Developing Dynamic Web Pages Using JSP

You Already Know

Course(s):

Java Certification Training Course

Recap

- Explain JSP life cycle
  - JSP life cycle

- Create JSP elements
  - JSP declaration, scriptlet, and expression element

- Demonstrate JSP standard actions
  - JSP standard actions

- Use JSTL and custom tag libraries
  - JSTL
  - Custom tag libraries

A Day in the Life of a Full Stack Developer

After observing Joe's incredible performance in the last sprint, the entire e-commerce website project is handed over to Joe.

In this sprint, he has to work on the product details of the website. Joe has to create a servlet-based functionality where the owner of the website can add the product details and display them.

In this lesson, we will learn how to solve this real-world scenario to help Joe complete his task effectively and quickly.

Learning Objectives

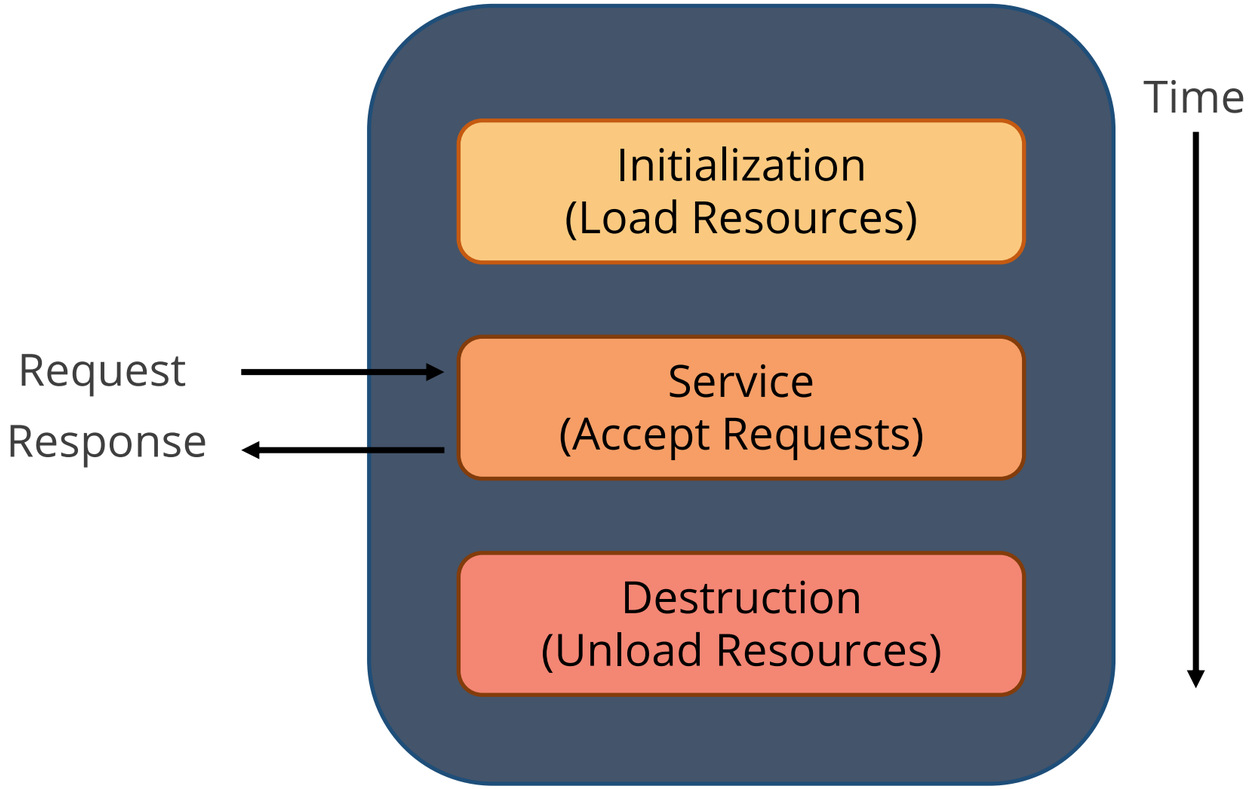
By the end of this lesson, you will be able to:

- Describe JSP in brief
- Create simple JSP pages by configuring JSP with Eclipse
- Explain JSP Scripting Elements
- Demonstrate JSP Implicit Objects
- Illustrate working of JSP Directives
- Utilize JSP action tags
- Illustrate session handling

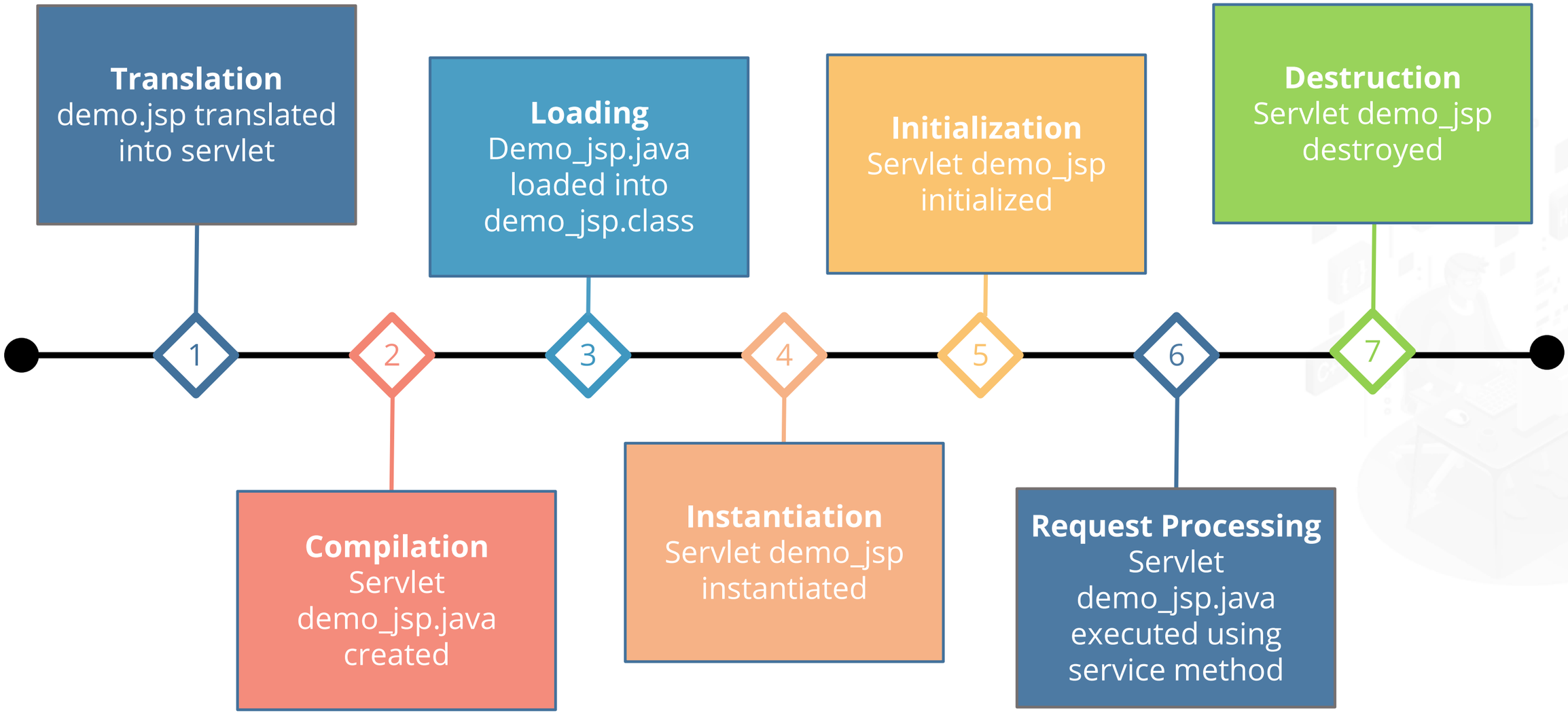
An Overview of JSP

JSP Life Cycle

A JSP life cycle is the translation of a JSP page into a servlet. The life cycle begins with the creation of a JSP and ends with its disintegration. A service request gets processed only after a JSP page gets converted into a servlet.



Steps in a JSP Life Cycle



Example of JSP Life Cycle

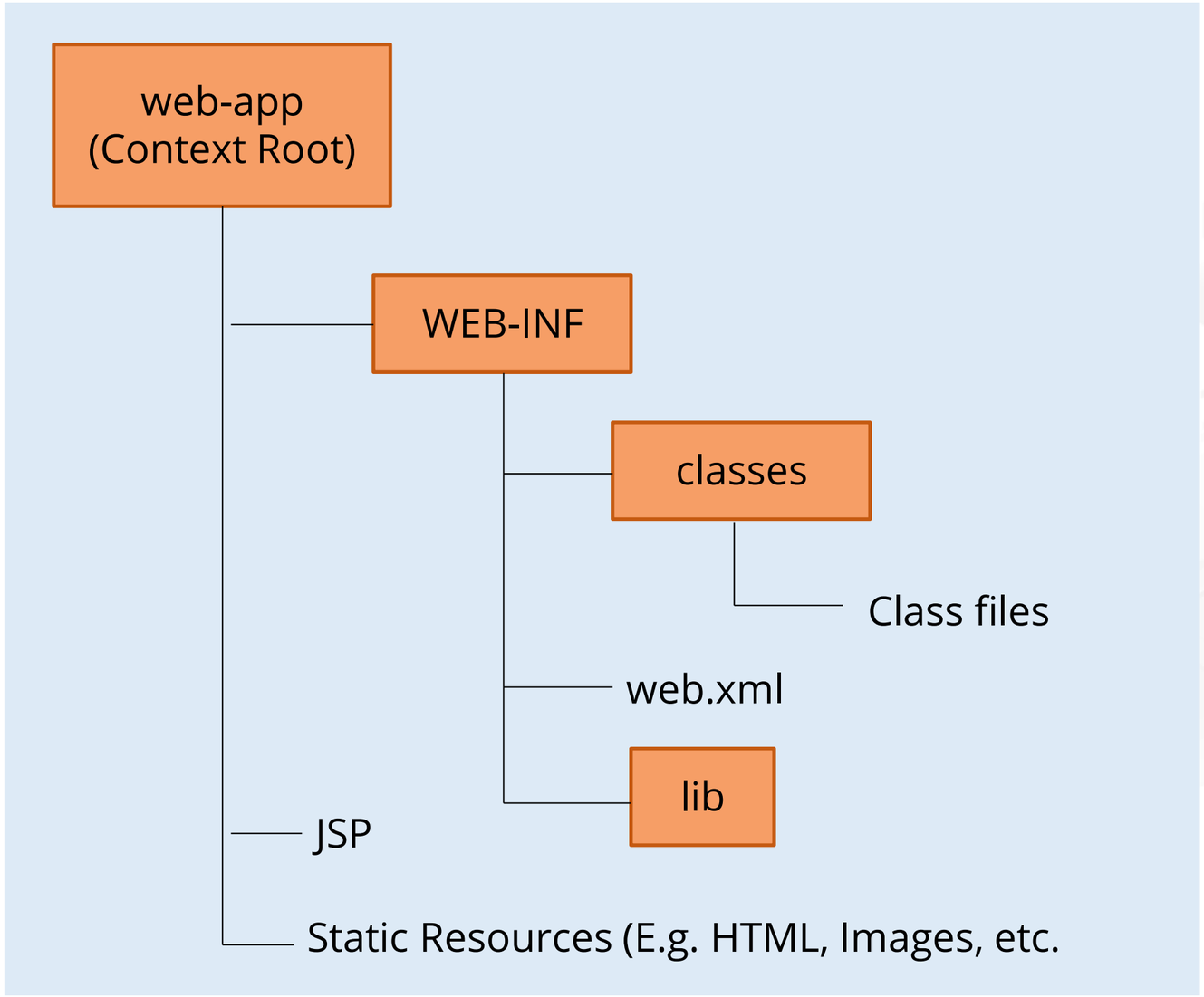
```
public void jspinit()
{
      //initializing the code
}

void _jspservice(HttpServletRequest request HttpServletResponse response)
{
      //handling all requests and responses
}

public void _jspdestroy()
{
            //Clean up code
}
```

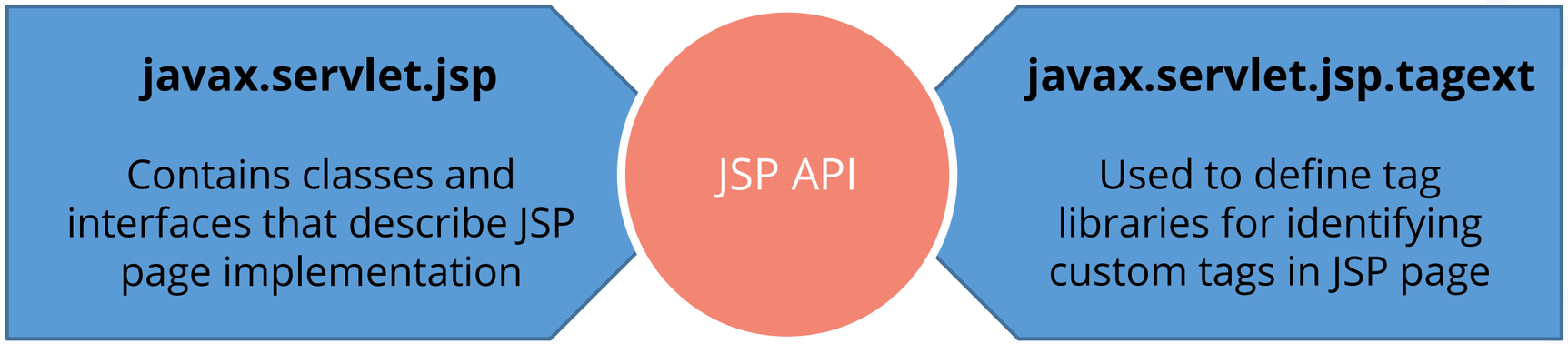
Directory Structure of JSP

The directory structure of the JSP page is the same as that of a servlet. The JSP page is placed outside the WEB-INF folder or in any directory.

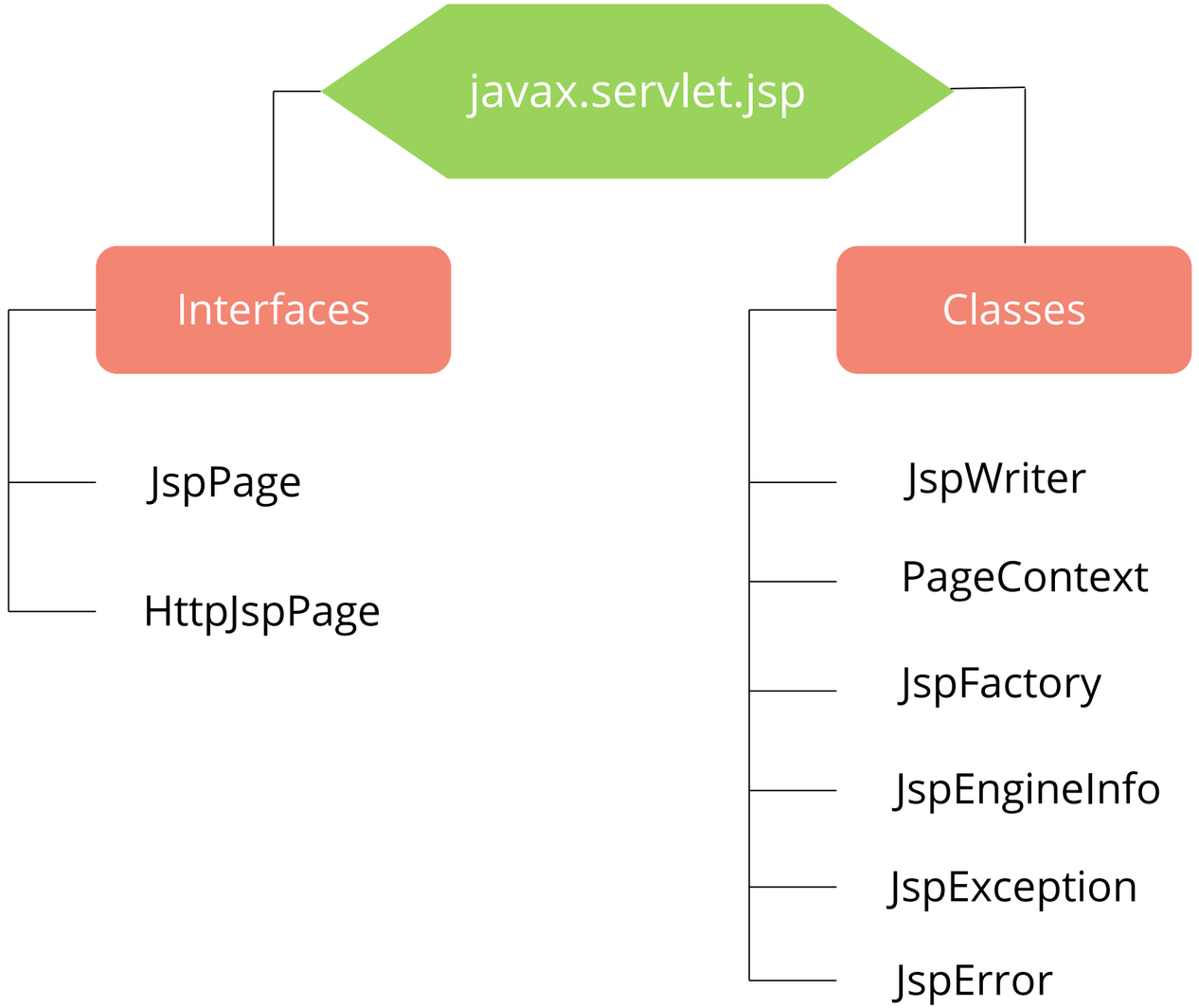


JSP API

API stands for **Application Programming Interface.** It allows programmers to build software applications in Java.

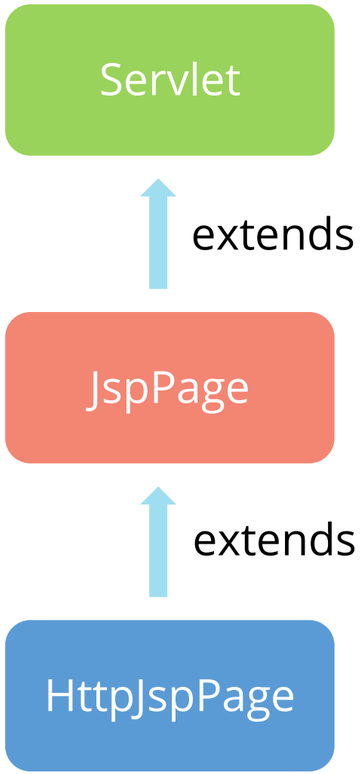


javax.servlet.jsp Package



JspPage Interface

All generated servlet classes implement the JspPage interface. It extends the Servlet interface.



Methods

public void jspInit()

Used to perform initialization of the JSP and is used only once

public void jspDestroy()

Invoked only once when the JSP is destroyed

HttpJspPage Interface

It extends the JspPage interface and provides one life cycle method.

public void jspServie()
Invoked each time when a request for a JSP comes to the container

Creating a Simple JSP Page

Duration: 30 min.

Problem Statement:

Write a program to configure JSP with Eclipse and create a simple JSP page.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to create a simple JSP page:

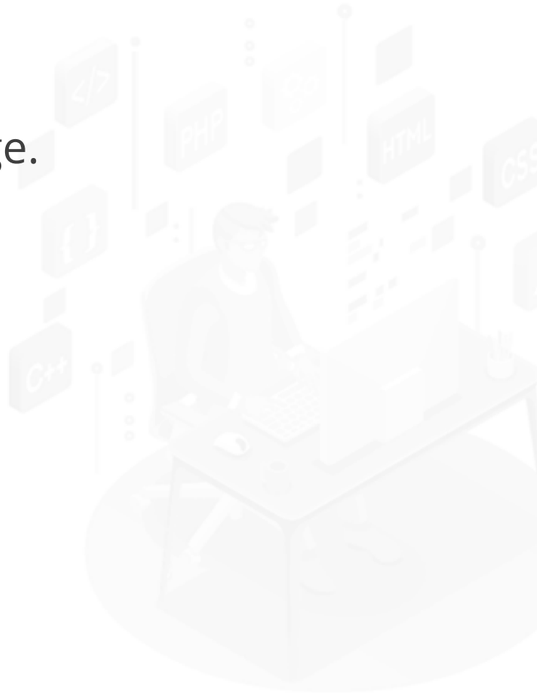
1. Create a Java project in your IDE.

2. Write a program in Java to configure JSP with Eclipse and create a simple JSP page.

3. Initialize the .git file.

4. Add and commit the program files.

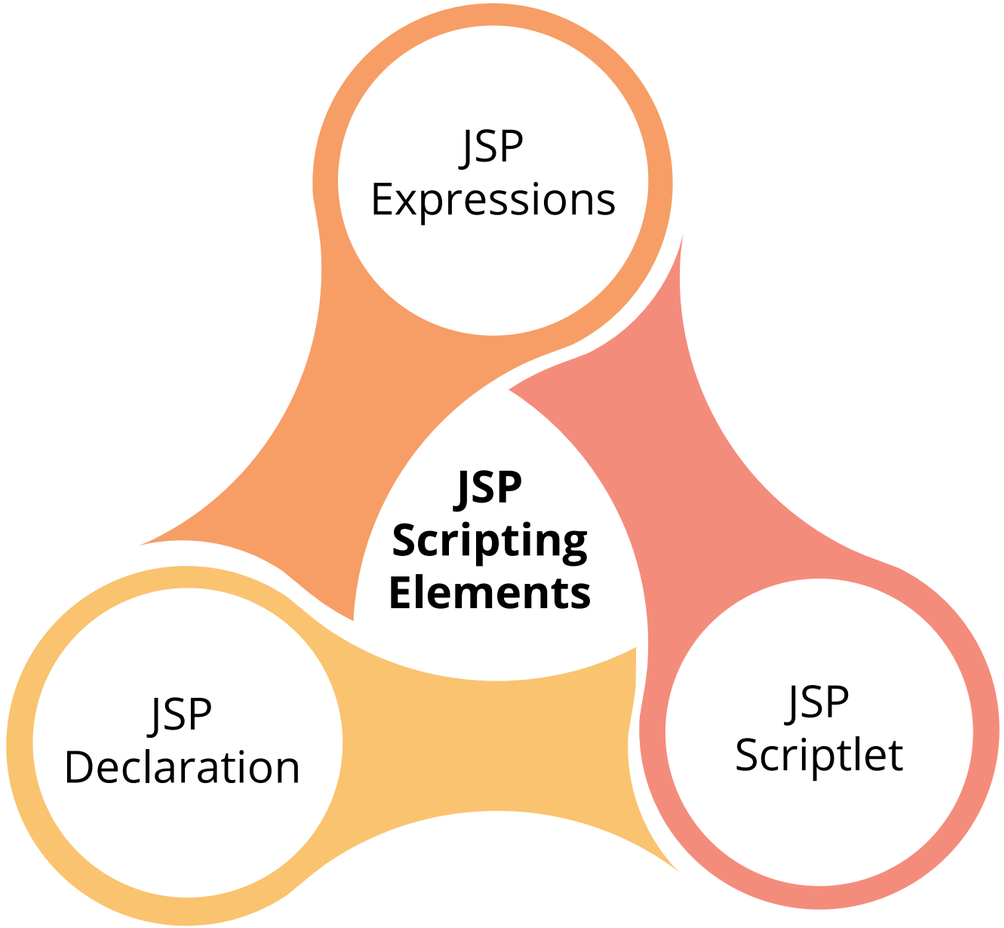
5. Push the code to your GitHub repository.



Scripting Elements or JSP Tags

JSP Scripting Elements

JSP Scripting Elements enable you to insert a Java code into the servlet. This code is generated from the current JSP.



JSP Scriptlet

A Scriptlet tag is used to execute a Java source code in JSP.

Syntax → <% java source code %>

Example to display a welcome message →

```
<html>
<body>
<% out.print("welcome to jsp"); %>
</body>
</html>
```

JSP Expressions

The code within JSP Expression tag is written to the output stream of the response.

Syntax → **<%= statement %>**

Example →

```
<html>
<body>
<%= java.util.Calendar.getInstance().getTime()%>
</body>
</html>
```

JSP Declaration

The JSP Declaration tag is used to declare fields and methods.

Syntax → <%! field or method declaration %>

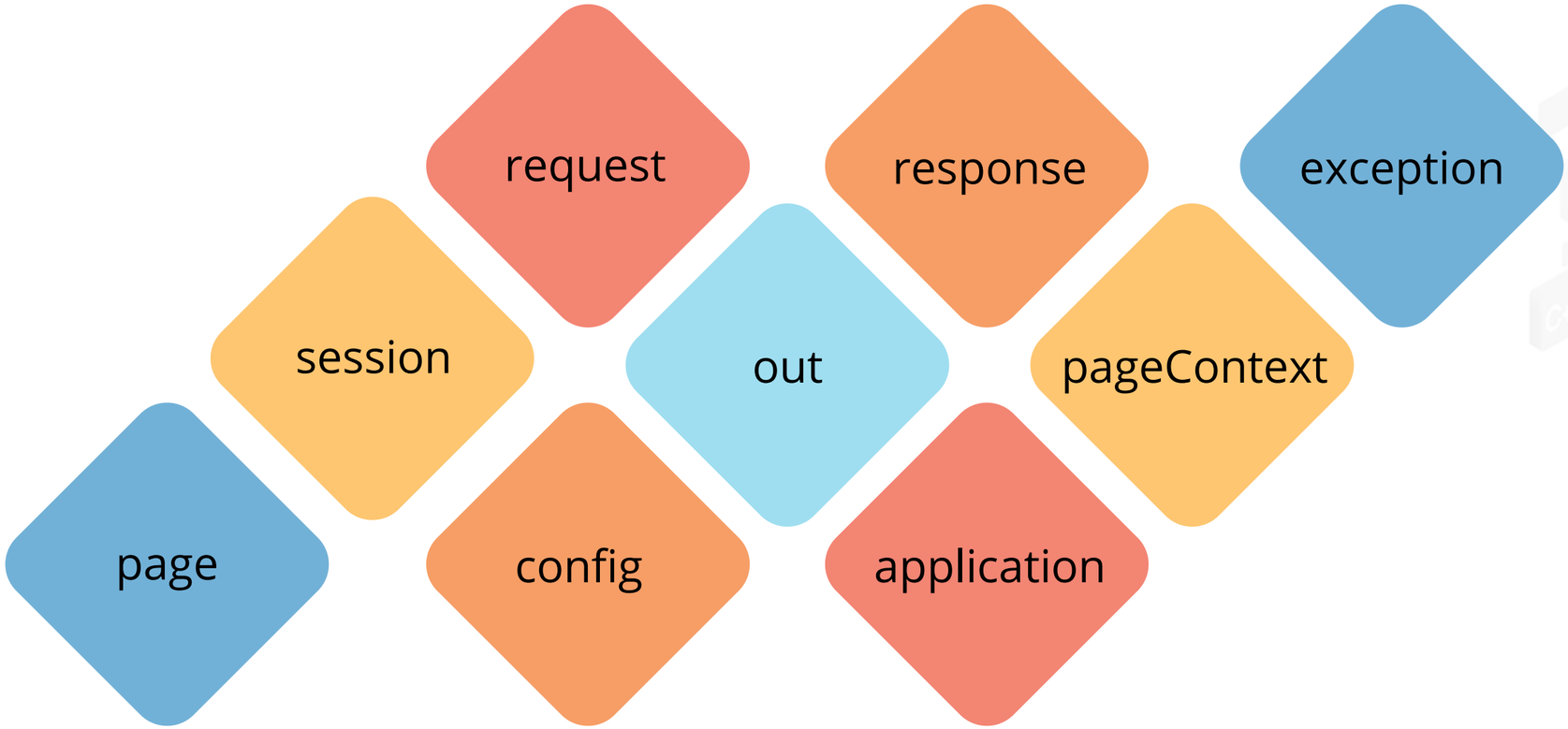
Example to declare a field

```
<html>
<body>
<%! int data=100; %>
<%= "Value is:"+data %>
</body>
</html>
```

JSP Implicit Objects

JSP Implicit Objects

JSP Implicit Objects are automatically created by the container when the JSP is translated to a servlet. These objects can be accessed directly in scriptlets used in the service method.



JSP Implicit Objects

JSP Implicit Object	Corresponding Classes	Description
out	javax.servlet.jsp.JspWriter	Used to access and print to the servlet's output stream
request	javax.servlet.http.HttpServletRequest	Used to get request information such as parameter, header information, remote address, server name, server port, content type, and character encoding
response	javax.servlet.http.HttpServletResponse	Used to add or manipulate responses such as redirect response to another resource, and send error
config	javax.servlet.ServletConfig	Used to get the initialization parameter in web.xml
application	javax.servlet.ServletContext	Used to get the context information and attributes in JSP
page	java.lang.Object	Used to get the currently executed servlet object for the corresponding JSP
exception	javax.servlet.jsp.JspException	Used to generate appropriate responses to error conditions
pageContext	javax.servlet.jsp.PageContext	Used to set, get, or remove attribute from one of the scopes – page, request, session, or application
session	javax.servlet.http.HttpSession	Used to set, get, or remove attribute or to get session information

JSP Implicit Objects

Duration: 90 min.

Problem Statement:

Write a program to demonstrate the function of JSP Implicit Objects.

ASSISTED PRACTICE

Assisted Practice: Guidelines

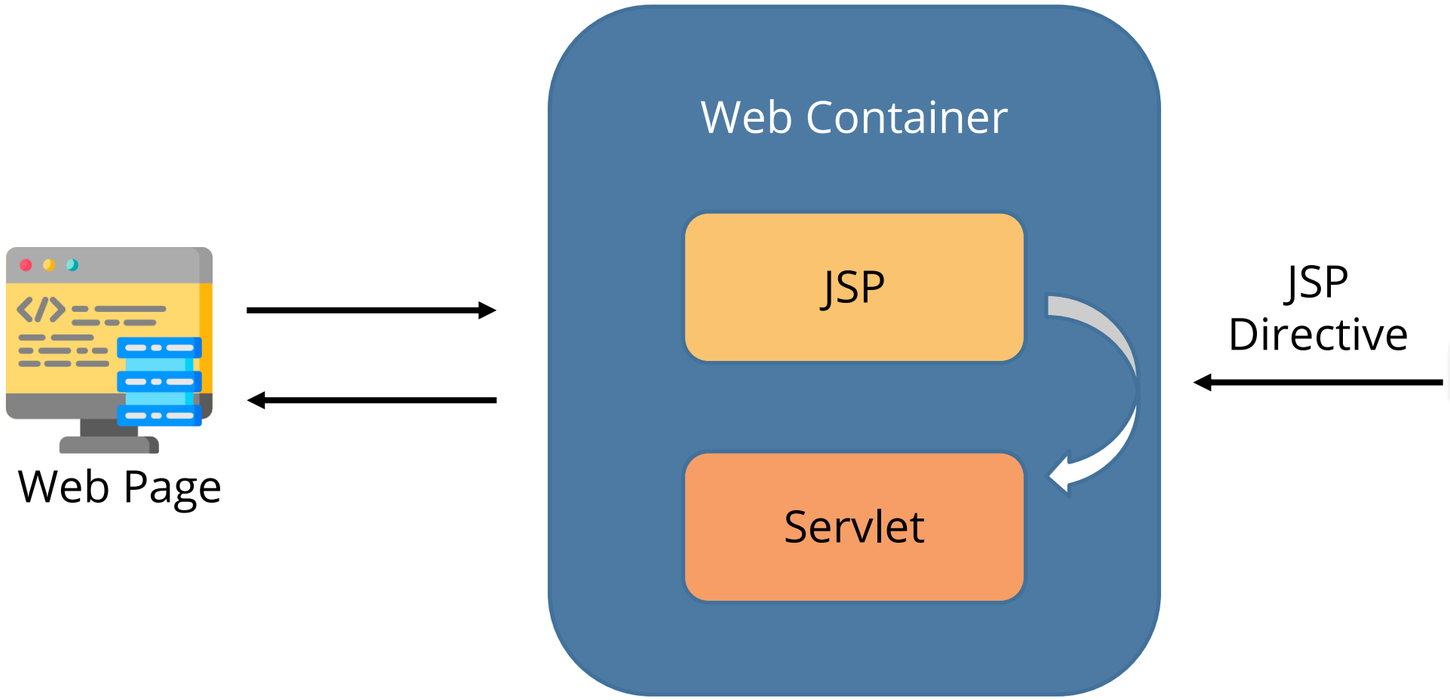
Steps to demonstrate JSP implicit objects:

1. Create a Java project in your IDE.
2. Write a program in Java to demonstrate the function of JSP Implicit Objects.
3. Initialize the .git file.
4. Add and commit the program files.
5. Push the code to your GitHub repository.

JSP Directives

JSP Directives

JSP Directives are messages that instruct the web container on how to translate a JSP into the corresponding servlet.



Directives and Descriptions

Syntax	`<%@ directive attribute = "value" %>`
<%@ page ... %>	Defines page-dependent attributes like scripting language, error page, and buffering requirements
<%@ include ... %>	Includes a file during the translation phase
<%@ taglib ... %>	Declares a tag library containing custom actions that are used in the page

Attributes of JSP Page Directive

The JSP Page Directive is used to define attributes that apply to an entire JSP.

Syntax

```
<%@ page attribute = "value" %>
```

Attribute	Description
buffer	Specifies the buffering characteristics for the server output response object
autoFlush	Controls the behavior of the servlet output buffer
contentType	Defines the character encoding for the JSP and the generated response page
errorPage	Specifies the page to display in case of an error condition
isErrorPage	Indicates that the current JSP is to be used as the error page for another JSP. Value is either TRUE or FALSE; default value being FALSE
extends	Specifies a superclass that the generated servlet should extend
import	Specifies the name of the package to be imported
info	Provides a description of the JSP

Attributes of JSP Page Directive

Attribute	Description
isThreadSafe	Marks a page as thread-safe. If the ThreadSafe option is set to false, the JSP engine ensures that only one thread at a time executes the JSP
language	Indicates the programming language used in scripting the JSP page
session	Indicates whether the JSP page uses HTTP sessions
isElIgnored	Provides option to disable the evaluation of Expression Language (EL) expressions
isScriptingEnabled	Determines if the scripting elements can be used

JSP Directives

Duration: 45 min.

Problem Statement:

Write a program to demonstrate the use of JSP Directives.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to demonstrate JSP directives:

1. Create a Java project in your IDE.
2. Write a program in Java to demonstrate the use of JSP Directives.
3. Initialize the .git file.
4. Add and commit the program files.
5. Push the code to your GitHub repository.

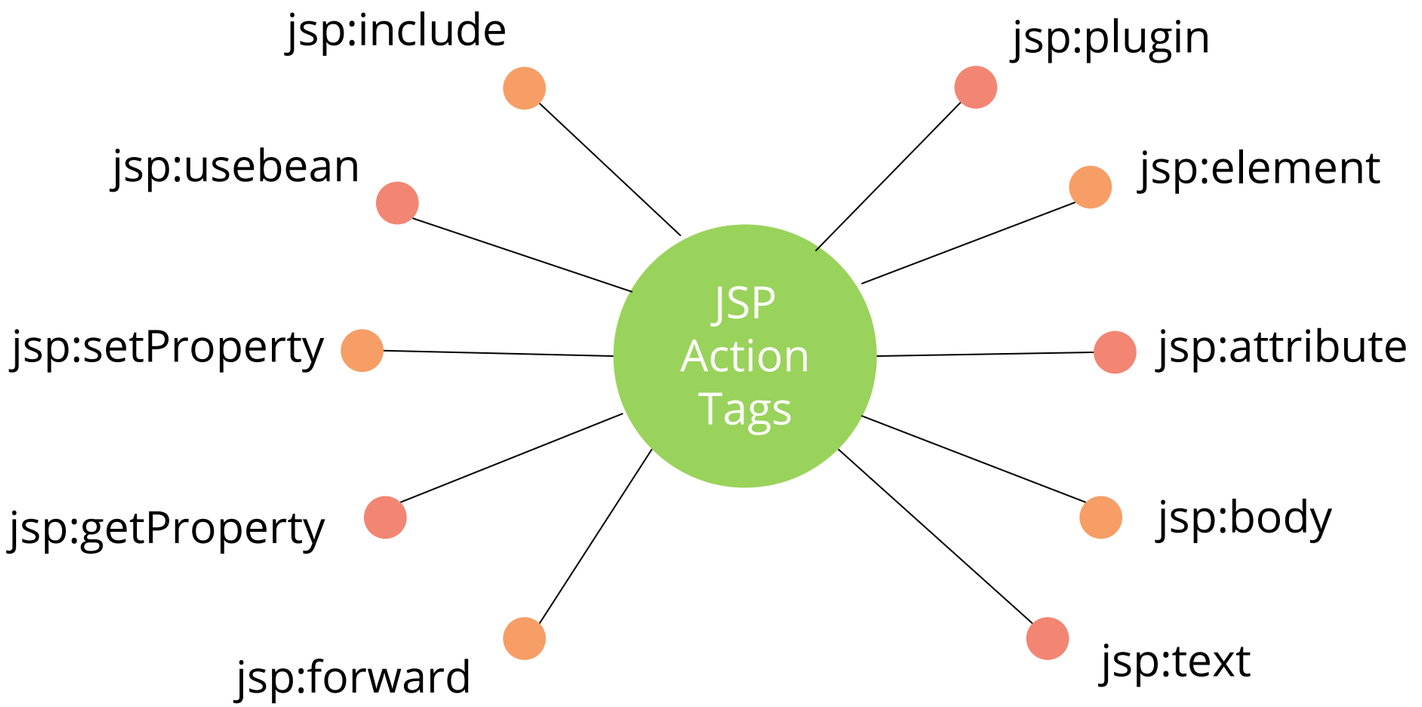
JSP Action Tags

JSP Action Tags

JSP action tags are used to control the flow between pages. JSP action tags are written as constructs in XML syntax.

Syntax

```
<jsp:action_name attribute = "value" />
```



JSP Action Tags: Common Attributes

Id Attribute

Uniquely identifies the action element

Scope Attribute

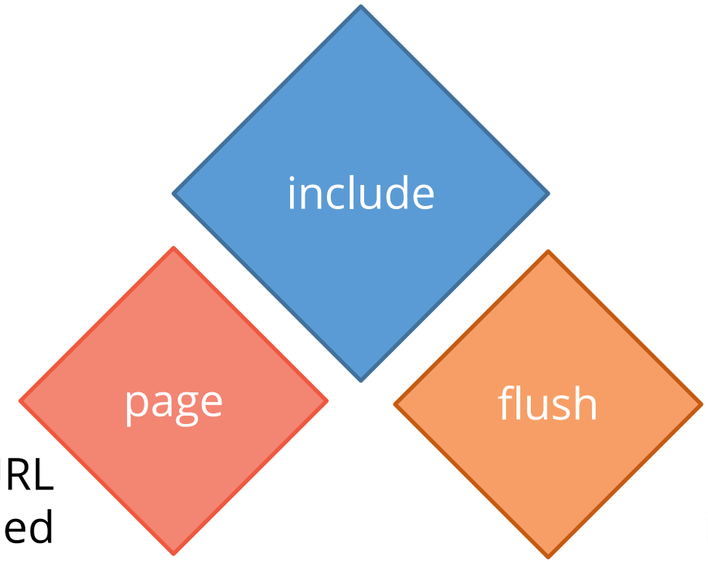
Identifies life cycle of the attribute

jsp:include

jsp:include inserts the specified file into the created JSP

Syntax

```
<jsp:include page = "relative URL" flush = "true" />
```



Identifies the relative URL of the page to be included

Determines if the included resource has its buffer flushed before it is included

jsp:useBean

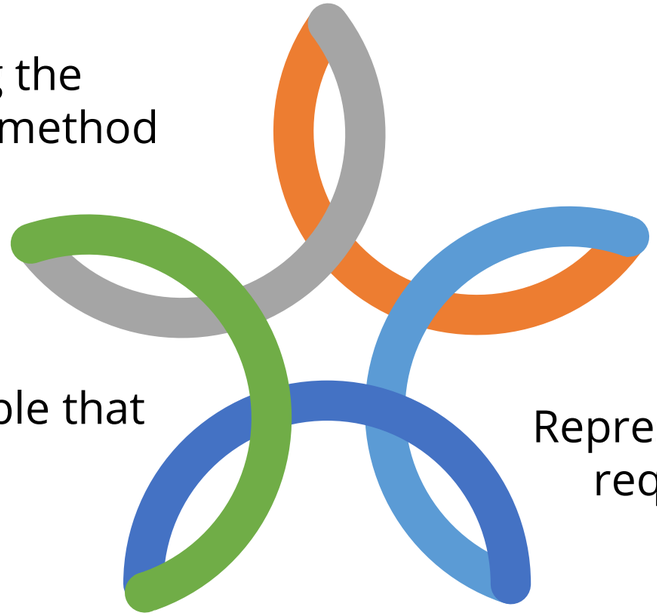
jsp:useBean is used to locate or instantiate a bean class

Syntax

<jsp:useBean id= “ " scope= " “ c**lass**= “ " type= “ “ beanName=“ " >

beanName
Instantiates the bean using the java.beans.Beans.instantiate() method

id
Identifies the bean in the specified scope



type
Specifies the type of the variable that will refer to the object

scope
Represents scope of the bean - page, request, session, or application

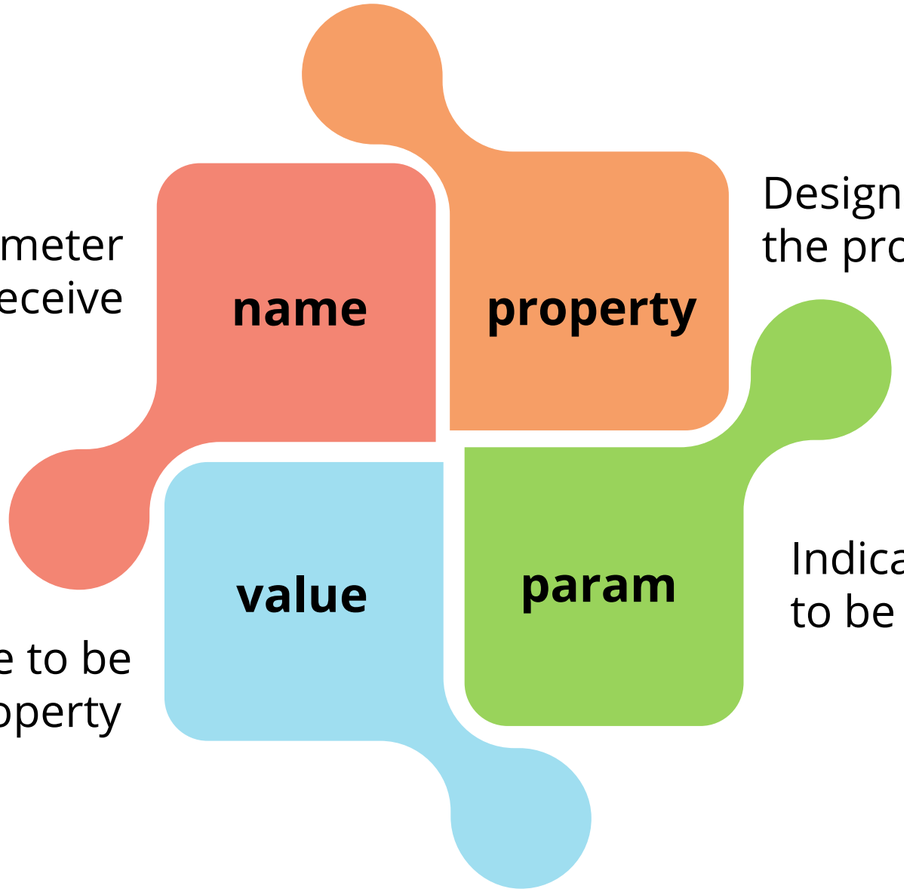
class
Creates an object of the bean class

jsp:setProperty

jsp:setProperty is used to set a property value or values in a bean

Syntax

<jsp:setProperty name = “ " property = “ " value = “ “ param = “ “/>



Sets the name of the request parameter whose value the property will receive

Designates the bean for which the property is to be set

Indicates the value to be assigned to a property

Indicates the property to be set

jsp:getProperty

jsp:getProperty is used to retrieve the value of a property

Syntax

```
<jsp:getProperty name="instanceOfBean" property="propertyName" />
```

Sets the name of the bean whose property needs to be retrieved

name

property

Indicates the name of the bean property to be retrieved

jsp:forward

jsp:forward is used to terminate the action of the current page and forward the request to another static page or a Java servlet

Syntax

<jsp:forward page = "Relative URL" />

Example

```
<jsp:forward page="includetest.jsp" />
```

jsp:plugin

jsp:plugin is used to insert Java components into a JSP

Example

```
<jsp:plugin type="applet" code="MyApplet.class" codebase="applet"
height="300" width="300">    <jsp:params>  <jsp:param name="productId"
value="19337" />    </jsp:params>
<jsp:fallback> Plugins are not supported in this browser.
</jsp:fallback></jsp:plugin>
```

jsp:text

jsp:text is used to write the template text in JSP pages and documents

Syntax

<jsp:text>Template data</jsp:text>

Example

```
<jsp:text>   <![CDATA[</ending>]]></jsp:text>
```

jsp:element, jsp:attribute, and jsp:body

<jsp:element>, <jsp:attribute>, and <jsp:body> actions are used to define XML elements dynamically

Example

```
<jsp:element name="a">  <jsp:attribute name="href">
<c:url value="https://www.microsoft.com" />
</jsp:attribute></jsp:element><jsp:plugin type="applet" code="MyApplet.class"
codebase="applet" >
<jsp:attribute name="width">160</jsp:attribute>
<jsp:attribute name="height">150</jsp:attribute>
<jsp:body>    <jsp:params>      <jsp:param name="productId" value="123413" />
</jsp:params>    <jsp:fallback>      Plugin tag not supported by browser.
</jsp:fallback>  </jsp:body></jsp:plugin>
```

JSP Action Tags

Duration: 75 min.

Problem Statement:

Write a program to demonstrate the use of JSP action tags.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to demonstrate JSP action tags:

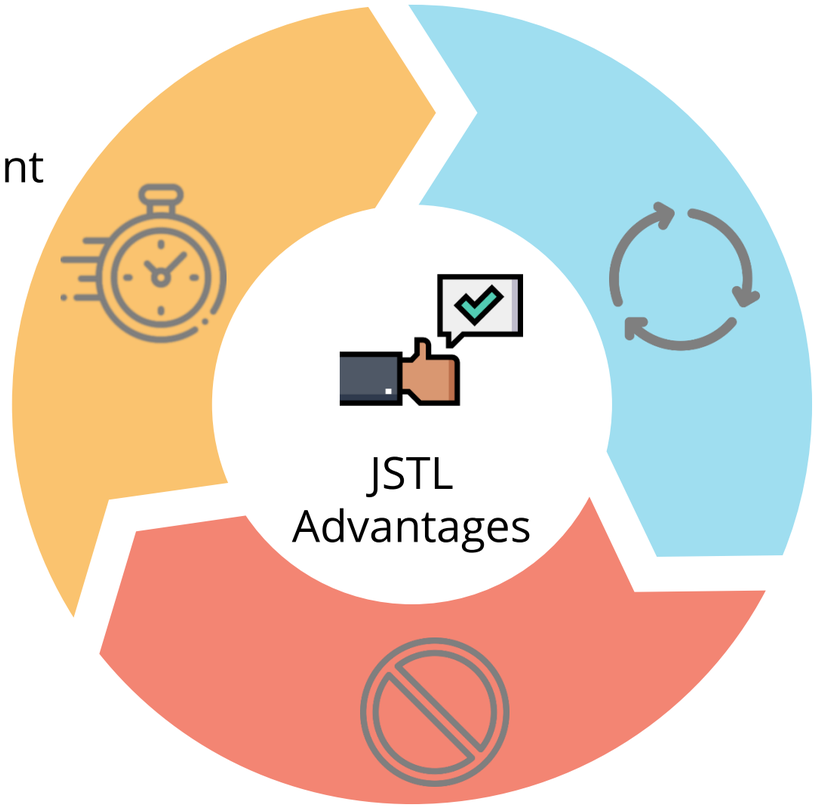
1. Create a Java project in your IDE.
2. Write a program in Java to demonstrate the use of JSP action tags.
3. Initialize the .git file.
4. Add and commit the program files.
5. Push the code to your GitHub repository.

FULL STACK

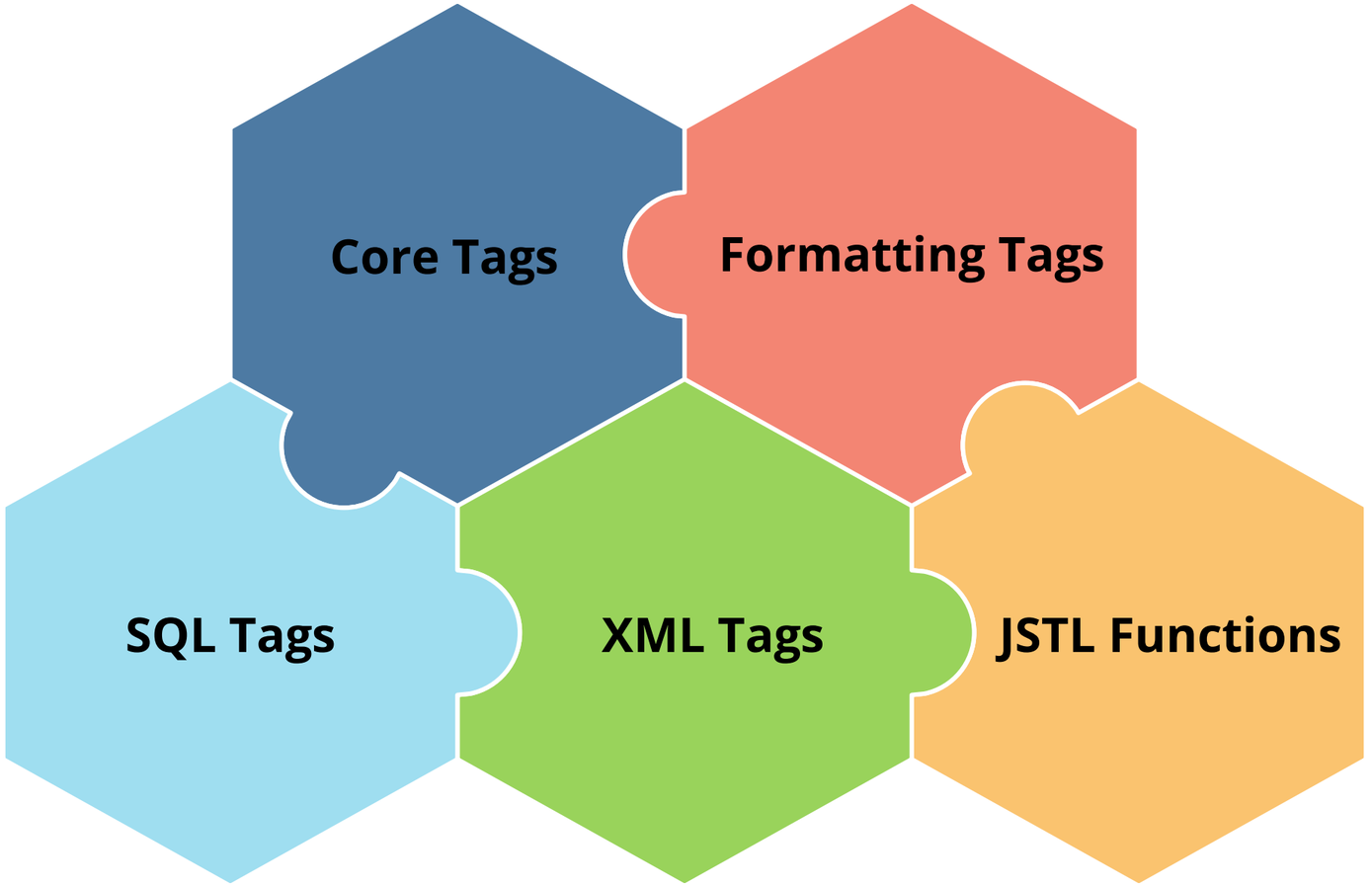
JSTL

JSTL

The JSP Standard Tag Library (JSTL) represents a set of tags to simplify the JSP development.



JSTL Classification



Core Tags

Core tags provide variable support, URL management, and flow control.

Syntax

```
<%@ taglib prefix = "c" uri = "http://java.sun.com/jsp/jstl/core" %>
```

Tag	Description
<c:out>	Displays the result of an expression
<c:set>	Sets the result of an expression evaluation in a **scope**
<c:remove>	Removes a scoped variable from a scope
<c:catch>	Catches any **Throwable** occurring in the body
<c:if>	Evaluates if the supplied condition is true
<c:choose>	Establishes context for mutually exclusive conditional operations
<c:when>	Subtag of <choose> that includes its body if its condition evaluates to be **true**

Core Tags

Tag	Description
<c:otherwise>	Subtag of <choose> that follows the <when> tag and runs only if the prior conditions evaluate to **false**
<c:import>	Retrieves an absolute or relative URL
<c:forEach>	Iteration tag, accepting many collection types and supporting subsetting and other functionality
<c:forTokens>	Iterates over tokens, separated by the supplied delimiters
<c:param>	Adds a parameter to a containing **import** tag's URL
<c:redirect>	Redirects to a new URL
<c:url>	Creates a URL with optional query parameters

Formatting Tags

Formatting tags are used to format and display text, the date, the time, and the numbers for internationalized websites.

Syntax

```
<%@ taglib prefix = "fmt" uri = "http://java.sun.com/jsp/jstl/fmt" %>
```

Tag	Description
<fmt:formatNumber>	Renders numerical value with specific precision or format
<fmt:parseNumber>	Parses the string representation of a number, currency, or percentage
<fmt:formatDate>	Formats a date and/or time using the supplied styles and pattern
<fmt:parseDate>	Parses the string representation of a date and/or time
<fmt:bundle>	Loads a resource bundle to be used by its tag body
<fmt:setLocale>	Stores the given locale in the locale configuration variable
<fmt:setBundle>	Loads a resource bundle and stores it in the named scoped variable or the bundle configuration variable

Formatting Tags

Tag	Description
<fmt:timeZone>	Specifies the time zone for time formatting or parsing actions nested in its body
<fmt:setTimeZone>	Stores the given time zone in the time zone configuration variable
<fmt:message>	Displays a message
<fmt:requestEncoding>	Sets the request character encoding

SQL Tags

SQL tags are used for interacting with relational databases (RDBMSs) such as Oracle, mySQL, or Microsoft SQL Server.

Syntax

```
<%@ taglib prefix = "sql" uri = "http://java.sun.com/jsp/jstl/sql" %>
```

Tag	Description
<sql:setDataSource>	Creates a simple DataSource suitable for prototyping
<sql:query>	Executes the SQL query defined in its body or through the SQL attribute
<sql:update>	Executes the SQL update defined in its body or through the SQL attribute
<sql:param>	Sets a parameter in an SQL statement to the specified value
<sql:dateParam>	Sets a parameter in an SQL statement to the specified java.util.Date value
<sql:transaction >	Provides nested database action elements with a shared connection, set up to execute all statements as one transaction

XML Tags

The XML tags provide flow control, transformation, and other features.

Syntax

```
<%@ taglib prefix = "x" uri = "http://java.sun.com/jsp/jstl/xml" %>
```

Tag	Description
<x:out>	Displays the result of an XPath expression
<x:parse>	Used to parse the XML data specified either via an attribute or in the tag body
<x:set >	Sets a variable to the value of an XPath expression
<x:if >	Evaluates a test XPath expression and if it is true, it processes its body; if the test condition is false, the body is ignored
<x:forEach>	To loop over nodes in an XML document

XML Tags

Tag	Description
<x:choose>	Establishes a context for mutually exclusive conditional operations
<x:when >	Subtag of <choose> that includes its body if its expression evaluates to **true**
<x:otherwise >	Subtag of <choose> that follows the <when> tags and runs only if the prior condition evaluates to **false**
<x:transform >	Applies an XSL transformation on a XML document
<x:param >	Used along with the transform tag to set a parameter in the XSLT stylesheet

JSTL Functions

JSTL includes standard functions, most of which are common string manipulation functions.

Syntax

```
<%@ taglib prefix = "fn" uri = "http://java.sun.com/jsp/jstl/functions" %>
```

Tag	Description
fn:contains()	Tests if an input string contains a specified substring
fn:containsIgnoreCase()	Tests if an input string contains a specified substring in a case insensitive way
fn:endsWith()	Tests if an input string ends with a specified suffix
fn:escapeXml()	Escapes characters that can be interpreted as XML markup
fn:indexOf()	Returns the index within a string of the first occurrence of a specified substring
fn:join()	Joins all elements of an array into a string
fn:length()	Returns the number of items in a collection, or the number of characters in a string

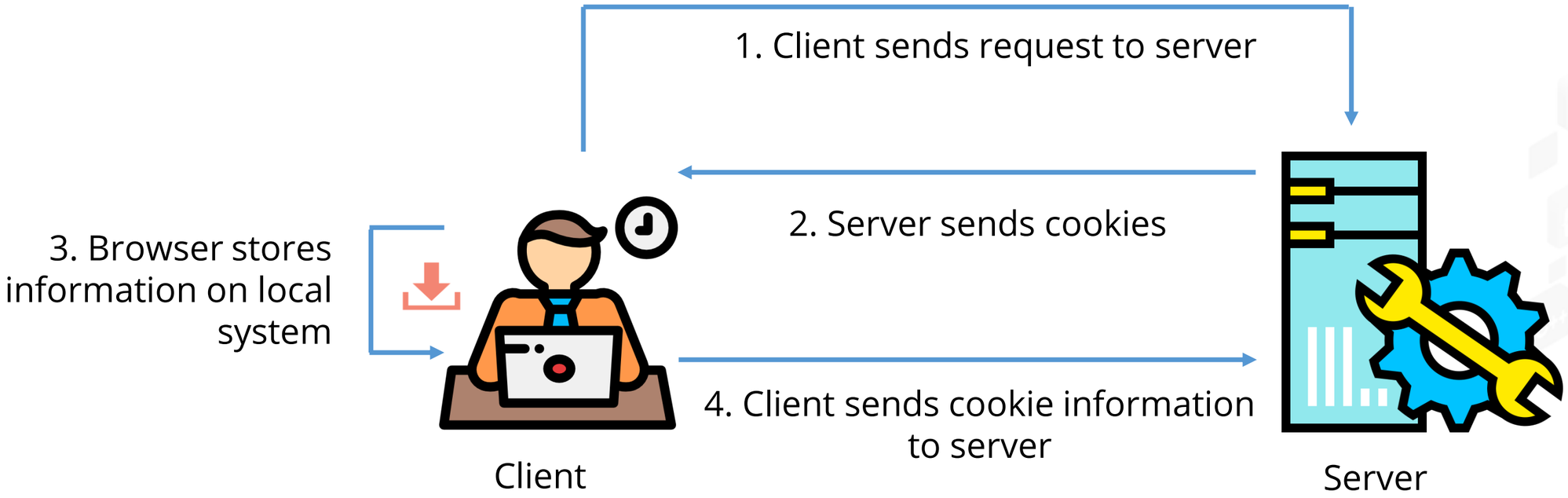
JSTL Functions

Tag	Description
fn:replace()	Returns a string resulting from replacing in an input string all occurrences with a given string
fn:split()	Splits a string into an array of substrings
fn:startsWith()	Tests if an input string starts with a specified suffix
fn:substring()	Returns a subset of a string
fn:substringAfter()	Returns a subset of a string following a specific substring
fn:substringBefore()	Returns a subset of a string before a specific substring
fn:toLowerCase()	Converts all the characters of a string to lowercase
fn:toUpperCase()	Converts all the characters of a string to uppercase
fn:trim()	Removes white spaces from both ends of a string

Session Handling

Cookies

Cookies are text files stored on the client system and kept for tracking purposes.



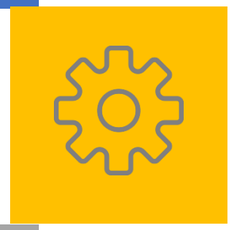
Accessing Cookies

Create Cookies

new Cookie("key","value");

Set Maximum Age

cookie.setMaxAge(60*60*24);



Send Cookies to HTTP Response Headers

response.addCookie(cookie);

Read Cookies

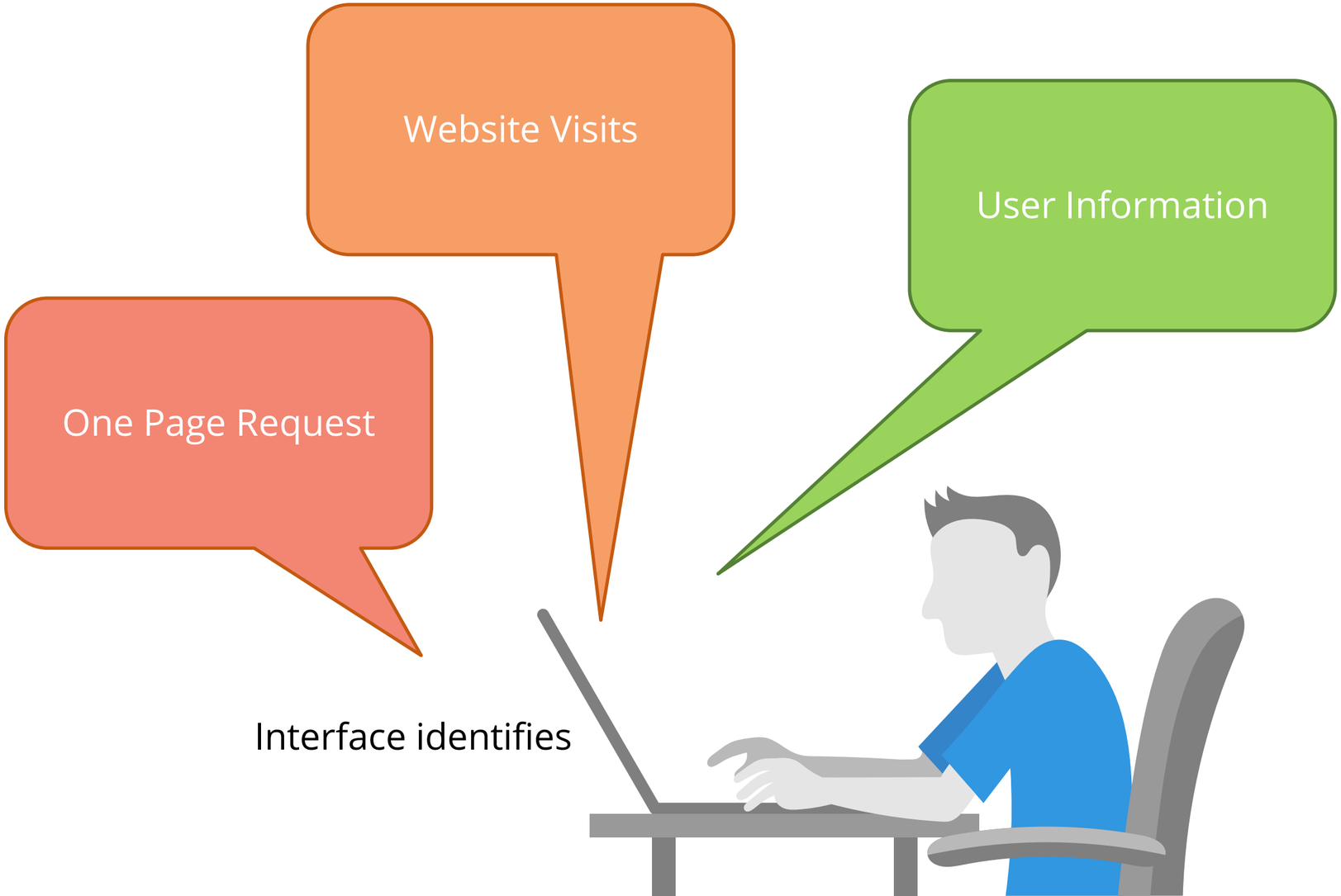
request.getCookies();

Delete Cookies

cookie.setMaxAge(0);

Handling a Session

In JSP, sessions are handled with the **Session Object.** It uses the servlet provided by HttpSession Interface.



Session Handling

Duration: 15 min.

Problem Statement:

Write a program to demonstrate session handling in JSP.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to demonstrate session handling in JSP:

1. Create a Java project in your IDE.
2. Write a program in Java to demonstrate session handling in JSP.
3. Initialize the .git file.
4. Add and commit the program files.
5. Push the code to your GitHub repository.

Key Takeaways

- JSP is used to create web applications using HTML tags and JSP tags.
- JSPs are easy to maintain and have less code.
- Scripting elements can insert the Java code inside the JSP. Types of scripting elements are scriptlet tag, expression tag, and declaration tag.
- Nine JSP Implicit Objects are created by web containers that are available in all the JSP pages.
- Messages that tell the web container how to translate a JSP page into corresponding servlets are known as JSP directives.
- Each action tag is used to perform a specific function. All action tags are used to control the flow between pages.
- Tracking is done by Cookies that are text files stored on the client system.

Product Details Portal

Duration: 45 min.

Problem Statement:
Write a Java program to add and display product details for an e-commerce site.

LESSON-END PROJECT

Before the Next Class

You should know:

- Fundamentals of Servlets
- Handling database
- Implementation of Hibernate
- Basics of JSP

Phase-end Projects

Project for Submission:

FlyAway is a ticket-booking portal that lets people book flights on their website. It should include a search form to allow entries of travel details, list the available flights with the ticket prices, and a confirmation of ticket and the journey details.

Project with Solution:

Learner's Academy is a school that has an online management system. The system keeps track of its classes, subjects, students, and teachers. It has a back-office application with a single administrator login. The administrator should be able to set up a master list of all the subjects and all the teachers. He/she should be able to assign subjects from the list and assign teacher to a class for a particular subject.

FlyAway (an Airline Booking Portal)

Duration: 240 min.

Project Objective:

FlyAway is a ticket-booking portal that lets people book flights on their website. It should include a search form to allow entries of travel details, list the available flights with the ticket prices, and a confirmation of ticket and the journey details.

PHASE-END PROJECT

Background of the Project Statement

FlyAway is a ticket-booking portal that lets people book flights on their website. As a Full Stack Developer, design and develop an airline booking portal named **FlyAway**. Use the GitHub repository to manage the project artifacts.

You Are Asked to Do

You are asked to develop a website which has following features:

- A search form on the homepage to allow entry of travel details, like the date of travel, source, destination, and the number of persons.
- Based on the travel details entered, it will show the available flights with their ticket prices.
- Once a person selects a flight to book, they will be taken to a registration page where they must fill in their details.
- On the next page, the details of the preferred flight will be displayed, and the payment will be done via a dummy payment gateway.
- On completion of the payment, a confirmation page will appear with the details of the booking.

You Are Asked to Do

You are asked to develop a website which has following features:

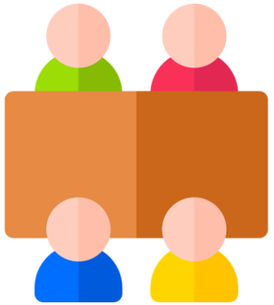
- An admin login page where the admin can change the password after login as per their wish
- A master list of places for source and destination
- A master list of airlines
- A list of flights where each flight has a source, destination, airline, and ticket price

You Must Use the Following

IDE: Eclipse or IntelliJ

Programming language: Java

Git and GitHub



Scrum

MySQL

Specification document

Developing a Backend Admin for Learner's Academy

Duration: 240 min.

Project Objective:

As a Full Stack Developer, design and develop a backend administrative portal for the Learner's Academy. Use the GitHub repository to manage the project artifacts.

PHASE-END PROJECT

Background of the Project Statement

Learner's Academy is a school that has an online management system. The system keeps track of its classes, subjects, students, and teachers. It has a back-office application with a single administrator login. As a Full Stack Developer, design and develop a backend administrative portal for the **Learner's Academy.** Use the GitHub repository to manage the project artifacts.

You Are Asked to Do

You are asked to develop a website which has following features:

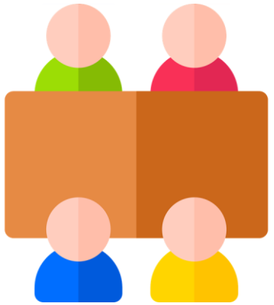
- Set up a master list of all the subjects for all the classes
- Set up a master list of all the teachers
- Set up a master list of all the classes
- Assign classes for subjects from the master list
- Assign teachers to a class for a subject (A teacher can be assigned to different classes for different subjects)
- Get a master list of students (Each student must be assigned to a single class)

You Must Use the Following

IDE: Eclipse or IntelliJ

Programming language: Java

Git and GitHub



Scrum

MySQL

Maven

Specification document