Struts 2 Tutorial

Apache **Struts 2** is an open-source web application framework for developing Java EE web applications. It uses and extends the Java Servlet API to encourage developers to adopt a model–view–controller (MVC) architecture

Apache Struts 2 is an elegant, extensible framework for creating enterprise-ready Java web applications. The framework is designed to streamline the full development cycle, from building, to deploying, to maintaining applications over time. Apache Struts 2 was originally known as WebWork 2

# Struts 2 History

Originally developed by the programmer and author Craig R. McClanahan, this was later taken over by the Apache Software Foundation in 2002. Struts have provided an excellent framework for developing application easily by organizing JSP and Servlet based on HTML formats and Java code. Strut1 with all standard Java technologies and packages of Jakarta assists to create an extensible development environment. However, with the growing demand of web application, **Strut 1** does not stand firm and needs to be changed with demand. This leads to the creation of Strut2, which is more developer friendly with features like Ajax, rapid development and extensibility.

Struts is a well-organized framework based on **MVC**architecture. With the development of new and lightweight MVC based framworks like Spring, Stripes and Tapestry, it becomes necessary to modify the Struts framework. So, the team of Apache Struts and another J2EE framework, **WebWork** of *OpenSymphony* joined hand together to develop an advanced framework with all possible developing features that will make it developer and user friendly

# Struts 1.x vs Struts 2.x Main Differences

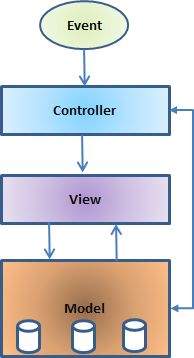
* In struts 1.x front controller is ActionServlet
  + In 2.x front controller is FilterDispatcher
* In struts 1.x we have RequestProcessor class
  + In 2.x we have Interceptors instead RequestProcessor
* In struts 1.x we have multiple tag libraries like, html, logic, bean..etc
  + In 2.x we do not have multiple libraries, instead we have single library which includes all tags
* In struts 1.x the configuration file name can be [any name].xml and we used to place in web-inf folder
  + In 2.x the configuration file must be struts.xml only and this must be in classes folder
* In struts 1.x we have form beans and Action classes separately
  + In 2.x form bean, Action classes are combined given as Action class only, of course we can take separately if we want ;)
* In struts 1.x properties file must be configured in struts-config.xml
  + But in 2.x we need to configure our resource bundle(s) in struts.properties file
* In struts 1.x we have programmatic and declarative validations only
  + In 2.x we have annotations support too along with programmatic and declarative validations

## Functional Differences

* In struts 1.x declarative validations are done by using validation frame work
  + In 2.x, declarative validations are done by using xwork2 frame work by webwork the reason being, its support valuations through Annotations
* In struts 1.x an Action class is a singleton class, so Action class object is not a thread safe, as a programmer we need to make it as thread safe by applying synchronization
  + In 2.x an Action class object will be created for each request, so it is by default thread safe, so no need to take care about safety issues here
* In struts 1.x we have only **jsp** as a view technology
  + In 2.x we have support of multiple view technologies like velocity, Freemarker, jasper reports, jsp bla bla
* In struts 1.x Action class is having servlet dependency, because in execute() method accepts req, res parameter right ! so.
  + In 2.x Action class doesn’t have any servlet dependency, because its execute() method doesn’t accepts any parameters, however we can access all servlet objects with dependency injection

# Basic MVC Architecture

MVC is popular as it isolates the application logic from the user interface layer and supports separation of concerns. Here the Controller receives all requests for the application and then prepares Model needed by the View. The View then uses the data/Model prepared by the Controller to generate a final presentable response. The MVC abstraction can be graphically represented as follows:



## Struts 2 framework features

Here are top ten features of Struts 2 that may force you to consider Struts2:

* **POJO forms and POJO actions** - Struts2 has done away with the Action Forms that were an integral part of the Struts framework. With Struts2, you can use any POJO to receive the form input. Similarly, you can now see any POJO as an Action class.
* **Tag support** - Struts2 has improved the form tags and the new tags allow the developers to write less code.
* **AJAX support** - Struts2 has recognised the take over by Web2.0 technologies, and has integrated AJAX support into the product by creating AJAX tags, that function very similar to the standard Struts2 tags.
* **Easy Integration** - Integration with other frameworks like Spring, Tiles and SiteMesh is now easier with a variety of integration available with Struts2.
* **Template Support** - Support for generating views using templates.
* **Plugin Support** - The core Struts2 behaviour can be enhanced and augmented by the use of plugins. A number of plugins are available for Struts2.
* **Profiling** - Struts2 offers integrated profiling to debug and profile the application. In addition to this, Struts also offers integrated debugging with the help of built in debugging tools.
* **Easy to modify tags** - Tag markups in Struts2 can be tweaked using Freemarker templates. This does not require JSP or java knowledge. Basic HTML, XML and CSS knowledge is enough to modify the tags.
* **Promote less configuration** - Struts2 promotes less configuration with the help of using default values for various settings. You don't have to configure something unless it deviates from the default settings set by Struts2.
* **View Technologies:** - Struts2 has a great support for multiple view options (JSP, Freemarker, Velocity and XSLT)

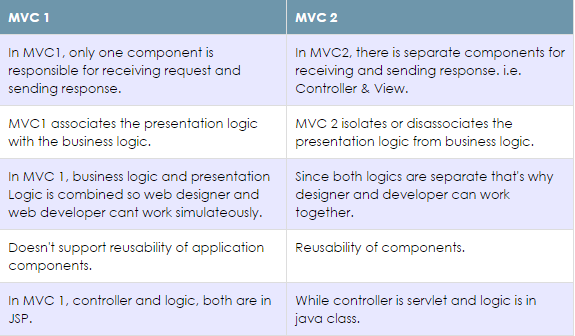
## Struts 2 disadvantages

Though Struts 2 comes with a list of great features but it is important to mention few negative points about Struts 2 and would need lots of improvements:

* **Bigger learning curve** - To use MVC with Struts, you have to be comfortable with the standard JSP, Servlet APIs and a large & elaborate framework.
* **Poor documentation** - Compared to the standard servlet and JSP APIs, Struts has fewer online resources, and many first-time users find the online Apache documentation confusing and poorly organized.
* **Less transparent** - With Struts applications, there is a lot more going on behind the scenes than with normal Java-based Web applications which makes it difficult to understand the framework.

Final note, A good framework should provide generic behavior that many different types of applications can make use of it. Struts 2 is one of the best web framework and being highly used for the development of Rich Internet Applications (RIA).

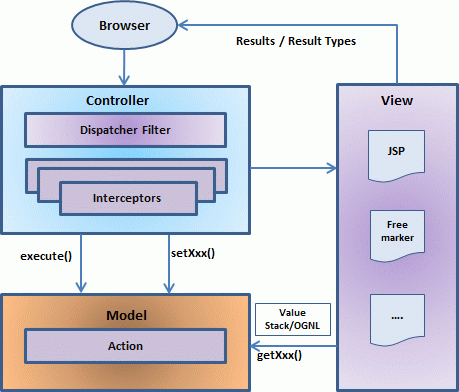
# MVC1 vs MVC2



# Struts 2 - Architecture

Struts2 is a MVC2 framework. The Model-View-Controller pattern in Struts2 is realized with following five core components:

* **Actions**
* **Interceptors**
* **Value Stack / OGNL**
* **Results / Result types**
* **View technologies**



The above diagram depicts the **M**odel, **V**iew and **C**ontroller to the Struts2 high level architecture. The controller is implemented with a Struts2 dispatch servlet filter as well as interceptors, the model is implemented with actions, and the view as a combination of result types and results. The value stack and OGNL provide common thread, linking and enabling integration between the other components.

Apart from the above components, there will be a lot of information that relates to configuration. Configuration for the web application, as well as configuration for actions, interceptors, results, etc.

## Request life cycle

Based on the above digram, one can explain the user's request life cycle in Struts 2 as follows:

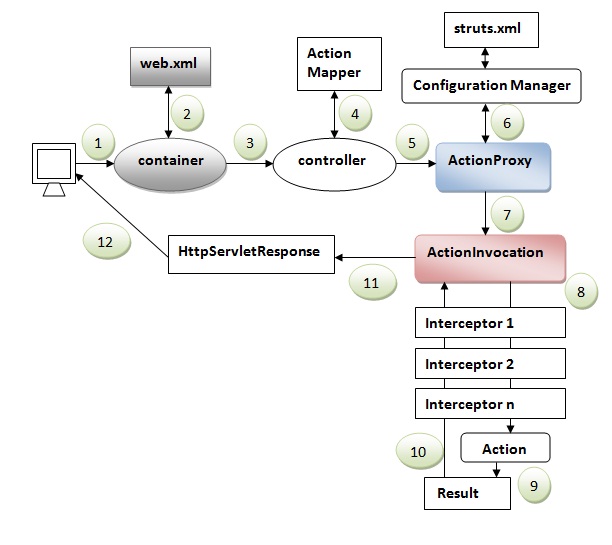
* User sends a request to the server for requesting for some resource (i.e pages).
* The FilterDispatcher looks at the request and then determines the appropriate Action.
* Configured interceptors functionalities applies such as validation, file upload etc.
* Selected action is executed to perform the requested operation.
* Again, configured interceptors are applied to do any post-processing if required.
* Finally the result is prepared by the view and returns the result to the user.

# Struts2 Basic Flow

# struts 2 basic flow

1. User sends a request for the action
2. Controller invokes the ActionInvocation
3. ActionInvocation invokes each interceptors and action
4. A result is generated
5. The result is sent back to the ActionInvocation
6. A HttpServletResponse is generated
7. Response is sent to the user

# Struts2 Standard Flow(Architecture)



1. User sends a request for the action
2. Container maps the request in the web.xml file and gets the class name of controller.
3. Container invokes the controller (StrutsPrepareAndExecuteFilter or FilterDispatcher). Since struts2.1, it is StrutsPrepareAndExecuteFilter. Before 2.1 it was FilterDispatcher.
4. Controller gets the information for the action from the ActionMapper
5. Controller invokes the ActionProxy
6. ActionProxy gets the information of action and interceptor stack from the configuration manager which gets the information from the struts.xml file.
7. ActionProxy forwards the request to the ActionInvocation
8. ActionInvocation invokes each interceptors and action
9. A result is generated
10. The result is sent back to the ActionInvocation
11. A HttpServletResponse is generated
12. Response is sent to the user

# Struts 2 - Configuration Files

* **web.xml**
* **struts.xml**
* **struts-config.xml**
* **struts.properties**

# Struts 2 - Interceptors

Interceptors are conceptually the same as servlet filters or the JDKs Proxy class

* Providing preprocessing logic before the action is called.
* Providing postprocessing logic after the action is called.
* Catching exceptions so that alternate processing can be performed.

|  |  |
| --- | --- |
| 1 | **Alias**  Allows parameters to have different name aliases across requests. |
| 2 | **Checkbox**  Assists in managing check boxes by adding a parameter value of false for check boxes that are not checked. |
| 3 | **conversionError**  Places error information from converting strings to parameter types into the action's field errors. |
| 4 | **createSession**  Automatically creates an HTTP session if one does not already exist. |
| 5 | **Debugging**  Provides several different debugging screens to the developer. |
| 6 | **execAndWait**  Sends the user to an intermediary waiting page while the action executes in the background. |
| 7 | **Exception**  Maps exceptions that are thrown from an action to a result, allowing automatic exception handling via redirection. |
| 8 | **fileUpload**  Facilitates easy file uploading. |
| 9 | **i18n**  Keeps track of the selected locale during a user's session. |
| 10 | **Logger**  Provides simple logging by outputting the name of the action being executed. |
| 11 | **Params**  Sets the request parameters on the action. |
| 12 | **Prepare**  This is typically used to do pre-processing work, such as setup database connections. |
| 13 | **Profile**  Allows simple profiling information to be logged for actions. |
| 14 | **Scope**  Stores and retrieves the action's state in the session or application scope. |
| 15 | **ServletConfig**  Provides the action with access to various servlet-based information. |
| 16 | **Timer**  Provides simple profiling information in the form of how long the action takes to execute. |
| 17 | **Token**  Checks the action for a valid token to prevent duplicate formsubmission. |
| 18 | **Validation**  Provides validation support for actions |

struts-default.xml file

<interceptor-stack name="basicStack">

<interceptor-ref name="exception"/>

<interceptor-ref name="servlet-config"/>

<interceptor-ref name="prepare"/>

<interceptor-ref name="checkbox"/>

<interceptor-ref name="params"/>

<interceptor-ref name="conversionError"/>

</interceptor-stack>

Use:

<action name="hello" class="com.tutorialspoint.struts2.MyAction">

<interceptor-ref name="basicStack"/>

<result>view.jsp</result>

</action

# Actions in Struts2

The action classes are core concepts in Struts2; it is Model class itself and contains the business logic for processing on the data. Stuts2 actions are not singleton, means instance of action is created on each request.

**There are three way to create actions in Struts2 :**

* **by extending ActionSupport class :**
* **by implementing Action interface :**
* **as POJO class :**
  + **com.opensymphony.xwork2.ActionSupport class**

It implements Action, Validateable, ValidationAware, TextProvider, LocaleProvider, Serializable interfaces

* + **com.opensymphony.xwork2.Action Interface**

|  |  |
| --- | --- |
| **Filed Name** | **Description** |
| **static String ERROR="error"** | **The action execution was a failure** |
| **static String INPUT="input"** | **The action execution require more input in order to succeed** |
| **static String LOGIN="login"** | **The action could not execute, since the user most was not logged in** |
| **static String NONE="none"** | **The action execution was successful but do not show a view** |
| **static String SUCCESS="success"** | **The action execution was successful** |

* + **com.opensymphony.xwork2.LocalProvider interface**

This interface provides a single method Locale getLocale() to get localized message with the help of TextProvider interface.

* + **com.opensymphony.xwork2.TextProvider interface**

provides methods to read localized message from resource bundle

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| **String getText(String key)** | **Gets a message based on a message key, or null if no message is found** |
| **String getText(String key, String defaultValue)** | **Gets a message based on a key, or, if the message is not found, a supplied default value is returned** |
| **ResourceBundle getTexts()** | **Get the resource bundle associated with the implementing class (usually an action)** |
| **ResourceBundle getTexts(String bundleName)** | **Get the named bundle, such as "pkg1/techmyguru/actions"** |
| **boolean hasKey(String key)** | **Checks if a message key exists** |

* + **com.opensymphony.xwork2.Validateable**

This interface provides a single method void validate(). It executes before the business logic ex- execute() method from action class.

* + **com.opensymphony.xwork2.ValidationAware**

This interface provides to add and retrieve action level or field level errors and action level messages also. Below are common methods to be used in the application:

|  |  |
| --- | --- |
| **Methods** | **Description** |
| **void addActionError(String anErrorMessage)** | **Add an Action-level error message to this Action** |
| **void addActionMessage(String aMessage)** | **Add an Action-level message to this Action** |
| **void addFieldError(String fieldName, String errorMessage)** | **Add an error message for a given field** |
| **void setActionErrors(Collection<String> errorMessages)** | **Set the Collection of Action-level String error messages** |
| **void setActionMessages(Collection<String> messages)** | **Set the Collection of Action-level String messages (not errors)** |
| **void setFieldErrors(Map<String,List<String>> errorMap)** | **Set the field error map of fieldname (String) to Collection of String error messages** |

## Struts 2 UI Tags

* TextBox example
  + Struts 2 <s:textfield>
* Password example
  + Struts 2 <s:password>
* Hidden value example
  + Struts 2 <s:hidden>
* Textarea example
  + Struts 2 <s:textarea>
* head example
  + Struts 2 <s:head>, render a HTML head component.