of ActionForm is passed to Business Logic as uid. The specified field must exist in form.

Return value uvo is stored in the session from Business Logic by considering USER_VALUE_OBJECT as key. blogic-result element is omitted when output value is not reflected.

- Struts Configuration File（struts-config.xml）
  - Business Logic Input/Output Function can be used by specifying BLogicIOPlugin as plugin element.

```
<struts-config>
  <form-beans>
    <form-bean name="logonForm"
        type="jp.terasoluna.fw.web.struts.form.DynaValidatorActionFormEx">
        <form-property name="usrId" type="java.lang.String"/>
        <form-property name="password" type="java.lang.String"/>
        ......
    </form-bean>
    ......
  </form-beans>

  <action-mappings type="jp.terasoluna.fw.web.struts.action.ActionMappingEx">
    <!—Logon process-->
    <action path="/logon" name="logonForm" scope="request">
      <forward name="success" path="/selectGroupSCR.do"/>
      <forward name="failure" path="/logonSCR.do"/>
    </action>
  </action-mappings>
  ......
```

ActionForm name defined by form-bean element is specified.

(Continued)

```
  <!-- BLogic-IO plugin -->
  <plug-in className="jp.terasoluna.fw.web.struts.plugins.BLogicIOPlugIn">
    <set-property property="resources"
                    value="/WEB-INF/blogic-io.xml,/WEB-INF/blogic-io2.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-config>
```

Digester Rule File

Mapper class that controls input/output information of Business Logic is specified. When nothing is specified, BLogicMapper is used by default. Default mapper class is used when extension is not specifically required.

Business Logic Input/Output Information Definition File (The file name is specified in comma-delimited format for file division.)

- Bean Definition File
  - Here, BLogicAction is used in Business Logic Execution Action Class. Set BLogic Interface Implementation Class LogonBLogic for BLogicAction. Please refer to “WH-01 Business Logic Execution” for Bean definition file description method in Business Logic execution.

BLogicAction is used for the process of action path “/logon” defined in struts-config.xml.

Business Logic is set in BLogicAction.

```
<bean name="/logon" scope="prototype"
      class="jp.terasoluna.fw.web.struts.actions.BLogicAction">
  <property name="businessLogic" ref="logonBLogic"/>
</bean>
<bean id="logonBLogic" scope="prototype" class="jp.terasoluna.sample.LogonBLogic" />
```

BLogic Interface Implementation Class is specified.

- Business Logic Input Value Object (LogonBLogicParam.java)
  - This object is used to store the value that is passed from presentation layer to Busines Logic and it is implemented by business developer. It is the argument of LogonBLogic#execute() method.

```
public class LogonBLogicParam {
    private String uid = null;
    private String pwd = null;
......
    public String getUid() {
        return this.uid;
    }
    public void setUid(String id) {
        this.uid = id;
    }
......
}
```

Field specified in blogic-property attribute of set-property element under blogic-param of blogic-io.xml is set.

getter/setter is set for the field.

- Business Logic Output Value Object (LogonBLogicResult.java)
  - This object is used to store the value of processing result of Business Logic and it is implemented by business developer. Return the object by storing in resultObject of return value BLogicResult of LogonBLogic#execute() method.

```
public class LogonBLogicResult {
    private SampleUserValueObject uvo = null;
......
    public SampleUserValueObject getUvo() {
        return this.uvo;
    }
......
    public void setUvo(SampleUserValueObject obj) {
        this.uvo = obj;
    }
......
}
```

Field specified in blogic-property attribute of set-property element under blogic-result of blogic-io.xml is set.

getter/setter is set for the field.

- Business Logic (LogonBLogic.java)
  - The description example of BLogic Interface Implementation Class is shown. Please refer to “WH-01 Business Logic Execution” for the details of Business Logic execution.
  - Description is omitted. However, it is necessary to access DB for the methods ‘isAuthenticated()’ and ‘getUserName()’, which are called from execute() method. Please refer to “CB-01 Database Access” for DB access method.

```
public class LogonBLogic implements BLogic<LogonBLogicParam> {
    ......
    public BLogicResult execute(LogonBLogicParam params) {
        // Get value from input value object.
        String uid = params.getUid();
        String pwd = params.getPwd();

        // Create output value.
        BLogicResult bResult = new BLogicResult();

        // Execute login authentication.
        if(isAuthenticated(uid, pwd)) {
          // Get value from DB.
          String userName = getUserName(uid);

          // Create UserValueObject.
          Sample1UserValueObject uvo = new Sample1UserValueObject();
```

The object specified in bean-name attribute of blogic-params element of blogic-io.xml is specified as input value object. Specification is not required in case of no input value.

Value can be obtained from input object with the name specified in blogic-property attribute of set-property element of blogic-io.xml.

(Continued)

```
        // Set value in UVO
        uvo.setId(uid);
        uvo.setName(userName);

        // Result object.
        LogonBLogicResult result = new LogonBLogicResult();
        result.setUvo(uvo);

        // Set result object (POJO) to output value
        sult.setResultObject(result);

        // Set the string indicating processing result of Business Logic to output value.
        bResult.setResultString("success");
      } else {
        // Create message.
        BLogicMessages messages = new BLogicMessages();
        messages.add("message", new BLogicMessage("blogic.message"));

        // Set message in output value.
        bResult.setMessages(messages);

        // Set the string indicating processing result of Business Logic to output value.
        bResult.setResultString("failure");
      }
      return result;
   }

   private boolean isAuthenticated(String userId, String password) {
      // Access DB and determine whether user exists or not.
      ......
   }

   private String getUserName(String userId) {
      // Access DB and get user name.
      .......
   }
}
```

The value in result object is set with the name specified in blogic-property attribute of set-property element of blogic-io.xml.

Execution result (POJO) of Business Logic is stored. It is processed by mapper class and reflected in presentation layer.

The next navigation destination is determined by specifying navigation destination logic name of Struts configuration file with the help of string that is set here.

BLogicMessages are created and stored in order to display messages on the screen.

As mentioned above, the information required for authentication is obtained from ActionForm in case of occurrence of /logon.do request, and results are stored in session when authentication is successful.

◆ Extension Points

BLogicMapper can be replaced when mapping rule of input/output information of Business Logic is to be customized. Create new AbstractBLogicMapper Implementation Class for mapper class or create by extending the default BLogicMapper. Please refer to Javadoc of each class for extension method of AbstractBLogicMapper and BLogicMapper.
In Struts configuration file (struts-config.xml), define the fully qualified name of mapper class to be used in the value of mapperClass property of <set-property> element. Then, that mapping class is processed when Business Logic is executed.

- Struts Configuration File (struts-config.xml)

```
<struts-config>
  ......
  <!-- BLogic-IO PlugIn  -->
  <plug-in className="jp.co.nttdata.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.example.web.plugin.BLogicMapperEx"/>
  </plug-in>
  ......
</struts-config>
```

Fully qualified name of extended mapper class is specified as the value of “mapperClass”.

※ Note that the target of Business Logic Input/Output Information Configuration File is the same module.
Further, when this property setting is
omitted, .terasoluna.fw.service.thin.BLogicMapper is set by default.

■ Classes

	Class Name	Overview
1	jp.terasoluna.fw.service.thin.AbstractBLogicMapper	Abstract class where functions that execute mapping of data between object of presentation layer and Business Logic, on the basis of BLogicResources stored in Business Logic Input/Output Information.
2	jp.terasoluna.fw.service.thin.BLogicMapper	Default AbstractBLogicMapper Implementation Class provided by TERASOLUNA.
3	jp.terasoluna.fw.web.struts.plugins.BLogicIOPlugin	It obtains the Input/Output information required for execution of Business Logic. Struts plugin that is executed when servlet is activated.
4	jp.terasoluna.fw.service.thin.BLogicResources	It stores the Business Logic input/output information of each module. It corresponds to information in <blogic-io> element of Business Logic Input/Output Information Definition File (blogic-io.xml).
5	jp.terasoluna.fw.service.thin.BLogicIO	It stores the Business Logic input/output information for each action. It corresponds to information in <action> element of blogic-io.xml.
6	jp.terasoluna.fw.service.thin.BLogicProperty	It corresponds to <set-property> element in <blogic-params> or <blogic-result> element of blogic-io.xml.

■ Related Function(s)

- “WB-01 User Information Retention”
- “WH-01 Business Logic Execution”
- “CB-01 Database Access”

■ Example

- Sample covering all functions in the TERASOLUNA Server Framework for Java (Web version)
  - “UC23 Business Logic Input/Output”
    - /webapps/blogicio/*
    - /webapps/WEB-INF/blogicio/*
    - jp.terasoluna.thin.functionsample.blogicio.*

- TERASOLUNA Server Framework for Java (Web version) Tutorial
  - “2.4 Logon”
  - /webapps/WEB-INF/blogic-io.xml

■ Remarks

- None

WI-01 List Display Function

■ Overview

◆ Functional Overview

- The <ts:pageLinks> tag for paging links can be used to display lists using the <logic:iterate> tag provided by Struts.
- Multiple <ts:pageLinks> elements can be specified on a single screen using the List display and Page link function.

◆ Schematic Diagram

◆ Description

- <logic:iterate>
  The list iteration tag <logic:iterate> of Struts is used. The List data is retrieved from the bean specified in the tag's attribute, and the list is looped over. Specified Bean may be a Collection, ArrayList, Vector, Enumeration, Iterator, Map, HashMap, Hashtable, TreeMap, array etc. The repetitive tags (<TR>~</TR> etc. in HTML table) are specified internally. Please refer to Struts for more details.
- <ts:pageLinks>
  Displays links to each page of the list defined in <logic:iterate> tag. To use the page link function, the ActionForm must have the following properties.
  - Property to store the displayed rows
  - Property to store the starting index
  - Property to store the total size of the list

■ Usage

◆ Coding points

- Example showing retrieving of list from the database
  - ➢ JSP
    - ✧ Please Refer to Attribute list for the details on each attribute.

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

Specify the action to get the list information

When name property is not set, it is directly retrieved from the request or session without retrieving from ActionForm or Bean

Entire list stored in the request is displayed.

  - ➢ Struts configuration file
    - ✧ ActionForm property required for list display and the action on clicking Page link, are set in Struts configuration file.

```
        <form-beans>
  <form-bean name="listForm"
    type="jp.terasoluna.fw.web.struts.form.DynaValidatorActionFormEx">
    <form-property name="row"
            type="java.lang.String" initial="10"/>
    <form-property name="startIndex"
            type="java.lang.String" initial="0"/>
  </form-bean>
</form-beans>

<action path="/list"
       name="listForm" scope="request">
  <forward name="success" path="/listSCR.do"/>
</action>
```

ActionForm item to get the rows and Start index of the request parameter, and pass it to business logic

Action mapping to call business logic that gets the list page-wise.

    - ✧ row, startIndex, and totalCount do not necessarily have to be in the actionform, but they may also be specified in the request and session. When the name attribute is not specified, the attribute in the specified scope is used.

- Bean definition file
  - In the bean definition file, specify the business logic to be class when specified action is executed.

```
        <bean name="/list" scope="prototype"
    class="jp.terasoluna.sample.web.action.ListAction">
  <property name="listService" ref="listService"/>
</bean>

<bean id="listService" scope="prototype"
     class="jp.terasoluna.sample.service.ListService">
  <property name="dao" ref="queryDAO"/>
</bean>
```

Business logic to be executed by action is set.

- BLogic input/output configuration file
  - Specify the value passed from action and the value returned to action.

```
          <action path="/list">
    <blogic-params bean-name="jp.terasoluna.sample.bean.ListBean">
      <set-property property="startIndex" source="form" />
      <set-property property="row" source="form" />
    </blogic-params>

    <blogic-result>
      <set-property property="userBeans" dest="request" />
      <set-property property="startIndex" dest="request" />
      <set-property property="row" dest="request" />
      <set-property property="totalCount" dest="request" />
    </blogic-result>
  </action>
```

Start index and rows are copied from ActionForm to a JavaBean and then passed to action.

Result object in BLogicResult returned by action is stored in session, request or form.
Here, when the start index, rows and total count (required for page link function) are set in the request, set the value in 'property' attribute only to retrieved directly from the request instead of using the name property of page link function.

- Action
  - ✧ It is the action to be executed on clicking the Page link. This action inherits AbstractBLogicAction.

```
public class ListAction extends AbstractBLogicAction<ListBean> {
private ListService
          listService = null;
......
public BLogicResult doExecuteBLogic(ListBean listBean)
    throws Exception {

    // execute business logic and get result
    ResultBean resultBean
       = listService.getUserList(listBean);
    resultBean.setStartIndex(listBean.getStartIndex());
    resultBean.setRow(listBean.getRow());

    //Create BLogicResult and set the result
    BLogicResult result = new BLogicResult();
    result.setResultString("success");
    result.setResultObject(resultBean);
    return result;
}
```

Specify Business logic class in Bean definition file.

List is retrieved by executing the business logic.

Result object is stored in BLogicResult.

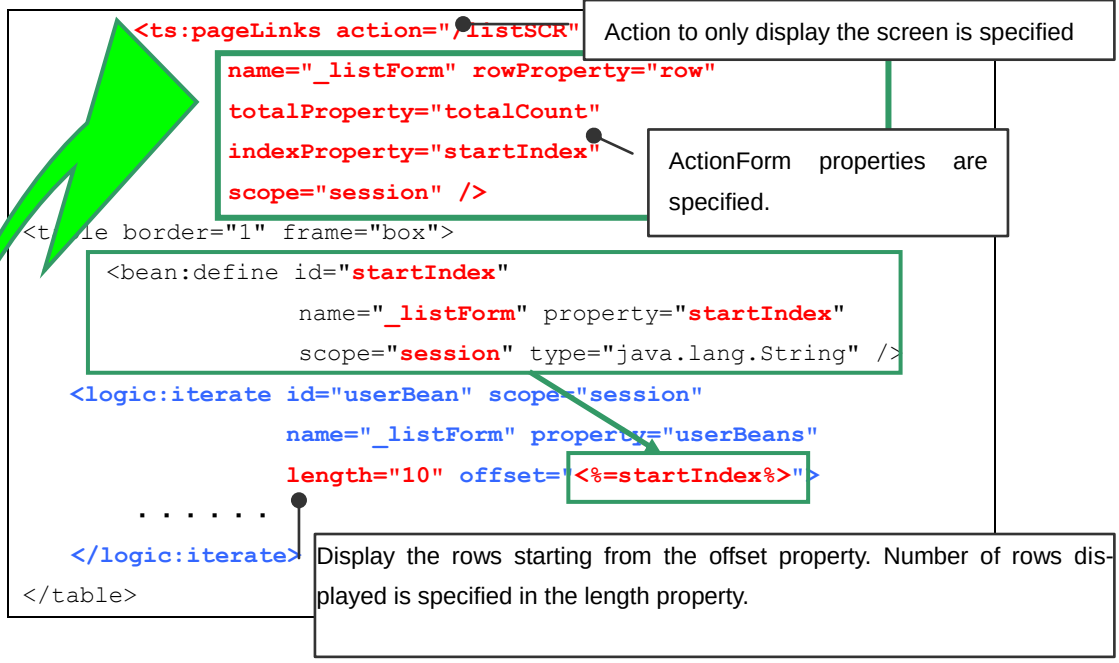
- Business logic
  - ✧ It is the business logic that is called from the action that is called on clicking the Page link.

```
public ResultBean getUserList(ListBean listBean) {
    //Acquire Start index and rows
    int startIndex = listBean.getStartIndex();
    int row = listBean.getRow();
    // Acquire Total count
    Integer totalcount = dao.executeForObject(
                        "getUserCount", null, Integer.class);
    //Acquire the list information only in screen display part.
    UserBean[] beans = dao.executeForObjectArray("getUserList",
                      null, UserBean.class, startIndex, row);
    ResultBean resultBean = new ResultBean();
    resultBean.setTotalCount(totalcount.intValue());
    resultBean.setUserBeans(beans);
    return resultBean;
```

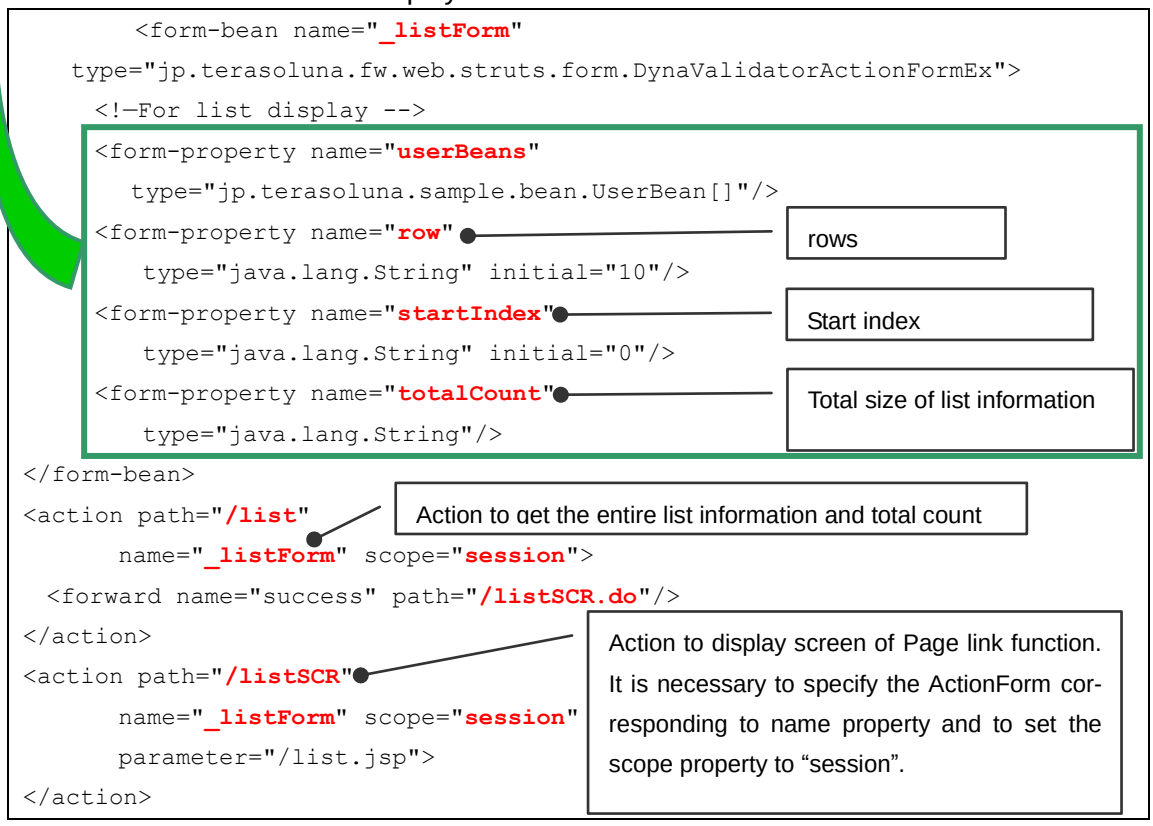
Start index and rows are got and set in DAO argument.

The total count and list information is stored in Bean and is returned to action.

- Example showing retrieving of list information from database
  - JSP
    - Refer to Attribute list for attribute details.



  - Struts configuration file
    - It is necessary to store the list information and related attribute in index in case of List display where ActionForm is used.



- Bean definition file
  - In the bean definition file, specify the business logic to be executed when the action is executed.

```
        <bean name="/list" scope="prototype"
    class="jp.terasoluna.sample.web.action.ListAction">
  <property name="listService">
    <ref local="listService"/>
  </property>
</bean>

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

Specify the action to get the entire list information and total count. It is called only in the beginning when list screen is displayed.

Specify ForwardAction to onlyy redisplay the list information.

- BLogic input / output configuration file
  - Specify the values passed from action and the values returned to action.

```
          <action path="/list">
     <blogic-params bean-name="java.util.HashMap">
     </blogic-params>

     <blogic-result>
       <set-property property="userBeans"  dest="form" />
       <set-property property="totalCount" dest="form" />
     </blogic-result>
   </action>
```

This example has no properties, however in case of search screens, etc describe the necessary properties.

Result object (BLogicResult) returned from action is stored in the form.

➢ Action

✧ This is the action to be executing on clicking a Page link. This action inherits AbstractBLogicAction.

```
public class ListAction extends AbstractBLogicAction {
private ListService
          listService = null;
......
public BLogicResult doExecuteBLogic(Object obj)
    throws Exception {

    //Execution of business logic and
    ResultBean resultBean = listService.getUserList();

    //Create BLogicResult and set the result
    BLogicResult result = new BLogicResult();
    result.setResultString("success");
    result.setResultObject(resultBean);
    return result;
}
```

Business logic is set in the Bean definition file.

List information is acquired by executing the business logic.

Result object is stored in BLogicResult.

➢ Business logic

✧ It is the business logic called from action that is executed on clicking the Page link.

```
public ResultBean getUserList() {
//Get the list information.
UserBean[] beans = dao.executeForObjectArray("getUserList",
                   null, UserBean.class);
ResultBean resultBean = new ResultBean();
resultBean.setTotalCount(beans.length);
resultBean.setUserBeans(beans);
return resultBean;
```

The acquired total count and list information is stored in Bean and is returned to action.

- To use submit

The page link function, by default, uses hyperlinks to change the page. However, it is also possible to use submit (using JavaScript), setting the submit attribute to true. Moreover, <ts:form> element should be specify to page to display when submit is clicked.

➢ JSP

✧ Please refer to Attribute list for attribute details.

```
<ts:pageLinks action="/list" submit="true" rowProper-
ty="row"
totalProperty="totalCount" indexProperty="startIndex" />
```

- When DispatchAction is used.
  When DispatchAction is used to dispatch to multiple destinations, use the submit, forward and event attributes.
  - JSP
    - The hidden value in the event attribute is output, if the “forward” attribute and “submit” attribute are set to true. In the Javascript of the Page link click event, set “forward_pageLinks” in the hidden value.
    - Default value of event attribute is “event’.
    - Refer to the Attribute list for attribute details.

```
        <ts:form action="/listDSP">
<ts:pageLinks action="/list" name="dynaFormBean" rowProperty="row"
            totalProperty="totalCount" indexProperty="startIndex"
            submit="true" forward="true" event="next"
......
</ts:form>
```

  - Struts configuration file
    - Set “pageLinks” as the name of forward for page navigation.

```
        <action path="/listDSP"
         name="dynaFormBean" scope="request">
    <forward name="pageLinks" path="/list.do"/>
    <forward name="delete"    path="/delete.do"/>
    <forward name="default"   path="/list.do"/>
</action>
```

  - Bean definition file
    - In the event property of DispatchAction, specify the same value as the the value in the event attribute of <ts:pageLinks> tag. Since the default value of event attribute of DispatchAction is also “event”, it is not required when the event attribute is not specified in the Page link function.

```
      <bean name="/listDSP" scope="prototype"
class="jp.terasoluna.fw.web.struts.actions.DispatchAction">
  <property name="event"><value>next</value></property>
</bean>
```

- Reset a specified range
  ”startindex” and “endIndex” are necessary in the request parameter to reset a specified range using the reset function. In Page link function, “startIndex” and “endIndex” are output as hidden items containing index of the page to be displayed when the resetIndex attribute is set to true. Further, when using the combination of current functionality and Reset functionality is used, the value of indexProperty attribute should be other than “startIndex”. If “startIndex” is set as the value of indexProperty, it will overlap with the parameter that is used in Reset Functionality.

  - JSP

```
<ts:pageLinks action="/list" rowProperty="row"
    totalProperty="totalCount"
    indexProperty="pageStartIndex"
    resetIndex="true"/>
```

Items used in List Display function are output as hidden

➢ Output result of HTML

```
<input type="hidden" name="row" value="10"/>
<input type="hidden" name="pageStartIndex" value="0"/>
<input type="hidden" name="startIndex" value="0"/>
<input type="hidden" name="endIndex" value="9"/>
```

Items used in Specified range reset functionality are also output as hid-

Please refer to “WB-04 Form Properties Resetting Function” for the details of the specified range reset function.

- Writing page links at desired location using the id attribute

  Normally, in the Page link function, the <ts:pageLinks> tag is used where links are to be outputted, however, when a string is specified in id attribute, page links are outputted in the Page context with that string as the key. The Page link saved in the page context can be later output using <bean:write> tag, the <ts:pageLinks> tags can be later output to the desired location.

  Note that sanitizing process is executed in case of output using <bean:write> tag. So, It is necessary to set the filter attribute to false in case of <bean:write> tag.

  ➢ JSP

```
<ts:pageLinks id="reservePageLinks" action="/list"
      rowProperty="row" totalProperty="totalCount"
      indexProperty="startIndex" resetIndex="true"/>
<bean:write name="reservePageLinks" filter="false"/>
```

- Output of current page, total page count and total count

  In Page link function, current page and total page count are set in Page context using the value in the currentPageIndex property and totalPageCount property as Key. Moreover, default values are set to these properties. When values are not specified, these default values are used and set to Page context. <bean:write> tag can be to output from Page context to screen. Total count should already be saved in form, session or request, as it is required in the Page link function. <bean:write> tag can be used to output to the screen.

  - JSP (When attribute is not specified)

```
        <ts: pageLinks action="/list" rowProperty="row"
        totalProperty="totalCount" indexProperty="startIndex"/>
<bean:write name="currentPageIndex" /> is the current page.
<bean:write name="totalPageCount" /> is the total page count.
<bean:write name="totalCount" /> is the total count.
```

  - JSP (When attribute is specified)

```
        <ts:pageLinks action="/list" rowProperty="row"
        totalProperty="totalCount" indexProperty="startIndex"
        currentPageIndex="nowPage" totalPageCount="totalPage"/>
<bean:write name="nowPage" />
<bean:write name="totalPage" />
```

- Modify the output link

  The format of outputted page links can be changed in a property file. Properties that can be set are described below.

Key	De-fault	Overview
pageLinks.prev<Numeric value>.char	-	Outputs the link to display the previous pages. Displays the previous page mentioned in the “<Numeric value>” part. When 10 is selected, it display a page, that is 10 pages before the current page. Multiple links can be s pecified by changing the “<Numeric value>” part.
pageLinks.next<Numeric value>.char	-	Outputs the link to display the next pages. Displays the next page mentioned in “<Numeric value>” part. When 10 is specified, it displays the page, that is 10 pages af ter the current page. Multiple links can be specified by changing “<Numeric value>” part.
pageLinks.maxDirectLink Count	10	Specifies the number of Page number links to be displa yed between previous page and next page. When “5” is set to this key and when 5th page is the current page in the list of 10 pages, the page number links “3 4 **5** 6 7” are displayed on the screen.

- Property setting example

```
        pageLinks.prev10.char=Go Back 10 pages
pageLinks.prev5.char=Go Back 5 pages
pageLinks.prev1.char=Prev page
pageLinks.next1.char=Next page
pageLinks.next5.char=Go Forward 5 pages
pageLinks.next10.char=Go Forward 10 pages
pageLinks.maxDirectLinkCount=5
```

- Output example with the above-mentioned settings
  - Output example (using the above property file) displaying the 13th page of the list screen.

```
Go Back 10 pages Go Back 5 pages Prev Page 11 12 13 14 15 Next page Go For-
                                                    ward 5 pages Go Forward 10 pages
```

- Use of image link

  Link part can be set as an image link in the property file of the Page link function as shown below.
  - Property setting example

    Further, note that a link border is displayed around the image, when image link is used and “border="0"” is not mentioned.

```
        pageLinks.prev5.char=<img src="image/stepPrevious.gif" bor-
        der="0"/>
pageLinks.prev1.char=<img src="image/previous.gif" border="0"/>
pageLinks.next1.char=<img src="image/next.gif" border="0"/>
pageLinks.next5.char=<img src="image/stepNext.gif" border="0"/>
```

◆ Extension point

None.

■ Classes

	Class Name	Overview
1	jp.terasoluna.fw.web.struts.taglib.PageLinksTag	Custom tag providing link for page navigation in page unit

■ Reference

◆ Tag Attributes

Attributes	Required	Overview
Id	-	When a value is specified for this attribute, the page links are written to the Page context, and not in the screen. The value of the attribute is used as the key in Page context.
Action	△	Specifies the action path name to display the list view. This attribute is mandatory when submit attribute is false.
Name	-	Specifies the bean to acquire rows, Start index and total count of list information. When not specified, rows, Start index and list information total count is acquired directly from the scope.
rowProperty	○	Specifies the number of rows.
indexProperty	○	Specifies the start index.
totalProperty	○	Specifies the total count.
Scope	-	Specifies the scope to acquire bean specified in name attribute and 3 properties.
Submit	-	Set true for submitting without links. By default it is false.
forward	-	This attribute is valid when the submit attribute is “true”. This used to dispatch using DispatchAction of TERASOLUNA.When true is specified, the hidden tag specified in the value of the event attribute is printed. On ‘submit’, the value of the hidden tag is set to “forward_pageLinks. By default, it is false.
Event	-	This attribute is valid when true is set to forward attribute. It is used to dispatch by forward using DispatchAction of TERASOLUNA Server Framework for Java (Web version). Hidden tag with the name specified in this attribute is written. By default, it is “event”.
resetIndex	-	Hidden tags of startIndex and endIndex that are used to reset the specified range are written. By default, it is false.
currentPageIndex	-	It is the key to save current page count of corresponding list in Page context. By default, it is "currentPageIndex”.
totalPageCount	-	It is the key to save total page count of corresponding list in Page context. By default, it is "totalPageCount“.

■ Related Function(s)

"WK-09 List Display Related Function(s)"

■ Example

- Sample covering all functions of TERASOLUNA Server Framework for Java (Web version)
  - "UC24 List Display"
    - /webapps/Page link/*
    - /webapps/WEB-INF/Page link/*
    - jp.terasoluna.thin.functionsample.Page link.*

- TERASOLUNA Server Framework for Java (Web version)
  - "2.5 List Display"
  - List View

- " Spring based framework Tutorial"

■ Remarks

None.

WJ-01 ~ WK-09 Screen Display

■ Overview

◆ Functional Overview

- Function for Screen Display (Custom Tag).
- TERASOLUNA Server Framework for Java (Web version) provides 2 types of Custom Tags as given below.
  - TERASOLUNA Server Framework for Java (Web version) specific custom tags
  - Extended Struts tags

◆ List of Custom Tags provided

- Custom Tags provided by terasoluna-thinclient

Function No.	Function Name	Tag	Functional Overview
WJ-01	Access Control	<t:ifAuthorized> <t:ifAuthorizedBlock>	It toggles between Show/Hide depending on the user's privileges.
WJ-02	Server Blockage Check	<t:ifPreBlockade>	It toggles between Show/Hide on checking whether the server is blocked or not.
WJ-03	Calendar Input	<t:inputCalendar>	This is a function(s) that allows you to enter the Year/Month/Day using the calendar screen.
WJ-04	Character String Display	<t:write>	It changes the specified bean property value and displays it.
WJ-05	Date Change	<t:date>	It formats the date and time according to the specified format.
WJ-06	Date in Japanese name of era Con-version	<t:jdate>	It formats the date and time data as per the Japanese Calendar.
WJ-07	Decimal Display	<t:decimal>	It formats and outputs the symbols and numerical values with decimal point, or defines it as a scripting variable.
WJ-08	Trim	<t:rtrim> <t:ltrim> <t:trim>	It deletes the spaces contained in the specified string.
WJ-09	Substring	<t:left>	It extracts the specified number of characters from the left of the character string.
WJ-10	Define Codelist	<t:defineCodeList>	It uses the loaded Codelist in jsp.
WJ-11	Write Codelist Count	<t:writeCodeCount>	It displays the count of Codelist.
WJ-12	Display value of specified code	<t:writeCodeValue>	It displays the value for the specified code on the screen.

- Custom Tags provided by terasoluna-thinclient-struts

Function No.	Function Name	Tag	Functional Overview
WK-01	Styleclass Change	<ts:changeStyleClass>	It changes the style sheet class in event of an error.
WK-02	Message Display	<ts:errors> <ts:messages>	It displays the Message or the Error message information.
WK-03	No Cache Form Tag	<ts:form>	This adds a random ID to the Action URL for avoiding cache.
WK-04	No Cache Link	<ts:link>	This adds a random ID to the Action URL for avoiding cache.
WK-05	Form Target Specification	<ts:submit>	It specifies the Form target.
WK-06	Message Popup	<ts:messagesPopup> <ts:body>	This displays the Message or an Error message on a popup screen.
WK-07	Error Message Check	<ts:ifErrors> <ts:ifNotErrors>	It identifies whether the error information has been configured to a request or a session, and toggles between Show / Hide.
WK-08	Client Check Extension	<ts:javascript>	It outputs the value to be used for validation of input from client at time of input check, as javascript variable.
WK-09	List Display Related Function(s)	<logic:iterate> <ts:pageLinks>	It uses the list display function <logic:iterate> element of Struts. It also provides a tag required for paging.

■ WJ-01 Access Control

◆ Overview

Determines whether the currently logged on user has access privilege to the specified path and accordingly switches between show/hide.

◆ Description

- <t:ifAuthorized>
  This tag evaluates the body of tag only when there is an access privilege for the path specified by path attribute. AuthorizationController executes the access privilege check. Please refer to “WA-02 Access Control” for further information on AuthorizationController.
- <t:ifAuthorizedBlock>
  This element controls the result of <t:ifAuthorized> element for each blockId, links <t:ifAuthorized> element with blockId, and determines whether to display it in the body. For details refer to the Example
  Access control by using multiple <t:ifAuthorized> tags can be implemented as below:
  As a prerequisite, it is required that blockId of <t:ifAuthorized> contains the same value as blockId of <t:ifAuthorizedBlock> for which linking is to be established.
  - If there exists even a single access priviledge for the path specified in <t:ifAuthorized> tag, the <t:ifAuthorizedBlock> tag will be evaluated.
  - If there is no access priviledge for the path specified in <t:ifAuthorized> tag, the <t:ifAuthorizedBlock> tag will be evaluated.

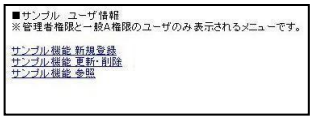
  If <t:ifAuthorizedBlock> is to be nested further, by linking the blockId attribute of parent with parentBlockId attribute of child, and determines whether to display it in the body.

◆ Example

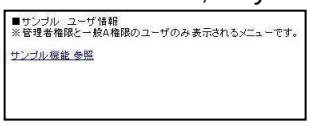
- Usage example of <t:ifAuthorized> tag and <t:ifAuthorizedBlock> tag
  To change the Menu contents on the basis of each access priviledge.
  The access prriviledges are as below:
  1. Administrator can access all paths
  2. General user A can access only "/sample/search.do" path
  3. General user B cannot access any path。

Below is the output of the above JSP

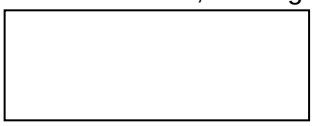
For Administrator, All links will be displayed



For General User A , only reference link will be displayed



For General User B, nothing will be displayed



For implementation details and Usage Example of Access Permissions Check functionality Refer to "WA-02 Access Permissions Check function"

◆ List of Tag Attributes

- List of Attributes of <t:ifAuthorized> element

Attribute	Mandatory	Overview
Path	○	Specifies targeted path
blockId	-	BlockId for association with <htmlx:IfAuthorizedBlock> element, which is parent of this tag is specified.

- List of attributes of <t:ifAuthorizedBlock>

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

◆ Example

- Sample covering all functions in the TERASOLUNA Server Framework for Java (Web Version)
  - "UC05 Access Authority Check"
    - /webapps/authorization/*
    - /webapps/WEB-INF/authorization/*
    - jp.terasoluna.thin.functionsample.authorization.*

WJ-02 Server Blockage Check

◆ Overview

Switches between show/hide depending on whether the server is in pre-blockage state or in 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 using ServerBlockageController.
  Please refer to “WA-02 Server Blockage Check” for further information on Server-BlockageController.

◆ Example

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

◆ List of Tag Attributes

None.

◆ Example

- Sample covering all functions of the TERASOLUNA Server Framework for Java (Web version)
  - “UC06 Server Blockage Check”
    - /webapps/serverblockage/*
    - /webapps/WEB-INF/serverblockage/*
    - jp.terasoluna.thin.functionsample.serverblockage.*

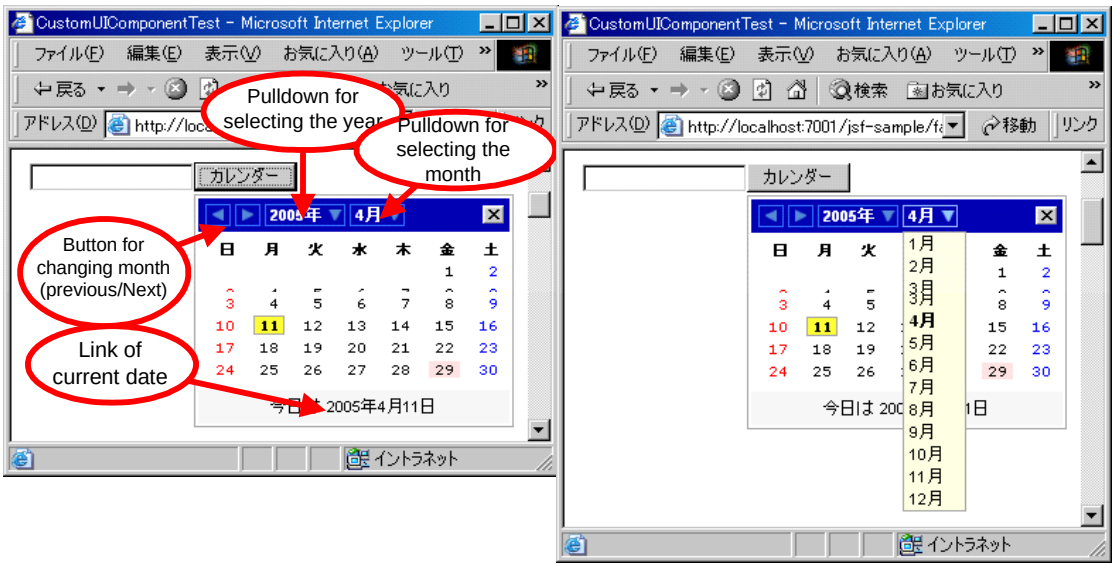
■ WJ-03 Calendar Input

◆ Overview

Provides Calendar Input Function(s) for the specified text field.

◆ Description

- <t:inputCalendar>
  Display the calendar in a screen using JavaScript, allowing to select a date in an input field



◆ Functional Details

- Configuration File
  Use message resource file as configuration file for Internationalization support
  Create message resource file with a file name as “Calendar properties” exactly under the class path

- Holiday definition
  In message resource file, holidays in the calendar can be specified as follows In message resource key “calendar.holiday.” part is fixed and after that, sequence number is added after “1”. Parameters are described by separating “Year”, ”Month”, ”Day” and ”Holiday description” with a “,(Comma)”. In case of holidays on same date every year, “0” is specified for “Year”, so that it can be identi-

fied.

```
calendar.holiday.1=0,1,1, New Year's Day
calendar.holiday.2=0,2,11, National Foundation Day
calendar.holiday.3=0,4,29, Greenery day
calendar.holiday.4=2005,1,10, Coming-of-Age Day
calendar.holiday.5=2005,3,20, Spring Equinox Day
calendar.holiday.6=2005,9,19, Respect for the Aged Day
calendar.holiday.7=2005,9,23, Autumn Equinox Day
calendar.holiday.8=2005,10,10, National-Sports Day
```

- Change 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
```

- 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 storage location of the image to be used in the calendar input function(s) can be changed by specifying the code in message resource file as given below.

  It should end with “/”. Storage 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/
```

- Change the storage location of style sheet

  The storage location of the style sheet to be used in calendar input function(s) can be changed by specifying the code in message resource file as given below.

  Storage location should end with “/”. File name of style sheet used by this function(s) is “<Prefix> + InputCalendar.css”.

  Set the message resource key, “calendar.stylesheet.dir”. Default is “css/”.

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

- Change the storage location of external JavaScript
  The storage location of external JavaScript to be used in calendar input function(s) can be changed by specifying the code in message resource file as given below. Storage location should end with "/". File name of the JavaScript used by this function(s) is "InputCalendar.js".
  Set the message resource key, "calendar.javascript.dir". Default is "js/".

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

◆ Example

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

◆ List of Tag Attributes

Attribute	Mandatory	Overview
For	○	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 "/", "-", ".", "Single byte 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 value for acquiring the calendar format from message resource.

◆ Notes

- **Attribute name "event" cannot be used**
  If an attribute with name "event" is defined in the screen using this functionality, then this functionality may behave erroneously.（Since this functionality uses event object of JavaScript）
  Specifically while setting up page navigation condition by using WE-02 Standard Dispatched functionality for radio button etc, the default forward destination is defined with the name "event". In this case, in Bean definition for DispatchAction, an event property should be set so as to change the default "event" item name.

```
<bean name="/sampleDSP" scope="prototype"
      Class="jp.terasoluna.fw.web.struts.actions.DispatchAction">
    <property name="event" value="dispatchName"/>
</bean>
```

Here "dispatchName" is the name assigned to radio button

◆ Example

- Sample covering all functions of TERASOLUNA Server Framework for Java (Web version)
  - "UC25 Calendar Input"
    - /webapps/calendar/*
    - /webapps/WEB-INF/calendar/*

- TERASOLUNA Server Framework for Java (Web version) tutorial:
  - "2.7 Registration"
  - Registration Screen

WJ-04 Character String Display

◆ Overview

Gets the value of specified bean property and displays it.

◆ Description

- <t:write>
  - ➢ Extract the specified bean property value and assign it to the current JspWriter as a String. Moreover, make the following replacements in attribute value.
  - ➢ Replace null or null character with“ ”
  - ➢ Replace single byte space with “ ”
  - ➢ Replace line feed code with
  - ➢ Ignore line feed character

◆ Example

```
Contents of primaryField
「" ABCD" + System.getProperty("line.separator") + " EFG"」

<t:write name="htmlxForm"
          property="primaryField" />
↓Output of above code
 ABCD<br> EFG
```

◆ List of Tag Attributes

Attribute	Mandatory	Overview
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 it as false..
replaceNullToNbsp	-	When this attribute is set to true and when the value of specified bean property is a Null Character or Null, is output. To disable this attribute, it is necessary to explicitly set it as false.
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 it as false.
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 it as false.
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 (consistent with the other tags in this tag library, a runtime exception is thrown).
Name	○	Specifies the name of the bean, whose property (if specified) is to be displayed. The value of this bean is displayed, if property is not specified.
property	-	Specifies the property name accessed on the bean specified by name. This value may be a simple, indexed, or nested property reference 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 single byte and double byte 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.

◆ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - "UC26 Character String Display"
    - /webapps/write/*
    - /webapps/WEB-INF/write/*
    - jp.terasoluna.thin.functionsample.write.*

■ WJ-05 Date Change

◆ Overview

Formats date and time as per the specified format.

◆ Description

- <t:date>
  Formats and renders the string as per the format specified in pattern attributes using the time pattern string of java.text.SimpleDateFormat class. For details of the time pattern string, please refer to the 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
```

◆ List of Tag Attributes

Attributes	Mandatory	Overview
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.
pattern	○	Output format to be formatted. The output format specified by pattern attribute is rendered in the subclass of DateFormatterTagBase class. For details, refer to the documents of subclass.
format	-	Format of the date If it is specified using value attribute Default value is “yyyy/MM/dd HH:mm:ss”

◆ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - “UC27 Date Change”
    - /webapps/date/*
    - /webapps/WEB-INF/date/*
    - jp.terasoluna.thin.functionsample.date.*

■ WJ-06 Date in Japanese name of era Conversion

◆ Overview

Formats date-time data in date in Japanese name of era.

◆ Description

- <t:jdate>
  When date-time data is to be formatted, converts it to Wareki Gengo (“Showa”, “S”etc) and, Wareki Year (Not Seireki “2002”, but Heisei “14” etc), Japanese notation of Yobi (“Getsuyobi”, “Getsu” etc).

◆ Functional Details

- Date in Japanese name of era Settings

Gets Date in Japanese name of era from the property file through DateUtil class.
Sets it in property file in the following format.

```
wareki.gengo.ID.name=Gengo name
wareki.gengo.ID.roman=Romaji expression of gengo
wareki.gengo.ID.startDate=Gengo start date(Year:yyyy/MM/dd format)
```

Following is the common setting example

```
wareki.gengo.0.name = Heisei
wareki.gengo.0.roman = H
wareki.gengo.0.startDate = 1989/01/08
wareki.gengo.1.name = Showa
wareki.gengo.1.roman = S
wareki.gengo.1.startDate = 1926/12/25
wareki.gengo.2.name = Taisho
wareki.gengo.2.roman = T
wareki.gengo.2.startDate = 1912/07/30
wareki.gengo.3.name =Meiji
wareki.gengo.3.roman = M
wareki.gengo.3.startDate = 1868/09/04
```

◆ 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
```

◆ List of Tag Attributes

Attributes	Mandatory	Overview
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.
pattern	-	Output format to be formated. The output format specified by pattern attribute is rendered in the subclass of DateFormatterTagBase class. Refer to JavaDoc of JDateTag for details.
format		Format of the date If it is specified using value attribute. Default value is “yyyy/MM/dd HH:mm:ss”

◆ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - "UC28 Japanese Calendar date Change"
    - /webapps/wareki/*
    - /webapps/WEB-INF/wareki/*
    - jp.terasoluna.thin.functionsample.wareki.*

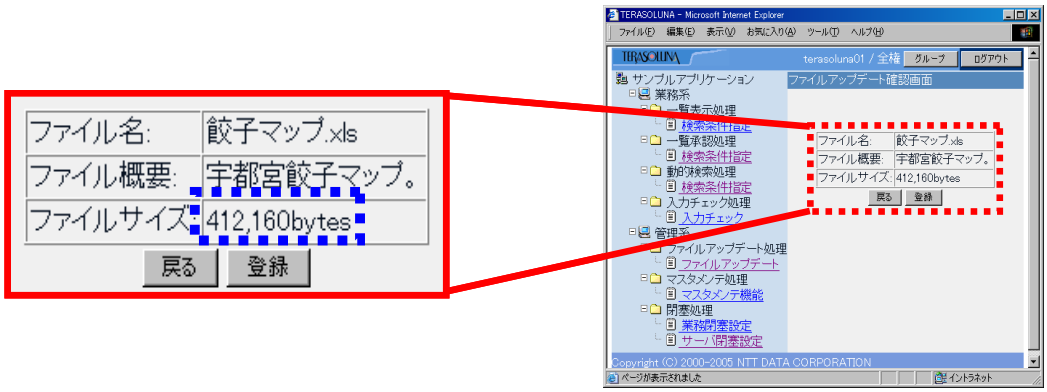
■ WJ-07 Decimal Display

◆ Overview

Formats and outputs the numeric value.

◆ Description

- <t:decimal>
  - ➢ Formats and outputs a symbol/decimal number or defines it as a scripting variable.



◆ Example

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

◆ List of Tag Attributes

Attributes	Mandatory	Overview
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 constructor 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.
scope	-	Scope while searching the bean specified by name attribute.
Value	-	String to be formatted. String can be rendered by the BigDecimal constructor 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.

◆ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - "UC29 Decimal display"
    - /webapps/decimal/*
    - /webapps/WEB-INF/decimal/*
    - jp.terasoluna.thin.functionsample.decimal.*

■ Trim

◆ Overview

Deletes single and/or double byte spaces from the specified string.

◆ Description

- <t:rtrim>
  Deletes spaces from the right side of a string.
  Example: “ string ” ⇒delete⇒ “ string”
- <t:ltrim>
  Deletes spaces from the left side of a string.
  Example: “ string ” ⇒delete⇒ “string ”
- <t:trim>
  Deletes spaces from both sides of a string.
  Example: “ string ” ⇒delete⇒ “string”
- Trim double byte space
  All single/double byte spaces can be deleted by specifying true to zenkaku attribute.

◆ Example

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

◆ List of Tag Attributes

- Common elements <t:rtrim>,<t:ltrim>,<t:trim>

Attributes	Mandatory	Overview
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 it when the bean specified by name attribute could not be found. When false is specified, JspException is thrown if bean could not be found.
Name	-	Name of bean containing the string to be formatted in property. The string representation (string returned by toString() method) of the instance specified by name attribute is to be formatted when property attribute is not specified. It is ignored when value attribute is specified.
property	-	Name of property to be accessed in bean that is specified in 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.
zenkaku	-	When this attribute is set to true, single / double byte spaces are deleted.

◆ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - "UC30 Trim"
    - /webapps/trim/*
    - /webapps/WEB-INF/trim/*
    - jp.terasoluna.thin.functionsample.trim.*

■ WJ-09 Substring

◆ Overview

■ Extracts the specified number of characters from left side of the string.

◆ Description

- <t:left>
  Substring() method of String class extracts the specified number of characters from the left side of string.

◆ Example

```
Contents of field001
""Java write once, Run anywhere.""

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

↓Output result of above code (10 characters from left side are dis-
played.)

"Java write"
```

◆ List of Tag Attributes

Attributes	Mandatory	Overview
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 it when the bean specified by name attribute could not be found. When false is specified, JspException is thrown if bean could not be found.
Name	-	Name of bean containing the string to be formatted in property. The string representation (string returned by toString() method) of the instance specified by name attribute is to be formatted when property attribute is not specified. It is ignored when value attribute is specified.
property	-	Name of property to be accessed in bean that is specified in 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.

◆ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - "UC31 Character string cutting off"
    - /webapps/left/*
    - /webapps/WEB-INF/left/*
    - jp.terasoluna.thin.functionsample.left.

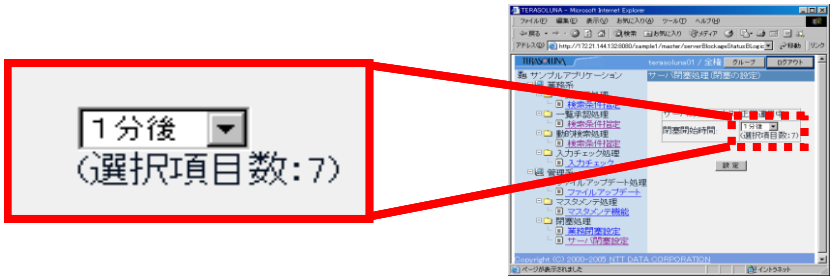
■ WJ-10 Define Codelist

◆ Overview

Used the read a codelist in jsp.
Refer to “WB-05 Codelist Function(s)” for details on reading a codelist.

◆ Description

- <t:defineCodeList>
  Gets the code list from CodeListLoader of specified id and defines it as 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><br>
(Number of selected items:<t:defineCodeCount id="loader"/>)
```

◆ List of Tag Attributes

Attributes	Mandatory	Overview
Id	○	Codelist is searched by this attribute. After tag declaration, Codelist can be referred in <logic:iterator>tag, <html:options> tag etc.

◆ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - "UC13 Codelist"
    - /webapps/codelist/*
    - /webapps/WEB-INF/codelist/*
    - jp.terasoluna.thin.functionsample.codelist.*

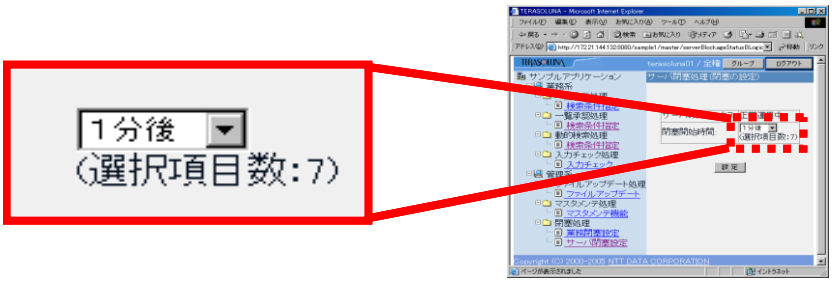
■ WJ-11 Write Codelist Count

◆ Overview

Outputs the number of elements in a Codelist. Please refer to “WB-05 Codelist Function” for details on reading codelists.

◆ Description

- <t:writeCodeCount>
  Outputs the number of elements of Codelist from the CodeListLoader of specified id.



◆ How to use

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

◆ List of Tag Attributes

Attributes	Mandatory	Overview
Id	○	Count of elements in the Codelist referred by this attribute is output. When Codelist is not found, 0 is returned.

◆ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - “UC13 Codelist”
    - /webapps/codelist/*
    - /webapps/WEB-INF/codelist/*

- jp.terasoluna.thin.functionsample.codelist.*

■ WJ-12 Display value for a specified code

◆ Overview

Displays the name for a given code in a code list. Usually, the list of codes will be displayed on one screen in a select box, the user will select an item and then the name for the selected code will be displayed on the next screen using this function..
Refer to “WB-05 Code List Function” for details on reading Codelist.

◆ Description

- <t:writeCodeValue>
  By specifying id of CodeListLoader and code value to be displayed, name for that code is displayed on the screen.

◆ Example

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

◆ List of Tag Attributes

Attributes	Mandatory	Overview
codeList	○	BeanId of CodeListLoader instance that stores the CodeBean to be output.
Key	-	Specifies the code whose display name is to be retrieved from the given codelist. If the key is not specified, then the code is re rieved from the bean name and property name
Name	-	Name of bean that stores the code whose display name is required. Not used when the key attribute is specified.
Property	-	Property of bean that stores the code whose display name is required. Not used when the key attribute is specified.
Scope	-	Scope which contains the Bean that stores the Code whose display name is required..

◆ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - "UC13 Codelist function(s)"
    - /webapps/codelist/*
    - /webapps/WEB-INF/codelist/*
    - jp.terasoluna.thin.functionsample.codelist.*

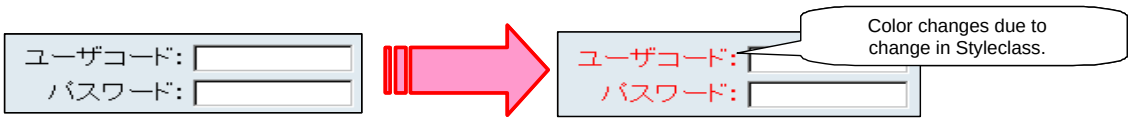
■ WK-01 Styleclass Change

◆ Overview

Changes the style sheet class, when an error occurs.

◆ Description

- <ts:changeStyleClass>
  Changes the style sheet class for a field, when the error information is set for that field in the request or session.



◆ Example

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

◆ List of Tag Attributes

Attributes	Mandatory	Overview
Name	○	Specifies the field name that determines 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.

◆ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - “UC33 Style Class change”
    - /webapps/styleclass/*
    - /webapps/WEB-INF/styleclass/*
    - jp.terasoluna.thin.functionsample.styleclass.*

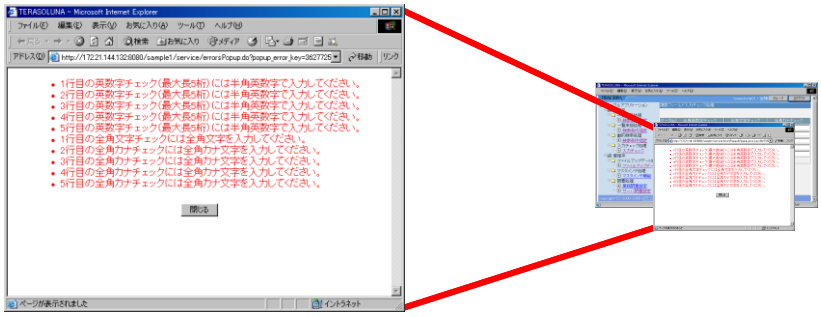
■ WK-02 Message Display

◆ Overview

Displays Message or Error Message information.

◆ Description

- <ts:errors>
  Extends ErrorsTag of Struts and gets / displays the error information in the session or request
- <ts:messages>
  Extends MessageTag of Struts and gets the information in session or request



◆ Example

- Example of <ts:errors>element

```
<ts:errors/>
<html:button property="forward_action" value="Close" on-
click="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" on-
click="window.close()"/>
```

<ts:messages>elements differ from <ts:errors>elements, and Bean in which the messages are stored should be defined for message display.

◆ Functionality Details

- Registering MessageResource file.
  Refer CE-01 Message Management Function and WG-01 Message Management

Function

- Defining MessageResource（application-message.properties）
  The display string is fetched from MessageResource using message key

application-message.properties

```
errors.alphaNumericString=Enter half width alphanumerics in {0}.
errors.hankakuKanaString=Enter kana in {0}.
errors.hankakuString=Enter hankaku string in {0}.
errors.zenkakuString=Enter full-width string in {0}.
```

Define pairs of message information and message key

- Layout at the time of displaying messages
  In the definition of message resource, by specifying the below attributes, messages can be displayed in various layouts

Attributes

・errors.header = Displayed before all the messages
・errors.footer = Displayed after all the messages
・errors.prefix = Displayed before each message
・errors.suffix = Displayed after each message

application-message.properties

```
errors.header=<div style="color.red">
errors.footer=</div>
errors.prefix=→
errors.suffix=<br/>
password.short=Password should be more than 5 characters
username.required=username is required。
```

Result of message display

```
<div style="color.red">
→username is required<br/>
→Password should be more than 5 characters<br/>
</div>
```

- Message Settings（Action class）
  All messages are added to BLogicMessages using add method. Each message is fetched from messageResource using BLogicMessage class.
  If Error information is to be displayed using </ts:errors>, store error messages in

setErrors method of BLogicResult
If message information is to be displayed using </ts:messages>, store messages in setMessages method of BLogicResult

Error message settings

```
// creating error information
BLogicMessages errors = new BLogicMessages();
errors.add(GLobals.ERROR_KEY, new BLogicMessage("errors.required",”User ID”);

//setting error information
BLogicResult result = new BLogicResult();
result.setErrors(errors);
```

◆ List of Tag Attributes

- List of attributes of <ts:errors> elemt
  Same as API of <html:errors> element

- List of Attributes of <ts:messages>element
  Same as API of <html:messages> element

  Notes for <ts:errors>element, <ts:messages>element
  In the Pop up screen display, the information in the session is not deleted unless this tag is used.

◆ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - “UC34 Message Display Function(s)”
    - /webapps/message/*
    - /webapps/WEB-INF/message/*

WK-03 No cache form tag

◆ Overview

Adds a random ID for no-cache to action URL.

◆ Description

- <ts:form>
  Extends <html: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=3336517264997268823">
 <input type="hidden" name="org.apache.struts.taglib.html.TOKEN"
        value="6235eb8a1f477895315e96be95bcc7f2">
 <div align="center">
```

◆ List of Tag Attributes

- Same as API of <html:form> element

◆ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - “UC35 Extension form･Link･submit”
    - /webapps/nocache/*
    - /webapps/WEB-INF/nocache/*
    - jp.terasoluna.thin.functionsample.nocache.*

- TERASOLUNA Server Framework for Java (Web version) Tutorial
  - Registration Screen

■ WK-04 No Cache Link

◆ Overview

Extends the <html:link> tag provided by Struts.

◆ Description

- <ts:link>
  Random ID for no cache is added to Action URL.

◆ Example

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

↓Random ID is added while displaying.

<a href="/hoge.do?r=3336517264997268823">href=/hoge.jsp</a>
```

◆ List of Tag Attributes

- Similar to API of <html:link> element

◆ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - "UC35 Extension form･Link･submit"
    - /webapps/nocache/*
    - /webapps/WEB-INF/nocache/*
    - jp.terasoluna.thin.functionsample.nocache.*

- TERASOLUNA Server Framework for Java (Web version) Tutorial
  - "2.7 Registration"
  - Registration Screen

■ WK-05 Form Target Specification

◆ Overview

Specifies the form target.

◆ Description

- <ts:submit>
  The form target is specified in the target attribute.

◆ Example

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

◆ List of Tag Attributes

Attributes	Mandatory	Overview
target	-	Specifies target destination. The Values that can be specified are as follows: • _blank. Dispays by opening a new browser • _top. Removes frame bifurcation and displays on complete • _self. displays in the same window (frame) in which the link is present • _parent. Displays in parent frame • Window/frame name. Displays in arbitrary frame/window

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

◆ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - ➢ “UC35 Extension form/Link/submit”
    - ✧ /webapps/nocache/*
    - ✧ /webapps/WEB-INF/nocache/*
    - ✧ jp.terasoluna.thin.functionsample.nocache.*

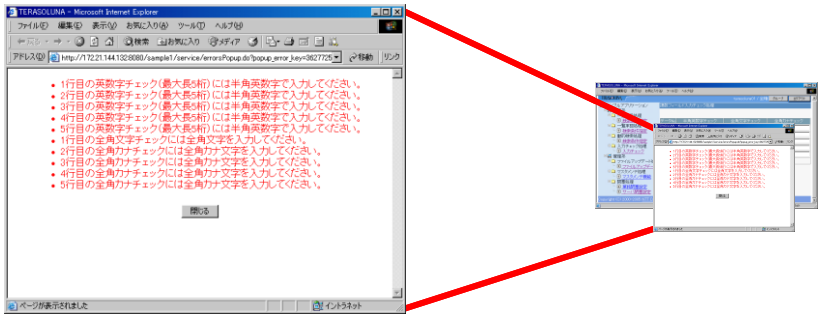
■ WK-06 Message Popup

◆ Overview

Opens the popup screen to display messages or error messages.

◆ Description

- <ts:messagesPopup>
  Sets the script for opening popup screen, to PageContex.
  To use this function(s), it should be linked with <ts:body>. <ts:messagesPopup> should be defined before <ts:body>.
- <ts:body>
  Adds the Script set by <ts:messagesPopup> tag, to the event that is executed while loading the screen.



◆ Example

```
<ts:messagesPopup popup="/service/errorsPopup.do" title="TERASOLUNA"/>
<ts:body>
↓Output of above code
<body onLoad="__onLoad__()">
  <script type="text/javascript">
  <!--
    function __onLoad__() {
      window.open("/sample1/service/errorsPopup.do"," TERASOLUNA","");
    }
  //-->
  </script>
```

◆ List of Tag Attributes

- <ts:messagesPopup>

Attributes	Mandatory	Overview
popup	○	URL to be displayed on Popup Screen. It corresponds to the first argument of window.open() of JavaScript.
title	-	Title of 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.

- <ts:body>

Attributes	Mandatory	Overview
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 the Background color.
background	-	Specifies the Image to be set at the background
text	-	Specifies the Text character color.
link	-	Specifies the Link color.
vlink	-	Specifies the visited link color
alink	-	Specifies the active link color.

◆ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - "UC32 Message Popup"
    - /webapps/popup/*
    - /webapps/WEB-INF/popup/*

WK-07 Error Message Check

◆ Overview

Determines whether the error information is set in the request or session, and accordingly switches between show/hide.

◆ Description

- <ts:ifErrors>
  Evaluates the tag body when there is a validation error or error information.
- <ts:ifNotErrors>
  Evaluates the tag body when there is no validation error and no error information.

◆ Example

```
<ts:ifErrors>
   ... // Items to be displayed when there is an error
</ts:ifErrors>
<ts:ifNotErrors>
 ... // Items to be displayed when there is an error
</ts:ifNotErrors>
```

◆ List of Tag Attributes

None.

◆ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - "UC34 Message Display"
    - /webapps/message/*
    - /webapps/WEB-INF/message/*

■ WK-08 Client Check Extension

◆ Overview

The value to be used for validation of input from client at the time of input check is output as javascript variable.
Please refer to “WF-01 Input Check Extension”, for further details on input check.

◆ Description

- <ts:javascript>
  The variables required by validation rules of Japanese system provided by TERASOLUNA Server Framework for Java (Web version) 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>
```

◆ List of Tag Attributes

Same as <html:javascript>

◆ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - “UC36 Client Check”
    - /webapps/clientvalidation/*
    - /webapps/WEB-INF/clientvalidation/*

WK-09 List Display Related Function(s)

◆ Overview

Page link function <ts:pageLinks> element can be used for the list displayed by using <logic:iterate> element provided by Struts. Multiple <ts:pageLinks> elements can be mentioned on a single screen by using List display function and Page link function.

◆ Description

- <logic:iterate>
  Strut's List display function <logic:iterate > tag is used. List information is acquired from the specified bean and the list is looped over. Specified Bean may be a Collection, ArrayList, Vector, Enumeration, Iterator, Map, HashMap, Hashtable, TreeMap, array etc. The repetitive items (<TR>~</TR> etc. in HTM table) should be specified within. Please refer to Struts for the detailed usage.
- <ts:pageLinks>
  Displays the Page navigation links of the list defined by <logic:iterate> tag. It is necessary to provide the following properties to ActionForm for using Page link function.
  - Attribute to store the display row count
  - Attribute to store start index
  - Attribute to store total count of list information

- Please refer to "WI-01 List View Function" for further details on the usage, tag attributes etc of list view related function.

◆ Example

- Sample covering all functions in TERASOLUNA Server Framework for Java (Web version)
  - "UC24 List Display"
    - /webapps/pagelink/*
    - /webapps/WEB-INF/pagelink/*
    - jp.terasoluna.thin.functionsample.pagelink.*

- TERASOLUNA Server Framework for Java (Web version) tutorial
  - "2.5 List Display"
  - List View