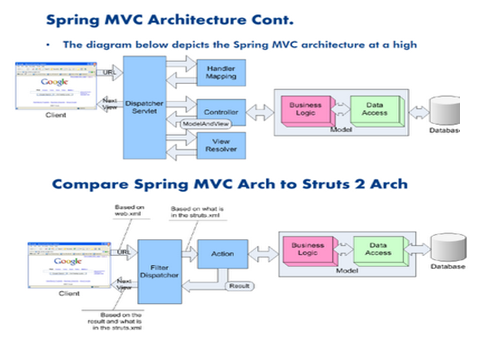
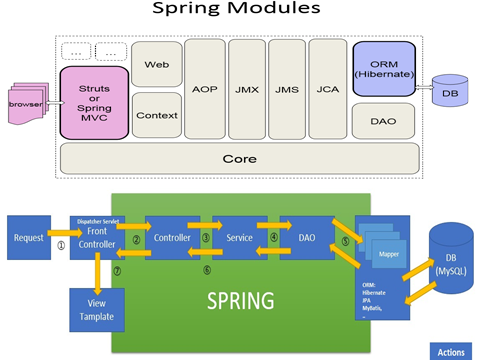
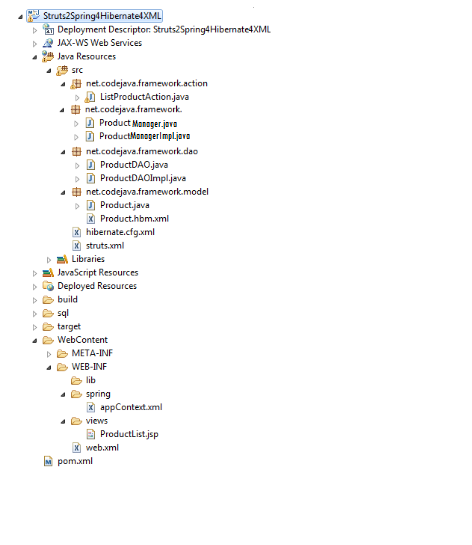
**HRS Kick Start and Struts 2 integrated with spring Session:**

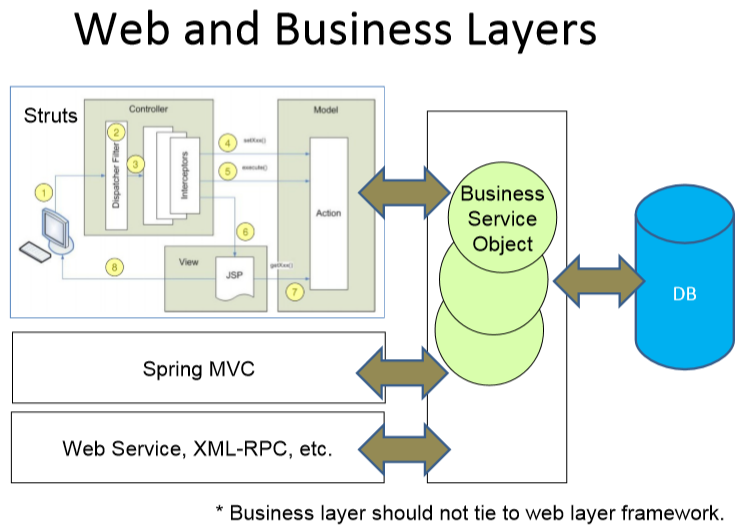
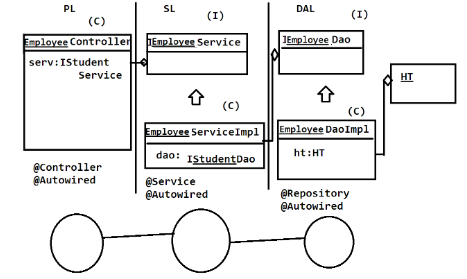


Spring is an enterprise application framework, have modular approach and it is invasive framework

Struts is a web application framework, a good frame work for front end based applications,,

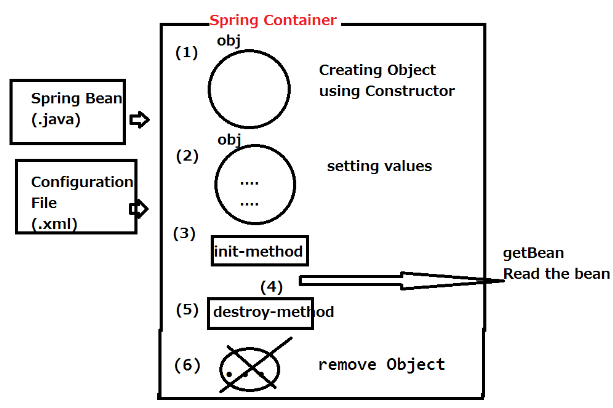
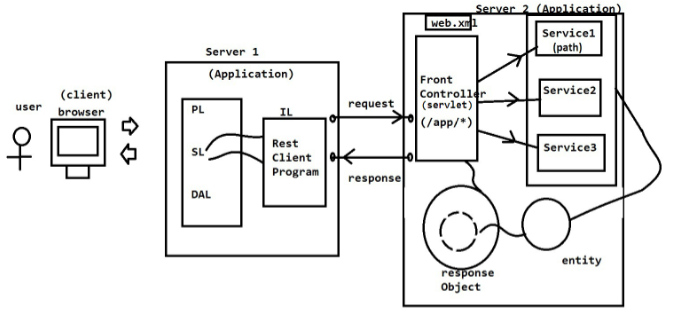
and Hibernate is an ORM framework.

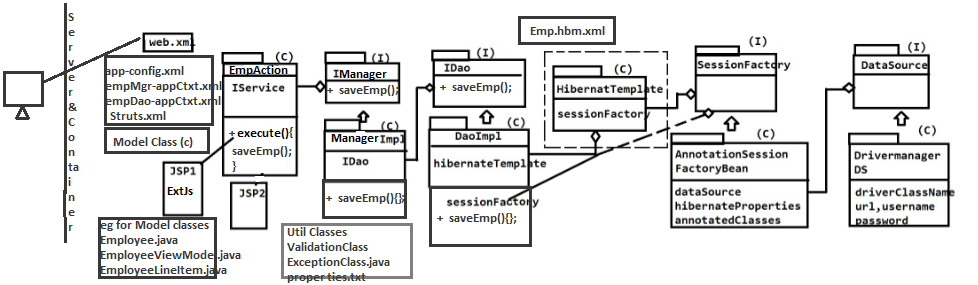
****

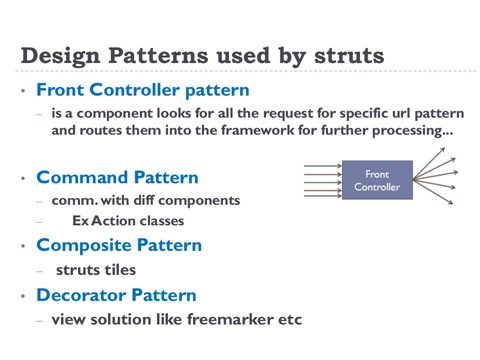
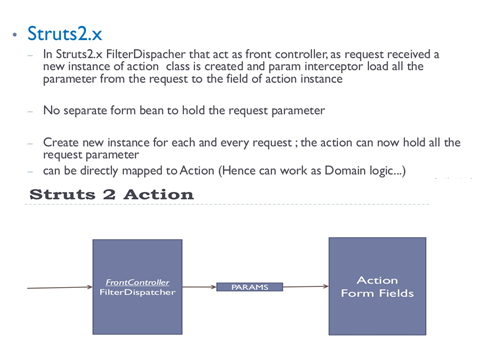


**why do we need integrate struts with spring**:

* spring has its own container capability
* easy dependency management (so easy to work with others layers)
* supports loose coupling
* testing easy
* supports modular approach and invasive framework
* reduces boiler plate code, RAD using Spring projects and Templates like REST, Spring boot, etc



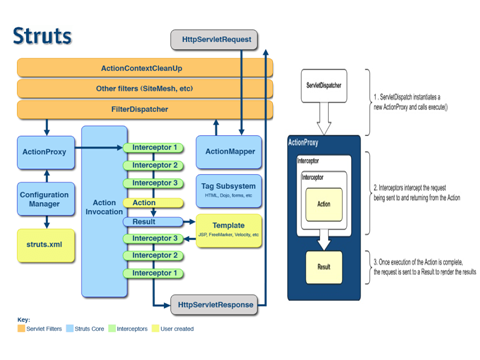
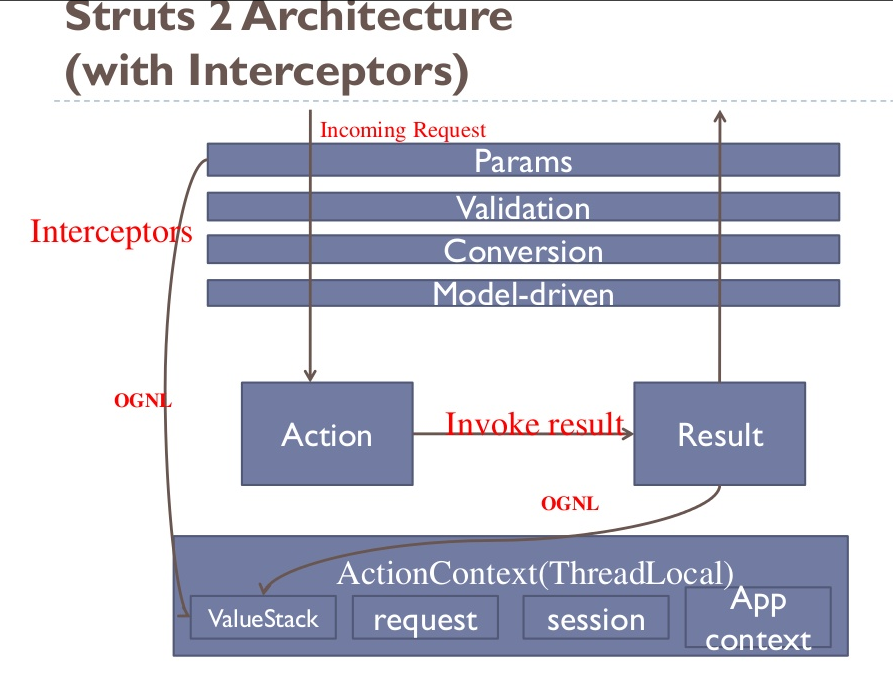
****

**Some of Struts 2 features are**:

* POJO (plain old Java Objects)-based actions
* Easy Integration & plugin support for REST, AJAX, Hibernate, Spring, etc
* Having inbuilt support for I18N
* Extensive Validation Support
* Various Tag support & Easy to Modify Tags
* AJAX Support
* Theme and Template support i.e support of various view-layer technologies
* ease of profiling and debugging
* -Various Result Types using:

**core components**

* Interceptors
* Value Stack

[[1]](#footnote-1)  ****In Struts2.x FilterDispacher that act as front controller, as request received a new instance of action class is created and param interceptor load all the parameter from the request to the field of action instance

– Create new instance for each and every request ;

– No separate form bean to hold the request parameter

the action can now hold all the request parameter can be directly mapped to Action (Hence can work as Domain logic...)

Flow

1. User Sends HTTP request

2. FilterDispatcher determines appropriate action

3. Interceptors are applied

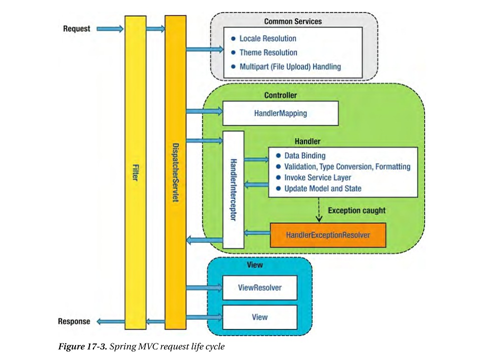
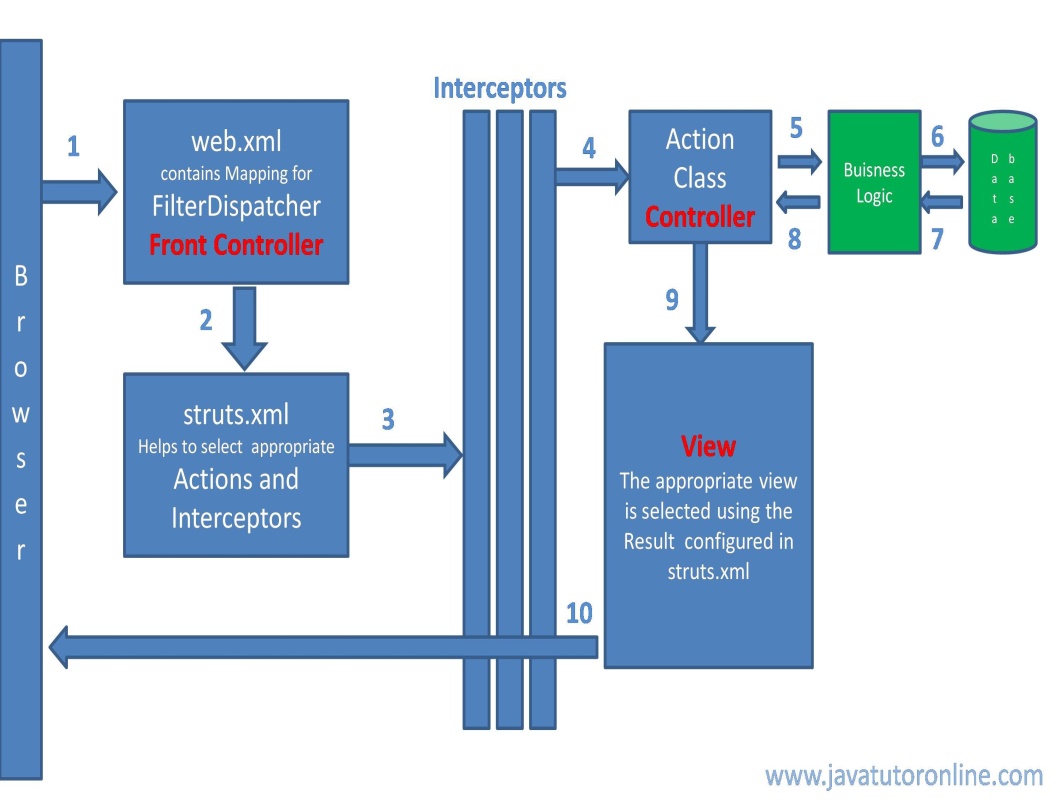
4. Action parameters are set (via params interceptor)

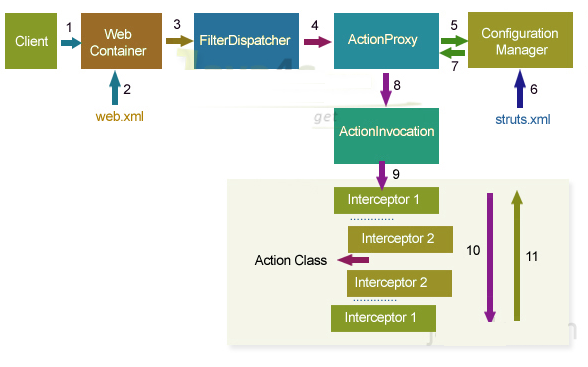
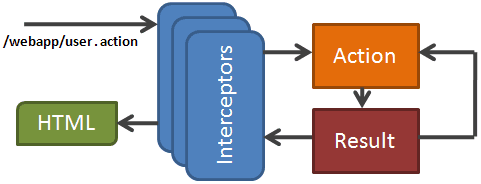
5. Action is being executed

6. The Result (view) renders the output

7. Optionaly: the view retrieves data from action

8. The result is displayed to the user

**** ****

**Working and flow of Struts 2:**

* The request is first sent from the browser to the server. Then the server & web container loads the web.xml if the request pattern matches then it is forwarded to the FilterDispatcher. In struts 2 the FilterDispatcher is the Front Controller.

----

* FilterDispatcher hand overs the request to ActionProxy, it is a proxy class which is responsible to apply before and after services to original business logic
* ActionProxy contacts ConfiguraionManager class, to know the suitable Action for the request and the needed services for the request
* ConfigurationManager class loads structs.xml and provides the required information back to ActionProxy
* ActionPorxy delegates the request along with its information to ActionInvocation
* ActionInvocation executes the interceptors added to an Action from 1 – N, after that it will call the business logic implemented from N – 1 in reverse order
* ActionInvocation receives finally result produced by an action aclass
* ActionProxy transfers the result back to FilterDispatcher
* FilterDispatcher selects an appropriate view, basing on the result
* Finally FilterDispatcher uses RequestDispatchers forwarding mechanism and forward a view as a response back to the client

---

* Based on the request url and it’s mapping in the struts.xml the appropriate action class to be executed is decided.
* Before the Action class is executed the request is passed through the interceptors.
* The action method of the action class (controller) is executed.
* The action class calls the business logic function.

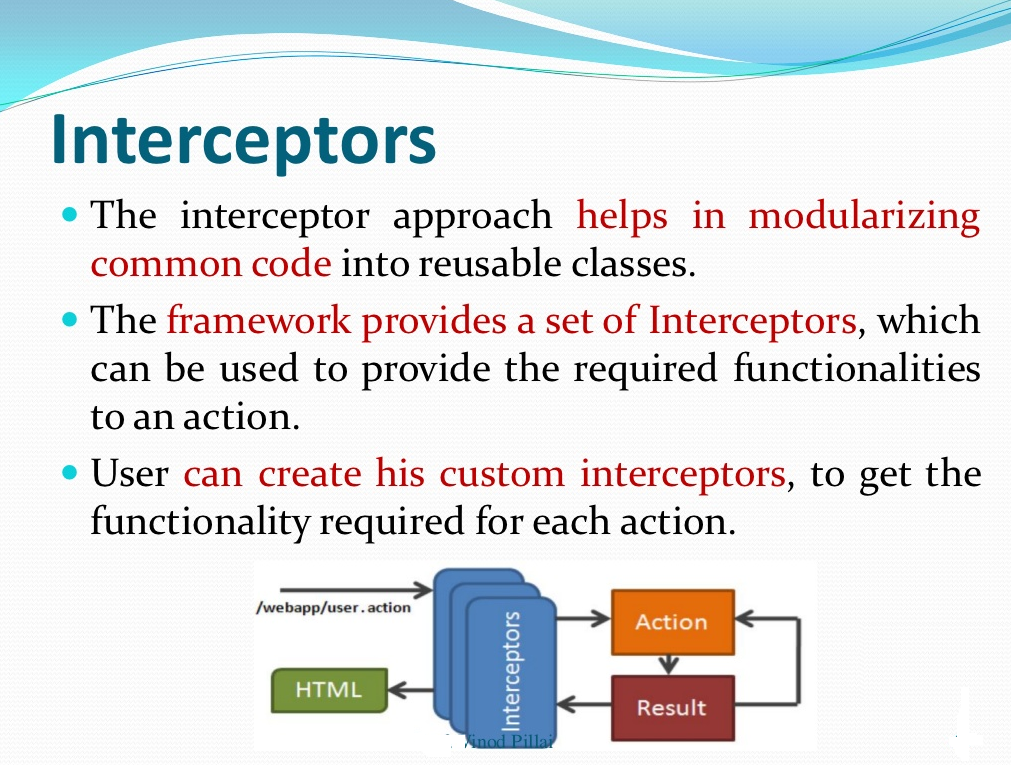
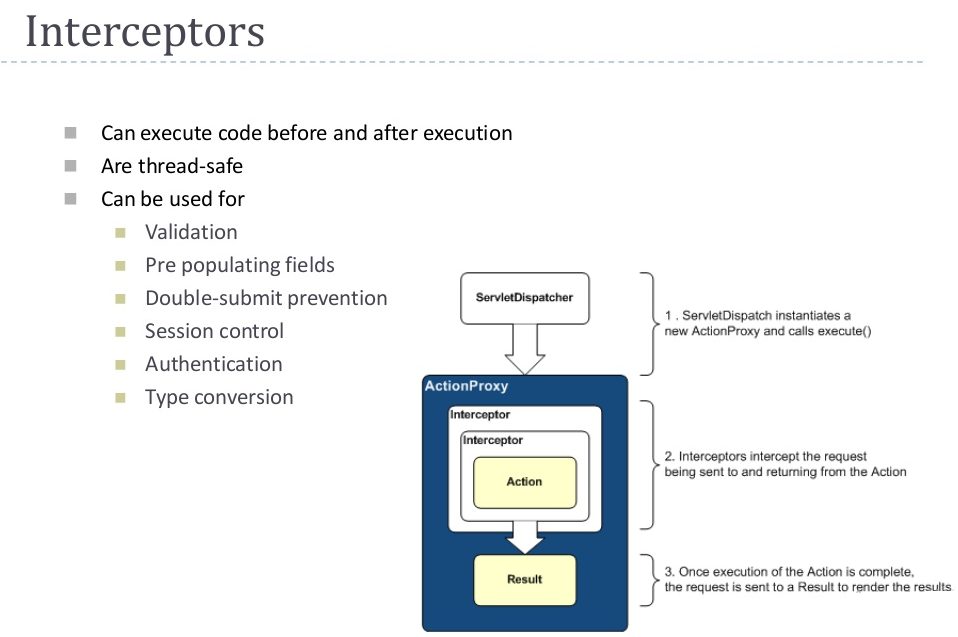
--

* The Business Logic class works with the database.
* Bushiness logic class gets the data from the database.
* Processed data from bushiness logic is sent back to the Action class or the controller.
* Depending on the result the Controller identifies the view to be rendered.

---

* Before the response/result is generated the interceptors are executed again.

 **Interceptors** are one of the most powerful features of Struts 2.

** **

**Xml/Annotation Based Configuration**

--------

Mapping Actions in struts.xml or on anyAction class

-------

<action>

<action name=“hello“ class="com.hrs.actions.HelloWorld">

<result name=“success“>/pages/HelloWorld.jsp</result>

<result name=“error“>/pages/logout.jsp</result>

</action>

Use action name in URL invocation

http://host:port/app/hello.action

--------

@Action(value = "login", results = {

@Result(name = "SUCCESS", location = "/welcome.jsp"),

@Result(name = "ERROR", location = "/error.jsp") })

@Namespaces(value={@Namespace("/login"),@Namespace("/")})

public class LoginAction {

-----------------------------------------

In Struts 2 any POJO can be an action class. "Actions and Results" it is convenient to extend the ActionSupport class

in Struts 2. On top of that, an action class can be used to service related actions.

**Custom Action Class:**

public class **HelloWorldAction** extend ActionSupport {

private String message="Hello World. Time is: ";

public String execute()

{

message += new Date();

return "success";

}

public String getMessage() {

return message;

}

}

**ActionSupport**

 ActionSupport class provides default implementaion for various services required by common actions classes...

class ActionSupport implements Validateable,ValidationAware, LocaleProvider, TextProvider,ValidationAware,Action,Serilizable {

}

Actions extending ActionSupport

- facilitates validation(field validation,expression validation,..)

- facilitates localization(I18n)

- facilitates Error feedback Messages

- facilitates ExceptionHandling

Action Interface

 Action interface define some useful constants, we can used these constant as return from action methods.

public interface Action {

public static final String SUCCESS = "success";

public static final String NONE = "none";

public static final String ERROR = "error";

public static final String INPUT = "input";

public static final String LOGIN = "login";

public String execute() throws Exception;

}

---

In Struts 2, an HTML form maps directly to a POJO.

By writing the validation logic in the action class programmatically validate user input in Struts 2

Struts 2 comes with a tag library that covers as like Jstl

Struts.xml

--

- Actions Mapping

- Results Mapping

- custom interceptors

- struts in built features extended class mapping

-include other xml files required

**struts.xml**

-----

<struts>

//action mappings

<package name="user/\*" namespace="/User" extends="struts-default">

<action name=“user“ class="com.hrs.actions.UserAction" method="getUser">

<param name="user.id">{1}</param>

<result type="success">results/login.jsp</result>

<result type="error">results/logout.sp</result>

</action>

//configurations

<constant name="struts.devMode" value="false" />

<constant name="struts.custom.i18n.resources" value="messages" />

..

// interceptors

<interceptors>

<interceptor name="params" class="…"/>

<interceptor-stack name="basicStack">

<interceptor-ref name="exception"/>

<interceptor-ref name="servletConfig"/>

<interceptor-ref name="prepare"/>

<interceptor-ref name="checkbox"/>

<interceptor-ref name="params"/>

<interceptor-ref name="conversionError"/>

</interceptor-stack>

</interceptors>

<default-interceptor-ref name="defaultStack"/>//include other struts.xml files

<include file="audit/struts-member.xml"></include>

<include file="audit/struts-audit.xml"></include>

</package>

</struts>

Above is the snippet from struts.xml file, notice that result element doesn’t start with a forward slash (/).

In this case, struts will look for result JSP page at {WEBAPP-ROOT}/{Namespace}/{result} path.

So our JSP file should be present at WEBAPP/User/results/ directory.

----

<!-- Normal Action mappings are defined here -->

<package name="default" namespace="" extends="struts-default">

In Struts 2 you use multiple configuration files too, however they must reside in or a

subdirectory of WEB-INF/classes

--

public interface Interceptor extends Serializable {

void destroy();

void init();

String intercept(ActionInvocation invocation)

throws Exception;

}

Custom interceptor:

public class MyInterceptor extends AbstractInterceptor {

public String intercept(ActionInvocation invocation)throws Exception {

/\* let us do some pre-processing \*/

String output = "Pre-Processing";

System.out.println(output);

/\* let us call action or next interceptor \*/

String result = invocation.invoke();

/\* let us do some post-processing \*/

output = "Post-Processing";

System.out.println(output);

return result;

}

}

**Themes**

- templates for customizing components markup

**Extensible via plugins**

– Other frameworks integration

– Embedding other application modules

- Jasper Reports

- JSON

- JFreeChart

**Supports Spring Integration**

- Dependency Injection interceptor

- Full Spring lifecycle management

**Tag Library:**

- Form Component tags

- Ajax UI component tags

- Control tags: iterator, if/else...

- Data tags: manipulate the ValueStack

**Internationalization(I18n)**

 For implement i18n in we need

1. resource bundles 2. interceptors and 3. tag libraries

Reasonable defaults: struts-defaults

-Result types

-Interceptor stacks

**Result types:**

- JSP : dispatcher

- Stream : PDF, MS Word etc.

- Velocity, FreeMarker,XSLT

The view part is highly configurable and it supports different result-types such as Velocity, FreeMarker, JSP

**Datatype Convertors ,Datatype conversion framework**

**Struts 2 Type Conversion**

 Struts2 provide automatically type conversion for basic data types such as ....

 Integer, Float, Double, Decimal  Date and Datetime  Arrays and Collections

 Enumerations  Boolean  BigDecimal

 What if we have user define object?

 In that cases Struts 2 Type Conversion is very handy.....

**Interceptors**

- will control actions pre and post processing

– can change the user’s workflow, modify the result and inject objects into the action

# Result Types – allows post-processing, and additional

result-based processing or rendering of information

returned by the action

# Plug-in packages – new interceptors, result types, results

and actions can be packaged together into a plug-in that

can be re-used across many projects

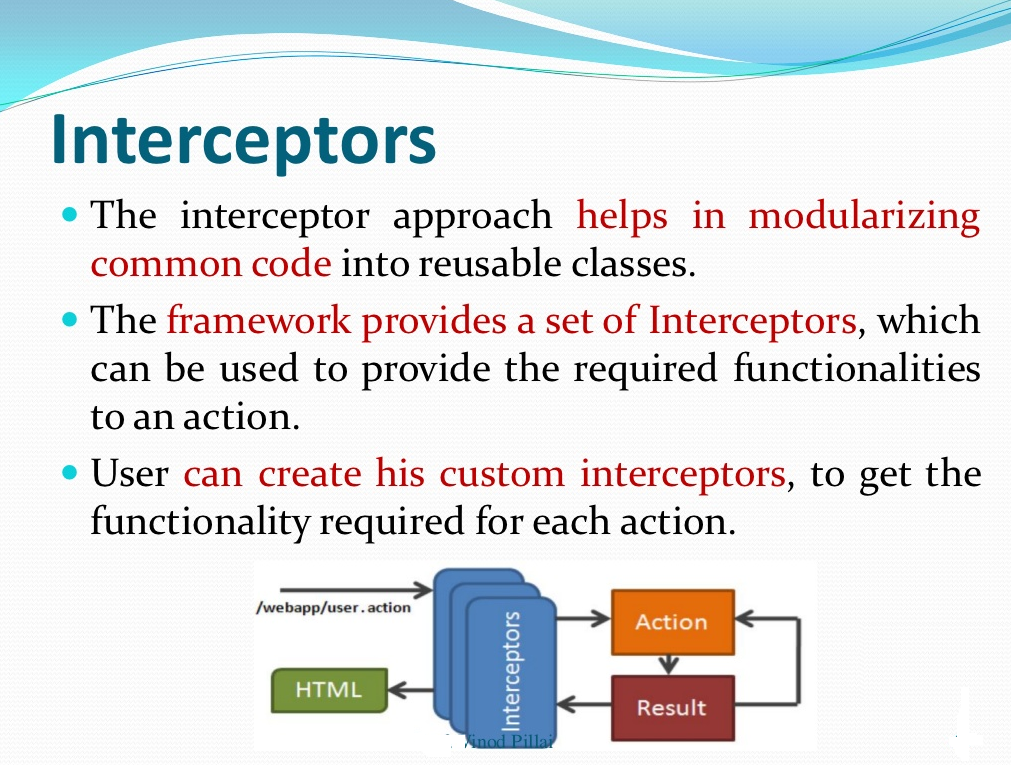
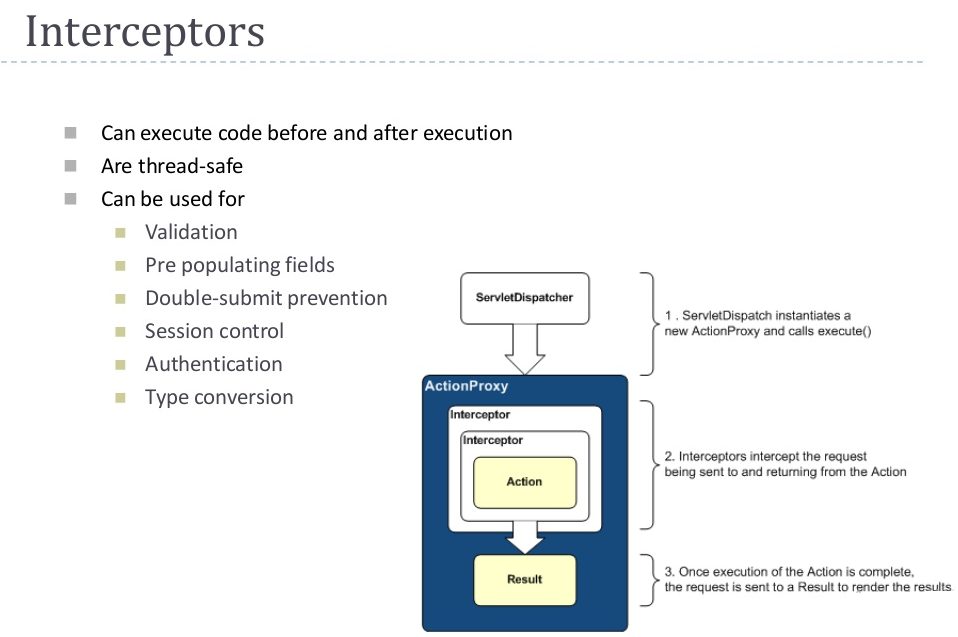
# Plug-in extensions points – allows new implementations

of the core framework classes to be substituted in to

Struts2, thus changing the way the framework behaves

--

**Interceptors**

** **

 Interceptors are one of the most powerful features of Struts 2.

**\*Result Types:**

 As Interceptor, as the name suggests, intercepts the request, and provides some additional processing before and after the execution of action and result.

**Plug-in packages :**

 The interceptor approach helps in modularizing common code into reusable classes.

 Common functionalities required by each action are implemented as Interceptors.

 These functionalities may include validation of input data, pre-processing of file upload, and sometimes pre-processing of controls with some data before the web page

Plug-in extensions points :

 The framework provides a set of Interceptors, which can be used to provide the required functionalities to an action.

 User can create his custom interceptors, to get the functionality required for each action.

--

**-Value Stack:**

- Central storage for request Domain model data

- serves as context per request

- "Glues" model and view

- Accessed by interceptors & views

- Stores action properties

- Also stores message bundles

- ValueStack is referenced and manipulated by OGNL

**OGNL Object Graph NavigationLanguage**

 OGNL is a powerful expression language that is used to reference and manipulate data on the ValueStack.

 OGNL also helps in data transfer and type conversion.

 The OGNL is very similar to the JSP Expression Language.

 OGNL is based on the idea of having a root or default object within the context Rajeev Gupta Java Training

OGNL is the interface between the Struts 2 framework string-based HTTP Input and Output and the Java-based internal processing

•we can use OGNL to display object models in JSPs.

1. Naresh Bisa (AF7561) [↑](#footnote-ref-1)