

MCV in JSP, and Struts 2

Model 1 and Model 2 (MVC) Architecture

1. [Model 1 and Model 2 \(MVC\) Architecture](#)
2. [Model 1 Architecture](#)
3. [Model 2 \(MVC\) Architecture](#)

Before developing the web applications, we need to have idea about design models. There are two types of programming models (design models)

1. Model 1 Architecture
2. Model 2 (MVC) Architecture

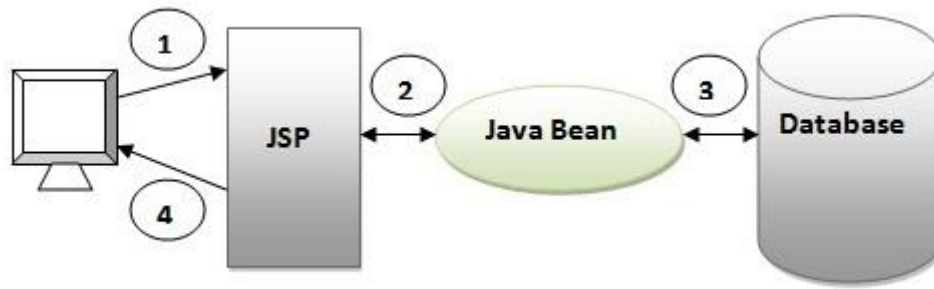
Model 1 Architecture

Servlet and JSP are the main technologies to develop the web applications.

Servlet was considered superior to CGI. Servlet technology doesn't create process, rather it creates thread to handle request. The advantage of creating thread over process is that it doesn't allocate separate memory area. Thus many subsequent requests can be easily handled by servlet.

Problem in Servlet technology Servlet needs to recompile if any designing code is modified. It doesn't provide separation of concern. Presentation and Business logic are mixed up.

JSP overcomes almost all the problems of Servlet. It provides better separation of concern, now presentation and business logic can be easily separated. You don't need to redeploy the application if JSP page is modified. JSP provides support to develop web application using JavaBean, custom tags and JSTL so that we can put the business logic separate from our JSP that will be easier to test and debug.



As you can see in the above figure, there is picture which show the flow of the model1 architecture.

1. Browser sends request for the JSP page
2. JSP accesses Java Bean and invokes business logic
3. Java Bean connects to the database and get/save data
4. Response is sent to the browser which is generated by JSP

Advantage of Model 1 Architecture

- Easy and Quick to develop web application

Disadvantage of Model 1 Architecture

- **Navigation control is decentralized** since every page contains the logic to determine the next page. If JSP page name is changed that is referred by other pages, we need to change it in all the pages that leads to the maintenance problem.
- **Time consuming** You need to spend more time to develop custom tags in JSP. So that we don't need to use scriptlet tag.
- **Hard to extend** It is better for small applications but not for large applications.

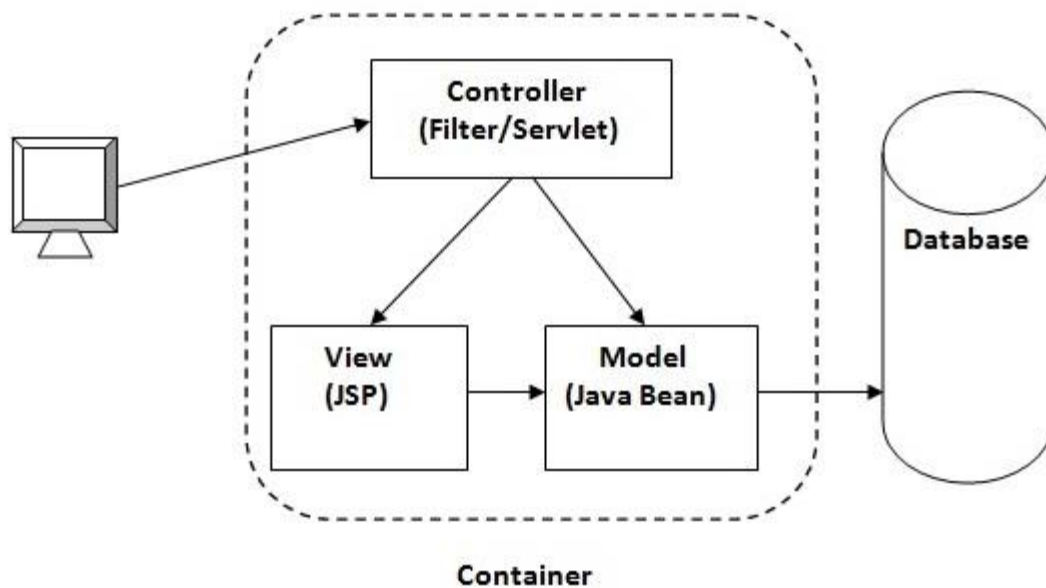
Model 2 (MVC) Architecture

Model 2 is based on the MVC (Model View Controller) design pattern. The MVC design pattern consists of three modules model, view and controller.

Model The model represents the state (data) and business logic of the application.

View The view module is responsible to display data i.e. it represents the presentation.

Controller The controller module acts as an interface between view and model. It intercepts all the requests i.e. receives input and commands to Model / View to change accordingly.



Advantage of Model 2 (MVC) Architecture

- **Navigation control is centralized** Now only controller contains the logic to determine the next page.
- **Easy to maintain**
- **Easy to extend**
- **Easy to test**
- **Better separation of concerns**

Disadvantage of Model 2 (MVC) Architecture

- We need to write the controller code self. If we change the controller code, we need to recompile the class and redeploy the application.

Solution of Model 2 Architecture: Configurable MVC Components

It uses the declarative approach for defining view components, request mapping etc. It resolves the problem of Model 2 architecture. The Struts framework provides the configurable MVC support. In struts 2, we define all the action classes and view components in struts.xml file.

MVC in JSP

MVC stands for Model View and Controller. It is a **design pattern** that separates the business logic, presentation logic and data.

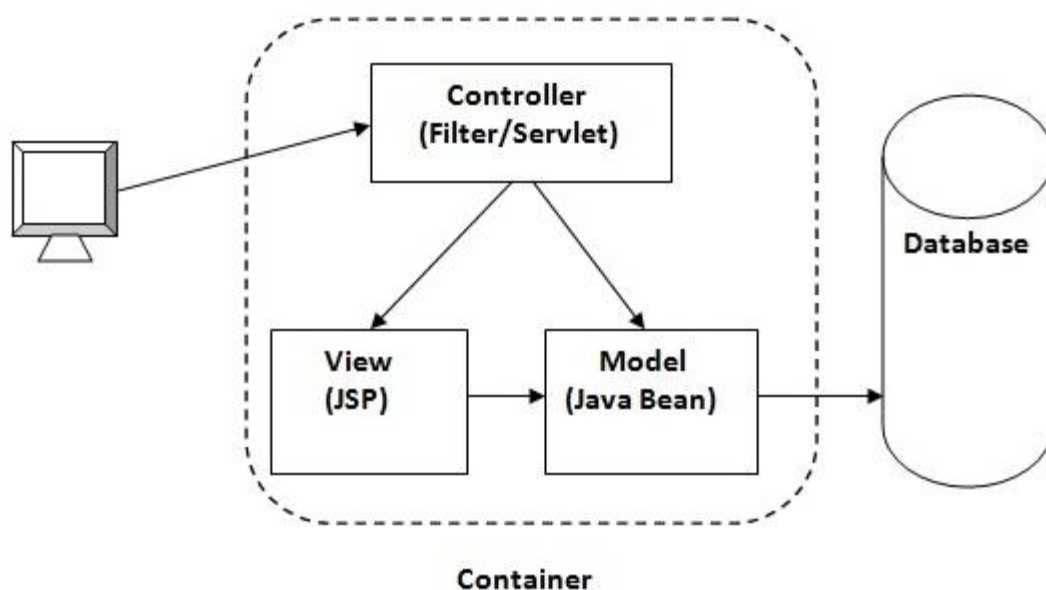
Controller acts as an interface between View and Model. Controller intercepts all the incoming requests.

Model represents the state of the application i.e. data. It can also have business logic.

View represents the presentaion i.e. UI(User Interface).

Advantage of MVC (Model 2) Architecture

1. Navigation Control is centralized
2. Easy to maintain the large application



Example of following MVC in JSP

In this example, we are using servlet as a controller, jsp as a view component, Java Bean class as a model.

In this example, we have created 5 pages:

- **index.jsp** a page that gets input from the user.
- **ControllerServlet.java** a servlet that acts as a controller.
- **login-success.jsp** and **login-error.jsp** files acts as view components.

- **web.xml** file for mapping the servlet.

File: index.jsp

```
1. <form action="ControllerServlet" method="post">
2. Name:<input type="text" name="name"><br>
3. Password:<input type="password" name="password"><br>
4. <input type="submit" value="login">
5. </form>
```

File: ControllerServlet

```
1. package com.javatpoint;
2. import java.io.IOException;
3. import java.io.PrintWriter;
4. import javax.servlet.RequestDispatcher;
5. import javax.servlet.ServletException;
6. import javax.servlet.http.HttpServlet;
7. import javax.servlet.http.HttpServletRequest;
8. import javax.servlet.http.HttpServletResponse;
9. public class ControllerServlet extends HttpServlet {
10.     protected void doPost(HttpServletRequest request, HttpServletResponse response)
11.         throws ServletException, IOException {
12.         response.setContentType("text/html");
13.         PrintWriter out=response.getWriter();
14.
15.         String name=request.getParameter("name");
16.         String password=request.getParameter("password");
17.
18.         LoginBean bean=new LoginBean();
19.         bean.setName(name);
20.         bean.setPassword(password);
21.         request.setAttribute("bean",bean);
22.
23.         boolean status=bean.validate();
24.
25.         if(status){
26.             RequestDispatcher rd=request.getRequestDispatcher("login-success.jsp");
27.             rd.forward(request, response);
28.         }
29.         else{
30.             RequestDispatcher rd=request.getRequestDispatcher("login-error.jsp");
31.             rd.forward(request, response);
32.         }
33.
34.     }
35.
36.     @Override
37.     protected void doGet(HttpServletRequest req, HttpServletResponse resp)
38.         throws ServletException, IOException {
39.         doPost(req, resp);
40.     }
41. }
```

File: LoginBean.java

```
1. package com.javatpoint;
2. public class LoginBean {
3.     private String name,password;
```

```

4.
5. public String getName() {
6.     return name;
7. }
8. public void setName(String name) {
9.     this.name = name;
10.}
11. public String getPassword() {
12.     return password;
13.}
14. public void setPassword(String password) {
15.     this.password = password;
16.}
17. public boolean validate(){
18.     if(password.equals("admin")){
19.         return true;
20.     }
21.     else{
22.         return false;
23.     }
24.}
25.}

```

File: login-success.jsp

```

1. <%@page import="com.javatpoint.LoginBean"%>
2.
3. <p>You are successfully logged in!</p>
4. <%
5. LoginBean bean=(LoginBean)request.getAttribute("bean");
6. out.print("Welcome, "+bean.getName());
7. %>

```

File: login-error.jsp

```

1. <p>Sorry! username or password error</p>
2. <%@ include file="index.jsp" %>

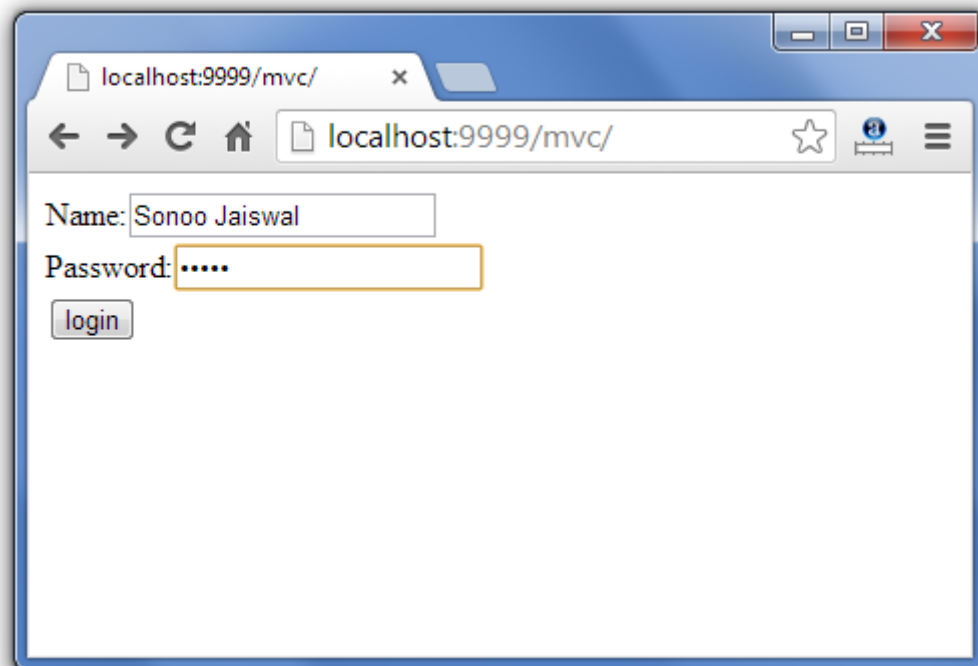
```

File: web.xml

```

1. <?xml version="1.0" encoding="UTF-8"?>
2. <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/ja
   vae/web-app_2_5.xsd"
4. xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/ja
   vae/web-app_3_0.xsd"
5. id="WebApp_ID" version="3.0">
6.
7.     <servlet>
8.         <servlet-name>s1</servlet-name>
9.         <servlet-class>com.javatpoint.ControllerServlet</servlet-class>
10.     </servlet>
11.     <servlet-mapping>
12.         <servlet-name>s1</servlet-name>
13.         <url-pattern>/ControllerServlet</url-pattern>
14.     </servlet-mapping>
15. </web-app>

```



Struts 2

The **struts 2 framework** is used to develop **MVC-based web application**.

The struts framework was initially created by **Craig McClanahan** and donated to Apache Foundation in May, 2000 and Struts 1.0 was released in June 2001.

[Struts 2](#) is a famous Model-View-Controller (MVC) framework, mainly found in the enterprise market since its release in 2000. Its goal is to separate the model (the business logic) from the view (the HTML pages) and the controller (the instance that manages the interaction between the model and the view). Struts provides the controller class, while we are in charge of creating the rest.

The current stable release of Struts is Struts 2.3.16.1 in March 2, 2014.

This struts 2 tutorial covers all the topics of Struts 2 Framework with simplified examples for beginners and experienced persons.

Struts 2 Framework

The Struts 2 framework is used to develop MVC (Model View Controller) based web applications. Struts 2 is the combination of **webwork framework** of opensymphony and **struts 1**.

1. struts2 = webwork + struts1

The Struts 2 provides supports to POJO based actions, Validation Support, AJAX Support, Integration support to various frameworks such as Hibernate, Spring, Tiles etc, support to various result types such as Freemarker, Velocity, JSP etc.

Struts 2 “HelloWorld” Example

In this example we will create a “Hello World Struts 2” application that will show you the basics of this framework.

Despite its simplicity, you will learn about the different parts of a Struts 2 applications and how you can use them as a base for your future projects.

1. Initial steps

If you haven't done this, go to <http://struts.apache.org/> and download the latest Struts 2 version. Once downloaded, unzip it and keep it at hand.
Open Eclipse and create a new Dynamic Web Project.

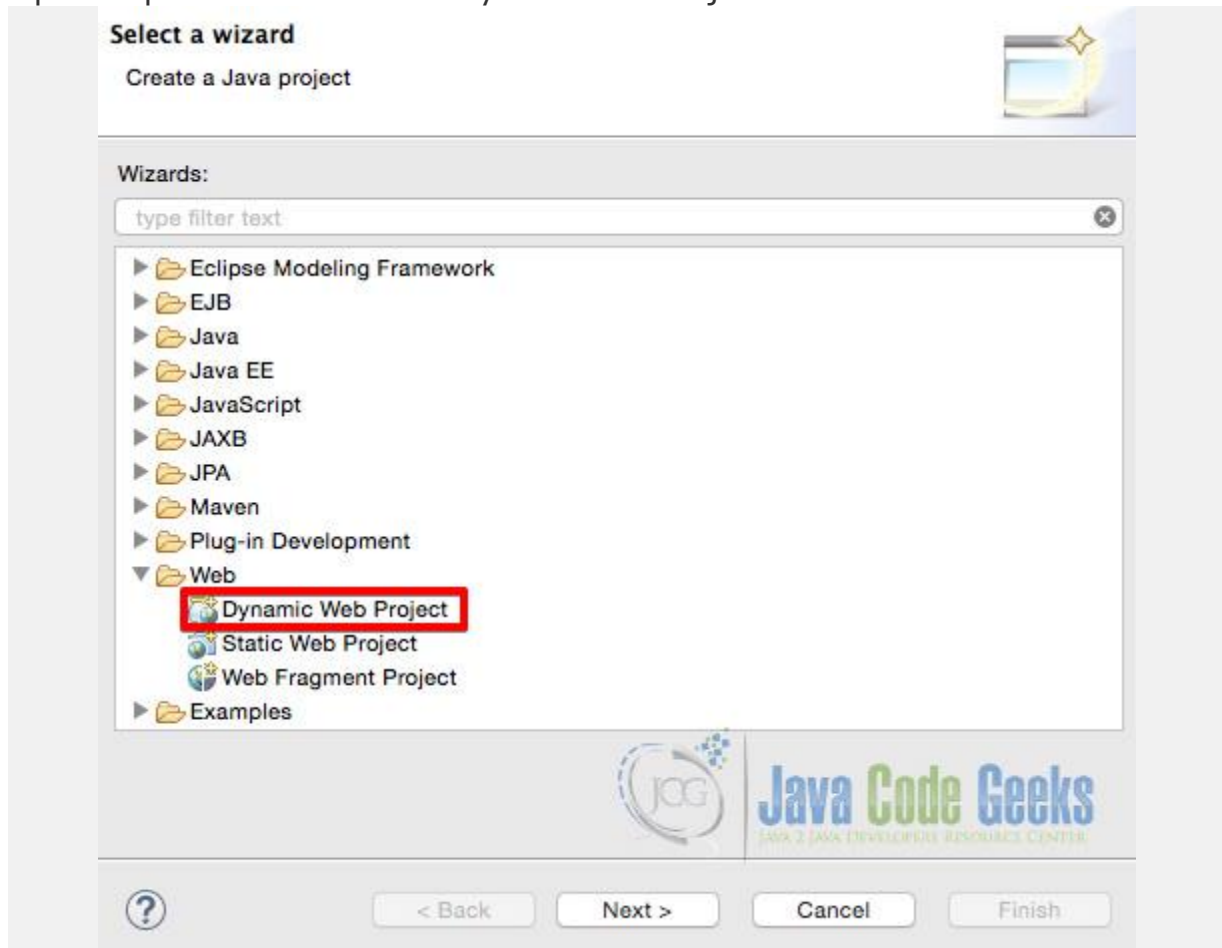
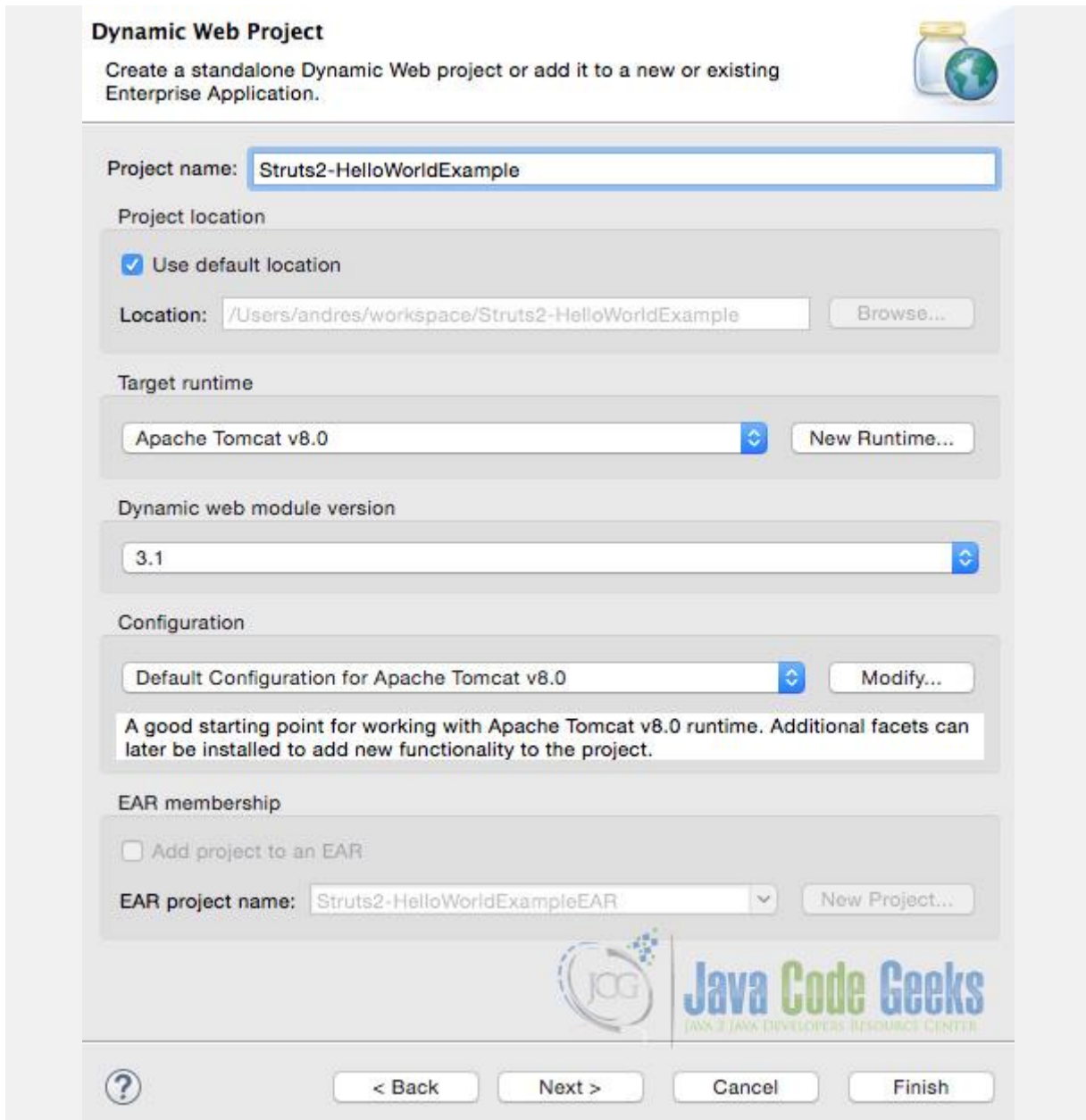


Figure 1: Dynamic Web Project

Fill the New Dynamic Web Project information window. You can use the following screenshot as a guide:



Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name:

Project location

☒ Use default location

Location:

Target runtime

Dynamic web module version

Configuration

A good starting point for working with Apache Tomcat v8.0 runtime. Additional facets can later be installed to add new functionality to the project.

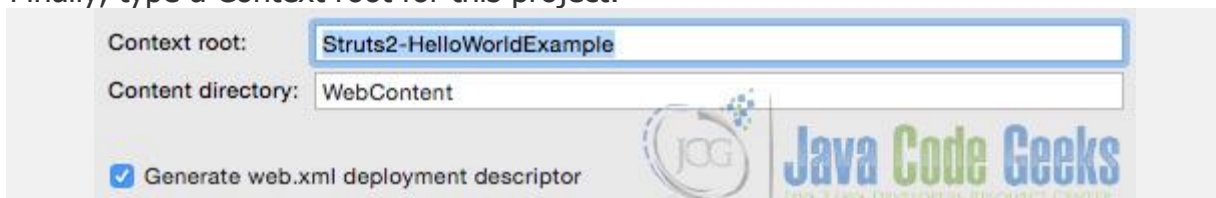
EAR membership

☐ Add project to an EAR

EAR project name:

Figure 2: Project Details

Finally, type a Context root for this project:



Context root:

Content directory:

☒ Generate web.xml deployment descriptor

Figure 3: Project Details

As a final step, add the following JAR files to the project (all of them are included in the Struts file you previously downloaded):



Figure 4: Libraries

Ok, it's time to begin coding.

2. Action Classes

The Struts controller calls **Actions Classes** (the model) to respond to web requests. A simple Action Class does not have any dependency.

In this example, we will create an `Action` class named `HelloWorldExample`. Go ahead and create a new `Class` file with the following contents:

HelloWorldExample.java:

```
01 package com.javacodegeeks;
02
03 public class HelloWorldExample {
04
05     private String name;
06
07     public String execute() throws Exception {
08         return "success";
09     }
```

```

10
11     public String getName() {
12         return name;
13     }
14
15     public void setName(String name) {
16         this.name = name;
17     }
18 }

```

This Action Class contains just a field called `name`. There's nothing really important about it; just a getter and a setter. Boring, right?. But wait! There is also a method called `execute`. This is very important because this is the one that Struts will call and the one that will define what is the next resource to bring into the game. The name of this method can be anything, but `execute` is a standard one.

In this example we just return the `success` string. What this action will bring next is defined later in the Struts configuration file.

3. Views – JSP Files

In this example we have two views (JSP files). The first one, `index.jsp`, shows a simple HTML form that prompts for a name. The important thing here is that the form's action must match the name we'll later use in our `struts.xml` configuration file. If they don't match, it will never work.

index.jsp:

```

01 <!DOCTYPE html>
02 <html>
03 <head>
04     <title>Hello World Example</title>
05 </head>
06 <body>
07     <h1>Hello World Example</h1>

```

```

08     <form action="hello">

09         <label for="name">Enter your name</label><br/>

10         <input type="text" name="name"/>

11         <input type="submit" value="Submit"/>

12     </form>

13 </body>

14 </html>

```

The second file, `hello.jsp`, will be in charge of displaying a gracious salutation to whomever dares to type a name in the previous HTML form. Pretty simple and to the point.

hello.jsp:

```

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

02 <!DOCTYPE html>

03 <html>

04 <head>

05     <title>Hello World Example</title>

06 </head>

07 <body>

08     Hello <s:property value="name"/>!

09 </body>

10 </html>

```

The HTML response uses a special Struts tag: `s:property`. This tag simply calls the corresponding getter (`getName`) from our `HelloWorldExample` action class.

4. Configuration Files

In a Servlet Container context, Struts plays the role of a `filter`. With this in mind, we can set it up as such in our `web.xml` file. A standard configuration looks like the following:

web.xml:

```

01 <?xml version="1.0" encoding="UTF-8"?>
02
03 <web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
04         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
05         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaeehttp://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"
06         version="3.1">
07
08     <display-name>Struts 2 - Hello World Example</display-name>
09     <welcome-file-list>
10         <welcome-file>index.jsp</welcome-file>
11     </welcome-file-list>
12     <filter>
13         <filter-name>struts2</filter-name>
14         <filter-class>
15             org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter
16         </filter-class>
17     </filter>
18     <filter-mapping>
19         <filter-name>struts2</filter-name>
20         <url-pattern>/*</url-pattern>
21     </filter-mapping>

```

An important part in this file is the `url-pattern` option. It instructs the Servlet Container to execute the given filter – Struts 2 – on any request made to the given url.

The second (and last) part of the configuration is the creation of a file named `struts.xml`. The main goal of this file is to set the relations

between URL's and our **action classes**; this means, what *action class* should Struts call for any requested URL.

It is very, very **important** that you place this file inside your `WEB-INF/classes` directory. Failing to do this will bring you many hours of frustration... I can tell you. Most IDE's do not automatically create this directory, so go ahead and create it. Once created, add a new XML file and name it `struts.xml`. Copy and paste the following configuration into it:

struts.xml:

```
01 <?xml version="1.0" encoding="UTF-8"?>

02

03 <!DOCTYPE struts PUBLIC

04 "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"

05 "http://struts.apache.org/dtds/struts-2.0.dtd
```

Let's check the `action` tag in it. This tag is in charge of matching URL's and actions classes. The `name` attribute defines the URL that Struts will try to match whenever a request is made (remember the form action value we used in `index.jsp`?, well, this is the one). Next we'll find the `class` attribute; this is the class that will be instantiated and executed whenever the value in the `name` attribute is matched. Finally we'll see the `result` sub-element. The given resource in it, in this case `hello.jsp`, will be called whenever `class` returns a `String` with a value of `success`. Now that everything is setup it is time to run it.

5. Running the Hello World Example

At this point your folder structure should look like the following:



Figure 5: Directory Structure

Run your project and open a web browser. Type the following address:

1 <http://localhost:8080/Struts2-HelloWorldExample/>

You should see the contents of the `index.jsp` file when you run the application:



Figure 6: Web Form

Type your name and click `Submit` to receive a nice surprise:



Figure 7: Web Result

Congratulations!

6. Common Problems

Most issues on Struts are related to a wrong configuration file or a missing one. Yes, even if Struts cannot find this (really) important file, you may see an error like the following:

```
1 There is no Action mapped for namespace / and action name hello
```


This can be tricky to tackle, since you might spend countless hours trying to look for errors inside this file, when the real reason is that Struts cannot locate it. To fix this error make sure `struts.xml` resides in your `/WEB-INF/classes` directory. Common editors like Eclipse and Netbeans do not automatically create this folder, so go ahead and create it.