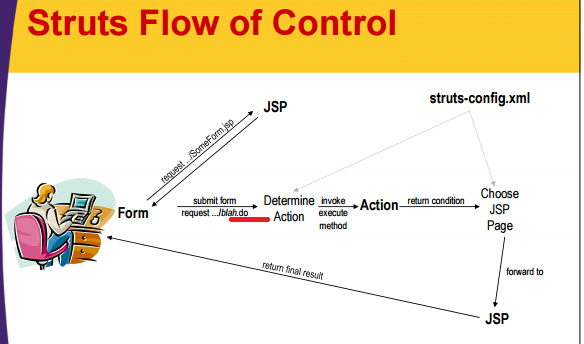
1. Struts provides utility classes to handle many of the most common tasks in Web application development
2. 
3. **Explanation of struts flow is**,

RED color🡺In struts action attribute of the form tag should have .do extension🡺MS

And –That address is mapped by struts-config xml to an Action class and then The execute method of the Action object is invoked

In our case PRHTMLAction extends BaseBondlineInputAction.

BBIAction is an abstract class and it extends BaseBondlineaction

BBAction implements execute() method and it’s an abstract class🡺 here logic is written in such a way that execute() method of BBAction calls performRequest() method which is implemented by its child

PRHTMLAction is the child class that implements performRequest () method.

Now let’s see the syntax for execute () 🡺 Book

Now let’s see the syntax for performRequest ()🡺 Book

The Action object then invokes business logic and data-access logic, placing the results in normal beans stored in request, session, or application scope.

The Action uses **actionMapping.findForward** to return a condition, and the conditions are mapped by struts-config.xml to various JSP pages🡺 Syntax for this is

Final ActionForward forward = actionMapping.findForward(“summary”);

Struts forwards request to the appropriate JSP page Struts forwards request to the appropriate JSP page

1. **The Six Basic Steps in Using Struts**

struts-config.xml 🡺 Use WEB-INF/struts-config.xml to:

Map incoming .do addresses to Action classes

Map return conditions to JSP pages Map return conditions to JSP pages

Declare any form beans that are being used.

Be sure to restart the server after modifying struts-config.xml; the file is read only when the Web application is first loaded.

Define a form bean🡺

This bean is a class extends ActionForm and will represent the data submitted by the user. It is automatically populated when the input form is submitted. Beans are postponed until the next section.

Create results beans🡺

In the MVC architecture, the business-logic and data-access code create the results and the JSP pages present them. To transfer the results from one layer to the other, they are stored in beans. These beans differ from form beans in that they need extend no particular beans differ from form beans in that they need extend no particular class, and they represent the output of the computational process, not the input to the process. Beans will be discussed in the next section.

NOTE 🡺 DIFFERENCE BETWEEN ABOVE POINT “C” AND “D” IS 🡺

The form bean represents the input data – I.e., the data that came from the HTML form.

The results beans represent the output data – I.e., the data created by the business logic to represent the results of the computation or database lookup.

**Note: 🡺 They are stored in request, session, or application scope with the setAttribute method of HttpServletRequest, HttpSession, or ServletContext, respectively.**

Define an Action class to handle requests🡺

The struts-config.xml file designates the Action classes that handle requests for various URLs The Action objects themselves need to requests for various URLs. The Action objects themselves need to do the real work: invoke the appropriate business- and data-accesslogic, store the results in beans, and designate the type of situation (missing data database error success category 1 success category (missing data, database error, success category 1, success category 2, etc.) that is appropriate for the results. The struts-config.xml file then decides which JSP page should apply to that situation.

**Note: Action Class🡺 Instead of reading form data explicitly with request.getParameter, the execute method uses a bean that is automatically q filled in from the request data.**

Create form that invokes blah.do🡺

Create an input form whose ACTION corresponds to one of the .do addresses listed in struts-config.xml.

Rather than using the standard HTML FORM and INPUT tags, we now use **html:form** and **html:text** (and related tags).

Display results in JSP🡺

Since Struts is built around MVC, **these JSP pages should avoid JSP scripting elements whenever possible**. For basic Struts, these pages usually use the bean:write tag, but in JSP 2.0 the JSP 2.0 expression language is a viable alternative.

In most cases the JSP pages only make sense when the request is In most cases, the JSP pages only make sense when the request is funneled through the Action, so the pages go in WEB-INF.

If the JSP pages makes sense independently of the Action (e.g., if they display session data), then the JSP pages should be placed in a regular subdirectory of the Web application, and the forward entries in struts-config.xml should say.

As before, the JSP page uses **bean:write** to output properties of the form and result beans, It may also use **bean:message** to output standard messages and text labels that are defined in a properties file.

struts-config.xml file🡺 Book

**Step 6 (Display results in JSP) in struts-config.xml🡺** In general, there can be several possible JSP pages🡺– Corresponding to the various possible return values of the execute method of the Action execute method of the Action. • In struts-config.xml, each JSP page is declared in a forward entry within the appropriate action.

**Chapter -3**

**Struts-Beans:**

One of the arguments to execute is a form bean that **is automatically created and whose properties are automatically populated based on incoming request parameters of the same name**

**Or**

A **form bean is a Java object that will be automatically filled in based on the incoming form parameters then filled in based on the incoming form parameters, then passed to the execute method.**

Bean must extend ActionForm

Bean must be declared in struts-config.xml with form-beans

For each incoming request parameter, the corresponding setter method and The validate method is called by struts framework itself

Ms🡺uses of the Beans we might see during the Struts tag reading

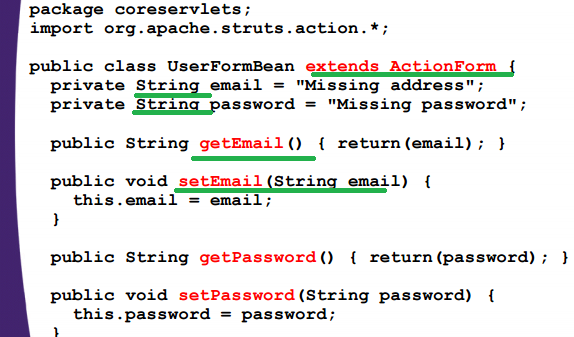
Rules for creating a bean class is,

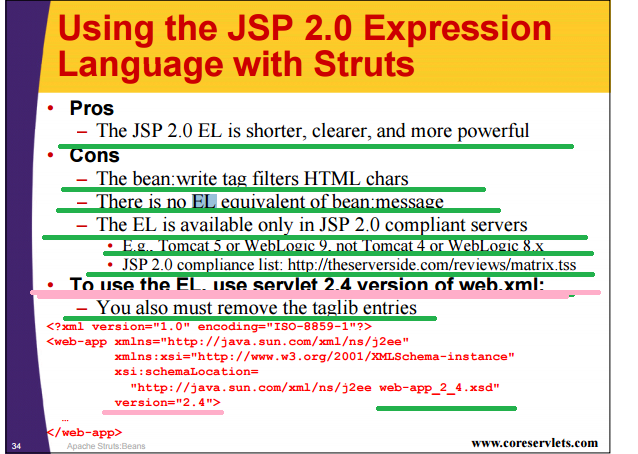
– It must extend ActionForm🡺 The argument to execute is of type ActionForm Cast the value to your The argument to execute is of type ActionForm. Cast the value to your real type, and then each bean property has the value of the request parameter with the matching name.

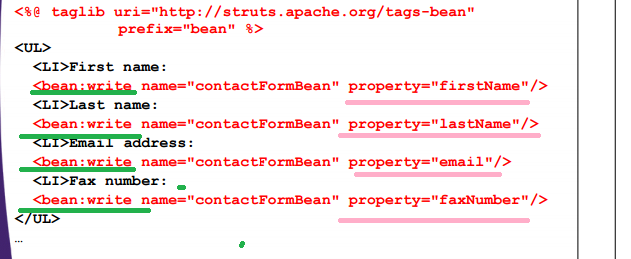
It must have a zero argument constructor🡺 The system will automatically call this default constructor.

It must have settable bean properties that correspond to the request parameter names🡺 That is, it must have a setBlah method corresponding to each incoming request parameter named blah.

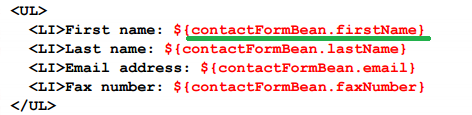
It must have gettable bean properties for each property that you want to output in the JSP page.





Input form example: 🡺

Output JSP example:



**Chapter -4**

**Struts –Forms**

Struts HTML tags

Three Characteristics of html tags:

The textfield names and the bean properties are guaranteed to stay in synch

The textfield values can be prepopulated based on the values in a bean🡺 That is, the initial values of the form elements can be taken from a Java object

--

Rather than using the standard HTML FORM and INPUT tags, we now use html:form and html:text (and related tags).

**The html:form tag associates a bean with the form, and html:text automatically uses bean property names for each textfield NAME and bean property values for each textfield VALUE.**

**Syntax for html:text is 🡺**

**<html:text property="firstName" />**

Explanation for above syntax 🡺Each textfield NAME will be taken from the bean property name, and each textfield VALUE will come from the bean property value.

**In addition, as in the previous section, this form can still use the bean:message tag to output standard messages and text labels.**

**Using the Struts html:form element instead of the standard HTML FORM element yields four results:**

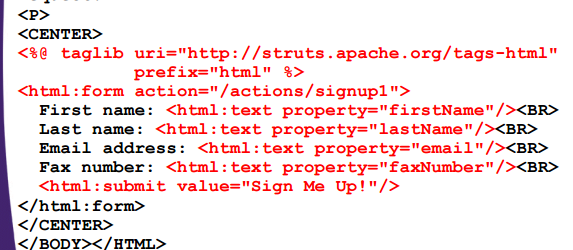
A bean is associated with the form.

The .do suffix is appended automatically

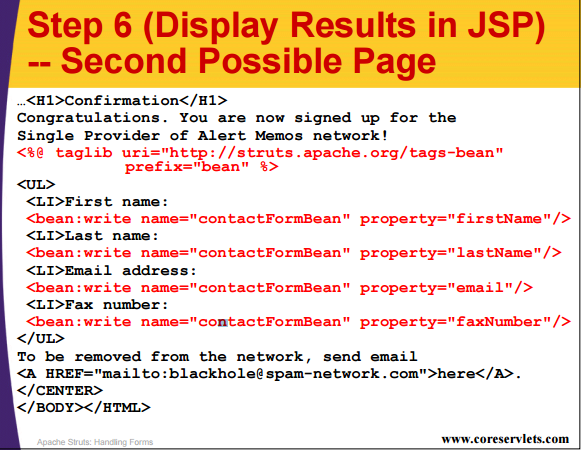
* You say<html:form action = "/actions/blah”> to get <html:form action = ""/webAppPrefix/actions/blah.do”>
* **POST, not GET, is the default METHOD**🡺You say <html:form action = "/actions/blah”> to get <html:form action = ""/webAppPrefix/actions/blah.do” **METHOD="POST"**>

1. Use **html:xxxx** for other input elements 🡺In your input form, use html:button, html:checkbox, html:textarea, etc., to declare submit buttons, checkboxes, text areas, etc
2. Syntax to get inputs from user:

* Top of Form



Output JSP



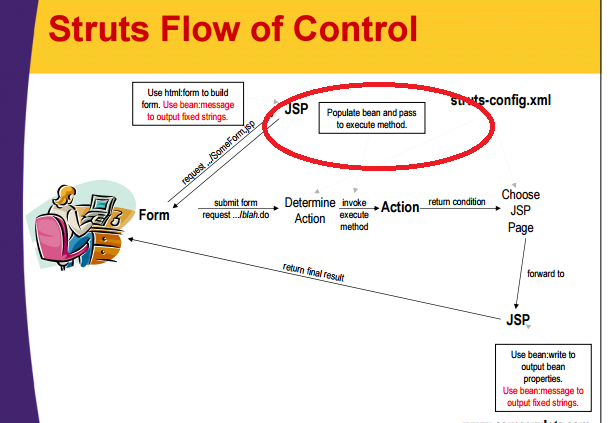
**Chapter 5🡺**

**Struts Messages**

MS 🡺 Difference between **bean:write** and **bean:message**

– Using bean:write to output bean properties

--Using bean:message to output constant strings



**PROPERTIES FILE IN STRUTS**

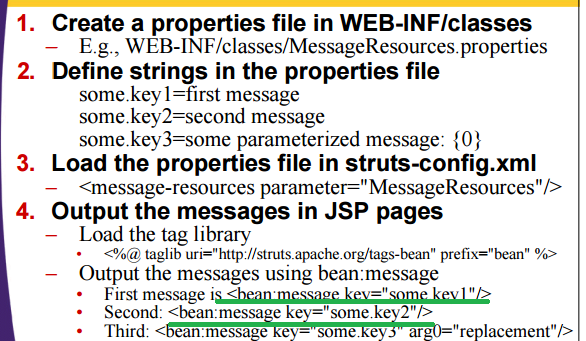
Advantage of using Properties file

Easy Maintenance

Centralized updates🡺If a message is used in several places, it can be updated with a single change.

Supports I18N

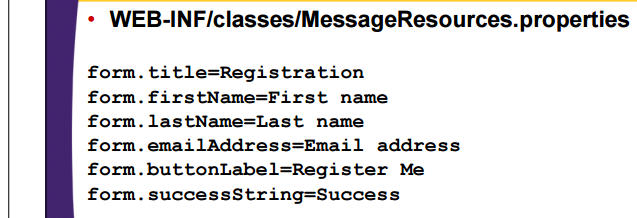
Steps to use the Properties file in struts,



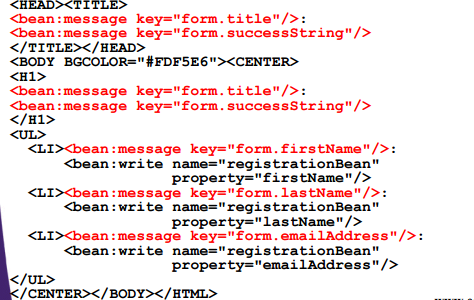
**Syntax for using the Property files,**

**<message-resources parameter="MessageResources" null="false"/>**

Complete struts-config.xml file with the <message-resources> see the Book or tutorials

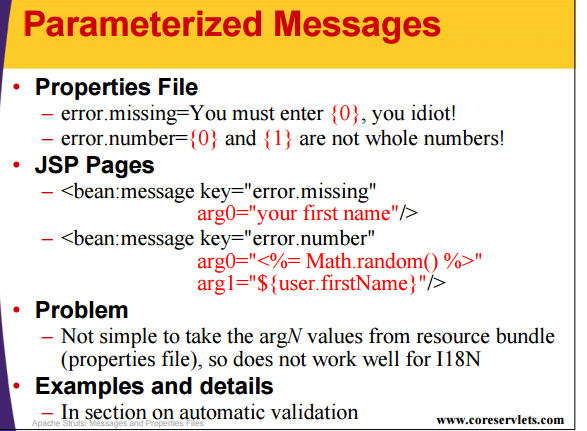
**Syntax for Property files 🡺**

**Example for How we use the content of property files**

**Example for both bean:write and bean:message elements**

Syntax for internationalization 🡺later we wills see

**Parameterized Messages**

1. 

**Chapter -6**

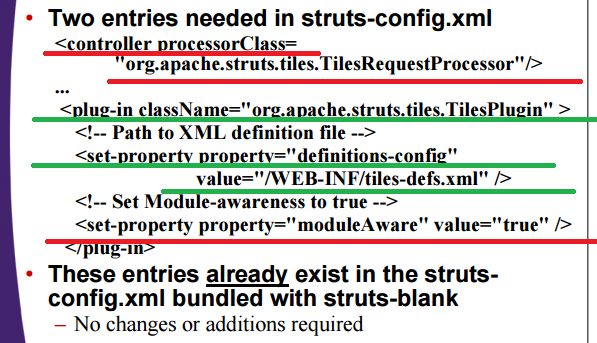
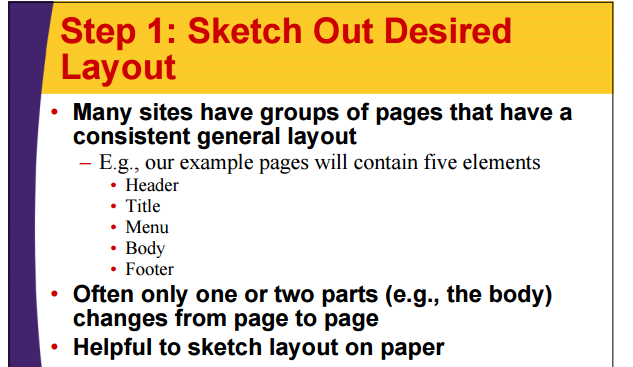
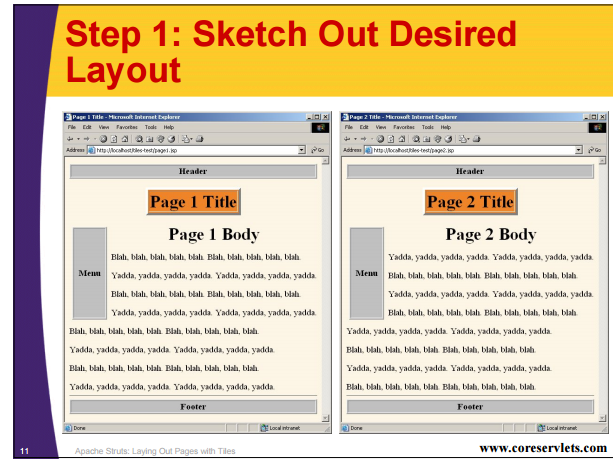
**Advanced \_Actions**

**DispatchAction**

1. In, HTML, one form has one ACTION
2. Needed to see the Syntax and Uses of this class

**Chapter – 10**

**Struts-Tiles**

1. Tiles basics
2. – Sketch out desired layout
3. – Make template file that represents layout
4. – Create JSP pages that define layout pieces
5. – Create JSP pages that populate layout
6. Tiles Motivations
7. **• Reuse (not rewrite) repeated sections of pages**
8. **• Simplify the creation of similar pages**
9. **• Increase flexibility and ease of maintenance compared to .**
10. Tiles Prerequisites
11. 
12. 
13. Example for Layout

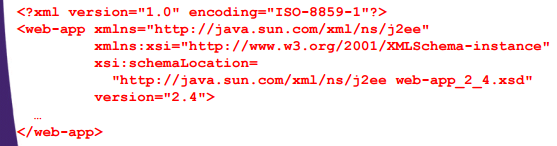
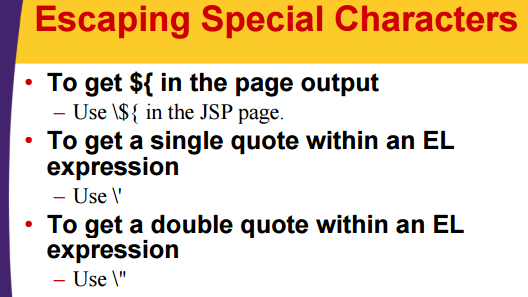
NOTE 🡺IN ORDER TO USE TILES WE HAVE TO ADD <taglib> directive like we do to use html, bean and logic tags of struts framework

Syntax: 🡺

**<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles" />**

1. **Our logic in application is different from one discussed here🡺NEEDED TO DISCUSS WITH TEAM**

**Struts –and-JSP2-EL**

1. Available only in servers that support JSP 2.0 (servlets 2.4) 🡺E.g., Tomcat 5, not Tomcat 4
2. You must use the JSP 2.0 web.xml file 🡺 
3. Syntax 🡺 
4. 

**NOTE🡺 TO GET MORE UNDERSTANDING ON THIS TOPIC FIRST SEE THE JSP AND THEN COME BACK AND PREPARE THE NOTES**

**Using the JSP Standard Tag Library (JSTL) with Struts**

1. JSTL is the recommended replacement for the Struts looping and logic tags
2. •JSTL is not part of the JSP 1.2 or 2.0 Specs
3. It is a separate specification that requires a separate download
4. Available only in servers that support servlets 2.3 and JSP 1.2 or later. Cannot be retrofitted into JSP 1.1.
5. **The JSTL expression language is now part of JSP 2.0**
6. Installing JSTL
7. Some servers come with JSTL preinstalled 🡺 – E.g., Caucho Resin
8. JSTL (like JSP) is a specification, not an implementation
9. To install:
10. Download zip file
11. Unzip into directory of your choice (e.g., C:\jakarta-taglibs)
12. Copy install\_dir/standard-1.0.1/lib/jstl.jar and install\_dir/standard-1.0.1/lib/standard.jar to the WEB-INF/lib directory of your Web application

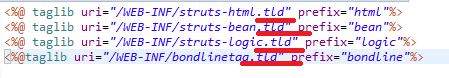
NOTE 🡺 UNDER THIS TOPIC ONLY JSP’S JSTL TAGS ARE EXPLAINED IT DON’T HAVE EXPLAINATION FOR STRUTS TAG

FOLLOWING NOTES ARE PREPARED FROM SOME OTHER TUTORIALS🡺

1. LOGIC TAGS 🡺
2. <logic:iterate>
3. <logic:present>
4. <logic:notPresent>
5. <logic:equal>
6. <logic:notEqual>
7. <logic:empty>
8. <logic:notEmpty>
9. <logic:messagesPresent>
10. <logic:greaterThan>

NOTE🡺IN ORDER TO USE THE STRUTS OR ANY USER DEFINED TAGS WE HAVE TO INCLUDE ITS .tld FILES by using the directive taglib

1. &nbsp🡺 ms 🡺is used to provide the space



1. BEAN TAGS 🡺
2. <bean:write>
3. <bean:message>
4. <bean:parameter>
5. <bean:define>
6. HTML TAGS🡺
7. <html:form>
8. <html:text>
9. <html:textarea>
10. <html:button>
11. <html:select>
12. <html:option>
13. <html:optionsCollection>🡺
14. <html:hidden>
15. <html:radio>

PENDING WORK WITH STRUTS 1.2 🡺

1. Needed to collect more Information🡺Struts tags
2. Needed to collect more Information🡺Tiles 🡺Syntax
3. DispatchAction
4. Error Handling and Validation