Developing Applications Using Java Web Frameworks

Session - 2

Working with Struts 2 Framework

Objectives

- Explain the features of Struts 2 framework
- Explain the architecture of Struts 2 framework
- Explain the components and request lifecycle of Struts 2 framework
- Explain the difference between Struts 1 and Struts 2 framework
- Explain how to implement Struts 2 actions using Action interface and ActionSupport class
- Explain the process of managing data from forms to JavaBean properties
- Explain how to configure Struts 2 components in the configuration file
- Explain Result and Result Types
- Explain the annotations provided by Struts 2 framework
- Explain how to develop a Java Web application using Struts 2 framework

Introduction 1-2

Apache Struts Framework

Java-based framework that separates the business logic from the presentation layer.

Based on MVC design paradigms, which separate the application development into three namely, **Model**, **View**, and **Controller**.

Struts 2 is a thorough revision of Struts architecture containing features of **WebWork** and **Struts 1**.

Introduction 2-2

- **Struts 2 = WebWork + Struts 1**

- **Struts 2 framework**
  - Is a flexible framework for creating enterprise-ready Java Web applications based on cleaner implementation of MVC.
  - Is designed to simplify the entire development cycle, including building, deploying, and maintaining applications over time.

Features of Struts 2

- Simplified Design
- Configurable MVC Components
- POJO-based Actions and Forms
- Enhanced Tags
- Object Graph Navigation Language (OGNL) and ValueStack
- Easy Integration
- Annotation Support
- Minimal Configurations
- AJAX Support
- View Technologies

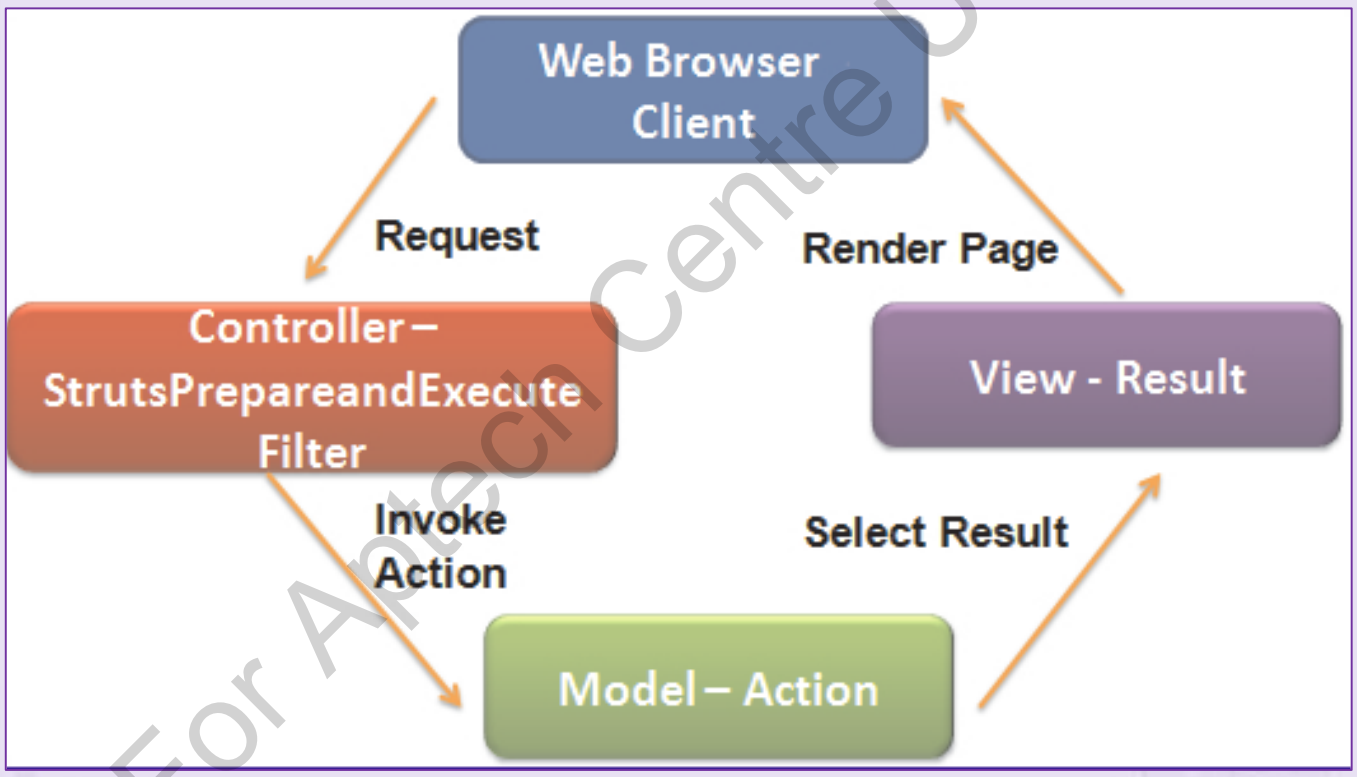
Architecture of Struts 2

Struts 2 is based on a Pull-MVC framework.

- Pull-MVC framework means:
  - Data that is to be displayed is pulled from Action class, which acts as a model in the Web application.
- Struts 2 also provides powerful APIs for configuring the Validators, Interceptors, and so on that reduces the processing on the Action class.

Struts 2: MVC Pattern 1-3

- Following figure shows Struts 2 MVC Pattern – Core components:



Struts 2: MVC Pattern 2-3

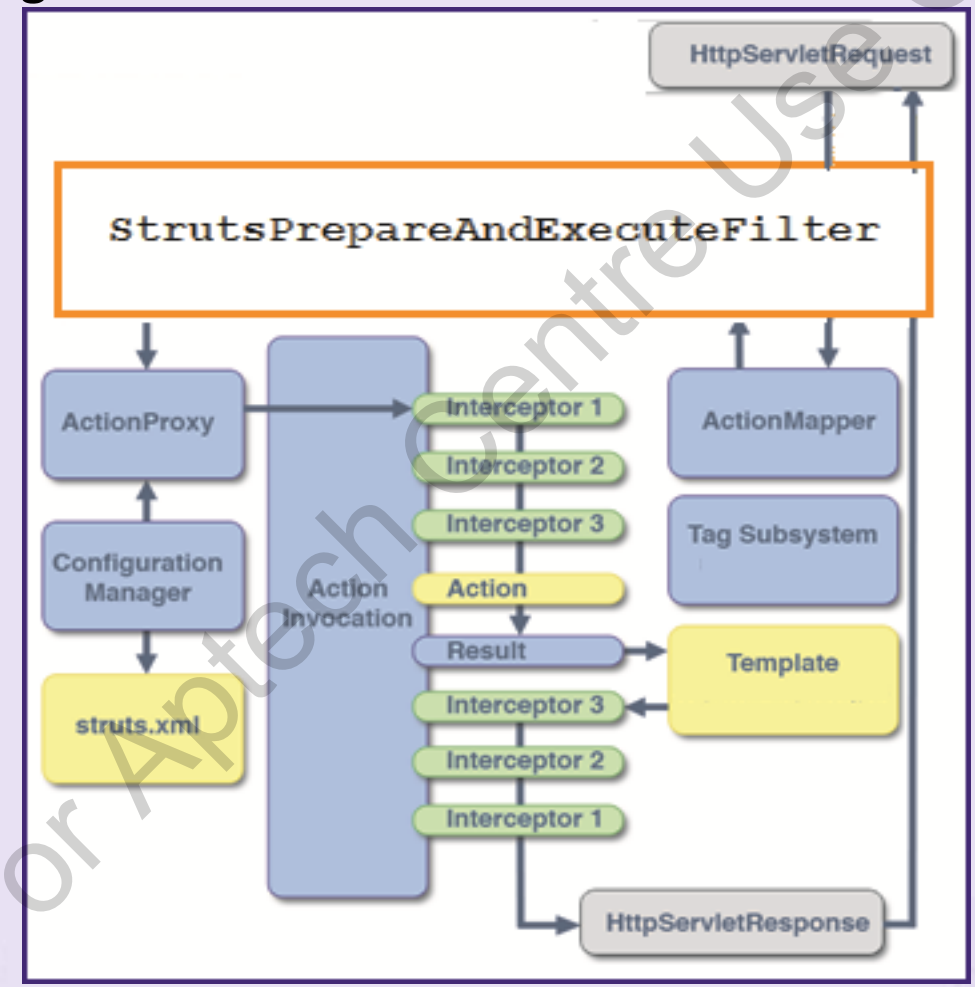
- Controller
  - Controls all the processing done by the application.
  - In Struts 2, `StrutsPrepareAndExecuteFilter` plays the role of a Controller.
  - The `StrutsPrepareAndExecuteFilter` is a Servlet filter that checks each incoming requests and determines the Struts 2 action to handle the request.
- Model
  - Is a business object and has no user interface.
  - Can be defined as the internal state of an application.
  - Is implemented by the action component.
  - Contains both the data model and the business logic.

Struts 2: MVC Pattern 3-3

- View
  - Represents the presentation component of the MVC design pattern.
  - Returns the result page to the Web browser.
  - Returns commonly a JSP pages to display the results.
  - Translates the state of the application to a visual presentation with which the user can interact.

Components of Struts Framework 1-5

- Following figure shows the Struts 2 Framework:



Components of Struts Framework 2-5

Request Initiation (HttpServletRequest)

- A request in a form of a URL initiates from and ends in a user's Web browser.
- A request initiated by a user is passed through a standard filter chain which makes the actual decision by invoking the required `StrutsPrepareAndExecuteFilter`.

Components of Struts Framework 3-5

- **StrutsPrepareAndExecuteFilter**
  - Plays a role of the Controller.
  - Is a Servlet filter that inspects each incoming request to decide which action should handle the request.
  - When called, the `FilterDispatcher` consults the `ActionMapper` to determine whether the request should invoke an Action.
- **Action Mapper**
  - Provides a mapping between HTTP requests and action invocation requests and vice-versa.
  - When `HttpServletRequest` comes, the `ActionMapper` tries to match an appropriate action invocation request.
  - If no action invocation request matches, it returns null and if the `ActionMapper` determines that an action should be invoked, the `StrutsPrepareAndExecuteFilter` delegates control to the `ActionProxy`.

Components of Struts Framework 4-5

❑ Action Proxy

- Reads the framework configuration files manager, such as the `struts.xml` file.
- Holds the configuration and context information to process the request and the execution results after the request has been processed.
- Creates an instance of `ActionInvocation` class and delegates the control.

❑ Action Invocation

- Responsible for command pattern implementation.
- In a request-response model, the request is routed to the invoker.
- Invoker sends it to the encapsulated command object and is then sent to the suitable method of the receiver to initiate proper action.
- The receiver object is then generated and attached to the command.
- The configured Interceptors are invoked one by one in sequence and then the Action is invoked.

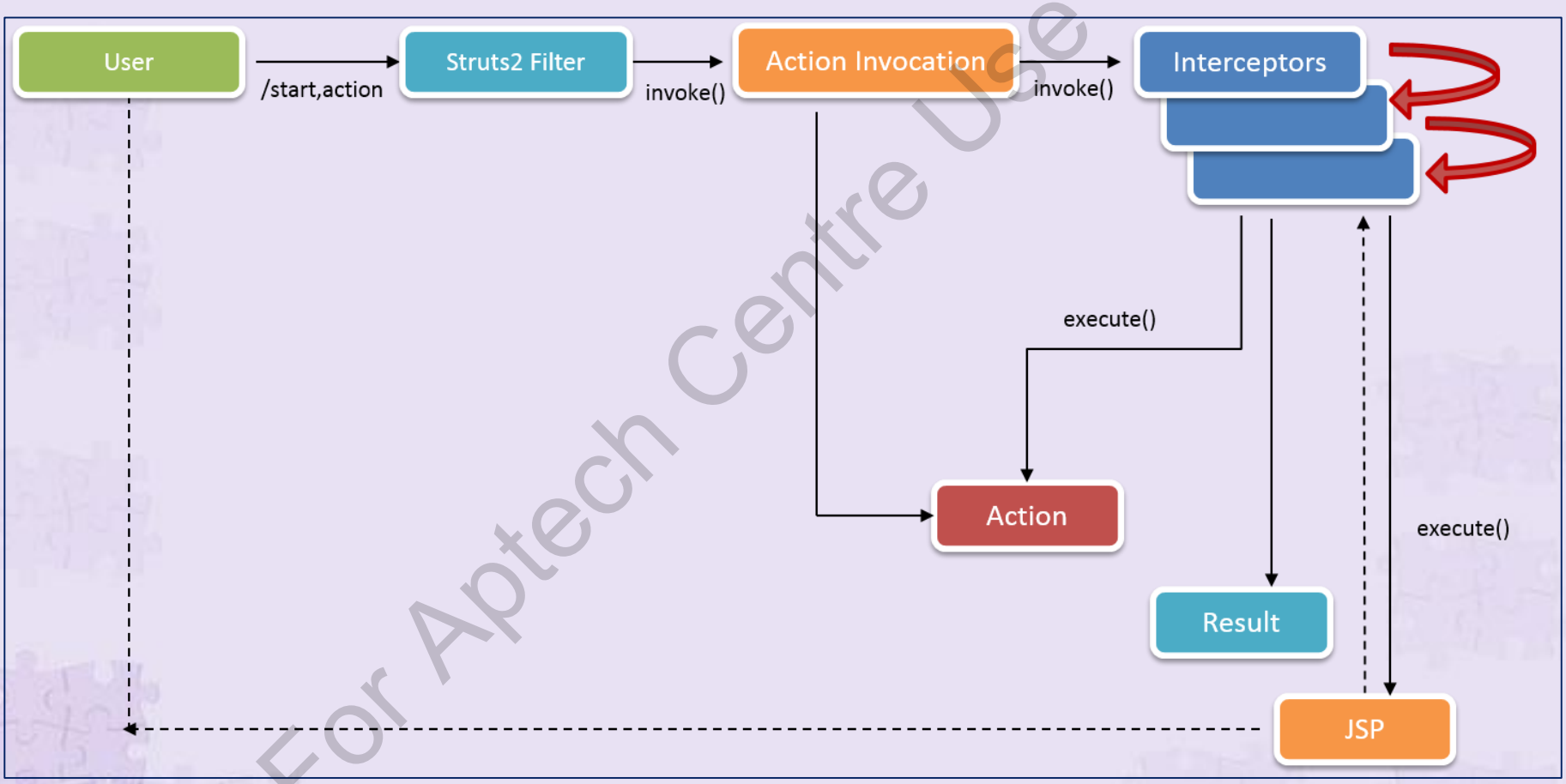
Components of Struts Framework 5-5

❑ Interceptors

- Provides pre-processing logic before invoking the action.
- Interacts with the action and set the request parameter on the action.
- Provides post-processing logic after invoking the action.
- Modifies the results.
- Catches exceptions to perform alternate processing or return different results.
- Core functionalities such as handling exception, type conversion, file upload, page preparation, and so on is implemented using pluggable Interceptors.
- Invokes both before and after the execution of the action and the results are rendered back to the user.

Request-Response Lifecycle 1-2

- Following figure shows the request-response lifecycle of Struts 2:



Request-Response Lifecycle 2-2

Step 1: User sends the request

Step 2: `FilterDispatcher` determines the appropriate action

Step 3: Interceptors are applied

Step 4: Action is executed

Step 5: Result is displayed

Configuring Struts 2 in web.xml File 1-2

The developer configures the Struts 2 Web application in `web.xml` file present in the WEB-INF folder of the application.

Struts 2 Web configuration uses `<filters>` in place of `<servlet>`.

The `URI` pattern should be specified as "`/*`" so as to ensure that all kind of request patterns are served by `org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter`.

Configuring Struts 2 in web.xml File 2-2

- Following figure shows the implementation of the Controller Servlet filter in `web.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
. . .
<filter>
    <filter-name>struts2</filter-name>
    <filter-class>
org.apache.struts2.dispatcher.ng.filter.StrutsPrepare
                                        AndExecuteFilter
</filter-class>
</filter>
<filter-mapping>
          <filter-name>struts2</filter-name>
           <url-pattern>/*</url-pattern>
</filter-mapping>

<session-config>
      <session-timeout> 30 </session-timeout>
</session-config>

<welcome-file-list>
     <welcome-file>example/index.jsp</welcome-file>
</welcome-file-list>
</web-app>
```

Difference Between Struts 1 and Struts 2 1-3

- Following Table lists the differences between Struts 1 framework and Struts 2 framework:

Feature	Struts 1	Struts 2
Controller	Struts 1 uses a Servlet Controller such as ActionServlet class.	Struts 2 uses a Filter to act as a Controller.
Form Input	In Struts 1 an HTML form mapped to an ActionForm object is used to capture inputs.	In Struts 2 Action properties are used for capturing input and thus eliminating the need for creating another class for obtaining input. The HTML form maps directly to a POJO class in Struts 2.

Difference Between Struts 1 and Struts 2 2-3

Feature	Struts 1	Struts 2
Model	In Struts 1 the Action interface is implemented. It acts as an adapter between the contents of an HTTP request and the corresponding business logic that executes to process the request.	Struts 2 action classes do not require to implement the Action interface. The Action object is a POJO object that can be easily tested.
Thread Model	In Struts 1, only one instance of the user-defined Action class is created for handling all incoming requests and so it is required to be thread-safe.	Struts 2 creates instance of user-defined Action class for each incoming request.
Tag Library	Struts 1 provides several tag libraries such as HTML, Bean, and Logic.	Struts 2 provide a single tag library.

Difference Between Struts 1 and Struts 2 3-3

Feature	Struts 1	Struts 2
Expression Language	Struts 1 provided Java Standard Tag Library (JSTL) Expression Language (EL).	Struts 2 uses OGNL as an expression language.
Type Conversion	Struts 1 uses JSP mechanism of binding object into the page contexts for access.	Struts 2 uses ValueStack for the taglibs to access values.
Control Of Action Execution	Struts 1 has a separate lifecycle for each of the module. All actions share same lifecycle in the module.	Struts 2 create separate lifecycles for each action using the Interceptor Stack.

Introduction to Struts 2 Action

- The action classes act as controller in the MVC pattern.
- The main functions of Struts 2 action classes includes responding to a user request, executing business logic, and then returning a result to the user based on configuration file (`struts.xml`) to render the view page.
- Struts 2 actions serve in two other important capacities:
  - Transferring the data from the request to the view, irrespective of the type of result.
  - Assisting the framework to determine which result should render the view that will be returned in response to the request.

Using Action Classes 1-6

- Contains business logic, handles client's data, retrieve resources, perform validation, and select appropriate result page for the view.
- As Struts 2 POJO-based actions, the class must implement an `execute()` method.
- The code specification inside the `execute()` method should only hold the logic of the work associated with the request.
- The `execute()` method returns a String value.
- The String value specifies the result page which has to be returned as a View to the client.

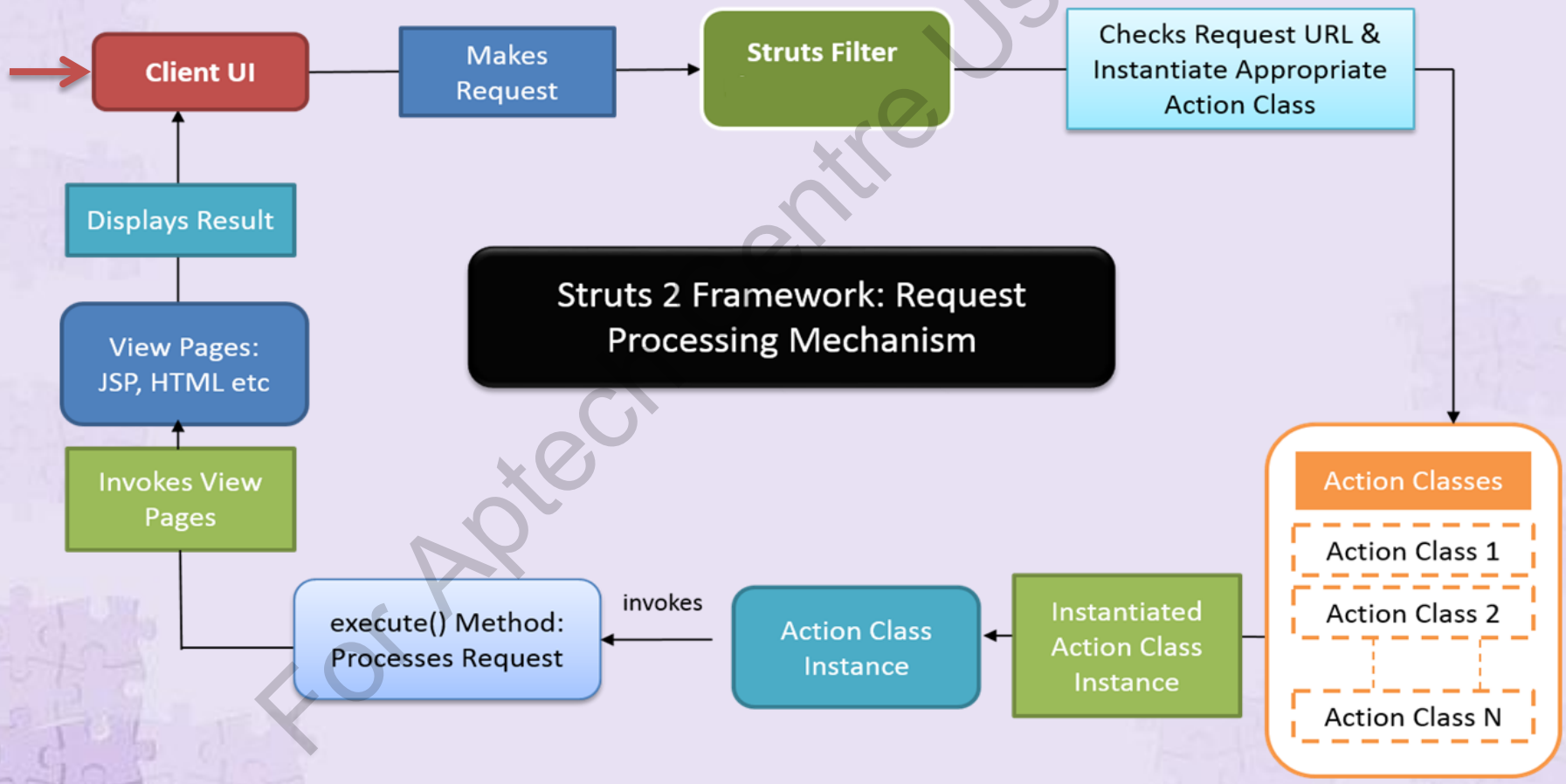
Using Action Classes 2-6

- Following code snippet shows how a POJO class can be implemented as Action class in Struts 2:

```
public class MessageAction{
        // execute() method containing business logic
        public String execute() {
                return "success";
        }
 }
```

Using Action Classes 3-6

- Following figure shows the Request Processing Mechanism of `Action` class.



Using Action Classes 4-6

- Following are helpers of Action class:
  - **Action interface**. – It can be implemented by the Action class. It is provided in the package `com.opensymphony.xwork2` in the Struts 2 framework.
  - **ActionSupport Class** – It is defined in the `com.opensymphony.xwork2` package and can be extended by the Action class.

Using Action Classes 5-6

❑ Action Interface

- Exposes the `execute()` method to the action class implementing it.
- Declares five constants that can be returned from the Action class.
- Following figure shows the Action interface defined by Struts framework:

```
public Interface Action {
 //States that the action execution was successful
   public static final String SUCCESS = "success";
//states that the action execution was successful
//but do not show a view.
   public static final String NONE = "none";
//States that the action execution was a failure
   public static final String ERROR = "error";
//States that the action execution requires
//more input in order to succeed
   public static final String INPUT = "input";
//States that the action could not be executed as
//the user was not logged in
   public static final String LOGIN = "login";
// Business logic
   public String execute() throws Exception;
}
```

Using Action Classes 6-6

❑ ActionSupport Class

- Adds useful utilities to the class that extends it.
- Provides a default implementation for the `execute()` method.
- The default implementation of the `execute()` method in the `ActionSupport` class returns `Action.SUCCESS` String object.
- Following figure shows the `ActionSupport` class declaration semantics.

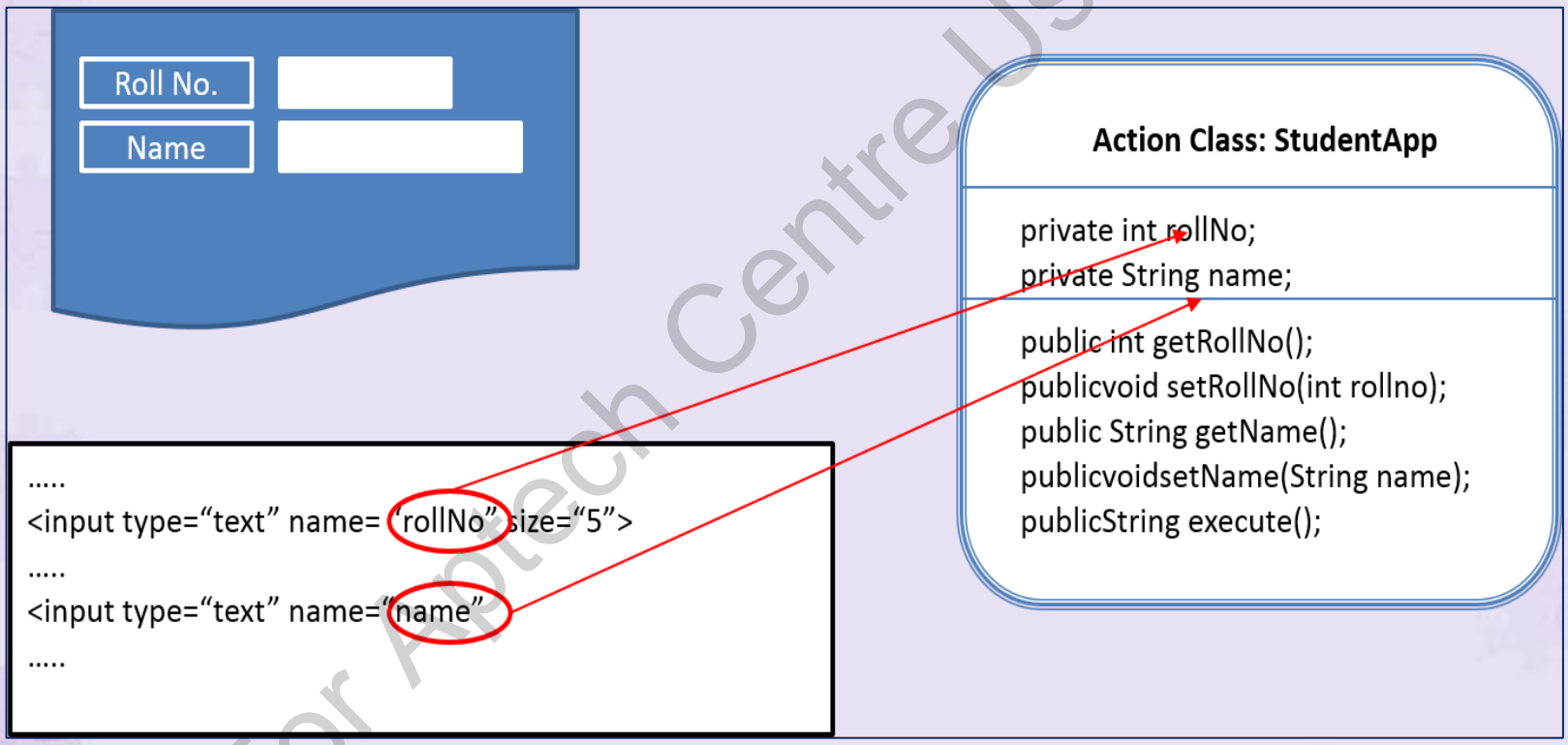
```
public ActionSupport implements Action,
Validateable, ValidationAware, TextProvider,
LocaleProvide, Serializable {
. . .
    public String execute()throws Exception {
            return SUCCESS;
    }
}
```

Actions and Form Data 1-2

- The Struts 2 framework follows the JavaBean paradigm.
- Struts 2 framework, automatically moves the request parameters from the form to the JavaBeans properties that have matching names.
- The data from the form fields is stored locally in the Action class by using the class instance variables or properties.
- The getter and setter methods help to retrieve and assign the data values to the properties respectively.
- The `execute()` method references the data using these properties.

Actions and Form Data 2-2

- Following figure shows the mapping of the form field with the JavaBeans properties:



Actions Return Value

- After the completion of action, the `execute()` method returns a control string.
- The string helps the Struts 2 filter to decide the result/view page that should be rendered.
- The mapping of the resultant string and the view page to be rendered is defined in the configuration file named struts.xml.
- Following figure displays the `execute()` method code specification:

```
……
publicString execute () {
        setMessage("Welcome"+getName());
        return "success";
}
……
```

Configuring Actions

- The `struts.xml` is the configuration file used in Struts 2.
- Following code snippet shows the implementation of `struts.xml` file:

```
<!DOCTYPE struts PUBLIC
. . .
<struts>
<include file="example.xml"/>
      <!-- Configuration for the default package. -->
      <package name="default" namespace="/" extends="struts-default"
method="execute">
        <action name="StudentApp"
                          class="com.demo.myApp.StudentApp">
            <result name="success">/SuccessView.jsp</result>
             <result name="error">/ErrorView.jsp </result>
        </action>
      </package>
</struts>
```

Result and ResultTypes

- Following table shows the configured Result Types provided by Struts 2 framework:

Method	Description
Dispatcher	Rendered using JSP.
Chain	Chains actions with each other.
HttpHeader	Returns a configured HTTP header response.
Redirect	Redirects the user to a configured URL.
RedirectAction	Redirects the user to a configured action.
Stream	Streams raw data to the browser.
PlainText	Returns the content as plain text.
FreeMaker	The page is rendered as FreeMaker.
XSL	Renders XML to the browser, which is transformed using XSLT.

Configuring Result

- The developer can set the result types in the `struts.xml` configuration file.
- Following code snippet shows the setting of default result type in `struts.xml`:

```
<result-types>
      <result-type name="dispatcher" default="true"

class="org.apache.struts2.dispatcher.ServletDispatcherResult"
/>
</result-types>
```

- The code declares dispatcher as the default result type for the package.
- This means each request will be forwarded to a Web resource such as JSP to render the result to the client.

User-defined Results 1-3

- Consider a scenario where the application needs to perform the Create, Read, Update, and Delete (CRUD) operations.
- Depending on the operation performed, the appropriate page needs to be rendered in the result.
- The `execute()` method of the action class defines the logical name of the action.
- Action class returns a bunch of logical result strings which can be mapped to appropriate result pages in the configuration file.

User-defined Results 2-3

- Following code snippet shows the `execute()` method with different logical results:

```
public class CRUDAction
{
    public String execute() {
     if(createOper())
     { return "create";  }
     else if (deleteOper())
     { return "delete"; }
     else if (readOper())
     { return "read"; }
     else if (writeOper())
     { return "write"; }
     else
      return "error";
     }
}
```

- The code evaluates the invoked methods and return the appropriate result value from the `execute()` method.

User-defined Results 3-3

- Following code snippet shows the mapping of the results with their corresponding view pages configured in the `struts.xml` file:

```
<struts>
    <action name="CRUDaction" class="actions.CRUDAction">
    <result name = "create">/create.jsp</result>
    <result name = "delete">/delete.jsp</result>
    <result name = "read">/read.jsp</result>
     <result name = "update">update.jsp</result>
</action>
```

- Based on the return result value, the appropriate View is displayed to the user.

Action Annotations 1-3

❑ Some of the Action annotations are as follows:

`@Action`

- Used to mark the action class.
- Maps to some URL called as action path.
- The action path is made up of package, class, and method name of an action.
- Struts 2 will automatically create action for classes name ending with Action.
- The action name is determined by removing the Action suffix and converting first letter to lowercase.
- If the class is not annotated with @Result annotation to provide the result, then result pages are looked into **WEB-INF/content** directory of the project directory.
- The name of the result page should be `{action}-{return_string}.jsp`.

Action Annotations 2-3

@Namespace

- Helps to set the path of the action URL.
- Placed before the action class.
- Applies to all actions defined in the class, even if the fully qualified URLs are not specified.
- If no namespace annotation is used, the namespace will be generated from the package name.

@Result and @Results

- Used for configuring a result for the return value.
- Can be declared at global or local level.
- If it is added at the class level, it is a global annotation.
- If it is defined at the method level, then it is local.
- The annotation accepts four parameters for configuration - name, value, type, and parameters.

Action Annotations 3-3

- To use Struts 2 annotations, we need to add `struts2-convention-plugin` library in the classpath and in `web.xml`.
- Following code snippet displays the configuration setting of the `web.xml` configuration file.

```
...
<filter>
   <filter-name> struts 2 </filter-name>
   <filter-class>org.apache.struts2.dispatcher.ng.filter.
StrutsPrepareAndExecuteFilter</filter-class>
<init-param>
   <param-name>actionPackages</param-name>
   <param-value>com.book.session2.actions</param-value>
</init-param>
</filter>
...
```

- The `<param-name>` and the `<param-value>` elements have been used to specify the location of the packages containing the action classes using annotations.

Creating a Web Application Using Struts 2

- Struts 2 application is based on MVC design pattern.
- To create a Web application in Struts 2, the steps to be followed are as follows:
  - Create views which will provide user interface for the applications.
  - Create an action class; define the input properties and its getter/setter methods within the action class and define the `execute()` method to perform the action implementation.
  - Establish relationship between the Views and Controllers using configuration file, `struts.xml`, or using annotations.
  - Modify the `web.xml` file to enable the Web application.
  - Build and run the Web application.

Create Views 1-4

- Views represent the presentation layer of the application.
- Consider that there are three views - `index.jsp`, `AcceptDetails.jsp`, and `DisplayDetails.jsp`.
- Following code snippet shows the inclusion of Struts 2 tag library in the JSP page:

```
...
<%@taglib prefix="s" uri="/struts-tags" %>
. . .
<%@taglib uri="/struts-tags" prefix="s" %>
```

Create Views 2-4

- Following code snippet shows the implementation of `index.jsp` page:

```
...
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Welcome Page</title>
</head>
<body>
<h1>Welcome to Struts 2 Example!</h1>
Click to enter <s:a href="example/AcceptDetails.jsp"> Details
</s:a>
</body>
</html>
...
```

- The `index.jsp` page uses Struts 2-specific HTML tag.
- The `<s:a href>` tag is used to emulate an HTML link.

Create Views 3-4

- Following code snippet shows the implementation of the `AcceptDetails.jsp` page:

```
. . .
<body>
<h1> User Details Page </h1>
<s:form action="WelcomeMessage">
<s:textfield name="FirstName" label="First Name"/>
<s:textfield key="LastName" label="Last Name"/>
<s:submit/>
</s:form>
</body>
</html>

...
```

- The `<s:textfield/>` emulates an HTML text field.
- The `<s:submit/>` tag emulates an HTML submit button.

Create Views 4-4

- Following code snippet shows the implementation of the `DisplayDetails.jsp` page:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib uri="/struts-tags" prefix="s" %>
<html>
. . .
<body>
<h1>User Details:</h1>
<h3> <s:property value="message"/></h3>
</body>
</html>
```

- The view is displayed only after the successful submission of form by the user.
- The `<s:property value="message"/>` tag is used to automatically find the message property and print the value in the message property.

Creating the Action Class 1-3

- The data sent from `AcceptDetails.jsp` page is processed by action class.
- The action class extends the `ActionSupport` class.
- Following code snippet shows the implementation of the `action` class:

```
...
public class WelcomeMessage extends ActionSupport {
 /*
  * Declaring properties
  */
private String firstName;
private String lastName;
private String message;
...
```

- All the input fields submitted in the `AcceptDetails.jsp` page are declared as properties in the `WelcomeMessage` action class.

Creating the Action Class 2-3

```
...
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 String getMessage() {
return message;
...
```

- The code declares the getter and setter methods to access the form data submitted by the client.

Creating the Action Class 3-3

```
...
public String getMessage() {
return message;
}
public void setMessage(String message) {
this.message = message;
}
public String execute() throws Exception {
   java.util.Date dt = new java.util.Date();
//Defining temporary variable for storing the result message
  String tmpmessage;
  tmpmessage = "Hello " + getFirstName() + " " + getLastName();
  tmpmessage = tmpmessage + " You have logged in at " + dt.toString();
...
//Setting the result message to the message variable
  setMessage(tmpmessage);
  return SUCCESS;   }
}
```

- The OGNL expression language maps the input fields of the `AcceptDetails.jsp` page with the properties in the `WelcomeMessage` action class.

Actions in struts.xml File 1-4

- Following code snippet shows the implementation of the `struts.xml` file.

```
. . .
<package name="example" namespace="/example" extends="struts-
default">
<action name="WelcomeMessage" class="example.WelcomeMessage">
<result name="success">/example/DisplayDetails.jsp</result>
</action>
</package>
</struts>
. . .
```

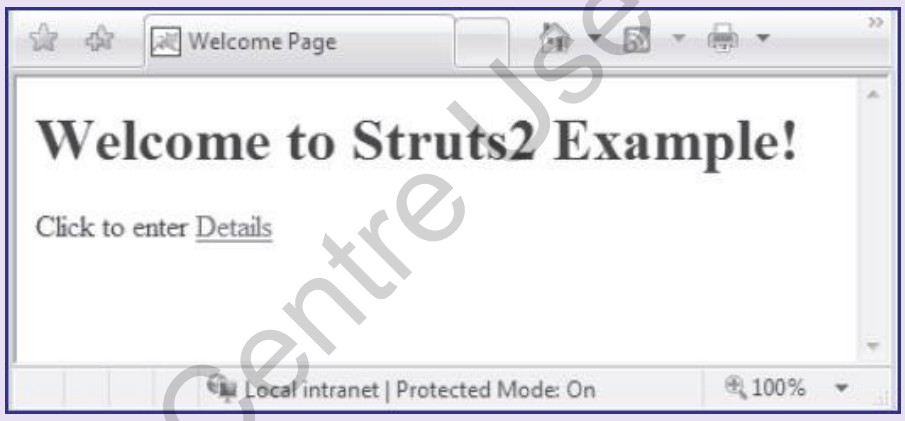
- The `<action />` element maps an identifier to handle an action class.
- The mapping determines how to process the request when a it is matched.

Actions in struts.xml File 2-4

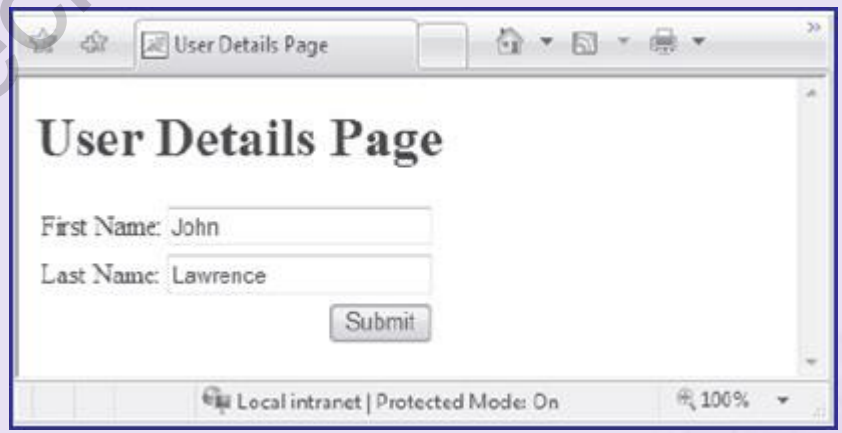
- The attributes of action element are name and class.
- The name attribute identifies the name of the action which matches a part from the URL.
- The class attribute identifies the action class to be executed with full package description. The sub-element of action is result.
- The `<result />` element displays the desired view.
- The `execute()` method of the action class returns a String value.
- The `execute()` method can return `ERROR` or `INPUT` for errors or conversion issues respectively.

Actions in struts.xml File 3-4

- Following figure displays the application startup page:

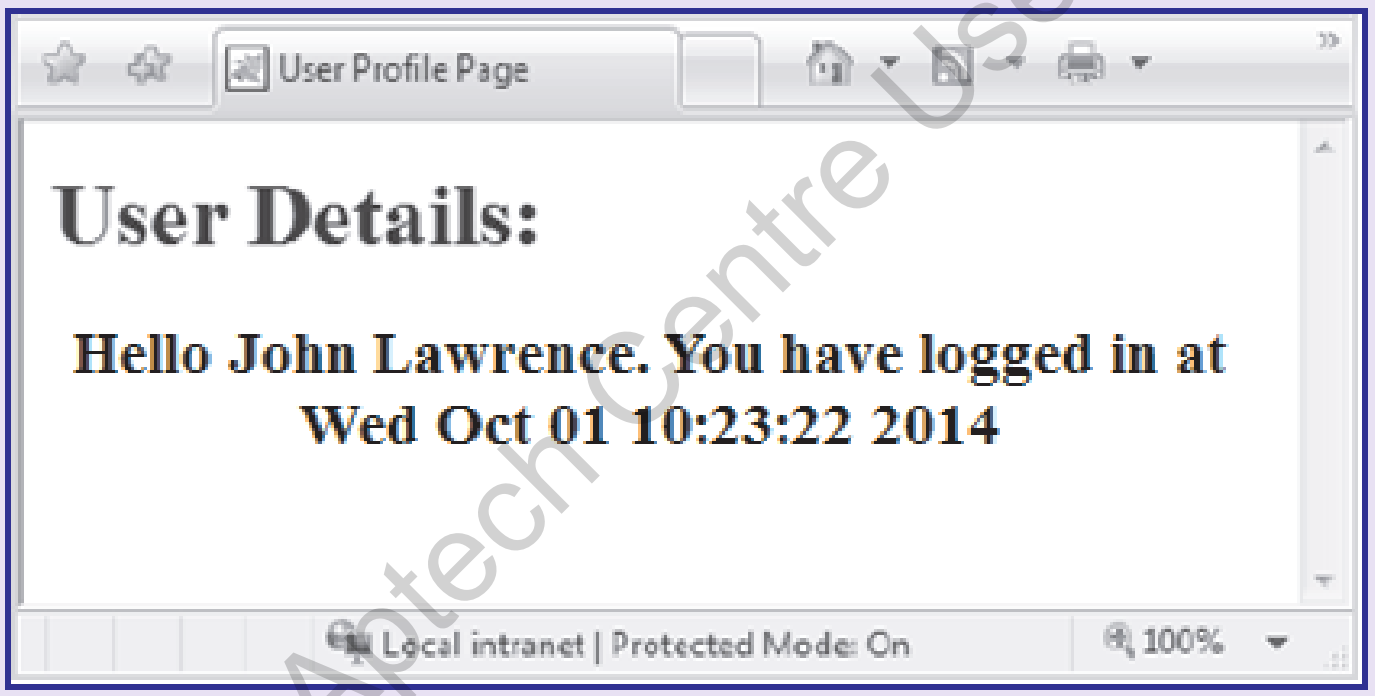


- Following figure displays the input page for entering the user details:



Actions in struts.xml File 4-4

- Following figure displays the Welcome page:



Summary

- Apache Struts is a Java-based framework that separates the business logic from the presentation layer.
- Some of the important features of Struts 2 framework are namely, simplified design, MVC components, POJO-based actions, enhanced tags, OGNL and ValueStack, easy integration, template support, annotations, AJAX support, and View technologies.
- In Struts 2, FilterDispatcher plays the role of a Controller. A Model is a business object and has no user interface. View component returns the result page to the Web browser.
- Various components of the Struts2 framework are namely FilterDispatcher, Action Mapper, Action Proxy, Action Invocation, and Interceptors.
- The FilterDispatcher accepts the request and then consult ActionMapper to determine the suitable Action.
- The main functions of Struts 2 action includes responding to a user request, executing business logic, and then returning a result to the user based on configuration file (struts.xml) to render the view page.
- Struts 2 framework also provides helper interfaces and classes which can be used for adding additional functionalities to an action class. They are Action interface and ActionSupport class.
- The data from the form fields is stored locally in the Action class by using the class instance variables or properties.