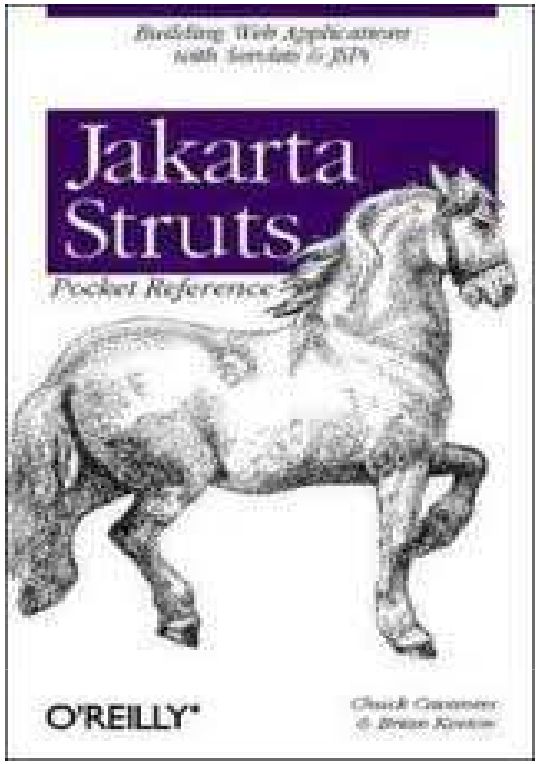


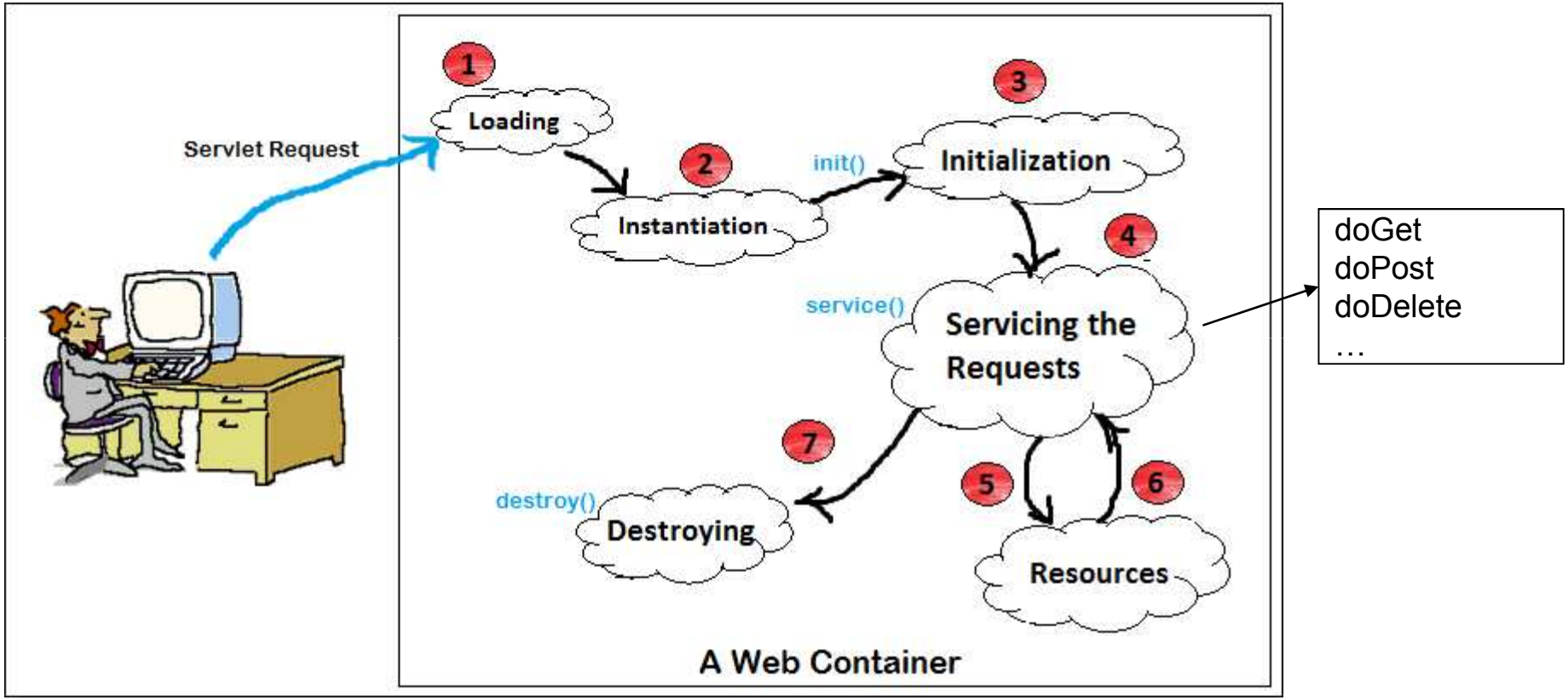
STRUTS 1.3

Dhanya Kumar KV

Agenda

1. Life Cycle of Servlet
2. Struts configuration files
3. Framework API: Class diagram and sequence diagram
4. Struts Tags
5. Using Tiles
6. Login example

Life Cycle Of Servlet



web.xml

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

<servlet>
     <servlet-name>ActionServlet</servlet-name>
     <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
     <init-param>
          <param-name>config</param-name>
          <param-value>
               /WEB-INF/classes/struts-config.xml,
               /WEB-INF/classes/struts-config-mappings.xml
          </param-value>
     </init-param>
     <load-on-startup>2</load-on-startup>
</servlet>
<servlet-mapping>
     <servlet-name>ActionServlet</servlet-name>
     <url-pattern>*.do</url-pattern>
</servlet-mapping>

<error-page>
     <exception-type>java.lang.Throwable</exception-type>
     <location>/error.do</location>
</error-page>
```

Struts-config.xml

```
<!DOCTYPE struts-config PUBLIC
   "-//Apache Software Foundation//DTD Struts Configuration 1.3//EN"
   "http://struts.apache.org/dtds/struts-config_1_3.dtd">

<struts-config>
    <global-exceptions>
        <exception key="MSG_000005" type="java.lang.Exception"
        handler="com.action.GlobalExceptionHandler" />
    </global-exceptions>

    <controller processorClass="com.cdrs.web.struts.StrutsRequestProcessor" />

    <message-resources parameter="com/cdrs/web/i18n/ResourceBundle" null="true" />

    <plug-in className="org.apache.struts.tiles.TilesPlugin">
        <set-property property="definitions-config"
                          value="/WEB-INF/classes/tiles-defs.xml" />
        <set-property property="moduleAware" value="true" />
    </plug-in>

    <plug-in className="org.springframework.web.struts.ContextLoaderPlugIn">
        <set-property property="contextConfigLocation"
                        value="/WEB-INF/classes/applicationContext.xml" />
    </plug-in>
</struts-config>
```

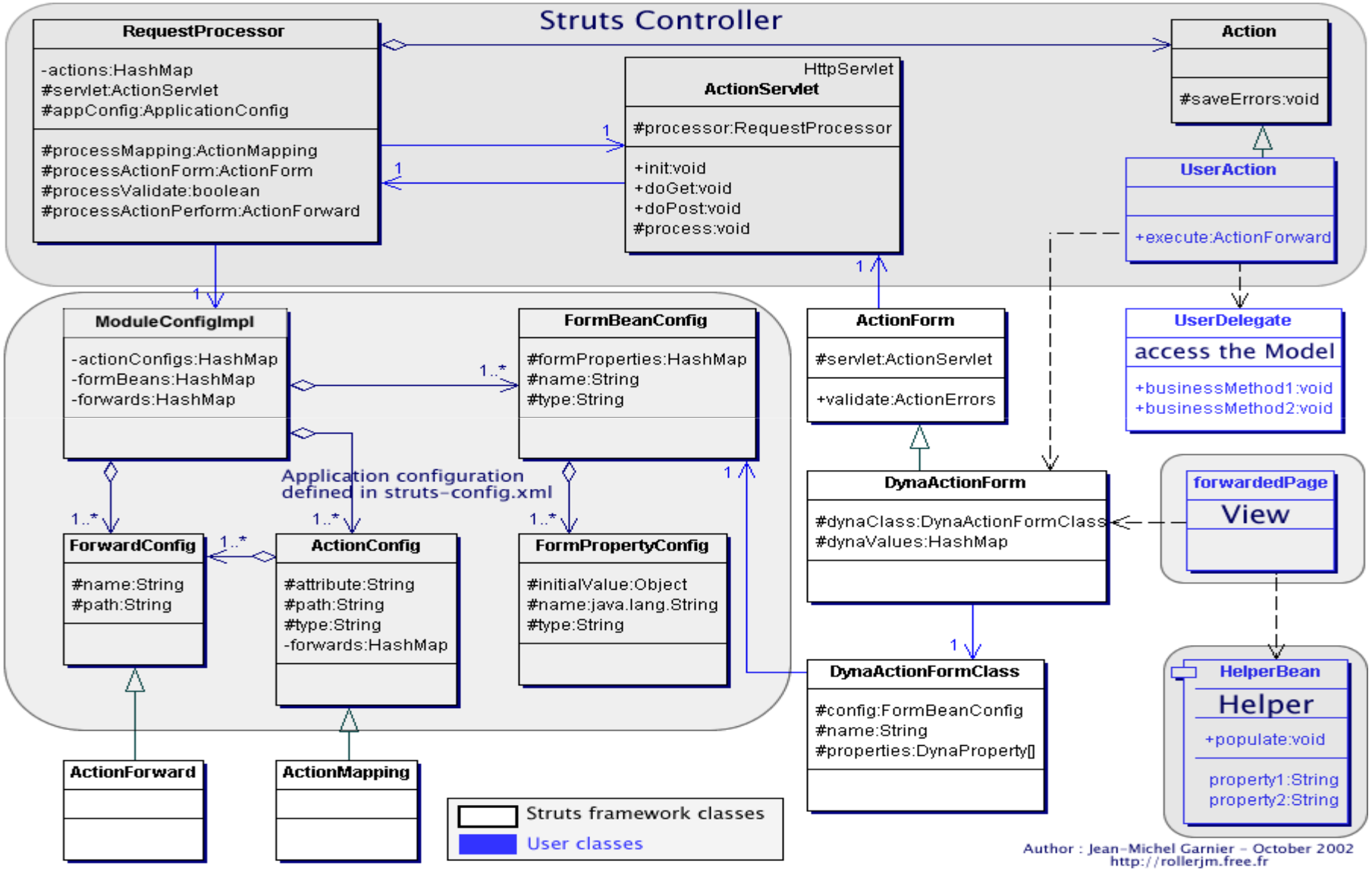
Struts-config-mappings.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!DOCTYPE struts-config PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 1.3//EN"
    "http://struts.apache.org/dtds/struts-config_1_3.dtd">

<struts-config>

    <form-beans>
        <form-bean name="userAccountForm" type="com.cdrs.beans.UserAccountForm" />
    </form-beans>

    <action-mappings>
        <action path="/Login" name="userAccountForm" scope="request"
              validate="true" parameter="method" input="/WEB-INF/view/index.jsp"
              type="com.oms.controller.action.useraccount.UserAccountAction">
              <forward name="FAILURE" path="loginpage" />
              <forward name="SUCCESS" path="homepage" />
        </action>
    </action-mappings>

</struts-config>
```

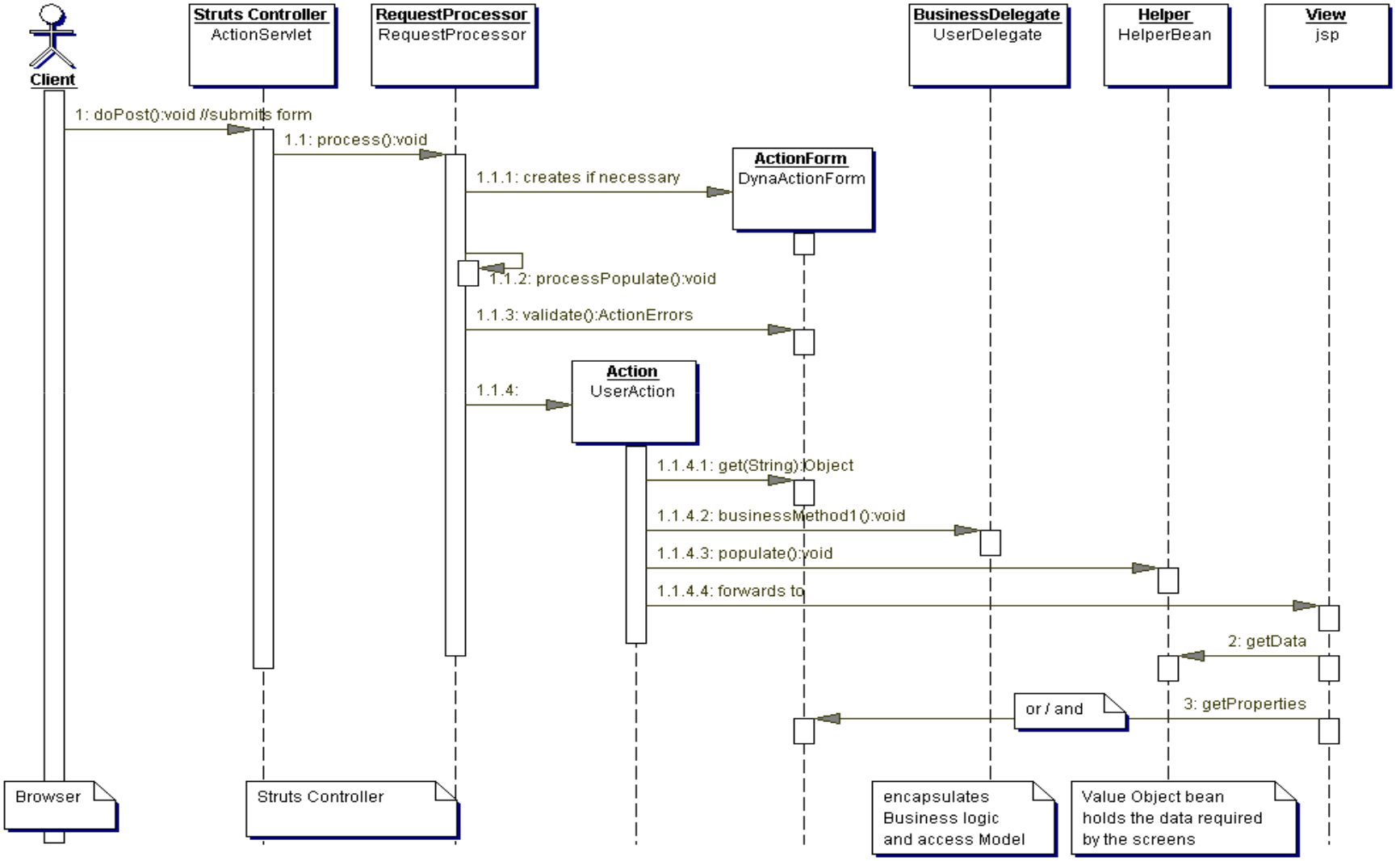
Tiles-def.xml

```
<tiles-definitions>

    <definition name="baseLayout" path="/WEB-INF/view/common/baseLayout.jsp">
        <put name="pageTitleKey" value="LABEL_000024" />
        <put name="pageHeaderKey" value="LABEL_000024" />
        <put name="header" value="/WEB-INF/view/common/header.jsp" />
        <put name="menu" value="/WEB-INF/view/login/loginLeftMenu.jsp" />
        <put name="body" value="/WEB-INF/view/login/login.jsp" />
        <put name="footer" value="/WEB-INF/view/common/footer.jsp" />
    </definition>

    <definition name="homepage" extends="baseLayout">
        <put name="pageTitleKey" value="LABEL_000030" />
        <put name="pageHeaderKey" value="LABEL_000030" />
        <put name="menu" value="/WEB-INF/view/common/menu.jsp" />
        <put name="body" value="/WEB-INF/view/item/showItems.jsp" />
    </definition>

    <definition name="loginpage" extends="baseLayout">
        <put name="pageTitleKey" value="LABEL_000024" />
        <put name="pageHeaderKey" value="LABEL_000024" />
        <put name="body" value="/WEB-INF/view/login/login.jsp" />
    </definition>

</tiles-definitions>
```

Struts API


Struts Controller
RequestProcessor
-actions:HashMap
#servlet:ActionServlet
#appConfig:ApplicationConfig
#processMapping:ActionMapping
#processActionForm:ActionForm
#processValidate:boolean
#processActionPerform:ActionForward
HttpServlet
ActionServlet
#processor:RequestProcessor
+init:void
+doGet:void
+doPost:void
#process:void
Action
#saveErrors:void
UserAction
+execute:ActionForward
UserDelegate
access the Model
+businessMethod1:void
+businessMethod2:void
ModuleConfigImpl
-actionConfigs:HashMap
-formBeans:HashMap
-forwards:HashMap
FormBeanConfig
#formProperties:HashMap
#name:String
#type:String
ActionForm
#servlet:ActionServlet
+validate:ActionErrors
Application configuration
defined in struts-config.xml
ForwardConfig
#name:String
#path:String
ActionConfig
#attribute:String
#path:String
#type:String
-forwards:HashMap
FormPropertyConfig
#initialValue:Object
#name:java.lang.String
#type:String
DynaActionForm
#dynaClass:DynaActionFormClass
#dynaValues:HashMap
forwardedPage
View
DynaActionFormClass
#config:FormBeanConfig
#name:String
#properties:DynaProperty[]
HelperBean
Helper
+populate:void
property1:String
property2:String
ActionForward
ActionMapping
Struts framework classes
User classes
Author : Jean-Michel Garnier - October 2002
http://rollerjm.free.fr

Struts Flow


Client
Struts Controller
ActionServlet
RequestProcessor
RequestProcessor
BusinessDelegate
UserDelegate
Helper
HelperBean
View
jsp
1: doPost():void //submits form
1.1: process():void
1.1.1: creates if necessary
ActionForm
DynaActionForm
1.1.2: processPopulate():void
1.1.3: validate():ActionErrors
1.1.4:
Action
UserAction
1.1.4.1: get(String):Object
1.1.4.2: businessMethod1():void
1.1.4.3: populate():void
1.1.4.4: forwards to
2: getData
3: getProperties
or / and
Browser
Struts Controller
encapsulates
Business logic
and access Model
Value Object bean
holds the data required
by the screens

ActionServlet

```
public class ActionServlet extends HttpServlet {

 public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        process(request, response);
 }

 public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        process(request, response);
 }

 protected void process(HttpServletRequest request,HttpServletResponse response)
        throws IOException, ServletException {

        ModuleUtils.getInstance().selectModule(request, getServletContext());
        ModuleConfig config = getModuleConfig(request);
        RequestProcessor processor = getProcessorForModule(config);
        if (processor == null) {
            processor = getRequestProcessor(config);
        }
        processor.process(request, response);
 }
}
```

RequestProcessor

```
public class RequestProcessor {
  public void process(HttpServletRequest request, HttpServletResponse response)
                                          throws IOException, ServletException {

        // Identify the path component we will use to select a mapping
        String path = processPath(request, response);

        // Identify the mapping for this request
        ActionMapping mapping = processMapping(request, response, path);

        // Process any ActionForm bean related to this request
        ActionForm form = processActionForm(request, response, mapping);
        processPopulate(request, response, form, mapping);
        processValidate(request, response, form, mapping)

        // Create or acquire the Action instance to process this request
        Action action = processActionCreate(request, response, mapping);

        // Call the Action instance itself. Invokes execute(..) method
        ActionForward forward =
            processActionPerform(request, response, action, form, mapping);

        // Process the returned ActionForward instance
        processForwardConfig(request, response, forward);
}
```

Action

```
public class Action {

 // override this method in used defined Action class.
 public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws Exception {
        return null;
 }

 protected void addMessages(HttpServletRequest request,ActionMessages messages) {
        // get any existing messages from the request, or make a new one
        // add incoming messages to ActionMessages object and Save the messages
        request.setAttribute(Globals.MESSAGE_KEY, requestMessages);
 }
 protected void addErrors(HttpServletRequest request, ActionMessages errors) {
         request.setAttribute(Globals.ERROR_KEY, requestErrors);
 }
 protected void saveErrors(HttpServletRequest request, ActionMessages errors) {
       // Remove any error messages attribute if none are required i.e.errors=null
       // Otherwise Save the error messages we need
       request.setAttribute(Globals.ERROR_KEY, errors);
 }
 protected void setLocale(HttpServletRequest request, Locale locale) {
         session.setAttribute(Globals.LOCALE_KEY, locale);
 }
```

Struts HTML Tags

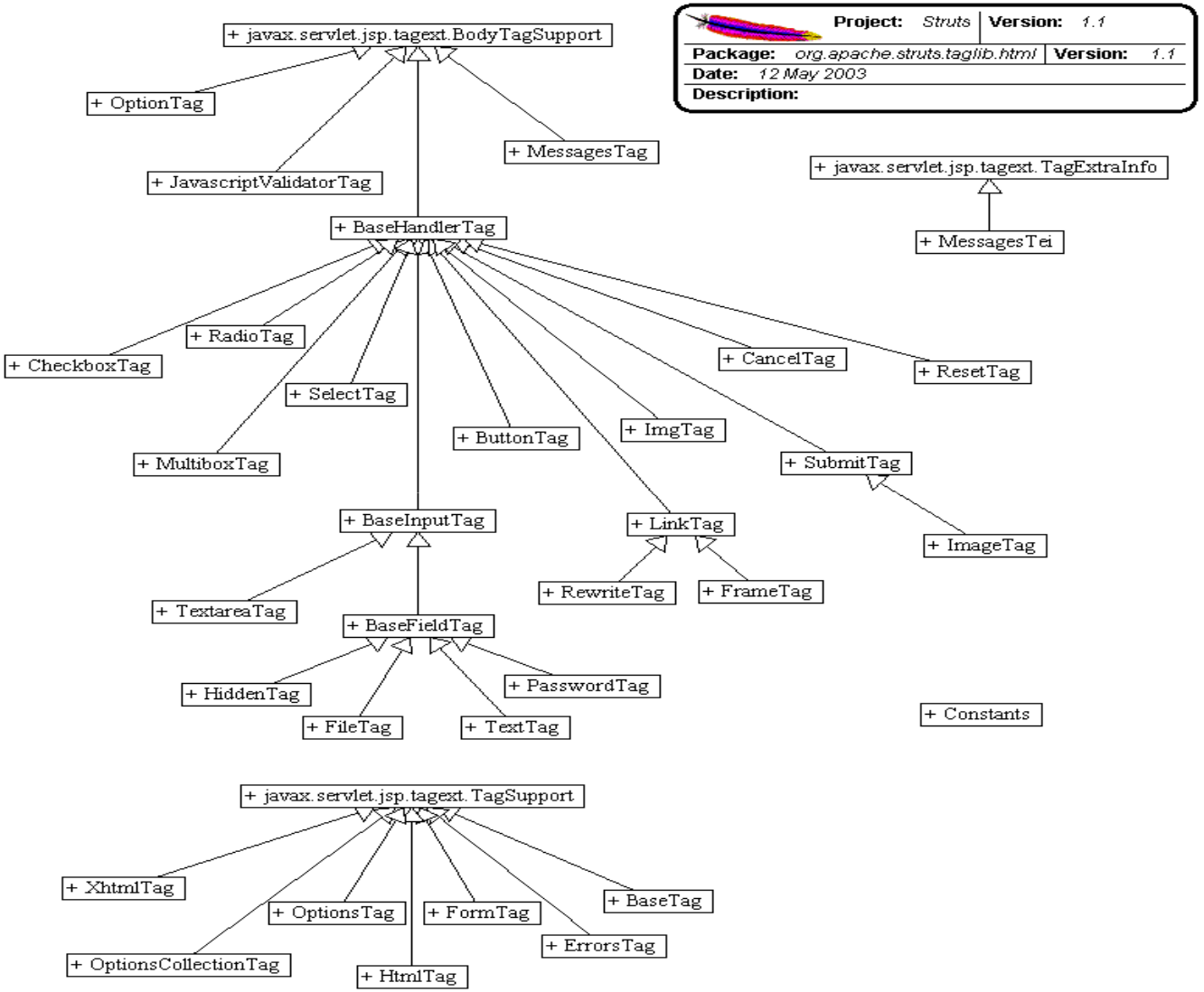
<html:message key="thekey"/>	Looks up the message corresponding to the given key in the message resources and displays it.
<html:password property="prop" size="10"/>	Tag creates the password field. The string is stored in the property named prop in the form bean.
<html:text property="text1" size="5"/>	Tag creates the text field. The string is retrieved from and later stored in the property named text1 in the form bean.
<html:submit>Submit</html:submit>	Tag creates a submit button with the provided content as the button text.
<html:reset>Reset</html:reset>	Tag creates a reset button with the provided content as the button text.
<html:errors/>	Tag prints all the available error on the page.
<html:file property="fileSelectionBox"/>	Tag creates the file upload element on the form. The property must be of the type org.apache.struts.upload.FormFile.
<html:checkbox property="myCheckBox"/>	Tag creates check box on the form.
<html:hidden property="hiddenfield"/>	Tag creates the hidden html element on the form.
<html:radio value="abc" property="myCheckBox"/>	Tag creates the check box on the form.
<html:select multiple="true" property="selectBox">	Tag creates list box on the form. The property selectBox must be an array of supported data-types, and the user may select several entries. Use <html:options> to specify the entries.
<html:textarea property="myTextArea" value="Hello Struts" />	Tag creates the text area on the form.
<html:form action="/Address" method="post">	Tag is used to create the HTML Form for posting the data on the server.
<html:base/>	Tag generates the base tag. <BASE ...> tells the browser to pretend that the current page is located at some URL other than where the browser found it. Any relative reference will be calculated from the URL given by <BASE HREF="..."> instead of the actual URL. <BASE ...> goes in the <HEAD> section.
<html:html>	Tag renders an HTML <html> Element.

http://www.roseindia.net/struts/strutsHtmlTags.shtml

Javascript Event Handlers

onblur	Executed when this element loses input focus.
onchange	Executed when this element loses input focus and its value has changed.
onclick	Executed when this element receives a mouse click.
ondblclick	Executed when this element receives a mouse - double click.
onfocus	Executed when this element receives input focus.
onkeydown	Executed when this element has focus and a key is depressed.
onkeypress	Executed when this element has focus and a key is depressed and released
onkeyup	Executed when this element has focus and a key is released
onmousedown	Executed when this element is under the mouse pointer and a mouse button is depressed.
onmousemove	Executed when this element is under the mouse pointer and the pointer is moved.
onmouseout	Executed when this element was under the mouse pointer but the pointer was moved outside the element.
onmouseover	Executed when this element was not under the mouse pointer but the pointer is moved inside the element.
onmouseup	Executed when this element is under the mouse pointer and a mouse button is released.
	"parent" form tag only
onreset	Executed if the form is reset.
onsubmit	Executed if the form is submitted.

Taglib Classes



Logic tags

Element	Description
empty	Evaluates the nested body content of this tag if the requested variable is either null or an empty string.
equal	Evaluates the nested body content of this tag if the requested variable is equal to the specified value.
forward	Forwards control to the page specified by the specified ActionForward entry.
greaterEqual	Evaluates the nested body content of this tag if the value of the requested variable is greater than or equal to the specified value.
greaterThan	Evaluates the nested body content of this tag if the value of the requested variable is greater than the specified value.
iterate	Repeat the nested body content of this tag over a specified collection.
lessEqual	Evaluates the nested body content of this tag if the value of the requested variable is less than or equal to the specified value.
lessThan	Evaluates the nested body content of this tag if the value of the requested variable is less than the specified value.
match	Evaluates the nested body content of this tag if the specified value is an appropriate substring of the requested variable.
messagesNotPresent	Generates the nested body content of this tag if the specified message is not present in this request.
messagesPresent	Generates the nested body content of this tag if the specified message is present in this request.
notEmpty	Evaluates the nested body content of this tag if the requested variable is neither null nor an empty string nor an empty java.util.Collection (tested by the .isEmpty() method on the java.util.Collection interface).
notEqual	Evaluates the nested body content of this tag if the requested variable is not equal to the specified value.
notMatch	Evaluates the nested body content of this tag if the specified value is not an appropriate substring of the requested variable.
notPresent	Generates the nested body content of this tag if the specified value is not present in this request.
present	Generates the nested body content of this tag if the specified value is present in this request.
redirect	Renders an HTTP Redirect.

For each element in the table, the tag has a `logic` prefix. For example, `empty` is `logic:empty`.

Bean tags

For each element in the table, the tag has a `bean` prefix. For example, `cookie` is `bean:cookie`.

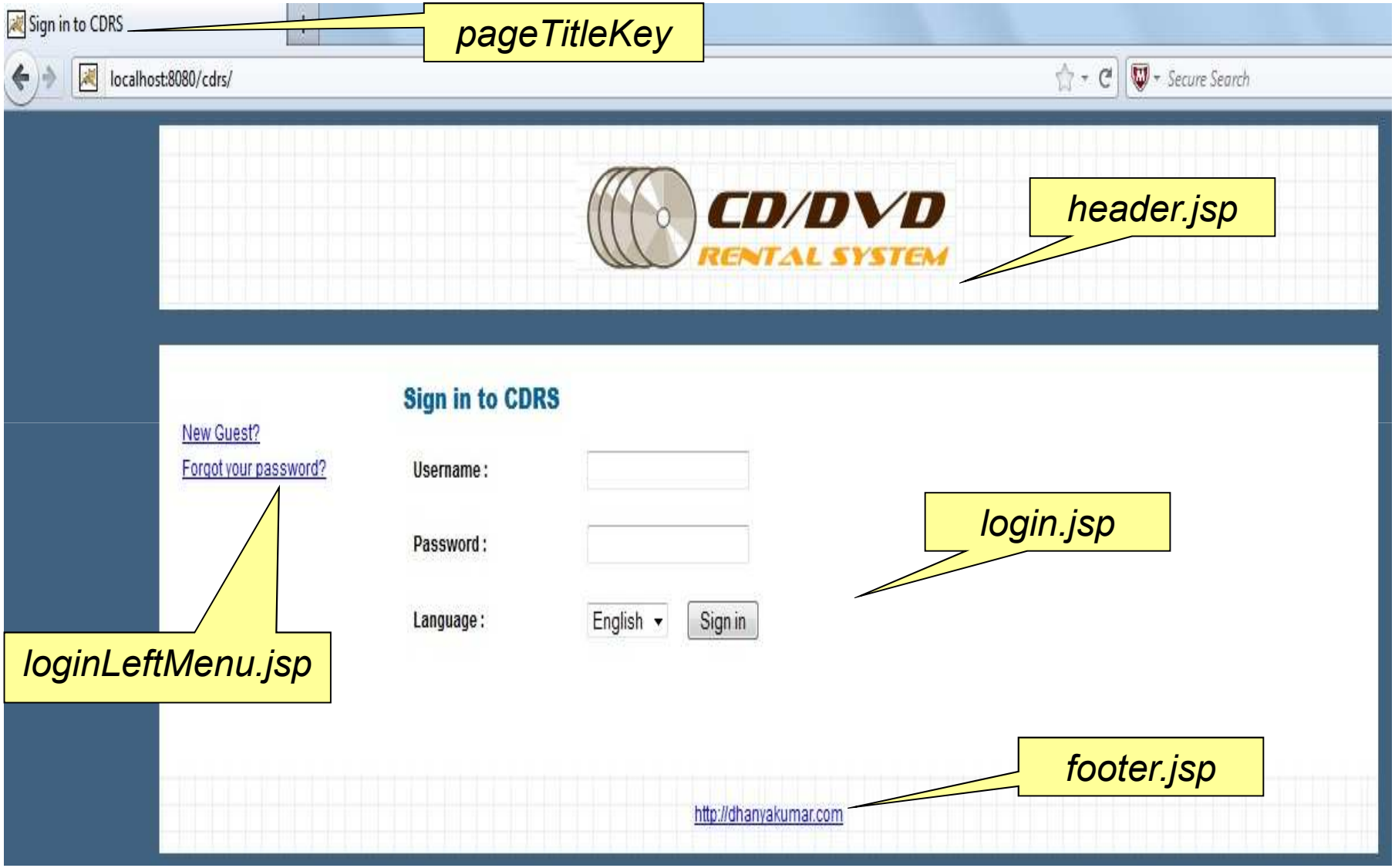
Element	Description
cookie	Defines a scripting variable based on the values of the specified request cookie.
define	Defines a scripting variable based on the values of the specified bean property.
header	Defines a scripting variable based on the values of the specified request header.
include	Loads the response from a dynamic application request and makes it available as a bean.
message	Renders an internationalized message string to the response.
page	Exposes a specified item from the page context as a bean.
parameter	Defines a scripting variable based on the values of the specified request parameter.
resource	Loads a Web application resource and makes it available as a bean.
size	Defines a bean that contains the number of elements in a Collection or Map class.
struts	Exposes a named Struts internal configuration object as a bean.
write	Inserts the value of the specified bean property into the current JSP page being rendered.

Tiles tags

For each element in the table, the tag has a `tiles` prefix. For example, `add` is `tiles:`

Element	Description
add	Adds an element to the surrounding list. This is equivalent to `put` but for a list element.
definition	Defines a tile.
get	Gets from a request scope the content that was put there by a `put` tag.
getAsString	Inserts the value of a specified tile attribute into the current JSP page being rendered.
importAttribute	Imports a tile attribute in a specified context.
initComponentDefinitions	Initializes a tile definitions factory.
insert	Inserts a tile.
put	Puts an attribute into a tile context.
putList	Declares a list to be passed as an attribute to a tile.
useAttribute	Uses an attribute value inside a page.

Tiles Example



BaseLayout.jsp

```
<head>
    <tiles:useAttribute name="pageTitleKey" ignore="true" scope="request" />
    <title><bean:message key="${requestScope.pageTitleKey}" /></title>
</head>

<body bgcolor="#00ADEF">

    <table border="0" cellpadding="0" cellspacing="0"
    style="border-collapse: collapse" bordercolor="#111111" width="100%"
    id="AutoNumber1">
        <tr>
         <td width="100%"><tiles:insert attribute="header" ignore="true" /></td>
        </tr>

        <tr>
         <td width="20%"><tiles:insert attribute="menu" ignore="true" /></td>
         <td width="80%"><tiles:insert attribute="body" ignore="true" /></td>
        </tr>

        <tr>
         <td width="100%"><tiles:insert attribute="footer" ignore="true" /></td>
        </tr>
     </table>

</body>
```

Index.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>

<tiles:insert page="/WEB-INF/view/common/baseLayout.jsp" flush="true">
    <tiles:put name="pageTitleKey" value="LABEL_000024" />
    <tiles:put name="pageHeaderKey" value="LABEL_000024" />
    <tiles:put name="header" value="header.jsp" />
    <tiles:put name="menu" value="/WEB-INF/view/login/loginLeftMenu.jsp" />
    <tiles:put name="body" value="/WEB-INF/view/login/login.jsp" />
    <tiles:put name="footer" value="footer.jsp" />
</tiles:insert>
```

Login.jsp

```
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<body>
<html:form action="/Login.do?method=validateUser">
<table border="0" cellpadding="0" cellspacing="0">
    <tr>
    <td width="17%">
    <p class="normal" align="justify">
        <b><bean:message key="LABEL_000001" style="" styleClass="" /> : </b></p>
    </td>
    <td width="1%"> </td>
    <td width="82%">
        <html:text name="userAccountForm" property="username" tabindex="1" /></td>
    </tr>
    <tr>
    <td width="17%" dir="ltr">
    <p class="normal" align="justify" dir="ltr">
        <b><bean:message key="LABEL_000002" /> : </b></p>
    </td>
    <td width="1%" dir="ltr"> </td>
    <td width="82%" dir="ltr">
    <html:password name="userAccountForm" property="password" tabindex="2" /></td>
    </tr>
```

Login.jsp

```
    <tr>
    <td width="17%">
    <p class="normal" align="justify">
        <b><bean:message key="LABEL_000021" /> : </b></p>
    </td>
    <td width="1%"> </td>
    <td width="82%">
    <select>
        <option value="en_US">English</option>
    </select>
       
    <input type="submit" value="<bean:message key="LABEL_000022"/>" name="login"/>
    </td>
    </tr>
</table>
</html:form>
</body>
```

ResourceBundle.properties

```
LABEL_000001           = Username
LABEL_000002           = Password
LABEL_000021           = Language
LABEL_000022           = Sign in
LABEL_000023           = New Guest?
LABEL_000024           = Sign in to CDRS
LABEL_000025           = Forgot your password?
MSG_000001             = {0} required.
MSG_000002             = {0} should be greater than {1}
error.required         = {0} is required.
error.minlength        = {0} can not be less than {1} characters.
error.maxlength        = {0} can not be greater than {1} characters.
error.invalid          = {0} is invalid.
error.invalid.number   = {0} is invalid. Only 0-9 characters allowed.
error.date             = {0} is not a valid date.
error.futuredate       = {0} should be greater than {1}.
error.futuredate.today= {0} should be greater than or equal to {1}.
error.pastdate         = {0} should be less than {1}.
error.todaysdate       = {0} should be equal to {1}.
error.range            = {0} is not in the range {1} to {2} characters.
error.email            = {0} is an invalid e-mail address.
error.greaterthan      = {0} should be greater than {1}.
error.lessthan         = {0} should less be greater than {1}.
```

UserAccountForm

```
public final class UserAccountForm extends ActionForm implements
                                           Serializable, Cloneable {

    private static final long serialVersionUID = -3048740746209508149L;
    private String username;
    private String password;
    private String method;

    public ActionErrors validate(ActionMapping mapping,HttpServletRequest request)
    {
        ActionErrors errors = new ActionErrors();
        if (StringUtils.isBlank(getUsername()) ||
            StringUtils.isBlank(getPassword())) {
            errors.add(Globals.MESSAGE_KEY,
                 new ActionMessage("MSG_0038",”Login credentials required.”
        }
        return errors;

        }

        // getters & setters methods
```

LoginAction

```
public class UserAccountAction extends Action {

 public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws Exception {
        UserAccountForm userAccountForm = (UserAccountForm) form;
        UserDto userDto = new UserDto();
        BeanUtils.copyProperties(userAccountForm, userDto);

        boolean regUser = getServiceManager().getUserAccountService()
                                             .validateUser(userDto);

        if(regUser){
           mapping.findForward(ActionForwardIds.SUCCESS)
         }else{
           addMessages(request, new ActionMessages("","Invalid Username/Pwd"));
           mapping.findForward(ActionForwardIds.FAILURE)
         }

}
```

UserAction

```
public class UserAction extends DispatchAction {

 public ActionForward validateUser(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws Exception {
        UserAccountForm userAccountForm = (UserAccountForm) form;
        UserDto userDto = new UserDto();
        BeanUtils.copyProperties(userAccountForm, userDto);

        boolean regUser = getServiceManager().getUserAccountService()
                                             .validateUser(userDto);

        if(regUser){
           mapping.findForward(ActionForwardIds.SUCCESS)
         }else{
           addMessages(request, new ActionMessages("","Invalid Username/Pwd"));
           mapping.findForward(ActionForwardIds.FAILURE)
         }

}

Note: method=validateUser hidden field in JSP page.
```

References

- http://onlinejava.co.in/products/struts_interview_QA.html
- http://struts.apache.org/1.2.x/userGuide/index.html
- http://www.roseindia.net/struts/
- http://www.vaannila.com/struts/struts-tutorial/struts-tutorial.html
- http://exadel.com/tutorial/struts/5.2/guess/strutsintro.html
- http://www.lalilуna.de/articles/posts/first-steps-using-struts-tutorial.html
- http://javaboutique.internet.com/tutorials/Struts/


Thank you!