



Laboratory Information Management System

Code Base



Struts Work Flow

Struts is purely based on the Model-View-Controller (MVC) architecture

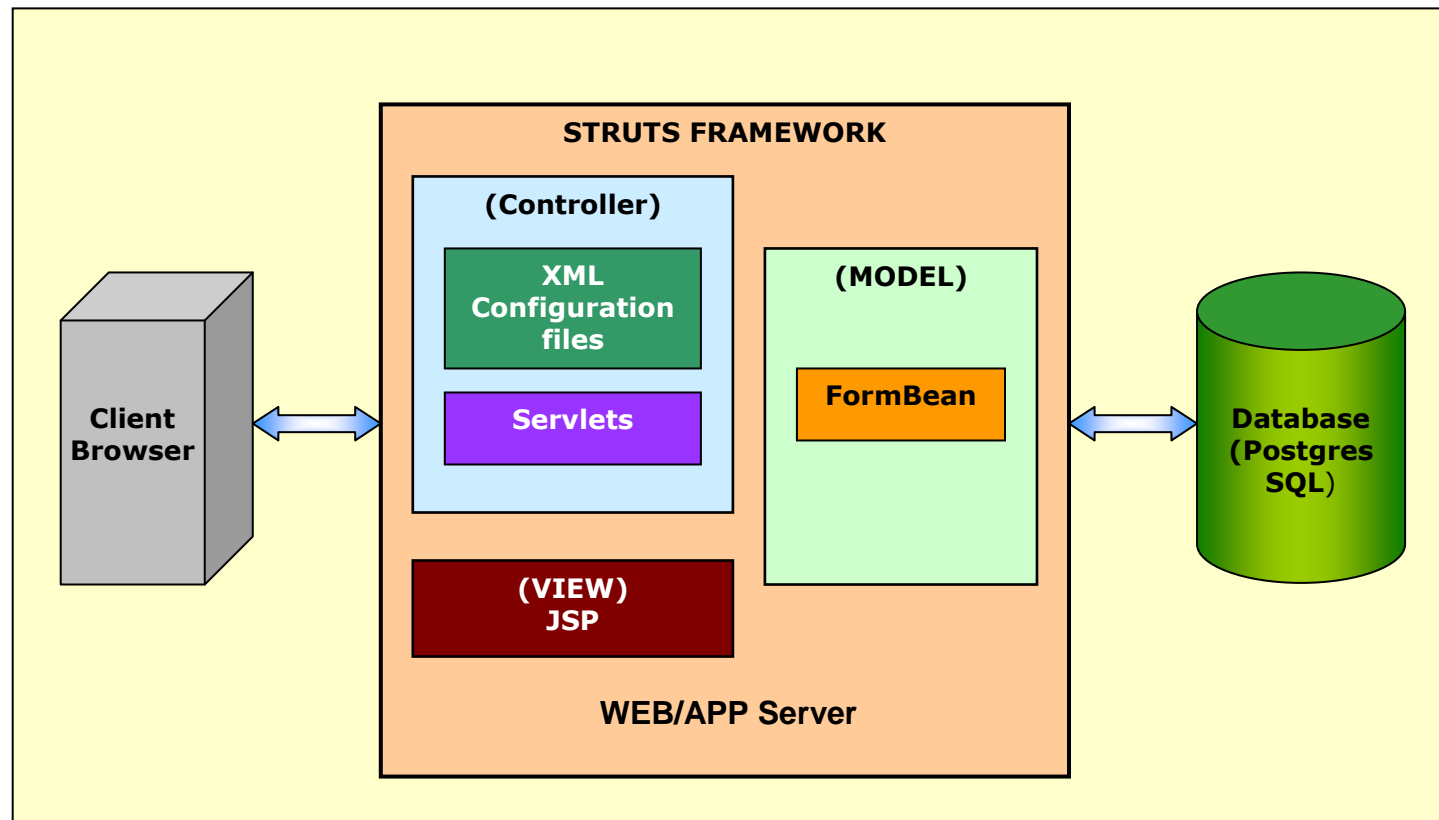
Model - application data and the business rules of an application.

View - represents the presentation of the application.

Controller - Action Servlet acts as the Controller.



Struts Work Flow





ActionServlet

- It is the main Controller component that handles client requests
- process() method is invoked.

```
<html:form action="/selectpsc" onsubmit="return  
validateSelectpsc(this)">
```

```
<action input="/jsp/login/ProjectStudyCrop.jsp" path="/selectpsc"  
scope="session" name="NewExperimentFormBean"  
type="org.icrisat.lims.startup.login.DetailsAction">  
  <forward name="psc" path="/jsp/common/mainlayout.jsp"/>  
</action>
```



ActionServlet

```
<form-bean name="NewExperimentFormBean" dynamic="true"
  type="org.apache.struts.action.DynaActionForm">
  <form-property name="project" type="java.lang.String"/>
  <form-property name="study" type="java.lang.String"/>
  <form-property name="crop" type="java.lang.String"/>
</form-bean>
```

- validate() method is invoked.
- execute() method determine the target of a transaction.



Action

- It is also a Controller component.

Extending & Configuring Action Class:

1. every class of an application extends the class `org.apache.struts.action.Action`.
2. `execute()` method with business logic.



Action

```
public class LoginAction extends org.apache.struts.action.Action
{
    public ActionForward execute(ActionMapping am,
        ActionForm af, HttpServletRequest req,
        HttpServletResponse res) throws Exception {
        -----
        -----
        return am.findForward("abc");
    }
}
```



Action

3. action element in struts-config.xml file describing new Action.

```
<action-mappings>  
  <action type="org.icrisat.lims.startup.login.LoginAction"  
    path="/login" input="/jsp/login/LoginForm.jsp"  
    name="loginform" validate="true">  
    <forward name="frontpage" path="/jsp/login/ProjectStudyCrop.jsp"/>  
    <forward name="frontpage1" path="/jsp/login/Adduser.jsp"/>  
  </action>  
</action-mappings>
```



struts-config.xml

- It is the central location for all of the application's configuration settings.

```
<struts-config>
  <data-sources>
    <data-source>
  </data-sources>
  <form-beans>
    <form-bean />
  </form-beans>
  <global-forwards>
    <forward />
  </global-forwards>
  <action-mappings>
    <action/>
  </action-mappings>
  <controller />
  <message-resource />
  <plug-in />
</struts-config>
```



Data-Sources

- This element contains n -number of `<data-source/>` subelements, which actually describe each DataSource instance.
- DataSource is a factory containing JDBC connection objects.



Data-Sources

We initialize the DataSource in struts config file as shown below:

```
<data-sources>
  <data-source>
    <set-property property = 'driverClass' value='org.postgresql.Driver' />
    <set-property property = 'url' value='jdbc:postgresql://localhost:5432/LIMS' />
    <set-property property = 'maxCount' value='10' />
    <set-property property = 'minCount' value='1' />
    <set-property property = 'user' value='postgres' />
    <set-property property = 'password' value='postgres' />
  </data-source>
  - - - - -
</data-sources>
```



Data-Sources

In the action class we create connection as follows:

```
try{
    ServletContext context = servlet.getServletContext();
    DataSource dataSource =
        (DataSource)context.getAttribute(Globals.DATA_SOURCE_KEY);
    Connection con=dataSource.getConnection();
    Statement st=con.createStatement();
    ----
    ----
    ----
}
```



Form-Beans

- An instance of a subclass of an ActionForm class, which stores input data from a Struts action link that a user clicked.



Form-Beans

DynaActionForm - eliminates the need of creating a FormBean class by the developer.



Form-Beans

- The form bean definition in struts-config.xml file using DynaActionForm is as follows:

```
<form-beans>
<form-bean name="loginform" type="org.apache.struts.action.DynaActionForm">
    <form-property name="uname" type="java.lang.String"/>
    <form-property name="pass" type="java.lang.String"/>
</form-bean>
-
-
</form-beans>
```



Form-Beans

- Including form-bean under the action tag:

```
<action input="/jsp/login/LoginForm.jsp" name="loginform" path="/login"
        scope="session" type="org.icrisat.lims.startup.login.LoginAction"
        validate="true">
    <forward name="frontpage" path="/jsp/login/ProjectStudyCrop.jsp"/>
    <forward name="frontpage1" path="/jsp/login/Adduser.jsp"/>
</action>
```



Form-Beans

- Form data can be retrieved in action class as follows:

```
DynaActionForm dyna=(DynaActionForm)af;  
String user=(String)dyna.get("uname");  
String pass=(String)dyna.get("pass");
```



Form-Beans

DynaValidatorForm - extends DynaActionForm and provides basic default behaviour of Validate() method.

```
<form-bean name="UploadRoboticsDynaForm"  
           type="org.apache.struts.validator.DynaValidatorForm">  
  <form-property name="volume" type="java.lang.String"/>  
  <form-property name="excelfile" type="org.apache.struts.upload.FormFile"/>  
</form-bean>
```



Form-Beans

- The following code is from validation.xml & ApplicationResources.properties file to validate using DynaValidatorForm.

ApplicationResources.properties:

```
csvfile.required=Please upload the .csv file
```



Form-Beans

Validation.xml:

```
<form name="QuantificationTracking6">
  <field depends="required,filetype" property="excelfile">
    <msg key="csvfile.required" name="required"/>
    <var>
      <var-name>fileformat</var-name>
      <var-value>.csv</var-value>
    </var>
  </field>
</form>
```



Global-Forwards

- Acts as a container for public `<forward />` subelements of an Action.
- The syntax of the `<global-forwards/>` subelement, including a sample nested `<forward/>` element, is shown here:

```
<global-forwards>  
    <forward name="Lcss" path="/jsp/common/Lcss.css"/>  
    <forward name="logout" path="/jsp/common/SessionTimeout.jsp"/>  
</global-forwards>
```



Global-Forwards

In every JSP page

```
<head>  
<link rel="stylesheet" href="<html:rewrite forward='Lcss'/>" type="text/css">  
</head>
```

In every action class

```
if((String)session.getAttribute("user")==null)  
{  
    String msg="logout";  
    return am.findForward(msg);  
}
```



Action Mappings

- Acts as a container for n- number of action subelements.
- The `<action/>` subelement has been used to describe an Action instance to the ActionServlet.



Action Mappings

- The syntax of the `<action-mappings/>` subelement, including a sample `<action />` subelement, is shown here:

```
<action-mappings>
  <action input="/jsp/login/LoginForm.jsp" name="loginform" path="/login"
          scope="session" validate="true"
          type="org.icrisat.lims.startup.login.LoginAction">
    <forward name="frontpage" path="/jsp/login/ProjectStudyCrop.jsp"/>
    <forward name="frontpage1" path="/jsp/login/Adduser.jsp"/>
  </action>
  -----
  -----
</action-mappings>
```



Message Resources

- Used to define the location of the resource bundle file.

```
<message-resources parameter="/ApplicationResources"/>
```



Plug-in

- enhances the performance of an application.

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



Validator Framework

- provides functionality to validate form data.
- Validates data on user browser as well as on server side.



Using Validator Framework

- The **validator-rules.xml** declares and assigns logical names to the validator routines.
- The **validation.xml** defines the validations applied to a form bean.



validator-rule.xml

- contains the client-side javascript code for each validation routine.
- Supplied with predefined set of commonly used validation rules.
- Basic set of rules can be extended with custom validator.



validation.xml

- defines which validation routines is used to validate Form Beans.
- The formset element contains multiple `<form>` elements.
- The field element inside the form element defines the validations to apply to specified form fields.



Example of validation.xml file

```
<formset>
  <form name=" QuantificationTracking6">
    <field depends="required,mask" property="volume">
      <msg key="volume.required" name="required"/>
    <var>
      <var-name>mask</var-name>
      <var-value>^[0-9.]*$</var-value>
    </var>
    </field>
    .....
  </form>
  .....
</formset>
```



Resource Bundle

- Message class containing Locale dependent strings.
- To include resource bundle we have added the following code to struts-config.xml



Client Side Validation

- validates the user data on the browser.
- Steps for client side validation
 - Enabling the Validator plug-in
 - Creating Message Resources
 - Developing the Validation rules
 - Applying rules to jsp



Enabling the Validator plug-in

- makes the Validator available to the system.

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



Creating Message Resources

- Message resources are used to generate the validation error messages.

labno.required=Enter Lab Number

year.required=Year is required

vol.required=Please enter the Total Volume



Developing the Validation rules

- define validation rules in the validation.xml for our form. These rules are used for generating the JavaScript for validation.

```
<form name="PCRPlate_PrepR">  
  <field depends="required" property="selPlates">  
    <arg0 key="selPlates" name="required"/>  
  </field>  
</form>
```



Applying rules to jsp

- Our jsp code contains two important tags that are very useful in validating a form. The tags are

```
<html:javascript formName="PCRPlate_PrepR"/>
```

```
<html:form method="POST" action="/PCRPlate_PrepR"  
onsubmit="return validatePCRPlate_PrepR(this)">
```



Error Management

- The `ActionError` class
- The `ActionErrors` class
- The `<html:errors />` tag



ActionError

- Represents an encapsulation of an error message.
- Example of constructing an ActionError

```
ActionError error=new ActionError("prompt.invalid.password");
```



ActionErrors

- acts as a container for a collection of ActionError instances and represents a collection of ActionError classes.
- an example of constructing an ActionError

```
ActionErrors ae=new ActionErrors();  
ActionError error=new ActionError("prompt.invalid.password")  
ae.add("myerrors",error);  
saveErrors(req, ae);  
return (new ActionForward(am.getInput()));
```



Error Management

```
<action input="/jsp/login/LoginForm.jsp"
  name="loginform"
  path="/login" scope="session"
  type="org.icrisat.lims.startup.login.LoginAction"
  validate="true">
  <forward name="frontpage"
    path="/jsp/login/ProjectStudyCrop.jsp"/>
  <forward name="frontpage1"
    path="/jsp/login/Adduser.jsp"/>
</action>
```



bean:message

- retrieve keyed values from resource bundle
- Example Snippet for `<bean:message>`

```
<bean:message key="error.nameExists"/>
```



Tag Libraries

- Bean Tag Library
- HTML Tag Library
- Logic Tag Library



Tag Libraries

- To use these tag library add the `<taglib>` sub element to the web.xml

```
<taglib>
```

```
<taglib-uri>/WEB-INF/struts-bean.tld</taglib-uri>
```

```
<taglib-location>/WEB-INF/struts-bean.tld</taglib-location>
```

```
</taglib>
```



Tag Libraries

- taglib directive for jsp

```
<%@ taglib uri="/WEB-INF/struts-bean" prefix="bean" %>
```

```
<%@ taglib uri="/WEB-INF/struts-html" prefix="html" %>
```

```
<%@ taglib uri="/WEB-INF/struts-logic" prefix="logic" %>
```



Bean Tag Libraries

- encapsulate logic necessary to access and manipulate JavaBeans, HTTP cookies, and HTTP headers using scripting variables.
- The custom tags of Bean tag library that we used in LIMS are
 - `<bean:define />`
 - `<bean:message />`
 - `<bean:parameter />`
 - `<bean:write />`



HTML Tag Libraries

- Helpful when creating HTML-based user interfaces
- The custom tags in HTML tag library that we used in LIMS are
 - **<html:rewrite/>**
 - **<html:file/>**
 - **<html:link/>**
 - **<html:submit/>**
 - **<html:text/>.....etc.**



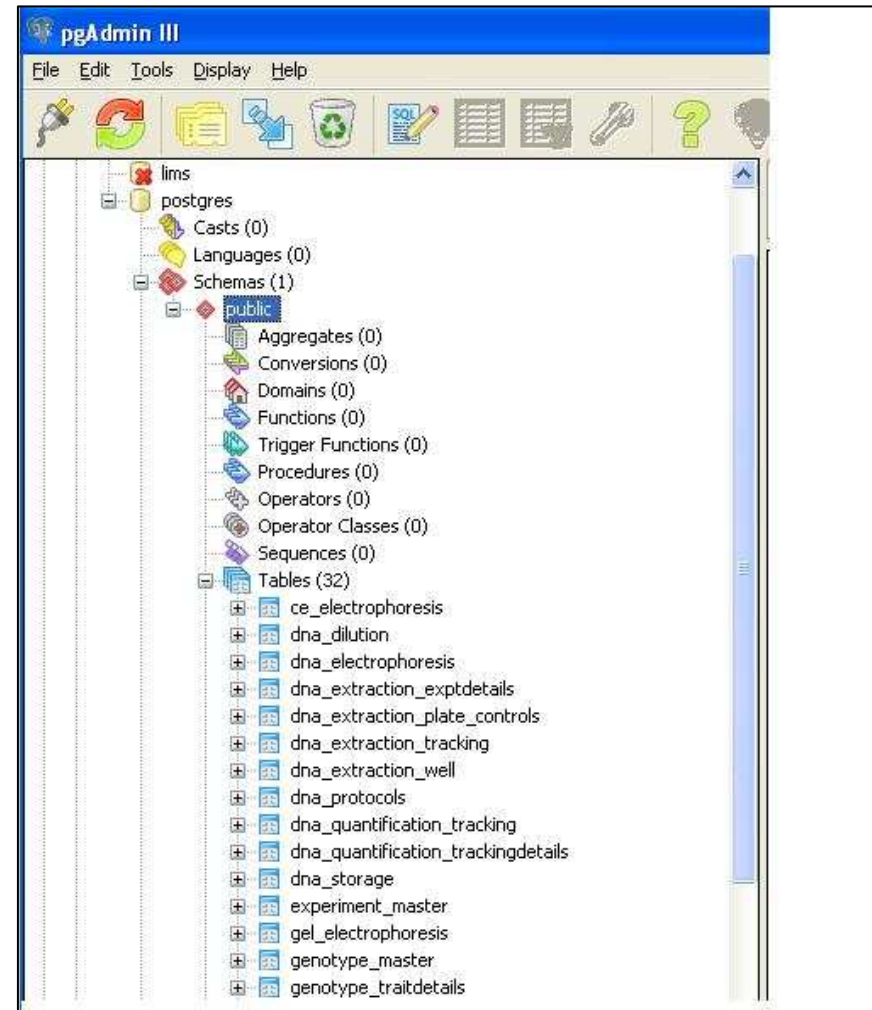
Logic Tag Library

- The focus of the Logic tag library is on decision-making and object evaluation.
- The custom tags in Logic tag library that we used in LIMS are
 - **<logic:empty />**
 - **<logic:notEmpty />**
 - **<logic:iterate />**
 - **<logic:equal/>**
 - **<logic:notEqual />**



Database structure

- PostgreSQL 8.1.3
- 32 tables
- 2 views





Eclipse

- Eclipse 3.2.1
- ExadelStudio 4.0.3 plugin.
- External jar files
 - postgresql-8.1-405.jdbc3.jar
 - ostermillerutils_1_05_00.jar
 - Jxl.jar



Naming Conventions

The screenshot displays the Exadel Studio IDE interface. The left-hand side shows a project navigator with a tree structure for 'NLIMS'. The main editor window is open to 'struts-config.xml', showing the following XML code:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE struts-config PUBLIC
"-//Apache Software Foundation//DTD Struts Configura
"http://jakarta.apache.org/struts/dtds/struts-config

<struts-config>
  <data-sources>
    <data-source>
      <set-property property = 'driverClass' value='org.
      <set-property property = 'url' value='jdbc:postgre
      <set-property property = 'maxCount' value='10' />
      <set-property property = 'minCount' value='1' />
      <set-property property = 'user' value='postgres' />
      <set-property property = 'password' value='postgre
    </data-source>
  </data-sources>

  <form-beans>
    <!-- For loginForm -->
    <form-bean name="loginform" type="org.apache.struts.ac
      <form-property name="pass" type="java.lang.String"
      <form-property name="uname" type="java.lang.String
    </form-bean>

    <!-- For change password page-->
    <form-bean name="changepasswd" type="org.apache.stru
      <form-property name="oldpass" type="java.lang.Stri
```



Thank You ...