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

1. Loading
2. Initialization
3. Servicing the Requests
4. Service
5. Resources
6. Destroying
7. Destroy

Servlet Request

A Web Container

doGet
doPost
doDelete
...

STRUTS 1.3
<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>
<!DOCTYPE struts-config PUBLIC
  "-//Apache Software Foundation//DTD Struts Configuration 1.3//EN"
  "http://struts.apache.org/dtlds/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>
<?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/dtlds/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-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

Author: Jean-Michel Garnier - October 2002
http://rollerjm.free.fr
Struts Flow

Diagram showing the Struts 1.3 flow, including components such as Client, Struts Controller, RequestProcessor, BusinessDelegate, Helper, and View, with actions like doPost, process, and validate.
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);
    }
}
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);
    }
}
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);
    }
}
<table>
<thead>
<tr>
<th>HTML Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;html:message key=&quot;theKey&quot;/&gt;</code></td>
<td>Looks up the message corresponding to the given key in the message resources and displays it.</td>
</tr>
<tr>
<td><code>&lt;html:password property=&quot;prop&quot; size=&quot;10&quot;/&gt;</code></td>
<td>Tag creates the password field. The string is stored in the property named prop in the form bean.</td>
</tr>
<tr>
<td><code>&lt;html:text property=&quot;text1&quot; size=&quot;5&quot;/&gt;</code></td>
<td>Tag creates the text field. The string is retrieved from and later stored in the property named text1 in the form bean.</td>
</tr>
<tr>
<td><code>&lt;html:submit&gt;Submit&lt;/html:submit&gt;</code></td>
<td>Tag creates a submit button with the provided content as the button text.</td>
</tr>
<tr>
<td><code>&lt;html:reset&gt;Reset&lt;/html:reset&gt;</code></td>
<td>Tag creates a reset button with the provided content as the button text.</td>
</tr>
<tr>
<td><code>&lt;html:errors/&gt;</code></td>
<td>Tag prints all the available error on the page.</td>
</tr>
<tr>
<td><code>&lt;html:file property=&quot;fileSelectionBox&quot;/&gt;</code></td>
<td>Tag creates the file upload element on the form. The property must be of the type org.apache.struts.upload.FormFile.</td>
</tr>
<tr>
<td><code>&lt;html:checkbox property=&quot;myCheckBox&quot;/&gt;</code></td>
<td>Tag creates check box on the form.</td>
</tr>
<tr>
<td><code>&lt;html:hidden property=&quot;hiddenfield&quot;/&gt;</code></td>
<td>Tag creates the hidden html element on the form.</td>
</tr>
<tr>
<td><code>&lt;html:radio value=&quot;abc&quot; property=&quot;myCheckBox&quot;/&gt;</code></td>
<td>Tag creates the check box on the form.</td>
</tr>
<tr>
<td><code>&lt;html:select multiple=&quot;true&quot; property=&quot;selectBox&quot;/&gt;</code></td>
<td>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 <code>&lt;html:options&gt;</code> to specify the entries.</td>
</tr>
<tr>
<td><code>&lt;html:textarea property=&quot;myTextArea&quot; value=&quot;Hello Struts&quot;/&gt;</code></td>
<td>Tag creates the text area on the form.</td>
</tr>
<tr>
<td><code>&lt;html:form action=&quot;/Address&quot; method=&quot;post&quot;/&gt;</code></td>
<td>Tag is used to create the HTML Form for posting the data on the server.</td>
</tr>
<tr>
<td><code>&lt;html:base/&gt;</code></td>
<td>Tag generates the base tag. <code>&lt;BASE ...&gt;</code> 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 <code>&lt;BASE HREF=&quot;...&quot;/&gt;</code> instead of the actual URL. <code>&lt;BASE ... /&gt;</code> goes in the <code>&lt;HEAD&gt;</code> section.</td>
</tr>
<tr>
<td><code>&lt;html:html/&gt;</code></td>
<td>Tag renders an HTML <code>&lt;html&gt;</code> Element.</td>
</tr>
</tbody>
</table>

http://www.roseindia.net/struts/strutsHtmlTags.shtml
<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onblur</td>
<td>Executed when this element loses input focus.</td>
</tr>
<tr>
<td>onchange</td>
<td>Executed when this element loses input focus and its value has changed.</td>
</tr>
<tr>
<td>onclick</td>
<td>Executed when this element receives a mouse click.</td>
</tr>
<tr>
<td>ondblclick</td>
<td>Executed when this element receives a mouse - double click.</td>
</tr>
<tr>
<td>onfocus</td>
<td>Executed when this element receives input focus.</td>
</tr>
<tr>
<td>onkeydown</td>
<td>Executed when this element has focus and a key is depressed.</td>
</tr>
<tr>
<td>onkeypress</td>
<td>Executed when this element has focus and a key is depressed and released</td>
</tr>
<tr>
<td>onkeyup</td>
<td>Executed when this element has focus and a key is released</td>
</tr>
<tr>
<td>onmousedown</td>
<td>Executed when this element is under the mouse pointer and a mouse button is depressed.</td>
</tr>
<tr>
<td>onmousemove</td>
<td>Executed when this element is under the mouse pointer and the pointer is moved.</td>
</tr>
<tr>
<td>onmouseout</td>
<td>Executed when this element was under the mouse pointer but the pointer was moved outside the element.</td>
</tr>
<tr>
<td>onmouseover</td>
<td>Executed when this element was not under the mouse pointer but the pointer is moved inside the element.</td>
</tr>
<tr>
<td>onmouseup</td>
<td>Executed when this element is under the mouse pointer and a mouse button is released.</td>
</tr>
<tr>
<td>onreset</td>
<td>Executed if the form is reset.</td>
</tr>
<tr>
<td>onsubmit</td>
<td>Executed if the form is submitted.</td>
</tr>
</tbody>
</table>

"parent" form tag only
Taglib Classes

STRUTS 1.3
### Logic tags

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>empty</td>
<td>Evaluates the nested body content of this tag if the requested variable is either null or an empty string.</td>
</tr>
<tr>
<td>equal</td>
<td>Evaluates the nested body content of this tag if the requested variable is equal to the specified value.</td>
</tr>
<tr>
<td>forward</td>
<td>Forwards control to the page specified by the specified ActionForward entry.</td>
</tr>
<tr>
<td>greaterEqual</td>
<td>Evaluates the nested body content of this tag if the value of the requested variable is greater than or equal to the specified value.</td>
</tr>
<tr>
<td>greaterThan</td>
<td>Evaluates the nested body content of this tag if the value of the requested variable is greater than the specified value.</td>
</tr>
<tr>
<td>iterate</td>
<td>Repeat the nested body content of this tag over a specified collection.</td>
</tr>
<tr>
<td>lessEqual</td>
<td>Evaluates the nested body content of this tag if the value of the requested variable is less than or equal to the specified value.</td>
</tr>
<tr>
<td>lessThan</td>
<td>Evaluates the nested body content of this tag if the value of the requested variable is less than the specified value.</td>
</tr>
<tr>
<td>match</td>
<td>Evaluates the nested body content of this tag if the specified value is an appropriate substring of the requested variable.</td>
</tr>
<tr>
<td>messagesNotPresent</td>
<td>Generates the nested body content of this tag if the specified message is not present in this request.</td>
</tr>
<tr>
<td>messagesPresent</td>
<td>Generates the nested body content of this tag if the specified message is present in this request.</td>
</tr>
<tr>
<td>notEmpty</td>
<td>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).</td>
</tr>
<tr>
<td>notEqual</td>
<td>Evaluates the nested body content of this tag if the requested variable is not equal to the specified value.</td>
</tr>
<tr>
<td>notMatch</td>
<td>Evaluates the nested body content of this tag if the specified value is not an appropriate substring of the requested variable.</td>
</tr>
<tr>
<td>notPresent</td>
<td>Generates the nested body content of this tag if the specified value is not present in this request.</td>
</tr>
<tr>
<td>present</td>
<td>Generates the nested body content of this tag if the specified value is present in this request.</td>
</tr>
<tr>
<td>redirect</td>
<td>Renders an HTTP Redirect.</td>
</tr>
</tbody>
</table>

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

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

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

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

- `pageTitleKey`
- `header.jsp`
- `login.jsp`
- `loginLeftMenu.jsp`
- `footer.jsp`
```xml
<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>
```
<%@ 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>
<%@ 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 align="justify">
            <b><bean:message key="LABEL_000001" style="" styleClass="" /> : </b></p>
    </td>
    <td width="1%">&nbsp;</td>
    <td width="82%">
        <html:text name="userAccountForm" property="username" tabindex="1" />
    </td>
</tr>
<tr>
    <td width="17%" dir="ltr">
        <p align="justify" dir="ltr">
            <b><bean:message key="LABEL_000002" /> : </b></p>
    </td>
    <td width="1%" dir="ltr">&nbsp;</td>
    <td width="82%" dir="ltr">
        <html:password name="userAccountForm" property="password" tabindex="2" />
    </td>
</tr>
</table>
</html:form>
</body>
<tr>
<td width="17%">
<p class="normal" align="justify">
  <b><bean:message key="LABEL_000021" /></b> : </p>
</td>
<td width="1%">&nbsp;</td>
<td width="82%">
<select>
  <option value="en_US">English</option>
</select>
</td>
</tr>
</table>
</html:form>
</body>
| 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}. |
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
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

```java
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://exadel.com/tutorial/struts/5.2/guess/strutsintro.html
- http://www.laliluna.de/articles/posts/first-steps-using-struts-tutorial.html
Thank you!