1. Introduction

Contents

[1. Introduction 1](#_Toc32507)

[2. JSP Life Cycle 1](#_Toc31031)

[3. JSP Development Models 1](#_Toc18094)

[4. Course Outline 1](#_Toc15608)

[5. Summary 2](#_Toc17532)

1. Introduction  
   =>slides: Pg. 1

This course consists of 8 modules which provides the complete information required for getting started with Java Server Pages programming for beginners.

=>slides: Pg. 2

This course covers from the fundamentals of JSP, that is from the lifecycle of JSP, basic soft JSP programming, then we shall understand JSP built-in objects, handling exceptions in JSP, standard actions supported by JSP, and slowly we will be graduating to advanced programming in JSP using Expression Language, JSP Standard Tag Library, properly known as JSTL, and also we shall learn how to create custom tags and use them within incorporate session handling in the JSP application. By completing this course you will be in a position to start developing dynamic web applications using JSP. =>slides: Pg. 3

This module can be considered as a prerequisite module for understanding this course where I will be explaining JSP lifecycle and the best development models to be followed while developing JSP application.

=>slides: Pg. 4

This course assumes that you have some fundamental knowledge on servlet programming. Let us get started.

# JSP Life Cycle

=>slides: Pg. 5

Before we start understanding JSP lifecycle, we need to know the answers for the following questions, what is JSP? When we already have servlets, then what does the need of JSP? Once we are clear with the answers for the above two questions, then we can continue with the JSP lifecycle.

=>slides: Pg. 6

Now let us understand what is JSP? Java Server Pages is a technology that helps software developers to create platform independent dynamic web applications rapidly based on HTML or XML document types.

=>slides: Pg. 7

JSP technology separates the user interface from content generation, which enables the designers to change the overall page layout without altering the underlying dynamic content.

=>slides: Pg. 8

When we already have servlets for the web development using Java, then why do we need JSP technology?

=>slides: Pg. 9

So let us understand some of the limitations of servlets. Web application development using servlets requires strong knowledge of Java. Writing the HTML code within the servlet page will be very difficult. Servlet is a picture of both presentation logic written using HTML and business logic written using Java. Whenever we make any changes or modifications to the servlet, it has to be recompiled and then redeployed to see the changes.

=>slides: Pg. 10

Now let us understand what is the advantage that developers will get using JSP. To develop a JSP program, it is not mandatory that the developer should have a strong programming knowledge. Writing the HTML code in JSP program is easy and it is similar to HTML. Modifications done to the JSP is recognized automatically and the changes will be effected immediate. JSP programs are by default thread safe. JSP programs can use standard tags, we can create our own custom tags using tag extension API. As we have reusable tag libraries, the amount of code needed to write powerful web applications will greatly reduces. It is easy to learn, implement, and to maintain JSP pages.

=>slides: Pg. 11

Now let us understand the lifecycle of JSP. Whenever a client sends a request to the JSP file to the server, then the web container translates the JSP page into Java source code for creating servlet class. If any JSP syntax errors are present, then they are caught in this phase, which is called as translation phase, the container tries to compile the servlet Java source into class file. Java language or syntax errors are caught at this phase and this phase is called as compilation phase. Once the Java source file is complied to class file, then the container loads the newly generated servlet class and then instantiates the servlet and calls the servlet's jspInit method to run, and then the container creates a new thread to handle the client's request and the servlet's jspService method runs and eventually the servlet sends a response back to the client or forwards the request to another web app component. When we deploy a web app with the JSP, the whole translation and compilation phase happens only when the JSP pages requested for the first time or the JSP page has been modified. Once it has been translated and compiled, it is just like any other servlet.

=>slides: Pg. 12

Once the servlet has been loaded and initialized, the only thing that happens at request time is creation or allocation of a thread for the service method.

=>slides: Pg. 13

And finally, whenever the container removes the servlet instance from service, it calls the JSP destroy method to perform any required cleanup.

=>slides: Pg. 14

Now let us understand in detail about JSP lifecycle methods. The generated servlet class for JSP file will be implementing HttpJspPage interface, which provides the jspService method and this method corresponds to the body of the JSP page, and this method is defined automatically by the JSP container and as a developer we should never provide the definition for the jspService method explicitly. HttpJspPage interface extends JspPage and the JspPage interface describes the generic interaction that a JspPage implementation class must satisfy, and this interface provides two methods, jspInit and jspDestroy. JspInit method is invoked when the JspPages initialize and the jspDestroy method is invoked when the JspPages about to be destroyed. JspPage interface extends Javax. Servlet. servlet. JspInit and JspDestroy methods can be overridden as per the requirement.

=>slides: Pg. 15

Whenever we use Eclipse and Tomcat server, then the generator servlet class will be extending org. apache. jasper. runtime. HttpJspBase where HttpJspBase class is an abstract class which extends javax. servlet. http. HttpServlet and implements javax. servlet. jsp. HttpJspPage. As we got a fair idea on JSP lifecycle, the next most important point that we need to know will be the JSP development models that we shall understand in the next clip.

# JSP Development Models

=>slides: Pg. 16

Whenever we develop an application, we need to first understand the best practices available for developing the application.

=>slides: Pg. 17

JSP supports two types of development models for developing the application, Model 1 Architecture, Model 2 Architecture.

=>slides: Pg. 18

Now let us understand Model 1 Architecture. In Model 1, a request is made to a JSP page and then that JSP page handles all the responsibilities for the request, including processing the request, validating data, handling the business logic, and generating response.

=>slides: Pg. 19

The advantage of using Model 1 Architecture, it is very easy to development web applications and also developing the applications will be faster, and hence Model 1 Architecture is commonly used in smaller and simple task applications. Although conceptually simple, Model 1 Architecture has many disadvantages. This architecture is not suitable for large scale application development because a great deal of functionality is duplicated in each JSP, and also the Model 1 Architecture unnecessarily ties together the business logic and presentation logic of the application. Combining business logic with presentation logic makes is hard to introduce a new view or access point in an application. There won't be no clear separation of logics as the JSP contains all mixed code. Modifications to one logic may affect the other logic. Every individual page should contain the logic to determine the next page and hence maintenance will be a problem.

=>slides: Pg. 20

Now let us understand Model 2 Development Architecture. Model 2 Architecture follows MVC architecture where M stands for model and V stands for view and C stands for controller. Model hosts the real business logic and the state. In other words, it knows the rules for retrieving and updating the data. It is the only part of the application that talks to the database. Views are responsible for the presentation. It gets the state of the model from the controller, although not directly, but the controller puts the model data in a place where the view can find it. Controllers responsibility is to accept the user input from the request and to figure out what this means to the model. Controller tells the model to update itself and also makes the model state available for the view.

=>slides: Pg. 21

Model to development process can be of two types, MVC 1 Architecture and MVC 2 Architecture.

=>slides: Pg. 22

Now let us understand Model 2 MVC 1 Architecture. In this model, JSP acts like the view and the controller, that is, if a request is made by the client it will be sent to the JSP and then the JSP handles the responsibility such as gathering the data, validating data, session management logic, presentation logic, et cetera. And a separate component as a model layer which handles the business logic, persistence logic, middleware services, et cetera will be used. This model is, again, suitable for developing simple web applications only. Following MVC 1 Architecture, Apache's software organization developed Jarkarta's ProjectsStruts framework. Struts offer many benefits to the web application developer including Model 2 implementation of model view controller design patterns in JSP web applications. Since the release of Struts, a number of competing frameworks have appeared. Many of these frameworks also claim to implement Model 2 and MVC. So the two terms have become synonymous in the minds of developers. This has lead to the use of the term MVC Model 2 or MVC 2 for short.

=>slides: Pg. 23

Now let us understand MVC 2 architecture in detail. The model portion of an MVC based application typically uses JavaBean classes to define the internal state of the system. They also specify the actions that can be taken to change that state. The view portion is constructed using JSP technology. JSP pages can contain static HTML or XML text, which is also called as template text, and also the view has the ability to insert dynamic content based on the interpretation at the page request time of special action packs. The JSP environment also includes a set of custom JSP tag libraries, standard JSP action tags, and a facility to install our own JSP custom tag libraries. The controller portion of the application is focused on receiving requests from the client, deciding what business logic function is to be performed, and delegating responsibility for producing the next phase of the user interface to an appropriate view component is handled by the controller. Whenever the client sends a request, it is first intercepted by a servlet, commonly referred as a controller servlet. This servlet handles the initial processing of the request and determines which JSP page to display next. Client can never send a request directly to JSP page in the MVC Model 2 Architecture. This allows the servlet to perform front end processing including authentication and authorization, centralized logging and help with internationalization, interacting with the model for the data. Once the request processing has been completed, the servlet directs the request to the appropriate JSP page, how the next page is determine varies widely across different applications. For example, in simple application the next JSP page to display may be hard coded in the servlet, based on the request parameters and current application state. And for the large web applications, a workflow or rules engine can also be used.

=>slides: Pg. 24

We have many advantages in using MVC 2 Architecture. Navigation control is now centralized, that is, only controllers contains the logic to determine the next page. It's supports the industry standard for developing the web applications. It is easy to maintain and also it is easy to extend. MVC 2 supports better separation of concerns. That is, modifications to one logic will not affect the other logics and also parallel development will be possible. But one small limitation we have is if we change the controller code, we need to recompile the servlet class and we need to redeploy it. These days Model 1 Architecture is not used in development. Developers usually use MVC 1 for the small applications or MVC 2 for medium and large applications development, and some developers even consider MVC 1 as Model 1 and MVC 2 as Model 2 development models.

# Course Outline

=>slides: Pg. 25

Let us understand what we will learn in this course.

=>slides: Pg. 26

In the first module we have understood the JSP lifecycle and the development models to be followed for developing an efficient JSP page.

=>slides: Pg. 27

In the second module of our course, we shall understand the scripting elements which can be considered as the basic building blocks for a JSP page with proper explanations and demos, then we shall also understand practically how the code gets generated within the servlet page. The first and second modules will provide a fair understanding of JSP fundamentals.

=>slides: Pg. 28

Then for learning JSP programming in an efficient manner, it is always advisable to learn along with coding, so from the third module of this course we shall start developing a web application using JSP and by the time we complete all the modules of this course we would have completed the entire web application. In the third module we shall learn JSP directives and we shall complete the home page of our application.

=>slides: Pg. 29

And then in the fourth module we shall learn JSP built-in objects and extend our application to support the login and registration forms using the JSP built-in objects.

=>slides: Pg. 30

And then in the fifth module, we shall learn how to handle the exceptions in JSP and we shall extend our application to provide the support of custom error pages

=>slides: Pg. 31

and then we shall learn how to work with JSP actions in our sixth module

=>slides: Pg. 32

and extend our application to provide the functionality to search for an individual course and to display that course.

=>slides: Pg. 33

In the seventh module we shall learn Expression Language and use that language to update the code to simplify the JSP development, and in the eighth module we shall learn how to encapsulate the core functionality using JSP Standard Tag Library and we shall extend our application

=>slides: Pg. 34

to display all the courses with the support of JSTL and Expression Language.

=>slides: Pg. 35

And finally in the ninth eighth module, we shall learn how to add session handling to our web application.create our own custom tags and enhance our application by creating a custom tag to generate the rating for the course, and use that rating in our course base to display the rating. A simple application to explore the concept of JSP in detail. Follow step by step, by the end of this course you will certainly feel comfortable with JSP development. Before jumping into the application development, in each module I will be providing a detailed explanation of the concept to understand better.

# Summary

=>slides: Pg. 36

In this module we have understood what does JSP, lifecycle of JSP, and development models supported by JSP. Knowing these details properly will help the developers to understand the JSP programming in better way. Hope you have got a fair idea on how JSP works. In the next module we shall understand JSP fundamentals by getting our hands dirty by writing some code and understand the basic fundamentals of JSP scriptlets, expressions, and declarations.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*