**What is MVC?**

Model-View-Controller (MVC) is a design pattern put together to help control change. MVC decouples interface from business logic and data.

* **Model :** The model contains the core of the application's functionality. The model encapsulates the state of the application. Sometimes the only functionality it contains is state. It knows nothing about the view or controller.
* **View:** The view provides the presentation of the model. It is the *look* of the application. The view can access the model getters, but it has no knowledge of the setters. In addition, it knows nothing about the controller. The view should be notified when changes to the model occur.
* **Controller:**The controller reacts to the user input. It creates and sets the model.

**2.What is a framework?**

A framework is made up of the set of classes which allow us to use a library in a best possible way for a specific requirement.

**3.What is Struts framework?**

Struts framework is an [open-source](http://www.developersbook.com/struts/interview-questions/struts-interview-questions-faqs.php) framework for developing the web applications in Java EE, based on MVC-2 architecture. It uses and extends the [Java Servlet](http://www.developersbook.com/struts/interview-questions/struts-interview-questions-faqs.php) API. Struts is robust architecture and can be used for the development of application of any size. Struts framework makes it much easier to design scalable, reliable Web applications with Java.

**4.What are the components of Struts?**

Struts components can be categorize into Model, View and Controller:

* **Model:** Components like business logic /[business processes](http://www.developersbook.com/struts/interview-questions/struts-interview-questions-faqs.php) and data are the part of model.
* **View:** HTML, JSP are the view components.
* **Controller:** Action Servlet of Struts is part of Controller components which works as front controller to handle all the requests.

**5.What are the core classes of the Struts Framework?**

Struts is a set of cooperating classes, servlets, and [JSP tags](http://www.developersbook.com/struts/interview-questions/struts-interview-questions-faqs.php) that make up a reusable MVC 2 design.

* JavaBeans components for managing application state and behavior.
* Event-driven development (via listeners as in traditional GUI development).
* Pages that represent MVC-style views; pages reference view roots via the JSF component tree.

**6.What is ActionServlet?**

ActionServlet is a simple servlet which is the backbone of all Struts applications. It is the main Controller component that handles client requests and determines which Action will process each received request. It serves as an Action factory – creating specific Action classes based on user’s request.

**7.What is role of ActionServlet?**

ActionServlet performs the role of Controller:

* Process user requests
* Determine what the user is trying to achieve according to the request
* Pull data from the model (if necessary) to be given to the appropriate view,
* Select the proper view to respond to the user
* Delegates most of this grunt work to Action classes
* Is responsible for initialization and clean-up of resources

**8.What is the ActionForm?**

ActionForm is javabean which represents the form inputs containing the request parameters from the View referencing the Action bean.

**9.What are the important methods of ActionForm?**

The important methods of ActionForm are : validate() & reset().

**10.Describe validate() and reset() methods ?**

***validate()*** : Used to validate properties after they have been populated; Called before FormBean is handed to Action. Returns a collection of ActionError as ActionErrors. Following is the method signature for the validate() method.

**public** ActionErrors validate(ActionMapping mapping,HttpServletRequest request)

***reset()***: reset() method is called by Struts Framework with each request that uses the defined ActionForm. The purpose of this method is to reset all of the ActionForm's data members prior to the new request values being set.

public void reset() {}

**11.What is ActionMapping?**

Action mapping contains all the deployment information for a particular Action bean. This class is to determine where the results of the Action will be sent once its processing is complete.

**12.How is the Action Mapping specified ?**

We can specify the action mapping in the configuration file called struts-config.xml. Struts framework creates ActionMapping object from <ActionMapping> configuration element of struts-config.xml file

<action-mappings>  
 <action path="/submit"  
 type="submit.SubmitAction"  
  name="submitForm"  
  input="/submit.jsp"  
  scope="request"  
 validate="true">  
 <forward name="success" path="/success.jsp"/>  
 <forward name="failure" path="/error.jsp"/>  
 </action>  
</action-mappings>

**13.What is role of Action Class?**

An Action Class performs a role of an adapter between the contents of an incoming HTTP request and the corresponding business logic that should be executed to process this request.

**14.In which method of Action class the business logic is executed ?**

In the execute() method of Action class the business logic is executed.

public ActionForward execute(   
 ActionMapping mapping,  
             ActionForm form,  
             HttpServletRequest request,  
             HttpServletResponse response)  
          throws Exception ;

execute() method of Action class:

* Perform the processing required to deal with this request
* Update the server-side objects (Scope variables) that will be used to create the next page of the user interface
* Return an appropriate ActionForward object

**15.What design patterns are used in Struts?**

Struts is based on model 2 MVC (Model-View-Controller) architecture. Struts controller uses the *command design pattern* and the action classes use the *adapter design pattern*. The process() method of the RequestProcessor uses the *template method design pattern*. Struts also implement the following J2EE design patterns.

* Service to Worker
* Dispatcher View
* Composite View (Struts Tiles)
* Front Controller
* View Helper
* Synchronizer Token

**.Can we have more than one struts-config.xml file for a single Struts application?**

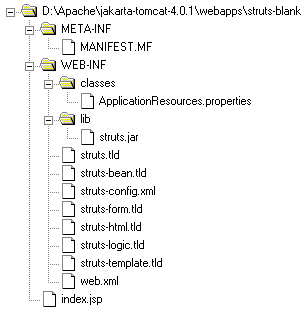
Yes, we can have more than one struts-config.[xml](http://www.developersbook.com/struts/interview-questions/struts-interview-questions-faqs-2.php) for a single Struts application. They can be configured as follows:

<servlet>  
<servlet-name>action</servlet-name>         
 <servlet-class>  
 org.[apache](http://www.developersbook.com/struts/interview-questions/struts-interview-questions-faqs-2.php)

.struts.action.ActionServlet  
 </servlet-class>  
<init-param>  
 <param-name>config</param-name> <param-value>  
  **/WEB-INF/struts-config.xml,                
 /WEB-INF/struts-admin.xml,  
 /WEB-INF/struts-config-forms.xml**  </param-value></init-param>  
.....  
<servlet>

**17.What is the directory structure of Struts application?**

The directory structure of Struts application :

  
  
**18.What is the difference between session scope and request scope when saving formbean ?**

when the scope is *request*,the values of formbean would be available for the current request.  
when the scope is *session*,the values of formbean would be available throughout the session.

**19.What are the important tags of struts-config.xml ?**

The five important sections are:



**20.What are the different kinds of actions in Struts?**

The different kinds of actions in Struts are:

* ForwardAction
* IncludeAction
* DispatchAction
* LookupDispatchAction
* SwitchAction

**21.What is DispatchAction?**

The DispatchAction class is used to group related actions into one class. Using this class, you can have a method for each logical action compared than a single execute method. The DispatchAction dispatches to one of the logical actions represented by the methods. It picks a method to invoke based on an incoming request parameter. The value of the incoming parameter is the name of the method that the DispatchAction will invoke.

**22.How to use DispatchAction?**

To use the DispatchAction, follow these steps :

* Create a class that extends DispatchAction (instead of Action)
* In a new class, add a method for every function you need to perform on the service – The method has the same signature as the execute() method of an Action class.
* Do not override execute() method – Because DispatchAction class itself provides execute() method.
* Add an entry to struts-config.xml

**23.What is the use of ForwardAction?**

The ForwardAction class is useful when you’re trying to integrate Struts into an existing application that uses Servlets to perform business logic functions. You can use this class to take advantage of the Struts controller and its functionality, without having to rewrite the existing Servlets. Use ForwardAction to forward a request to another resource in your application, such as a Servlet that already does business logic processing or even another [JSP](http://www.developersbook.com/struts/interview-questions/struts-interview-questions-faqs-2.php) page. By using this predefined action, you don’t have to write your own Action class. You just have to set up the struts-config file properly to use ForwardAction.

**24.What is IncludeAction?**

The IncludeAction class is useful when you want to integrate Struts into an application that uses Servlets. Use the IncludeAction class to include another resource in the response to the request being processed.

**25.What is the difference between ForwardAction and IncludeAction?**

The difference is that you need to use the IncludeAction only if the action is going to be included by another action or jsp. Use ForwardAction to forward a request to another resource in your application, such as a Servlet that already does business logic processing or even another JSP page.

**26.What is LookupDispatchAction?**

The LookupDispatchAction is a subclass of DispatchAction. It does a reverse lookup on the resource bundle to get the key and then gets the method whose name is associated with the key into the Resource Bundle.

**27.What is the use of LookupDispatchAction?**

LookupDispatchAction is useful if the method name in the Action is not driven by its name in the front end, but by the Locale independent key into the resource bundle. Since the key is always the same, the LookupDispatchAction shields your application from the side effects of I18N.

**28.What is difference between LookupDispatchAction and DispatchAction?**

The difference between LookupDispatchAction and DispatchAction is that the actual method that gets called in LookupDispatchAction is based on a lookup of a key value instead of specifying the method name directly.

**29.What is SwitchAction?**

The SwitchAction class provides a means to switch from a resource in one module to another resource in a different module. SwitchAction is useful only if you have multiple modules in your Struts application. The SwitchAction class can be used as is, without extending.

**30.What if <action> element has <forward> declaration with same name as global forward?**

In this case the global forward is not used. Instead the <action> element’s <forward> takes precendence.

**What is DynaActionForm?**

A specialized subclass of ActionForm that allows the creation of form beans with dynamic sets of properties (configured in configuration file), without requiring the developer to create a Java class for each type of form bean.

**32.What are the steps need to use DynaActionForm?**

Using a DynaActionForm instead of a custom subclass of ActionForm is relatively straightforward. You need to make changes in two places:

* In struts-config.xml: change your <form-bean> to be an org.apache.struts.action.DynaActionForm instead of some subclass of ActionForm

<form-bean name="loginForm"type="org.apache.struts.action.DynaActionForm" >  
    <form-property name="userName" type="[java](http://www.developersbook.com/struts/interview-questions/struts-interview-questions-faqs-3.php).lang.String"/>  
    <form-property name="password" type="java.lang.String" />  
</form-bean>

* In your Action subclass that uses your form bean:
  + import org.apache.struts.action.DynaActionForm
  + downcast the ActionForm parameter in execute() to a DynaActionForm
  + access the form fields with get(field) rather than getField()

import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import org.apache.struts.action.Action;  
import org.apache.struts.action.ActionForm;  
import org.apache.struts.action.ActionForward;  
import org.apache.struts.action.ActionMapping;  
import org.apache.struts.action.ActionMessage;  
import org.apache.struts.action.ActionMessages;  
  
  
**import org.apache.struts.action.DynaActionForm;**  
  
public class DynaActionFormExample extends Action {  
 public ActionForward execute(ActionMapping mapping, ActionForm form,  
 HttpServletRequest request, HttpServletResponse response)  
            [throws Exception](http://www.developersbook.com/struts/interview-questions/struts-interview-questions-faqs-3.php) {          
 **DynaActionForm loginForm = (DynaActionForm) form;**  
                ActionMessages errors = new ActionMessages();          
        if (((String) loginForm.get("userName")).equals("")) {  
            errors.add("userName", new ActionMessage(  
                            "error.userName.required"));  
        }  
        if (((String) loginForm.get("password")).equals("")) {  
            errors.add("password", new ActionMessage(  
                            "error.password.required"));  
        }  
        ...........

**33.How to display validation errors on jsp page?**

<html:errors/> tag displays all the errors. <html:errors/> iterates over ActionErrors request attribute.

**34.What are the various Struts** [**tag libraries**](http://www.developersbook.com/struts/interview-questions/struts-interview-questions-faqs-3.php)**?**

The various Struts tag libraries are:

* HTML Tags
* Bean Tags
* Logic Tags
* Template Tags
* Nested Tags
* Tiles Tags

**35.What is the use of <logic:iterate>?**

<logic:iterate> repeats the nested body content of this tag over a specified collection.

<table border=1>   
 <logic:iterate id="**customer**" name="customers">   
 <tr>   
 <td><bean:write name="customer" property="firstName"/></td>   
 <td><bean:write name="customer" property="lastName"/></td>   
 <td><bean:write name="customer" property="address"/></td>   
 </tr>   
 </logic:iterate>   
</table>

**36.What are differences between <bean:message> and <bean:write>**

**<bean:message>**: is used to retrive keyed values from resource bundle. It also supports the ability to include parameters that can be substituted for defined placeholders in the retrieved string.

<bean:message key="prompt.customer.firstname"/>

**<bean:write>**: is used to retrieve and [print](http://www.developersbook.com/struts/interview-questions/struts-interview-questions-faqs-3.php) the value of the bean property. <bean:write> has no body.

<bean:write name="customer" property="firstName"/>

**37.How the exceptions are handled in struts?**

Exceptions in Struts are handled in two ways:

* **Programmatic exception handling** : Explicit try/catch blocks in any code that can throw exception. It works well when custom value (i.e., of variable) needed when error occurs.
* **Declarative exception handling** :You can either define <global-exceptions> handling tags in your struts-config.xml or define the exception handling tags within <action></action> tag. It works well when custom page needed when error occurs. This approach applies only to exceptions thrown by Actions.

<global-exceptions>  
 <exception key="some.key"  
 type="java.lang.NullPointerException"  
 path="/WEB-INF/errors/null.jsp"/>  
</global-exceptions>

**or**

<exception key="some.key"   
 type="package.SomeException"   
 path="/WEB-INF/somepage.jsp"/>

**38.What is difference between ActionForm and DynaActionForm?**

* An ActionForm represents an HTML form that the user interacts with over one or more pages. You will provide properties to hold the state of the form with getters and setters to access them. Whereas, using DynaActionForm there is no need of providing properties to hold the state. Instead these properties and their type are declared in the struts-config.xml
* The DynaActionForm bloats up the Struts config file with the xml based definition. This gets annoying as the Struts Config file grow larger.
* The DynaActionForm is not strongly typed as the ActionForm. This means there is no compile time checking for the form fields. Detecting them at runtime is painful and makes you go through redeployment.
* ActionForm can be cleanly organized in packages as against the flat organization in the Struts Config file.
* ActionForm were designed to act as a Firewall between HTTP and the Action classes, i.e. isolate and encapsulate the HTTP request parameters from direct use in Actions. With DynaActionForm, the property access is no different than using request.getParameter( .. ).
* DynaActionForm construction at runtime requires a lot of Java Reflection (Introspection) machinery that can be avoided.

**39.How can we make message resources definitions file available to the Struts framework environment?**

We can make message resources definitions file (properties file) available to Struts framework environment by adding this file to struts-config.xml.

<message-resources parameter="com.login.struts.ApplicationResources"/>

**40.What is the life cycle of ActionForm?**

The lifecycle of ActionForm invoked by the RequestProcessor is as follows:

* Retrieve or Create Form Bean associated with Action
* "Store" FormBean in appropriate scope (request or session)
* Reset the properties of the FormBean
* Populate the properties of the FormBean
* Validate the properties of the FormBean
* Pass FormBean to Action

What is Action Class? Explain with Example

An Action class in the struts application extends Struts 'org.apache.struts.action.Action" Class.   
Action class acts as wrapper around the business logic and provides an inteface to the application's Model layer.   
An Action works as an adapter between the contents of an incoming HTTP request and the business logic that corresponds to it.   
Then the struts controller (ActionServlet) slects an appropriate Action and Request Processor creates an instance if necessary,   
and finally calls execute method of Action class.   
To use the Action, we need to Subclass and overwrite the execute() method. and your bussiness login in execute() method.   
The return type of the execute method is ActionForward which is used by the Struts Framework to forward the request to the JSP as per the value of the returned ActionForward object.   
ActionForward JSP from struts\_config.xml file.   
  
Developing our Action Class :   
  
Our Action class (EmpAction.java) is simple class that only forwards the success.jsp.   
Our Action class returns the ActionForward called "success", which is defined in the struts-config.xml file (action mapping is show later in this page).   
Here is code of our Action Class   
public class EmpAction extends Action   
{   
public ActionForward execute(   
ActionMapping mapping,   
ActionForm form,   
HttpServletRequest request,   
HttpServletResponse response) throws Exception{   
return mapping.findForward("success");   
}   
}   
  
mapping.findForward("success"); forward to JSP mentioned in struts\_config.xml.   
struts\_config.xml configuration is :   
  
<action   
path="/EmpAction"   
type="com.techfaq.EmpAction">   
<forward name="success" path="/success.jsp"/>   
</action>   
mapping.findForward("success") method forward to success.jsp (mentioned in struts\_config.xml);   
  
  
Here is the signature of the execute() method Action Class.   
  
public ActionForward execute(ActionMapping mapping,   
ActionForm form,   
javax.servlet.http.HttpServletRequest request,   
javax.servlet.http.HttpServletResponse response)   
throws java.lang.Exception   
  
Where   
mapping - The ActionMapping used to select this instance   
form - The optional ActionForm bean for this request (if any)   
request - The HTTP request we are processing   
response - The HTTP response we are creating   
Throws:   
Action class throws java.lang.Exception - if the application business logic throws an exception   
  
In the browser : http://localhost:8080/testApp/EmpAction.do   
This will call to execute() method of EmpAction and after that based on mapping.findForward("success") forward to success.jsp.

### **Q.How you can do Exception Handling in Struts ?**

There are two approaches available for the exception handling in struts.   
  
  
Declarative:   
Exceptions are defined in the struts-config.xml file and   
if the exception occurs the control is automatically passed to the appropriate error page.   
The <exception> tag is used to define the exception in the struts-config.xml file.   
  
For Example : (( If RuntimeException in SaveEmpAaction class , control goes to exception.jsp)   
  
<action path="/saveEmp"   
type="com.techfaq.SaveEmpAaction"   
input="/empform.jsp" >   
<exception key="error.system" type="java.lang.RuntimeException" path="/exception.jsp" />   
</action>   
  
Where   
  
Key: The key defines the key present in MessageResources.properties file to describe the exception occurred.   
  
Type: The class of the exception occurred.   
  
Path: The page where the control is to be followed in case exception occurred.   
  
Handler: The exception handler which will be called before passing the control to the file specified in path attribute   
  
OR   
  
  
Defining the Exceptions Globally for the struts-config.xml : ( If RuntimeException in any Action class , control goes to exception.jsp)   
<global-exceptions>   
<exception key="error.system" type="java.lang.RuntimeException" path="/exception.jsp" />   
</global-exceptions>   
  
  
Programmatic:   
In this approach the exceptions are caught using normal java language try/catch block and instead of showing the exception some meaningful messages are displayed.   
In this approach the flow of control is also maintained by the programs.   
The main drawback of the approach is the developer has to write the code for the flow of the application.

What are the Advantages of Struts ?

Struts follow MVC framework. So the JSP, Java and Action classes are organized and easily maintainable.   
Struts offers many advantages to the application programmer while reducing the development time and making the manageability of the application easier.   
Advantages of Struts :   
  
  
Centralized Configuration :   
Rather than hard coding information into java programs,many Struts values are represented in XML or property files.   
Struts\_config.xml file is the place from where you can get all information?s about your web application. This is organized.   
Your Action class , Form bean and JSP page information is in Struts\_config.xml so don't need to search . All info in one place.   
  
  
  
Form Beans :   
Don't need to set the form vales to your value object . When you want to capture data from a form ( In the servlet you do request.getParameter()).   
In the struts you don't need to do explicitly request.getParameter(). Struts request processor will do for you. All the input data will be set to form bean.   
  
  
  
Bean Tags :   
Struts provides a set of custom JSP tags (bean:write,in particular) that let you easily output the properties of JavaBeans components.   
Basically,these are concise and powerful variations of the standard jsp:useBean and jsp:getProperty tags.   
  
  
HTML tags :   
Struts provides a set of custom JSP tags to create HTML forms that are associated with JavaBeans components.   
  
  
Form Field Validation :   
  
Apache Struts has built-in capabilities for checking that form values are in the required format.   
If values are missing or in an improper format,the form can be automatically redisplayed with error messages and with the previously entered values maintained.   
This validation can be performed on the server (in Java),or both on the server and on the client (in JavaScript).

How Iterate Tag used with a Map ?

<logic:iterate id="user" name="userForm" property="usermap">   
<tr>   
<td>User Id: <bean:write name=" user" property="key"/></td>   
<td>Password: <bean:write name=" user" property="value"/></td>   
</tr>   
</logic:iterate>

How Iterate Tag used with a Map ?

<logic:iterate id="user" name="userForm" property="usermap">   
<tr>   
<td>User Id: <bean:write name=" user" property="key"/></td>   
<td>Password: <bean:write name=" user" property="value"/></td>   
</tr>   
</logic:iterate>

How does client side validation using validator framework work in struts ?

There are two configuration files used.   
One if called validator-rules.xml and the other is validation.xml.   
This is example of client side validation :Java Script message.   
  
Step 1.   
In the JSP page : empform.jsp   
<html:form action="/EmpSaveAaction" method="post" onsubmit="return validateEmpForm(this);">   
  
<html:text property="firstName" size="30" maxlength="30"/>   
<html:text property="lastName" size="30" maxlength="30"/>   
  
<html:submit>Save</html:submit>   
  
<!-- Begin Validator Javascript Function-->   
<html:javascript formName="empForm"/>   
<!-- End of Validator Javascript Function-->   
</html:form>   
  
  
You have to add   
<html:javascript formName="empForm"/> in JSP for activate client side validation .   
  
Step 2.   
Add action mapping in struts-config.xml   
  
<action   
path="/EmpSaveAaction"   
type="empForm"   
name="AddressForm"   
scope="request"   
validate="true"   
input="/empform.jsp">   
<forward name="success" path="/success.jsp"/>   
</action>   
  
and add the form bean inside <form-beans> </form-beans> tag   
<form-beans>   
<form-bean name="empForm" type="com.techfaq.form.EmpForm" />   
</form-beans>   
  
  
Step 3.   
validator-rules.xml :   
  
The validator-rules.xml file defines the Validator definitions available for a given application.   
The validator-rules.xml file acts as a template, defining all of the possible Validators that are available to an application.   
Example validator-rules.xml File :   
<form-validation>   
<global>   
<validator   
name="required"   
classname="org.apache.struts.util.StrutsValidator"   
method="validateRequired"   
methodparams="java.lang.Object,   
org.apache.commons.validator.ValidatorAction,   
org.apache.commons.validator.Field,   
org.apache.struts.action.ActionErrors,   
javax.servlet.http.HttpServletRequest"   
msg="errors.required"/>   
  
</global>   
</form-validation>   
  
Step 4.   
validation.xml File :   
The validation.xml file is where you couple the individual Validators defined in the validator-rules.xml to components within your application.   
validation.xml File   
  
<form-validation>   
<formset>   
<form name="empForm">   
<field   
property="firstName"   
depends="required">   
<arg key="label.firstName"/>   
</field>   
  
<field   
property="lastName"   
depends="required">   
<arg key="label.lastName"/>   
</field>   
</form>   
</formset>   
</form-validation>   
  
In the empForm firstName and lastName are the required filed.   
So in the above configuration you can see we add for both firstName and lastName.   
You can see depends="required" - "required" property is defind in validator-rules.xml.   
  
In the resource bundle : application\_resource.propertis file   
label.firstName=First Name   
label.lastName=Last Name   
  
#Error messages used by the Validator   
errors.required={0} is required.   
  
Based on the validation.xml File configuration .   
Error in jsp will be : Java Script message.   
First Name is required.   
Last Name is required.   
  
{0} will be filled by (First Name or Last Name) because validation.xml above configuration you have defind <arg key="label.firstName"/>.

How to do File Upload in Struts ?

Step 1.   
Create a form bean   
  
public class FileUploadForm extends ActionForm   
{   
private FormFile file;   
  
public FormFile getFile() {   
return file;   
}   
  
public void setFile(FormFile file) {   
this.file = file;   
}   
}   
  
  
Step 2.   
  
In the struts-config.xml file add   
<form-bean   
name="FileUploadForm"   
type="com.techfaq.form.FileUploadForm"/>   
  
Step 3.   
  
add action mapping entry in the struts-config.xml file:   
  
<action   
path="/FileUploadAndSave"   
type="com.techfaq.action.FileUploadAndSaveAction"   
name="FileUploadForm"   
scope="request"   
validate="true"   
input="/pages/fileupload.jsp">   
<forward name="success" path="/jsp/success.jsp"/>   
</action>   
  
Step 4.   
In the JSP   
<html:form action="/FileUploadAndSave" method="post" enctype="multipart/form-data">   
File Name   
  
<html:file property="file"/>   
  
<html:submit>Upload File</html:submit>   
  
</html:form>   
  
Step 5.   
In the Action class write the code   
  
public ActionForward execute(   
ActionMapping mapping,   
ActionForm form,   
HttpServletRequest request,   
HttpServletResponse response) throws Exception{   
FileUploadForm myForm = (FileUploadForm)form;   
  
// Process the FormFile   
FormFile file = myForm.getFile();   
String contentType = file.getContentType();   
//Get the file name   
String fileName = file.getFileName();   
int fileSize = file.getFileSize();   
byte[] fileData = file.getFileData();   
//Get the servers upload directory real path name   
String filePath = getServlet().getServletContext().getRealPath("/") +"uploadfile";   
/\* Save file on the server \*/   
if(!fileName.equals("")){   
System.out.println("Server path:" +filePath);   
//Create file   
File fileToCreate = new File(file, fileName);   
//If file does not exists create file   
if(!fileToCreate.exists()){   
FileOutputStream fileOutStream = new FileOutputStream(fileToCreate);   
fileOutStream.write(file.getFileData());   
fileOutStream.flush();   
fileOutStream.close();   
}   
  
  
}   
  
return mapping.findForward("success");   
}   
  
  
File will be oploaded to "uploadfile" directory og your server.

What is DynaActionForm ? and How you can retrive the value which is set in the JSP Page in case of DynaActionForm ?

DynaActionForm is specialized subclass of ActionForm that allows the creation of form beans with dynamic sets of properties,   
without requiring the developer to create a Java class for each type of form bean.   
DynaActionForm eliminates the need of FormBean class and now the form bean definition can be written into the struts-config.xml file. So, i   
t makes the FormBean declarative and this helps the programmer to reduce the development time.   
  
For Example : you have a EmpForm and you don't want a java class (EmpForm).   
EmpForm has propertis   
firstName, lastName, country   
  
In the struts-config.xml file , declare the form bean   
<form-bean name="EmpForm"   
type="org.apache.struts.action.DynaActionForm">   
<form-property name="firstName" type="java.lang.String"/>   
<form-property name="lastName" type="java.lang.String"/>   
<form-property name="country" type="java.lang.String" />   
</form-bean>   
  
Add action mapping in the struts-config.xml file:   
  
<action path="/saveEmp" type="com.techfaq.action.EmpSaveAction"   
name="EmpForm"   
scope="request"   
validate="true"   
input="/pages/empform.jsp">   
  
<forward name="success" path="/jsp/success.jsp"/>   
<forward name="failure" path="/jsp/error.jsp" />   
  
</action>   
  
In the Action class.   
public class EmpSaveAction extends Action   
{   
public ActionForward execute(   
ActionMapping mapping,   
ActionForm form,   
HttpServletRequest request,   
HttpServletResponse response) throws Exception{   
  
DynaActionForm empForm = (DynaActionForm)form;   
// this is the way you can retrive the value which is set in the JSP Page   
String firstName = (String)empForm.get("firstName");   
String lastName = (String)empForm.get("lastName");   
return mapping.findForward("success");   
}   
}   
}   
  
In the JSP page   
<html:text property="firstName" size="30" maxlength="30"/>   
<html:text property="lastName" size="30" maxlength="30"/>

How to Setup validator framework in Struts ?

Step 1.   
place validator-rules.xml and validation.xml in the WEB-INF directory.   
  
Step 2.   
Add the blow lines to struts-config.xml   
<plug-in className="org.apache.struts.validator.ValidatorPlugIn">   
<set-property property="pathnames" value="/WEB-INF/validator-rules.xml,   
/WEB-INF/validation.xml"/>   
</plug-in>   
  
Step 3.   
In the jsp add the tag   
<html:errors />   
  
Step 4.   
Add   
struts.jar   
commons-beanutils.jar   
commons-collections.jar   
commons-digester.jar   
commons-fileupload.jar   
commons-lang.jar   
commons-logging.jar   
commons-validator.jar   
  
into WEB-INF/lib directory   
  
Explanation :   
  
There are two configuration files used.   
One if called validator-rules.xml and the other is validation.xml.   
This is example of server side validation :   
  
  
validator-rules.xml :   
  
The validator-rules.xml file defines the Validator definitions available for a given application.   
The validator-rules.xml file acts as a template, defining all of the possible Validators that are available to an application.   
Example validator-rules.xml File :   
<form-validation>   
<global>   
<validator   
name="required"   
classname="org.apache.struts.util.StrutsValidator"   
method="validateRequired"   
methodparams="java.lang.Object,   
org.apache.commons.validator.ValidatorAction,   
org.apache.commons.validator.Field,   
org.apache.struts.action.ActionErrors,   
javax.servlet.http.HttpServletRequest"   
msg="errors.required"/>   
  
</global>   
</form-validation>   
  
validation.xml File :   
The validation.xml file is where you couple the individual Validators defined in the validator-rules.xml to components within your application.   
validation.xml File   
  
<form-validation>   
<formset>   
<form name="empForm">   
<field   
property="firstName"   
depends="required">   
<arg0 key="label.firstName"/>   
</field>   
  
<field   
property="lastName"   
depends="required">   
<arg0 key="label.lastName"/>   
</field>   
</form>   
</formset>   
</form-validation>   
  
In the empForm firstName and lastName are the required filed.   
So in the above configuration you can see we add for both firstName and lastName.   
You can see depends="required" - "required" property is defind in validator-rules.xml.   
  
In the resource bundle : application\_resource.propertis file   
label.firstName=First Name   
label.lastName=Last Name   
  
#Error messages used by the Validator   
errors.required={0} is required.   
  
Based on the validation.xml File configuration .   
Error in jsp will be :   
First Name is required.   
Last Name is required.   
  
{0} will be filled by (First Name or Last Name) because validation.xml above configuration you have defind <arg0 key="label.firstName"/>.

What helpers in the form of JSP pages are provided in Struts framework?

--struts-html.tld   
--struts-bean.tld   
--struts-logic.tld

How to Setup validator framework in Struts ?

Step 1.   
place validator-rules.xml and validation.xml in the WEB-INF directory.   
  
Step 2.   
Add the blow lines to struts-config.xml   
<plug-in className="org.apache.struts.validator.ValidatorPlugIn">   
<set-property property="pathnames" value="/WEB-INF/validator-rules.xml,   
/WEB-INF/validation.xml"/>   
</plug-in>   
  
Step 3.   
In the jsp add the tag   
<html:errors />   
  
Step 4.   
Add   
struts.jar   
commons-beanutils.jar   
commons-collections.jar   
commons-digester.jar   
commons-fileupload.jar   
commons-lang.jar   
commons-logging.jar   
commons-validator.jar   
  
into WEB-INF/lib directory   
  
Explanation :   
  
There are two configuration files used.   
One if called validator-rules.xml and the other is validation.xml.   
This is example of server side validation :   
  
  
validator-rules.xml :   
  
The validator-rules.xml file defines the Validator definitions available for a given application.   
The validator-rules.xml file acts as a template, defining all of the possible Validators that are available to an application.   
Example validator-rules.xml File :   
<form-validation>   
<global>   
<validator   
name="required"   
classname="org.apache.struts.util.StrutsValidator"   
method="validateRequired"   
methodparams="java.lang.Object,   
org.apache.commons.validator.ValidatorAction,   
org.apache.commons.validator.Field,   
org.apache.struts.action.ActionErrors,   
javax.servlet.http.HttpServletRequest"   
msg="errors.required"/>   
  
</global>   
</form-validation>   
  
validation.xml File :   
The validation.xml file is where you couple the individual Validators defined in the validator-rules.xml to components within your application.   
validation.xml File   
  
<form-validation>   
<formset>   
<form name="empForm">   
<field   
property="firstName"   
depends="required">   
<arg0 key="label.firstName"/>   
</field>   
  
<field   
property="lastName"   
depends="required">   
<arg0 key="label.lastName"/>   
</field>   
</form>   
</formset>   
</form-validation>   
  
In the empForm firstName and lastName are the required filed.   
So in the above configuration you can see we add for both firstName and lastName.   
You can see depends="required" - "required" property is defind in validator-rules.xml.   
  
In the resource bundle : application\_resource.propertis file   
label.firstName=First Name   
label.lastName=Last Name   
  
#Error messages used by the Validator   
errors.required={0} is required.   
  
Based on the validation.xml File configuration .   
Error in jsp will be :   
First Name is required.   
Last Name is required.   
  
{0} will be filled by (First Name or Last Name) because validation.xml above configuration you have defind <arg0 key="label.firstName"/>.

Q.How does validator framework work in Struts ?

There are two configuration files used.   
One if called validator-rules.xml and the other is validation.xml.   
This is example of server side validation :   
  
  
validator-rules.xml :   
  
The validator-rules.xml file defines the Validator definitions available for a given application.   
The validator-rules.xml file acts as a template, defining all of the possible Validators that are available to an application.   
Example validator-rules.xml File :   
<form-validation>   
<global>   
<validator   
name="required"   
classname="org.apache.struts.util.StrutsValidator"   
method="validateRequired"   
methodparams="java.lang.Object,   
org.apache.commons.validator.ValidatorAction,   
org.apache.commons.validator.Field,   
org.apache.struts.action.ActionErrors,   
javax.servlet.http.HttpServletRequest"   
msg="errors.required"/>   
  
</global>   
</form-validation>   
  
validation.xml File :   
The validation.xml file is where you couple the individual Validators defined in the validator-rules.xml to components within your application.   
validation.xml File   
  
<form-validation>   
<formset>   
<form name="empForm">   
<field   
property="firstName"   
depends="required">   
<arg0 key="label.firstName"/>   
</field>   
  
<field   
property="lastName"   
depends="required">   
<arg0 key="label.lastName"/>   
</field>   
</form>   
</formset>   
</form-validation>   
  
In the empForm firstName and lastName are the required filed.   
So in the above configuration you can see we add for both firstName and lastName.   
You can see depends="required" - "required" property is defind in validator-rules.xml.   
  
In the resource bundle : application\_resource.propertis file   
label.firstName=First Name   
label.lastName=Last Name   
  
#Error messages used by the Validator   
errors.required={0} is required.   
  
Based on the validation.xml File configuration .   
Error in jsp will be :   
First Name is required.   
Last Name is required.   
  
{0} will be filled by (First Name or Last Name) because validation.xml above configuration you have defind <arg0 key="label.firstName"/>.

What are Validators? and What are Basic Validators provided by the framework ?

Validator is a Java class that, when called by the Validator framework, executes a validation rule. The framework knows how to invoke a Validator class based on its method signature, as defined in a configuration file.   
  
1) byte,short,integer,long,float,double   
  
2) creditCard - Checks if the field is a valid credit card number.   
3) date - Checks if the field is a valid date.   
4) email - Checks if the field is a valid email address.   
4) mask - Succeeds if the field matches the corresponding regular expression mask.   
5) maxLength - Checks if the value's length is less than or equal to the given maximum length.   
6) minLength - Checks if the value's length is greater than or equal to the given minimum length.   
7) range - Checks if the value is within a minimum and maximum range.   
8) required - Checks if the field isn't null and length of the field is greater than zero, not including whitespace.

What are Validators? and What are Basic Validators provided by the framework ?

Validator is a Java class that, when called by the Validator framework, executes a validation rule. The framework knows how to invoke a Validator class based on its method signature, as defined in a configuration file.   
  
1) byte,short,integer,long,float,double   
  
2) creditCard - Checks if the field is a valid credit card number.   
3) date - Checks if the field is a valid date.   
4) email - Checks if the field is a valid email address.   
4) mask - Succeeds if the field matches the corresponding regular expression mask.   
5) maxLength - Checks if the value's length is less than or equal to the given maximum length.   
6) minLength - Checks if the value's length is greater than or equal to the given minimum length.   
7) range - Checks if the value is within a minimum and maximum range.   
8) required - Checks if the field isn't null and length of the field is greater than zero, not including whitespace.

What is the Difference between DispatchAction and LookupDispatchAction ?

LookupDispatchAction is subclass of DispatchAction.   
In case of DispatchAction, you have to declare the method name in the JSP page.   
For Example :   
http://localhost:8080/emp/empaction.do?step=add // IN CASE OF DispatchAction   
here step=add , we are delaring method name in JSP.   
or   
<html:submit property="step">Add</html:submit> //IN CASE OF DispatchAction   
so we can't use localization for button.   
  
  
To over come both the issues below   
a)method name declaration in JSP   
b)can't use localization for button   
We will go for LookupDispatchAction.   
  
In the LookupDispatchAction :   
For example :   
If there are three operation like add, delete, update employee.   
You can create three different Actions ?   
AddEmpAction, DeleteEmpAction and UpdateEmpAction.   
This is a valid approach, although not elegant since there might be duplication of code across   
the Actions since they are related. LookupDispatchAction is the answer to this   
problem. With LookupDispatchAction, you can combine all three Actions into one.   
  
Example :   
Step 1.   
three buttons might be   
<html:submit property="step">   
<bean:message key="button.add"/>   
</html:submit>   
<html:submit property="step">   
<bean:message key="button.delete"/>   
</html:submit>   
<html:submit property="step">   
<bean:message key="button.update"/>   
</html:submit>   
  
//No need method name declaration in JSP . Button name from resource bundle.   
  
Step 2.   
In the the Resource Bundle. //Here you can add localization.   
button.add=Add   
button.delete=Delete   
button.update=Update   
  
Step 3.   
Implement a method named getKeyMethodMap() in the subclass of the   
LookupDispatchAction. The method returns a java.util.Map. The   
keys used in the Map should be also used as keys in Message Resource   
Bundle.   
  
public class EmpAction extends LookupDispatchAction {   
  
public Map getKeyMethodMap()   
{   
Map map = new HashMap();   
map.put("button.add", "add");   
map.put("button.delete", "delete");   
map.put("button.update", "update");   
}   
  
  
public ActionForward add(ActionMapping mapping,   
ActionForm form, HttpServletRequest request,   
HttpServletResponse response) throws Exception   
{   
//your logic   
mapping.findForward("add-success");   
}   
  
public ActionForward delete(ActionMapping mapping,   
ActionForm form, HttpServletRequest request,   
HttpServletResponse response) throws Exception   
{   
//your logic   
mapping.findForward("delete-success");   
}   
  
public ActionForward update(ActionMapping mapping,   
ActionForm form, HttpServletRequest request,   
HttpServletResponse response) throws Exception   
{   
//your logic   
mapping.findForward("update-success");   
}   
  
}   
  
in struts-config.xml   
  
<action path="/empaction"   
input="/empform.jsp"   
type="list.EmpAction"   
parameter="step"   
scope="request"   
validate="false">   
<forward name="add-success"   
path="addEmpSuccess.jsp"   
redirect="true"/>   
<forward name="delete-success"   
path="deleteEmpSuccess.jsp"   
redirect="true"/>   
<forward name="update-success"   
path="updateEmpSuccess.jsp"   
redirect="true"/>   
</action>   
  
for every form submission, LookupDispatchAction does the   
reverse lookup on the resource bundle to get the key and then gets the method   
whose name is associated with the key from   
getKeyMethodmap().   
  
  
Flow of LookupDispatchAction:   
a) Form the button name "Add" get the key "button.add" fro resource bundle.   
b) then in the Map getKeyMethodMap() find the value associated with the key "button.add".   
c) value from the Map is "add" . then call add() method of the same action class.

Q. What is LookupDispatchAction?

When a set of actions is closely related and   
separating them into multiple Actions would result in duplication of code you can use LookupDispatchAction.   
LookupDispatchAction is subclass of DispatchAction.   
  
In case of DispatchAction, you have to declare the method name in the JSP page.   
For Example :   
http://localhost:8080/emp/empaction.do?step=add // IN CASE OF DispatchAction   
here step=add , we are delaring method name in JSP.   
or   
<html:submit property="step">Add</html:submit> //IN CASE OF DispatchAction   
so we can't use localization for button.   
  
  
To over come both the issues below   
a)method name declaration in JSP   
b)can't use localization for button   
We will go for LookupDispatchAction.   
  
For example :   
If there are three operation like add, delete, update employee.   
You can create three different Actions ?   
AddEmpAction, DeleteEmpAction and UpdateEmpAction.   
This is a valid approach, although not elegant since there might be duplication of code across   
the Actions since they are related. LookupDispatchAction is the answer to this   
problem. With LookupDispatchAction, you can combine all three Actions into one.   
  
Example :   
Step 1.   
three buttons might be   
<html:submit property="step">   
<bean:message key="button.add"/>   
</html:submit>   
<html:submit property="step">   
<bean:message key="button.delete"/>   
</html:submit>   
<html:submit property="step">   
<bean:message key="button.update"/>   
</html:submit>   
  
//No need method name declaration in JSP . Button name from resource bundle.   
  
Step 2.   
In the the Resource Bundle. //Here you can add localization.   
button.add=Add   
button.delete=Delete   
button.update=Update   
  
Step 3.   
Implement a method named getKeyMethodMap() in the subclass of the   
LookupDispatchAction. The method returns a java.util.Map. The   
keys used in the Map should be also used as keys in Message Resource   
Bundle.   
  
public class EmpAction extends LookupDispatchAction {   
  
public Map getKeyMethodMap()   
{   
Map map = new HashMap();   
map.put("button.add", "add");   
map.put("button.delete", "delete");   
map.put("button.update", "update");   
}   
  
  
public ActionForward add(ActionMapping mapping,   
ActionForm form, HttpServletRequest request,   
HttpServletResponse response) throws Exception   
{   
//your logic   
mapping.findForward("add-success");   
}   
  
public ActionForward delete(ActionMapping mapping,   
ActionForm form, HttpServletRequest request,   
HttpServletResponse response) throws Exception   
{   
//your logic   
mapping.findForward("delete-success");   
}   
  
public ActionForward update(ActionMapping mapping,   
ActionForm form, HttpServletRequest request,   
HttpServletResponse response) throws Exception   
{   
//your logic   
mapping.findForward("update-success");   
}   
  
}   
  
in struts-config.xml   
  
<action path="/empaction"   
input="/empform.jsp"   
type="list.EmpAction"   
parameter="step"   
scope="request"   
validate="false">   
<forward name="add-success"   
path="addEmpSuccess.jsp"   
redirect="true"/>   
<forward name="delete-success"   
path="deleteEmpSuccess.jsp"   
redirect="true"/>   
<forward name="update-success"   
path="updateEmpSuccess.jsp"   
redirect="true"/>   
</action>   
  
for every form submission, LookupDispatchAction does the   
reverse lookup on the resource bundle to get the key and then gets the method   
whose name is associated with the key from   
getKeyMethodmap().   
  
Flow of LookupDispatchAction:   
a) Form the button name "Add" get the key "button.add" fro resource bundle.   
b) then in the Map getKeyMethodMap() find the value associated with the key "button.add".   
c) value from the Map is "add" . then call add() method of the same action class.

.How to create a multiple selections list in Struts? and retrive seleted values ?

This is the code to display multiple selections list and retrive seleted values in struts.   
medList list contains list of Medium objects.   
  
Java :   
bean class.   
public class Medium {   
    int medId;   
    String medName;   
  
    /\*\*   
     \* @return Returns the medId.   
     \*/   
    public int getMedId() {   
        return medId;   
    }   
    /\*\*   
     \* @param medId The medId to set.   
     \*/   
    public void setMedId(int medId) {   
        this.medId = medId;   
    }   
    /\*\*   
     \* @return Returns the medName.   
     \*/   
    public String getMedName() {   
        return medName;   
    }   
    /\*\*   
     \* @param medName The medName to set.   
     \*/   
    public void setMedName(String medName) {   
        this.medName = medName;   
    }   
}   
  
In the Form Class :   
  
public class MediumForm {   
private List medList;   
private String[] med;   
    public void setMedList(List medList){   
        this.medList = medList;   
    }   
       
    public List getMedList(){   
        return this.medList;   
    }   
  
    public String[] getMed() {   
        return med;   
    }   
       
    public void setMed(String[] med) {   
        this.med = med;   
    }   
       
  
}   
  
In the Action class :   
  
List medList = DAO.getMediums();   
form.setMedList(medList);   
  
DAO Class :   
  
DAO Class to retrive mediums in data base.   
public static List getMediums(){   
        PreparedStatement pStmt = null;   
     Connection conn = null;   
     boolean success = false;   
     ResultSet rs = null;   
     List medList = new ArrayList();   
       
     try{   
         conn = getConnection();   
            
         String sql = " select \* from MEDIUM ";   
         pStmt = conn.prepareStatement(sql);   
                     
         rs = pStmt.executeQuery();   
         while(rs.next()){   
             Medium med = new Medium();   
             med.setMedId(rs.getInt("MED\_ID"));   
             med.setMedName(rs.getString("MEDIUM\_NAME"));   
             medList.add(med);   
             }   
           
           
           
     }catch(Exception e){   
         e.printStackTrace();   
            
     }finally{   
         closeConnectionProp(conn,pStmt,rs);   
     }   
           
     return medList;   
       
    }   
  
JSP Struts:   
  
<html:select name="MediumForm" property="med" multiple="true">   
  
<bean:define name="MediumForm" property="medList" id="mlist" />   
<html:options collection="mlist" property="medId" labelProperty="medName" />   
</html:select>   
  
  
  
In the Action class again :   
how to retrive the selected values.   
  
String[] med = form.getMed();

.How to create a drop down list in Struts?

|  |
| --- |
| This is the code to display drop down list and retrive seleted value in struts.  medList list contains list of Medium objects.   Java :  bean class.  public class Medium {      int medId;      String medName;       /\*\*       \* @return Returns the medId.       \*/      public int getMedId() {          return medId;      }      /\*\*       \* @param medId The medId to set.       \*/      public void setMedId(int medId) {          this.medId = medId;      }      /\*\*       \* @return Returns the medName.       \*/      public String getMedName() {          return medName;      }      /\*\*       \* @param medName The medName to set.       \*/      public void setMedName(String medName) {          this.medName = medName;      }  }   In the Form Class :   public class MediumForm {  private List medList;  private String med;      public void setMedList(List medList){          this.medList = medList;      }            public List getMedList(){          return this.medList;      }       public void setMed(String medList){          this.med = med;      }            public String getMed(){          return this.med;      }         }   In the Action class :   List medList = DAO.getMediums();  form.setMedList(medList);   DAO Class :   DAO Class to retrive mediums in data base.  public static List getMediums(){          PreparedStatement pStmt = null;       Connection conn = null;       boolean success = false;       ResultSet rs = null;       List medList = new ArrayList();             try{           conn = getConnection();                      String sql = " select \* from MEDIUM ";           pStmt = conn.prepareStatement(sql);                               rs = pStmt.executeQuery();           while(rs.next()){               Medium med = new Medium();               med.setMedId(rs.getInt("MED\_ID"));               med.setMedName(rs.getString("MEDIUM\_NAME"));               medList.add(med);               }                                     }catch(Exception e){           e.printStackTrace();                  }finally{           closeConnectionProp(conn,pStmt,rs);       }                 return medList;            }   JSP Struts:   <html:select name="MediumForm" property="med" >   <bean:define name="MediumForm" property="medList" id="mlist" />  <html:options collection="mlist" property="id" labelProperty="medName" />  </html:select>     In the Action class again :  how to retrive the select value.   String med = form.getMed(); |
| This is the code to display drop down list and retrive seleted value in struts.  medList list contains list of Medium objects.   Java :  bean class.  public class Medium {      int medId;      String medName;       /\*\*       \* @return Returns the medId.       \*/      public int getMedId() {          return medId;      }      /\*\*       \* @param medId The medId to set.       \*/      public void setMedId(int medId) {          this.medId = medId;      }      /\*\*       \* @return Returns the medName.       \*/      public String getMedName() {          return medName;      }      /\*\*       \* @param medName The medName to set.       \*/      public void setMedName(String medName) {          this.medName = medName;      }  }   In the Form Class :   public class MediumForm {  private List medList;  private String med;      public void setMedList(List medList){          this.medList = medList;      }            public List getMedList(){          return this.medList;      }       public void setMed(String medList){          this.med = med;      }            public String getMed(){          return this.med;      }         }   In the Action class :   List medList = DAO.getMediums();  form.setMedList(medList);   DAO Class :   DAO Class to retrive mediums in data base.  public static List getMediums(){          PreparedStatement pStmt = null;       Connection conn = null;       boolean success = false;       ResultSet rs = null;       List medList = new ArrayList();             try{           conn = getConnection();                      String sql = " select \* from MEDIUM ";           pStmt = conn.prepareStatement(sql);                               rs = pStmt.executeQuery();           while(rs.next()){               Medium med = new Medium();               med.setMedId(rs.getInt("MED\_ID"));               med.setMedName(rs.getString("MEDIUM\_NAME"));               medList.add(med);               }                                     }catch(Exception e){           e.printStackTrace();                  }finally{           closeConnectionProp(conn,pStmt,rs);       }                 return medList;            }   JSP Struts:   <html:select name="MediumForm" property="med" >   <bean:define name="MediumForm" property="medList" id="mlist" />  <html:options collection="mlist" property="id" labelProperty="medName" />  </html:select>     In the Action class again :  how to retrive the select value.   String med = form.getMed(); |

What is DispatchAction ?

When a set of actions is closely related and   
separating them into multiple Actions would result in duplication of code you can use DispatchAction.   
For example :   
If there are three operation like add, delete, update employee.   
You can create three different Actions ?   
AddEmpAction, DeleteEmpAction and UpdateEmpAction.   
This is a valid approach, although not elegant since there might be duplication of code across   
the Actions since they are related. DispatchAction is the answer to this   
problem. With DispatchAction, you can combine all three Actions into one.   
  
three urls might be   
http://localhost:8080/emp/empaction.do?step=add   
http://localhost:8080/emp/empaction.do?step=delete   
http://localhost:8080/emp/empaction.do?step=update   
  
public class EmpAction extends DispatchAction {   
public ActionForward add(ActionMapping mapping,   
ActionForm form, HttpServletRequest request,   
HttpServletResponse response) throws Exception   
{   
//your logic   
mapping.findForward("add-success");   
}   
  
public ActionForward delete(ActionMapping mapping,   
ActionForm form, HttpServletRequest request,   
HttpServletResponse response) throws Exception   
{   
//your logic   
mapping.findForward("delete-success");   
}   
  
public ActionForward update(ActionMapping mapping,   
ActionForm form, HttpServletRequest request,   
HttpServletResponse response) throws Exception   
{   
//your logic   
mapping.findForward("update-success");   
}   
  
}   
  
in struts-config.xml   
  
<action path="/empaction"   
input="/empform.jsp"   
type="list.EmpAction"   
parameter="step"   
scope="request"   
validate="false">   
<forward name="add-success"   
path="addEmpSuccess.jsp"   
redirect="true"/>   
<forward name="delete-success"   
path="deleteEmpSuccess.jsp"   
redirect="true"/>   
<forward name="update-success"   
path="updateEmpSuccess.jsp"   
redirect="true"/>   
</action>   
  
Notice that one of the HTTP request parameter in the four URLs is also named "step". Now, it is all coming   
together. DispatchAction knows what parameter to look for in the incoming   
URL request through this attribute named parameter in struts-config.xml.

Q. What is IncludeAction ?

IncludeAction included resulting resource in the HTTP response .   
In the JSP which don't use struts for include we do   
<jsp:include page="/test/ServletA"/>   
  
In struts JSP   
<jsp:include page="/App1/legacyA.do" />   
  
In the struts-config.xml   
<action path="/legacyA"   
parameter="/test/LegacyServletA"   
type="org.apache.struts.actions.IncludeAction" />

Q. How to Protect JSPs from direct access ?

JSPs located in the WEB-INF and its sub-directories are protected from outside access.   
If you want to go pageB.jsp from pageA.jsp   
<html:link action="gotoPageB">Go to Page B</html:link>   
  
in the struts-config.xml   
<action path="/gotoPageB"   
parameter="/WEB-INF/pageB.jsp"   
type="org.apache.struts.actions.ForwardAction"/>

What is ForwardAction ?

Suppose you want to go from PageA.jsp to PageB.jsp in your Struts   
application. The easy way of achieving this is to add a hyperlink in PageA.jsp as   
follows:   
<a href="PageB.jsp">Go to Page B</a>   
or even better, as follows:   
<html:link page="/PageB.jsp">Go to Page B</html:link>   
However this violates the MVC spirit by directly accessing   
the JSP.   
Struts provides a built-in Action class called ForwardAction to address this   
issue. With ForwardAction, the Struts Controller is still in the loop while   
navigating from PageA to PageB. There are two steps involved in using the   
ForwardAction. They are:   
\_ First, declare the PageA hyperlink that takes you to PageB as follows:   
<html:link page="/gotoPageB.do">Go to Page B</html:link>   
\_ Next, add an ActionMapping in the Struts Config file as follows:   
<action path="/gotoPageB"   
parameter="/PageB.jsp"   
type="org.apache.struts.actions.ForwardAction" />

How does reset() and Validate() method struts work ?

reset(): reset() method is called by Struts Framework with each request that uses the defined ActionForm. The purpose of this method is to reset all of the ActionForm's data members prior to the new request values being set.   
Example:   
/\*\*   
         \* Reset the form.   
         \*/   
        public void reset(ActionMapping mapping, HttpServletRequest request) {   
               
            super.reset(mapping,request);   
         this.password = null;   
         this.username = null;   
         this.guest = null;   
        }   
  
validate() : Used to validate properties after they have been populated; Called before FormBean is handed to Action. Returns a collection of ActionError as ActionErrors. Following is the method signature for the validate() method.   
  
Example :   
/\*\*   
         \* Validate the form's input.   
         \*/   
        public ActionErrors validate(ActionMapping mapping,   
                HttpServletRequest request) {   
                           
            ActionErrors errors = new ActionErrors();   
  
if(username == null){   
                    errors.add(ActionErrors.GLOBAL\_ERROR, new ActionError("error.username.required"));   
                }   
return errors;   
}

Multiple buttons in struts using java script?

<html:form action="/organization.do?method=createPage" >   
<tr>   
            <td>Fax</td>   
            <td><html:text property="fax" size="20"/></td>   
        </tr>   
        <tr>   
            <td>Email</td>   
            <td><html:text property="email" size="20"/></td>   
        </tr>   
  
<input type="button" value="Save and Continue" name="B1" onClick="save()">   
    &nbsp; <input type="button" value="Discard Changes" name="B2" onClick="discard()">   
</html:form>   
  
In the Java Script   
<script type="text/javascript" >   
  
    function save() {   
           
        document.OrganizationForm.action ="<%=request.getContextPath()%>/organization.do?method=save";   
        document.OrganizationForm.submit();   
    }   
  
    function discard() {   
           
        document.OrganizationForm.action ="<%=request.getContextPath()%>/organization.do?method=discard";   
        document.OrganizationForm.submit();   
    }   
</script>

Q.Mutli-click prevention using struts tokens with code example.

|  |
| --- |
| Struts has 3 methods use for the token, saveToken(), isTokenValid() and resetToken().  saveToken() - generate the token key and save to request/session attribute.  isTokenValid() - validate submitted token key against the 1 store in request/session.  resetToken() - reset the token key   Example :  Step 1.  Action Class where saveToken() before JSP Page.  First saveToken() then forward to your jsp.  Upon loading the form, invokes saveToken() on the action class to create and store the token key. Struts will store the generated key in request/session.   public class LoadAction extends Action  {  public ActionForward execute(ActionMapping mapping,ActionForm form,HttpServletRequest request,HttpServletResponse response)  {    ActionForward forward;       forward=mapping.findForward("FirstPage");// this is the jsp page where you want to struts tokens.       saveToken(request);             return forward;  }  }   Step 2.  If the token successfully created, when view source on the browser you will see the token, the token key is stored as a hidden field:  In jsp page :   <%@ page import="org.apache.struts.action.Action"%>  <%@ page import="org.apache.struts.taglib.html.Constants"%>   <%@ taglib uri="/WEB-INF/struts-tiles.tld" prefix="tiles" %>  <%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>  <%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>  <%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>  <html>  <head> <title> First Page </title> </head>   <body>  <form name="MyForm" method="post" action="/dpsubm/getForm/submit.do">  <input type="text" name="name" >  <input type="hidden" name="<%= Constants.TOKEN\_KEY %>"  value="<%= session.getAttribute(Action.TRANSACTION\_TOKEN\_KEY) %>" >  <input type="submit" value="submit">  </form>  </body>  </html>   Step 3. Your logic  Once the form submitted, invokes isTokenValid() on the action class, it will validate the submitted token key(hidden field) with the token key stored previously on request/session. If match, it will return true.   public class SubmitAction extends Action  {      public ActionForward execute(ActionMapping mapping ,ActionForm form ,HttpServletRequest request,HttpServletResponse response)      {          ActionForward forward=mapping.findForward("submitForm");          DupSubmitForm frm=(DupSubmitForm)form;                    if(isTokenValid(request))          {                            System.out.println("frm.getName()"+frm.getName());              resetToken(request);          }          else          {              System.out.println("frm.getName()"+frm.getName());              System.out.println("Duplicate Submission of the form");                        }          return forward;      }  }   Code you can get from:  http://www.techfaq360.com/viewTutorial.jsp?tutorialId=62 |

The information you are posting should be related to java and ORACLE technology. Not political. 

How to prevent mutli-click using struts tokens ?

Struts has 3 methods use for the token, saveToken(), isTokenValid() and resetToken().   
saveToken() - generate the token key and save to request/session attribute.   
isTokenValid() - validate submitted token key against the 1 store in request/session.   
resetToken() - reset the token key.   
  
How it works:   
1. Upon loading the form, invokes saveToken() on the action class to create and store the token key. Struts will store the generated key in request/session. If the token successfully created, when view source on the browser you will see something similar to the following, the token key is stored as a hidden field:   
<form action="myaction.do" method="post">   
<input type="hidden"   
name="<%= Constants.TOKEN\_KEY %>"   
value="<%= session.getAttribute(Action.TRANSACTION\_TOKEN\_KEY) %>" >   
  
2. Once the form submitted, invokes isTokenValid() on the action class, it will validate the submitted token key(hidden field) with the token key stored previously on request/session. If match, it will return true.   
  
.   
In Action Class.   
  
public final ActionForward perform(ActionMapping mapping,   
ActionForm form,   
HttpServletRequest request,   
HttpServletResponse response)   
throws IOException, ServletException   
{   
saveToken(request);   
if (!tokenIsValid(request))   
{   
//forward to error page saying "your transaction is already being processed"   
else   
{   
//process action   
//forward to jsp   
}   
  
// Reset token after transaction success.   
resetToken(request);   
  
}

How you will enable front-end validation based on the xml in validation.xml?

The <html:javascript> tag to allow front-end validation based on the xml in validation.xml. For example the code: <html:javascript formName=?logonForm? dynamicJavascript=?true? staticJavascript=?true? /> generates the client side java script for the form ?logonForm? as defined in the validation.xml file. The <html:javascript> when added in the jsp file generates the client site validation script.

What is new in ServletRequest interface ?

The following methods have been added to ServletRequest 2.4 version:   
public int getRemotePort()   
public java.lang.String getLocalName()   
public java.lang.String getLocalAddr()   
public int getLocalPort()

Q.Struts Action Chaining?

Chaining actions can be done by simply using the   
proper mapping in your forward entries in the struts-config.xml file.   
  
public class AAction extends Action   
{   
public ActionForward   
execute(ActionMapping mapping,   
ActionForm form,   
HttpServletRequest request,   
HttpServletResponse response) throws   
Exception   
{   
// Do something   
  
return mapping.findForward("success");   
}   
}

Q.How can I avoid validating a form before data is entered?

validate="false" ..   
The simplest way is to have two actions. The first one has the job of setting the form data, i.e. a blank registration screen. The second action in our writes the registration data to the database. Struts would take care of invoking the validation and returning the user to the correct screen if validation was not complete.   
The EditRegistration action in the struts example application illustrates this:   
< action path="/editRegistration">   
type="org.apache.struts.webapp.example.EditRegistrationAction"   
attribute="registrationForm"   
scope="request"   
validate="false">   
<forward name="success path="/registration.jsp"/>   
</action>   
When the /editRegistration action is invoked, a registrationForm is created and added to the request, but its validate method is not called. The default value of the validate attribute is true, so if you do not want an action to trigger form validation, you need to remember to add this attribute and set it to false

Q.Can I have an Action without a form?

Yes. If your Action does not need any data and it does   
not need to make any data available to the view or   
controller component that it forwards to, it doesn't need   
a form. A good example of an Action with no ActionForm is   
the LogoffAction in the struts example application:   
  
  
<action path="/logoff"   
type="org.apache.struts.webapp.example.LogoffAction">   
<forward name="success" path="/index.jsp"/>   
</action>   
  
This action needs no data other than the user's session, which   
it can get from the Request, and it doesn't need to prepare any   
view elements for display, so it does not need a form.   
  
  
However, you cannot use the <html:form> tag without   
an ActionForm. Even if you want to use the <html:form>   
tag with a simple Action that does not require input,   
the tag will expect you to use some type of ActionForm,   
even if it is an empty subclass without any properties.

Q.What is Struts Validator Framework?

Struts Framework provides the functionality to validate the form data. It can be use to validate the data on the users browser as well as on the server side. Struts Framework emits the java scripts and it can be used validate the form data on the client browser. Server side validation of form can be accomplished by sub classing your From Bean with DynaValidatorForm class.   
  
Client side validation : validation-rule.xml (java script code should be in this file and in JSP file   
<html:javascript formName="CreateForm"/>)   
  
Server side Validation :   
JSP page - should cointain   
< html:errors/ >   
Aactionform.validate() method   
or   
The Validator Framework uses two XML configuration files validator-rules.xml and validation.xml. The validator-rules.xml defines the standard validation routines, these are reusable and used in validation.xml. to define the form specific validations. The validation.xml defines the validations applied to a form bean.   
How you will display validation fail errors on jsp page? -   
The following tag displays all the errors: < html:errors/ >

Q.How you will make available any Message Resources Definitions file to the Struts Framework Environment?

Message Resources Definitions file are simple .properties files and these files contains the messages that can be used in the struts project. Message Resources Definitions files can be added to the struts-config.xml file through < message-resources / > tag. Example: < message-resources parameter= MessageResources / >

Q.What helpers in the form of JSP pages are provided in Struts framework?

--struts-html.tld   
--struts-bean.tld   
--struts-logic.tld

Q.How you will enable front-end client side validation based on the xml in validation.xml?

Validation Framework-Client Side   
Validation Framework provides the functionality to validate the form data. It can be use to validate the data on the client side. Errors will be displayed like java script.   
  
Struts has a class called ValidatorForm in org.apache.struts.validator package. This is a subclass of ActionForm and implements the validate() method. The validate() method invokes the Commons Validator, executes the rules using the two xml files (validator-rules.xml and validation.xml) and generates ActionErrors using the Message Resources defined in the struts-config.xml.   
validator-rules.xml :   
The validator-rules.xml file defines the Validator definitions available for a given application.   
The validator-rules.xml file acts as a template, defining all of the possible Validators that are available to an application.   
validation.xml File :   
The validation.xml file is where you couple the individual Validators defined in the validator-rules.xml to components within your application.   
  
  
Follow the below steps to setup Validation Framework in Struts (server side validation ).   
  
Step 1. Add validator plugin into struts-config.xml   
  
<plug-in className="org.apache.struts.validator.ValidatorPlugIn"> <set-property property="pathnames" value="/WEB-INF/validator-rules.xml,   
/WEB-INF/validation.xml"/>   
</plug-in>   
  
Step 3. In the JSP page (EmpForm.jsp)- Add onsubmit="return validateempForm(this);" and <html:javascript formName="empForm"/> for Client Side Java Script Validation.   
  
<%@ taglib uri="/WEB-INF/struts-tiles.tld" prefix="tiles" %>   
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>   
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>   
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>   
<html:form>   
<html:form action="/submitForm.do" method="post" onsubmit="return validateempForm(this);">   
<html:errors/> // this is for error message from resource bundle in JSP   
<html:text property="firstName" size="20" maxlength="50"/>   
<html:text property="lastName" size="20" maxlength="50"/>   
<html:submit >Save</html:submit>   
<!-- Begin Validator Javascript Function-->   
<html:javascript formName="empForm"/>   
<!-- End of Validator Javascript Function-->   
</html:form>   
  
  
Step 4. Add Message Resources location in struts-config.xml   
  
<!-- Message Resources --> <message-resources parameter="application\_resource"/>   
  
Step 5. In the the Resource Bundle. application\_resource.properties file //Here you can add localization.   
  
label.firstName=First Name   
label.lastName=Last Name   
errors.required={0} is required.   
  
  
Step 7. In the EmpForm   
  
package com.techfaq.form;   
public class EmpForm extends ValidatorForm {   
int empId;   
String firstName;   
String lastName;   
public String getFirstName() {   
return firstName;   
}   
public void setFirstName(String firstName) {   
this.firstName = firstName;   
}   
public String getLastName() {   
return lastName;   
}   
public void setLastName(String lastName) {   
this.lastName = lastName;   
}   
public int getEmpId() {   
return empId;   
}   
public void setEmpId(int empId) {   
this.empId = empId;   
}   
}   
  
Step 6. In the validation.xml - The validation.xml file is where you couple the individual Validators defined in the validator-rules.xml to components within your application   
  
<form-validation>   
<formset>   
<form name="empForm">   
<field property="firstName" depends="required">   
<arg0 key="label.firstName"/>   
</field>   
<field property="lastName" depends="required"> <arg0 key="label.lastName"/>   
</field>   
</form>   
</formset>   
</form-validation>   
  
  
Step 6. In the validator-rules.xml - The validator-rules.xml file defines the Validator definitions available for a given application.   
  
<form-validation>   
<global>   
<validator   
name="required"   
classname="org.apache.struts.util.StrutsValidator"   
method="validateRequired"   
methodparams="java.lang.Object,   
org.apache.commons.validator.ValidatorAction,   
org.apache.commons.validator.Field,   
org.apache.struts.action.ActionErrors,   
javax.servlet.http.HttpServletRequest"   
msg="errors.required"/>   
</global>   
</form-validation>   
  
  
Step 5. Add Action mapping and form entry into the stuts-confix.xml and validate="true" is for validation framework to validate the form input.   
  
<?xml version="1.0" encoding="UTF-8"?>   
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 1.1//EN"   
"http://jakarta.apache.org/struts/dtds/struts-config\_1\_1.dtd">   
<struts-config>   
<form-beans>   
<form-bean name="EmpForm" type="com.techfaq.form.EmpForm"> </form-bean>   
</form-beans>   
<action-mappings>   
<action path="/submitForm"   
type="com.techfaq.action.EmpAction"   
name="EmpForm"   
scope="request"   
validate="true" // This need for validation framework to validate the form input   
input="EmpForm.jsp">   
<forward name="success"   
path="success.jsp"/>   
<forward name="failure" path="failure.jsp" />   
</action>   
</action-mappings>   
</struts-config>   
  
  
Now in the browser type http://localhost:8080/testApp/EmpForm.jsp   
  
Don't Enter firstName and lastName in the text box and submit the "Save" BUTTON. the RequestProcessor checks for the validateattribute in the ActionMapping.   
If the validate is set to true, the RequestProcessor invokes the validate() method of the ValidatorForm instance.   
If Validate fail the RequestProcessor looks for the input attribute and return to JSP page mentioned in input tag.   
If Validate pass goto Action Class execute() method..   
If Validate fail , In the browser (EmpForm.jsp) you can see. Java Script pop up Message:   
  
  
First Name is required.   
Last Name is required.   
  
  
In the empForm firstName and lastName are the required filed. So in the above configuration you can see we add for both firstName and lastName. You can see depends="required" - "required" property is defind in validator-rules.xml. In the resource bundle : application\_resource.propertis file   
label.firstName=First Name   
label.lastName=Last Name   
#Error messages used by the Validator   
errors.required={0} is required.   
{0} will be filled by (First Name or Last Name) because validation.xml above configuration you have defind   
<arg0 key="label.lastName"/>. and <arg0 key="label.lastName"/>.

### **What is the difference between session scope and request scope when saving formbean ?**

when the scope is request,the values of formbean would be available for the current request.   
when the scope is session,the values of formbean would be available throughout the session.

Q.What design patterns are used in Struts

Struts is based on model 2 MVC (Model-View-Controller) architecture. Struts controller uses the command design pattern and the action classes use the adapter design pattern. The process() method of the RequestProcessor uses the template method design pattern. Struts also implement the following J2EE design patterns.   
  
Service to Worker   
Dispatcher View   
Composite View (Struts Tiles)   
Front Controller   
View Helper   
Synchronizer Token

Q.What is role of Action Class?

An Action Class performs a role of an adapter between the contents of an incoming HTTP request and the corresponding business logic that should be executed to process this request.   
  
In the execute() method of Action class the business logic is executed.   
  
  
public ActionForward execute(      ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response) throws Exception ;   
  
  
execute() method of Action class:   
  
Perform the processing required to deal with this request   
Update the server-side objects (Scope variables) that will be used to create the next page of the user interface   
Return an appropriate ActionForward object

Q.What is ActionMapping and is the Action Mapping specified?

Action mapping contains all the deployment information for a particular Action bean. This class is to determine where the results of the Action will be sent once its processing is complete.   
We can specify the action mapping in the configuration file called struts-config.xml. Struts framework creates ActionMapping object from <ActionMapping> configuration element of struts-config.xml file   
  
  
<action-mappings>   
<action path="/submit"    type="com.SubmitAction" name="submitForm" input="/submit.jsp" scope="request" validate="true"> <forward name="success" path="/success.jsp"/> <forward name="failure" path="/error.jsp"/> </action>   
</action-mappings>

What is ActionMapping?

ActionMapping object contains all the mapping information in struts\_config.xml.   
  
ActionServlet create the instance of ActionMapping and load all the struts\_config.xml data to ActionMapping object.   
  
Action mapping contains all the deployment information for a particular Action bean. This class is to determine where the results of the Action will be sent once its processing is complete.   
  
ActionMapping object contains all the mapping information in struts\_config.xml.   
  
For Example :   
  
<action-mappings>   
<action path="/submitForm"   
type="com.techfaq.action.EmpAction"   
name="EmpForm"   
scope="request"   
validate="true" // This need for validation framework to validate the form input   
input="EmpForm.jsp">   
<forward name="success"   
path="success.jsp"/>   
</action>   
  
  
What is role of ActionServlet?

The class org.apache.struts.action.ActionServlet is the called the ActionServlet. In the the Jakarta Struts Framework this class plays the role of controller. All the requests to the server goes through the controller. Controller is responsible for handling all the requests.   
  
  
  
Struts Flow start with ActionServlet then call to process() method of RequestProcessor.   
  
  
  
Step 1. Load ActionServlet using load-on-startup and do the following tasks.   
  
  
  
Any struts web application contain the ActionServlet configuration in web.xml file.   
  
On load-on-startup the servlet container Instantiate the ActionServlet .   
  
First Task by ActionServlet : The ActionServlet takes the Struts Config file name as an init-param.   
  
At startup, in the init() method, the ActionServlet reads the Struts Config file and load into memory.   
  
Second Task by ActionServlet : If the user types http://localhost:8080/app/submitForm.do in the browser URL bar, the URL will be intercepted and processed by the ActionServlet since the URL has a pattern \*.do, with a suffix of "do". Because servlet-mapping is   
  
<servlet-mapping>   
  
<servlet-name>action</servlet-name>   
  
<url-pattern>\*.do</url-pattern>   
  
</servlet-mapping>   
  
Third Task by ActionServlet : Then ActionServlet delegates the request handling to another class called RequestProcessor by invoking its process() method.   
  
  
  
Step 2. ActionServlet calls process() method of RequestProcessor

What are the core classes of the Struts Framework?

Struts follow Model View Controller Architecture. Components is divided based on MVC.   
  
Controller :   
  
The ActionServlet and the collaborating classes are RequestProcessor, ActionForm, Action, ActionMapping and ActionForward in in Controller category.   
  
ActionServlet :   
  
The central component of the Struts Controller is the ActionServlet. It is a concrete class and extends the javax.servlet.HttpServlet. It performs two important things.   
1. On startup, its reads the Struts Configuration file and loads it into memory in the init() method.   
2. In the doGet() and doPost() methods, it intercepts HTTP request and handles it appropriately.   
  
  
View :   
  
The View category contains utility classes ? variety of custom tags (html,bean,logic) and JSP.   
  
Model :   
  
Struts does not offer any components in the Model Category. Your Action class contains business logic may be model.

How you will display validation fail errors on jsp page?

Following tag displays all the errors:   
<html:errors/>

Give the Details of XML files used in Validator Framework?

The Validator Framework uses two XML configuration files validator-rules.xml and validation.xml. The validator-rules.xml defines the standard validation routines, these are reusable and used in validation.xml. to define the form specific validations. The validation.xml defines the validations applied to a form bean.   
  
Details :   
<http://www.techfaq360.com/tutorial/validation_framework.jsp>

### **Q.Struts Flow In Depth?**

Steps 1.   
ActionServlet   
The central component of the Struts Controller is the ActionServlet. It is   
a concrete class and extends the javax.servlet.HttpServlet. It performs   
two important things.   
On startup, its reads the Struts Configuration file and loads it into memory in   
the init() method.   
In the doGet() and doPost() methods, it intercepts HTTP request and   
handles it appropriately.   
  
In the web.xml   
  
<servlet>   
<servlet-name>action</servlet-name>   
<servlet-class>org.apache.struts.action.ActionServlet   
</servlet-class>   
<init-param>   
<param-name>config</param-name>   
<param-value>/WEB-INF/struts-config.xml</param-value>   
</init-param>   
<load-on-startup>1</load-on-startup>   
</servlet>   
  
<servlet-mapping>   
<servlet-name>action</servlet-name>   
<url-pattern>\*.do</url-pattern>   
</servlet-mapping>   
  
If the user types http://localhost:8080/App1/submitDetails.do in the   
browser URL bar. Server will call ActionServlet class because in the url-pattern the mapping is   
<url-pattern>\*.do</url-pattern>. Any \*.do will call ActionServlet class.   
ActionServlet calls the process() method of RequestProcessor class   
  
Step 2.   
ActionServlet calls the process() method of RequestProcessor class.   
The RequestProcessor first retrieves appropriate XML block for   
the URL from struts-config.xml. This XML block is referred to as   
ActionMapping in Struts terminology. In fact there is a class called   
ActionMapping in org.apache.struts.action package.   
ActionMapping is the class that does what its name says ? it holds the mapping   
between a URL and Action.   
  
A sample ActionMapping from struts-config.xml   
  
<action path="/submitDetails"   
type="mybank.example.CustomerAction"   
name="CustomerForm"   
scope="request"   
validate="true"   
input="CustomerDetailForm.jsp">   
<forward name="success"   
path="ThankYou.jsp"   
redirect=?true?/>   
<forward name="failure" path="error.jsp" />   
</action>   
  
Step 3.   
  
The RequestProcessor looks up the configuration file for the URL   
pattern /submitDetails. and finds the XML block (ActionMapping) shown above.   
The type attribute tells Struts which Action class has to be instantiated.   
  
Step 4.   
The RequestProcessor instantiates the CustomerForm and puts   
it in appropriate scope ? either session or request. The RequestProcessor   
determines the appropriate scope by looking at the scope attribute in the same   
ActionMapping.   
  
Step 5.   
Next, RequestProcessor iterates through the HTTP request parameters   
and populates the CustomerForm properties of the same name as the HTTP   
request parameters using Java Introspection.   
  
Step 6.   
Next, the RequestProcessor checks for the validate attribute in the   
ActionMapping. If the validate is set to true, the RequestProcessor invokes   
the validate() method on the CustomerForm instance. This is the method   
where you can put all the html form data validations. If any error then   
RequestProcessor checks for the input attribute in the ActionMapping   
and forward to page mentioned in the input tag.   
If no error in validate() method then continue.   
  
Step 7.   
  
The RequestProcessor instantiates the Action class specified in the   
ActionMapping (CustomerAction) and invokes the execute() method on   
the CustomerAction instance. The signature of the execute method is as   
follows.   
public ActionForward execute(ActionMapping mapping,   
ActionForm form,   
HttpServletRequest request,   
HttpServletResponse response) throws Exception   
  
The execute() method returns ActionForward.   
  
ActionForward forward = mapping.findForward(?success?);   
return forward. will forward to ThankYou.jsp.   
ActionForward forward = mapping.findForward(failure);   
return forward. will forward to error.jsp.

### **Q.What is Struts Validator Framework?**

Struts Framework provides the functionality to validate the form data. It can be use to validate the data on the users browser as well as on the server side. Struts Framework emits the java scripts and it can be used validate the form data on the client browser. Server side validation of form can be accomplished by sub classing your From Bean with DynaValidatorForm class.   
  
Client side validation : validation-rule.xml (java script code should be in this file and in JSP file   
<html:javascript formName="CreateForm"/>)   
  
Server side Validation :   
JSP page - should cointain < html:errors/ >   
Aactionform.validate() method or   
  
The Validator Framework uses two XML configuration files validator-rules.xml and validation.xml. The validator-rules.xml defines the standard validation routines, these are reusable and used in validation.xml. to define the form specific validations. The validation.xml defines the validations applied to a form bean. How you will display validation fail errors on jsp page? - The following tag displays all the errors: < html:errors/ >

nelluri642: hi

nelluri642: <action-mappings>

<action path="/temp"

type="org.apache.struts.actions.SwitchAction"/>

</action-mappings>

Referring to the above, which one of the following examples correctly switches to tempModule?

nelluri642: a.) /temp.do?prefix=/tempt&page=/index.do

b.) Zhtml:link module="/tempModule" path="/index.do"/>

c.) /tempModule.do?prefix=/tempModule&page=/index.do

d.) /tempModule.do?prefix=/temp&page=/index.do

e.) Zhtml:link module="/temp" path="/index.do"/>

nelluri642: Which one of the following statements about an ActionServlet plugin is true?

nelluri642: a.) It can be used as a last resort to extend ActionServlet functionality.

b.) It can be used instead of RequestProcessor configuration to extend the ActionServlet.

c.) It is configured in web.xml using ActionServlet initialization parameter.

d.) It can only implement the init() method to perform correctly.

e.) It can be used before extending the ActionServlet class.

venki alla: c

nelluri642: Which one of the following codes retrieves the same form information for every request made by all clients?

nelluri642: a.) MyForm form = (MyForm)pageContext.getAttribute("form\_name", PageContext.SERVLET\_SCOPE);

b.) MyForm form = (MyForm)pageContext.getAttribute("form\_name", PageContext.REQUEST\_SCOPE);

c.) MyForm form = (MyForm)pageContext.getAttribute("form\_name", PageContext.PAGE\_SCOPE);

d.) MyForm form = (MyForm)pageContext.getAttribute("form\_name", PageContext.SESSION\_SCOPE);

e.) MyForm form = (MyForm)pageContext.getAttribute("form\_name", PageContext.APPLICATION\_SCOPE);

nelluri642: The Struts message resources can be loaded either by 1) an init parameter in the servlet config in the web.xml or 2) by a <message-resources> tag in the struts-config.xml.

Referring to the above, when is using option 2 required in a Struts application?

nelluri642: a.) Option 2 is required when using Struts v1.2.x or greater.

b.) Option 2 is required when the application supports database access.

c.) Option 2 is required when the application uses the Spring framework.

d.) Option 2 is required when the application uses the Struts Tiles framework.

e.) Option 2 is required when the application supports more than one application resource file.

nelluri642: A developer has designed a banking Struts application that requires user authentication (userid and password). This application contains 30 Action classes. At a minimum, what does each of the Action classes implement?

nelluri642: a.) Secure Sockets Layer (SSL) code

b.) Code to verify balance and payment information

c.) Public key infrastructure code

d.) Code to check for session state

e.) Local variables shared by each server thread

venki alla: d

nelluri642: resource bundle:

Cannot find message resources under key org.apache.struts.action.MESSAGE

Which one of the following explains the error shown above?

nelluri642: a.) The ApplicationResources resides in the application classpath.

b.) The init parameter definition for the application property specifies the location of the file.

c.) The <message-resources> tag has an invalid parameter setting.

d.) The forward config is missing from the struts-config.xml.

e.) The struts-config.xml contains multiple message resource configurations.

venki alla: c

nelluri642: <logic:test1 message="true">

Zhtml:messages id="tst" message="true">

<div class="test">

<bean:write name="tst"/>

</div><br/>

</html:messages>

</logic:test1>

Referring to the above, assuming myAction contains an instance of the Action class, which one of the following clears messages from the session attribute for this tag?

nelluri642: a.) myAction.saveMessages(session, message);

b.) myAction.saveMessages(session, null);

c.) myAction.saveErrors(session, null);

d.) myAction.saveErrors(session, message);

e.) myAction.saveToken(request);

nelluri642: Referring to the above, assuming myAction contains an instance of the Action class to access this message tag, which one of the following saves error messages to the session attribute for this tag?

venki alla: d

nelluri642: <plug-in

className="org.apache.struts.validator.ValidatorPlugIn">

[PLACE CODE HERE]

</plug-in>

Referring to the above, which one of the following configures the ValidatorPlugIn with the rules contained in the files:

/WEB-INF/validator-rules1.xml

/WEB-INF/validator-rules2.xml

nelluri642: a.) <set-property property="pathnames"

value="/WEB-INF/validator-rules1.xml />

<set-property property="pathnames" value=

"/WEB-INF/validator-rules2.xml"/>

b.) <set-property property="pathnames"

value="/WEB-INF/validator-rules1.xml,/WEB-INF/validator-

rules2.xml"/>

c.) <set-property property="pathnames"

value="/WEB-INF/validator-rules1.xml />

<set-property property="pathnames" append="true"

value="/WEB-INF/validator-rules2.xml"/>

nelluri642: d.) <set-property property="pathnames"

value="/WEB-INF/validator-rules1.xml />

<set-property property="pathnames" concat="true"

value="/WEB-INF/validator-rules2.xml"/>

e.) <set-property property="pathnames"

value="/WEB-INF/validator-rules1.xml />

<set-property property="pathnames" merge="true"

value="/WEB-INF/validator-rules2.xml"/>

nelluri642: RedirectingActionForward = new

RedirectingActionForward("/home/test.jsp");

Referring to the above, which one of the following creates an equivalent instance of RedirectingActionForward class?

nelluri642: a.) ActionForward af = new ActionForward

("/home/test.jsp");

ActionForward af2 = af1.clone();

b.) ActionForward af = new ActionForward(null,

"/home/test.jsp", false);

c.) ActionConfig af = new ActionConfig(null,

"/home/test.jsp", true);

d.) ActionForward af = new ActionForward

("/home/test.jsp", true);

e.) ActionForward af = new ActionForward(/home/test.jsp,

true);

nelluri642: Which one of the following do you NOT need to do in order to construct a typical Struts application configuration that accepts user input from a Web browser?

nelluri642: a.) Modify struts-config.xml to map requests to an action class.

b.) Create a DispatchAction to handle several related tasks.

c.) Define a bean that extends ActionForm.

d.) Create an Action class to handle the request.

e.) Create an HTML form that invokes the action mapping in struts-config.xml.

venki alla: c

venki alla: questions pastem cheyi

nelluri642: Which one of the following advantages do the Struts <nested> tags have over the Zhtml> tags?

nelluri642: a.) The <nested> tags offer more stability.

b.) The <nested> tags are Struts v1.0 backwards compatible.

c.) The <nested> tags offer better coding management with objects that contain many properties.

d.) The <nested> tags have better page rendering performance.

e.) The <nested> tags offer greater integration with the <logic> tags.

venki alla: c

nelluri642: <action-mappings>

<action path="/test"

type="org.apache.struts.actions.SwitchAction"/>

</action-mappings>

Referring to the above, which one of the following URL examples correctly switches to testModule?

nelluri642: a.) /test.do?prefix=/test&page=/index.do

b.) Zhtml:link module="/moduleB" path="/index.do"/>

c.) /test.do?prefix=/testModule&page=/index.do

d.) /testModule.do?prefix=/test&page=/index.do

e.) Zhtml:link module="/test" path="/index.do"/>

venki alla: e

venki alla: repeat

nelluri642: Design Requirement 3.a - Widget - add a custom property called "widget" to the mapping properties

Referring to the above, which one of the following techniques does the developer perform to accomplish the task?

nelluri642: a.) The developer retrieves the source code for org.apache.struts.action.ActionMapping and adds the custom property.

b.) The developer uses the ActionMappingBase class and calls the addProperty() method to add the property.

c.) The developer subclasses the ActionMapping class and adds the appropriate getter and setter methods for the property.

d.) The developer upgrades Struts to version 1.2.x or greater since the widget property is standard in this release.

e.) The developer modifies the ActionMappingBase class and adds the custom property.

venki alla: d

nelluri642: <struts-config>

<action-mappings>

<action path="/login" type="test.LoginAction" >

<forward name="success" path="/homepage.jsp"/>

<forward name="failure" path="/login.jsp"/>

</action>

<action path="/logout" type="test.LogoutAction">

<forward name="success" path="/login.jsp"/>

</action>

<action path="/test" forward="/test.jsp"/>

</action-mappings>

</struts-config>

Referring to the above, which one of the following allows the LoginAction to be executed for all undefined path mappings?

nelluri642: a.) <action path="/login" type="test.LoginAction" scope="request">

b.) <action path="/login" type="test.LoginAction" unknown="true">

c.) <action path="/login" type="test.LoginAction" configured="false">

d.) <action path="/login\*" type="test.LoginAction" unknown="true">

e.) <action path="/login" type="test.LoginAction" configured="true">

venki alla: a

nelluri642: <beanefine id="copy" name="test"/>

Referring to the above, which one of the following changes the scope of the copy bean from request to session?

nelluri642: a.) <beanefine id="copy" name="test"

fromScope="request" toScope="session"/>

b.) <beanefine id="copy" name="test"

scope="request" toScope="session"/>

c.) <bean:copy id="copy" name="test"

fromScope="request" toScope="session"/>

d.) <beanefine id="copy" name="test" copy="true"

scope="request" toScope="session"/>

e.) <beanefine id="copy" name="test" copy="clone"

scope="request" toScope="session"/>

nelluri642: Which one of the following is an attribute of action-mapping in the struts-config.xml?

nelluri642: a.) actionClass

b.) formClass

c.) actionFormBean

d.) formAttribute

e.) unknown

venki alla: e

nelluri642: Technique 1:

return (new ActionForward("/test.jsp"));

Technique 2:

<action path="/login" type="com.test.struts.LoginAction">

<forward name="success" path="/test.jsp"/>

<forward name="failure" path="/login.jsp"/>

</action>

Referring to the above, which one of the following is an advantage of using Technique 2 over Technique 1?

nelluri642: a.) A change to the test.jsp file requires a recompile for Technique 1.

b.) A change to the test.jsp file requires an application restart for Technique 1.

c.) A change to the test.jsp file name requires a recompile for Technique 1.

d.) A change to the test.jsp file name requires an application restart for Technique 1.

e.) A change to the test.jsp file name requires a recompile for Technique 2.

venki alla: d

nelluri642: The definition of the Struts <controller> was moved from servlet initialization parameters to the struts-config.xml file.

Based on the scenario above, for which one of the following reasons was the <controller> element in struts-config moved to struts-config.xml?

nelluri642: a.) To allow specification of the forwardPattern for the application.

b.) To allow different module configurations for the application.

c.) To allow faster servlet startup for the application.

d.) To allow faster servlet shutdown for the application.

e.) To allow different inputForward configurations for the application.

venki alla: c

nelluri642: This session is complete.

Please contact your session administrator if you have further questions.

nelluri642: This session is complete.

Please contact your session administrator if you have further questions.