Step1: struts-confiq.xml

This file contains the following global-forward element which allows to specify forward name (the action) and the matching path. This allows us to specify this action anywhere in the application and to set to the path for the action specified

<global-forwards>

<forward name="gotoScrFLStartAppl" path="/default.do" redirect="true" />

<forward name="error" path="/error.do" redirect="true" />

</global-forwards>

Step1: In SSP-Access application, above mentioned global action is specified in the Index.jsp. This jsp contains the following forward tag which contains the initial forward name. Once the control servlet get a request from this JSP, it sets the default path mentioned in the element <global-forwards>

<body>

<logic:forward name=*"gotoScrFLStartAppl"* />

</body>

Step3: When the action =*"gotoScrFLStartAppl"* In the index.jsp page the path "/default" is being performed. Whenever this path is matched, the system forwarded the application to the URL "/scrflhomepage.do?performAction=init" because of the attribute of the following “<action>” element in the struts-config.xml.

<action forward="/scrflhomepage.do?performAction=init" path="/default" unknown="true" />

**Example**: The second way to use the **action** tag is shown here:

<action-mappings>

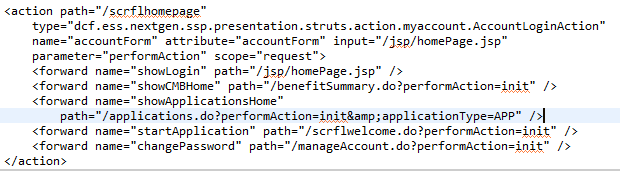
 <action path="/search"

      forward="/search.jsp"/>

</action-mappings>

This example uses the **forward** attribute to specify that the “/search.jsp” URL will be forwarded by using **RequestDispatcher.forward( )** when the specified path is matched.

Step4: Action mapping for "/scrflhomepage”



**Integrate Struts, Spring, and Hibernate**

The overall integration architecture

Struts (Webpages) 🡨🡪 Spring DI 🡨🡪 Hibernate (DAO) 🡨 🡪 Database (Oracle)

To integrate all these technologies together, we should

1. Integrate spring with Struts via spring’s ready make plug-in “ContextLoaderPlugin”. Specify this plugin in the Struts-Config.xml as shown below. In Action class, it have to extends the Spring’s “**ActionSupport**” class, and you can get the Spring bean via **getWebApplicationContext()**.

<plug-in className="org.springframework.web.struts.ContextLoaderPlugIn">

<set-property property="contextConfigLocation" value="/WEB-INF/spring\_app\_config.xml"/>

</plug-in>

In addition to the above we also need to

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/spring\_app\_config.xml</param-value>

</context-param>

<listener>

<listener-class>

org.springframework.web.context.ContextLoaderListener

</listener-class>

</listener>

<!-- Spring WS -->

<listener>

<listener-class>

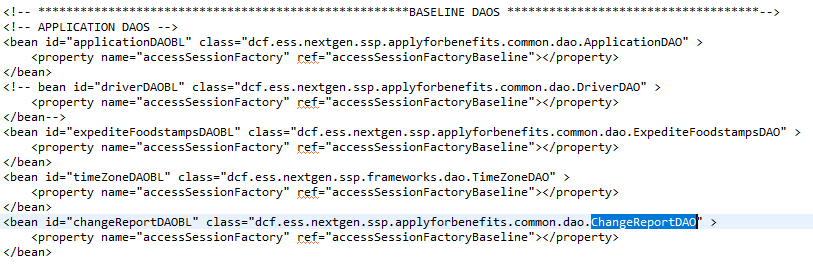
com.sun.xml.ws.transport.http.servlet.WSServletContextListener

</listener-class>

</listener>

<!-- Spring WS -->

1. Integrate spring with Hibernate with spring’s “LocalSessionFactory” bean class. In SSP access, we need to create this information in the spring.app.config.xml



1. Integrate Hibernate and the database. Create xxxxx.hbm.xml and add it in the hibernate.cfg.xml.

How to implement the SQL constraints in the Hibernate?

1. The SQL Constraints:
2. Equql (eq)
3. Not equal (ne)
4. Less than (lt)
5. Less than or equal (le)
6. Greater than (gt)
7. Greater than or equal (ge)
8. Ordering the result

Suppose we have a the following table

Create TBALE EMPLOYEE (

Id number,

Name varchar,

Ager number);

The employee.hbm.xml which maps EMPLOYEE table and Java is

<?xml version="1.0"?>

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name="Employee" table="EMPLOYEE">

<id name="id" column="id" type="long">

<generator class="increment"/> // This generates the primary key

</id>

<property name="name" column="name"/>

<property name="age" column="age" type="int"/>

</class>

</hibernate-mapping>

Coding Standard for DataAccess Layer (Java + Hybernate + Database(oracle) )

|  |  |  |
| --- | --- | --- |
| **Java Type** | **Hibernate Type** | **SQL Type** |
|  |  |  |
| Integer, int, long short | integer, long, short | number |
|  |  |  |
| char | character | char |
|  |  |  |
| java.math.BigDecimal | big\_decimal | NUMERIC, NUMBER |
|  |  |  |
| float, double | float, double | float, double |
|  |  |  |
| java.lang.Boolean, boolean | boolean | boolean, int |
|  |  |  |
|  |  |  |
| java.lang.string | string | varchar, varchar2 |
|  |  |  |
| Very long strings | text | CLOB, TEXT |
|  |  |  |
| java.util.Date | date, time, timestamp | DATE, TIME, TIMESTAMP |
|  |  |  |
| java.util.Calendar | calendar, calendar\_date | TIMESTAMP, DATE |
|  |  |  |
| java.util.Locale | locale | varchar,varchar2 |
|  |  |  |
| java.util.TimeZone | timezone | varchar, varchar2 |
|  |  |  |
| java.util Currency | Currency | varchar, varchar2 |
|  |  |  |
| java.sql.Clob | clob | CLOB |
|  |  |  |
| java.sql.Blob | blob | BLOB |
|  |  |  |
| Java object | serializable | binary field |
|  |  |  |
| byte array | binary | binary field |
|  |  |  |
| java.lang.Class | class | varchar, varchar2 |