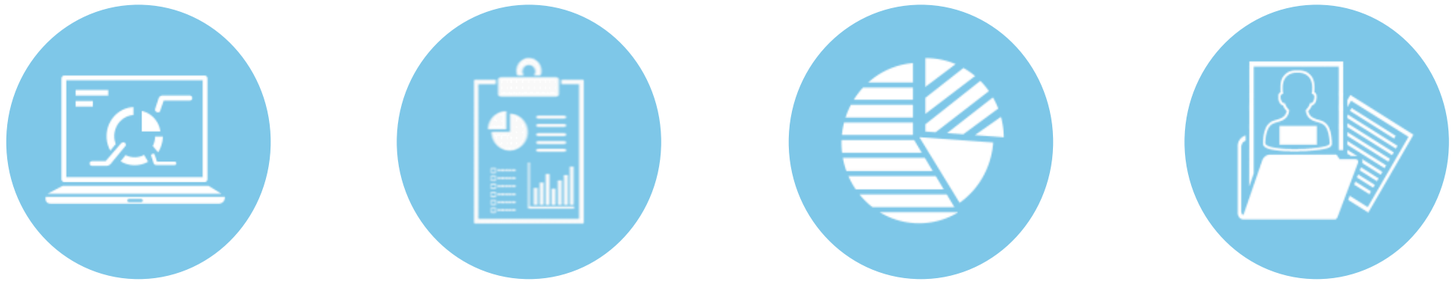
Advanced Java

Lesson 4—Java JSP



Learning Objectives

- Discuss JSP Architecture
- Understand JSP Lifecycle
- Explain how to create a JSP and run it in a Web Application
- Discuss JSP elements and Tags
- Describe JSTL and Custom Tag Libraries

Advanced Java

Topic 1—Introduction to JSP

What Is JSP?

JSP (Java Server Pages) is a text document consisting of Hyper Text Markup Language(HTML), Extensible Markup language(XML), and JSP elements which can be expressed in standard and XML syntax.

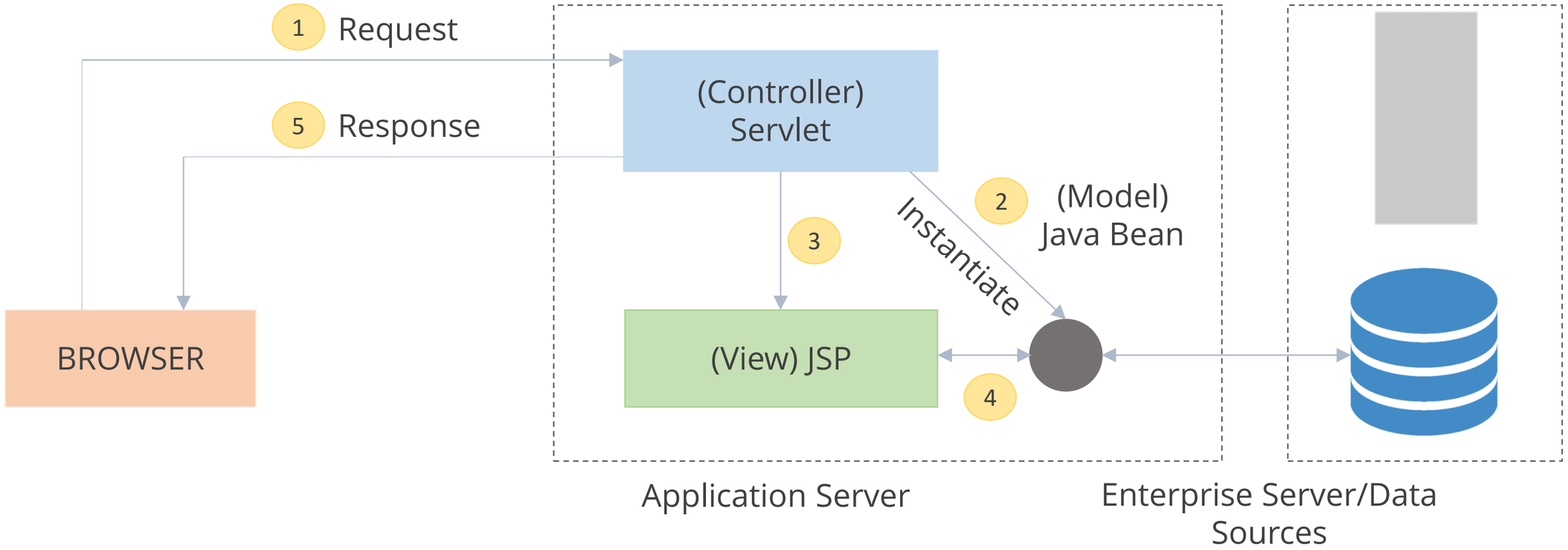
Advantages of JSP

JSP is used to develop web based applications. It supports the separation of presentation and business logic as follows:

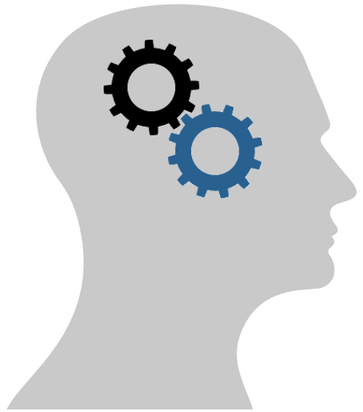
- Web designers can design and update pages without learning the Java programming language.
- Java technology programmers can write code without having to be concerned with web page design.
- It provides the optional mechanism to configure web application file (web.xml).
- JSP programming eliminates the need for repeated deployment (saving the JSP and making a request whenever the JSP is changed). The automatic deployment is taken care by container JASPER [JSP Execution Environment].
- JSP programming provides custom tags development.
- JSP programming environment provides automatic page compilation.

To configure JSP, create a dynamic web project in eclipse, configure Apache Tomcat, and create JSP file in web content. The steps are already discussed in the installation guide in your LMS.

JSP Architecture



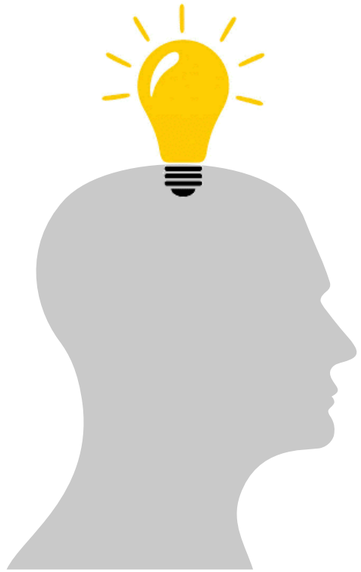
Think Through!



JSP is used to dynamically generate web content on a server and return it to the client. This can also be done using servlet.

Why do we need JSP?

Think Through!



JSP performs a number of tasks.

To understand this, let's learn about the differences between Java JSP and Servlets.

JSP vs. Servlet

JSP	Servlets
JSP is a web page scripting language that can generate dynamic content.	Servlets are Java programs that are Compiled. They create dynamic pages.
It is easier to code in JSP than in Java Servlet.	Involves writing a lot of code
In MVC model, JSP acts as a view.	In MVC model, Servlets act as controllers.
Custom tag can directly call Java beans.	There is no such custom tag.
JSP is generally preferred when there is not much processing of data required. It runs slower compared to Servlet as it takes compilation time to convert into Java Servlet.	Servlets are best for use when there is more processing and manipulation involved. Servlet runs faster than JSP.

Web Container Responsibilities

What does a web container do in a JSP?

- It provides standard libraries required for coding.
- It provides a suitable environment for execution.
- It calls the call-back methods at appropriate times.
- It maintains the life cycle of the program.

JSP API

JSP technology is based on JSP API. It consists of the following packages:

- javax.servlet.jsp (Classes and interface for core JSP)
- javax.servlet.jsp.tagext (Java server pages library)
- javax.servlet.jsp.el (Expression language)

Advanced Java

Topic 2—JSP Lifecycle

JSP Lifecycle

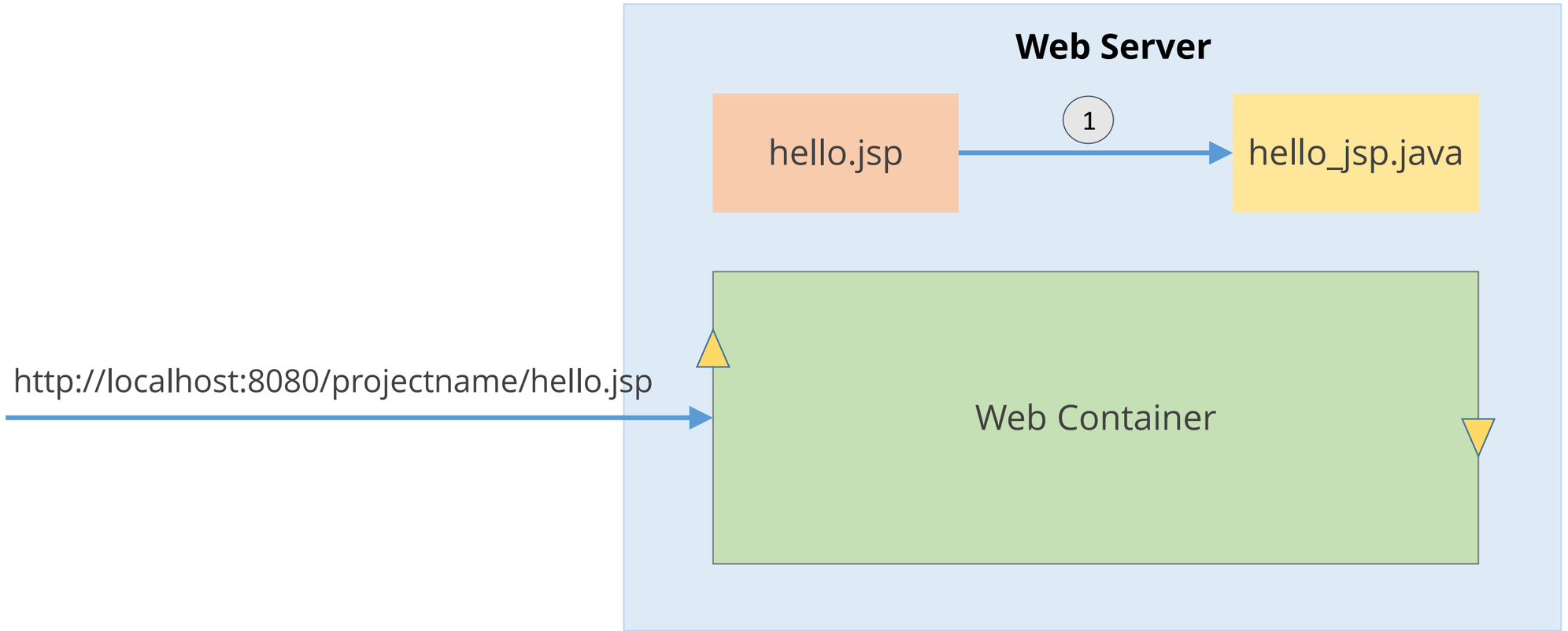
1. Translation of JSP to Servlet code
2. Compilation of Servlet to bytecode
3. Loading Servlet class
4. Creating Servlet instance
5. Initialization by calling _jspInit() method
6. Request processing by calling _jspService() method
7. Destroying object by calling _jspDestroy() method

JSP Lifecycle

- Translation of JSP to Servlet code
- Compilation of Servlet to bytecode
- Loading Servlet class
- Creating Servlet instance
- Initialization by calling _jspInit() method
- Request processing by calling _jspService() method
- Destroying object by calling _jspDestroy() method

TRANSLATION OF JSP TO SERVLET CODE

In the first step, the web container translates the JSP file into a Java source file that contains a servlet class definition.



JSP Lifecycle

- Translation of JSP to Servlet code
- Compilation of Servlet to bytecode
- Loading Servlet class
- Creating Servlet instance
- Initialization by calling _jspInit() method
- Request processing by calling _jspService() method
- Destroying object by calling _jspDestroy() method

COMPILATION OF SERVLET TO BYTECODE

hello.jsp

```
<html>
<head>
<title> My First JSP</title>
</head>
<%
int count=0;
%>
<body>
Page count is:
<% out.println(++count); %>
<body>
</html>
```

Translation to servlet

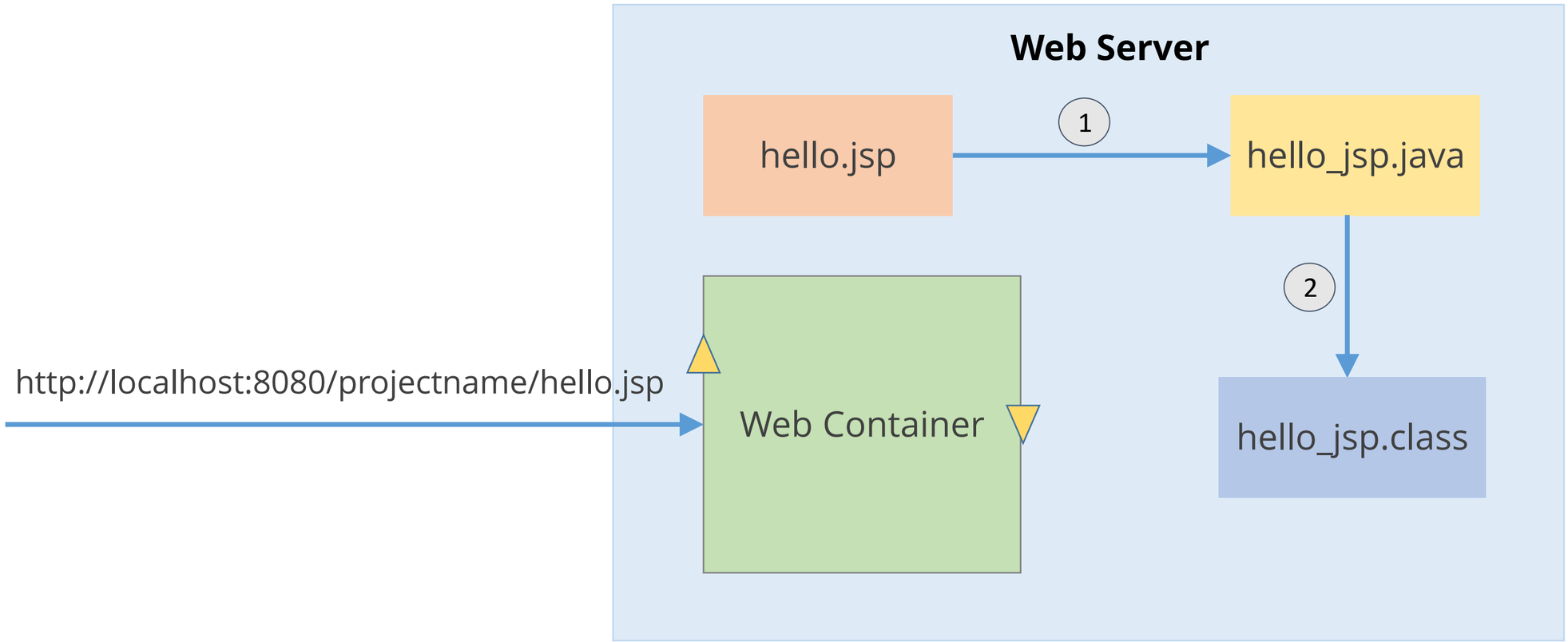
```
public class hello_jsp extends
HttpServlet
{
public void
_jspService(HttpServletRequest
request,
HttpServletResponse response)
{
PrintWriter out= response.getWriter();
response.setContentType("text/html");
out.write("<html><body>");
int count=0;
out.write("Page count id ");
out.write(++count);
out.write("</body></html>");
}
}
```

JSP Lifecycle

- Translation of JSP to Servlet code
- Compilation of Servlet to bytecode
- Loading Servlet class
- Creating Servlet instance
- Initialization by calling _jspInit() method
- Request processing by calling _jspService() method
- Destroying object by calling _jspDestroy() method

COMPILATION OF SERVLET TO BYTECODE

In the second step, the web container compiles the servlet source code into a Java class file.

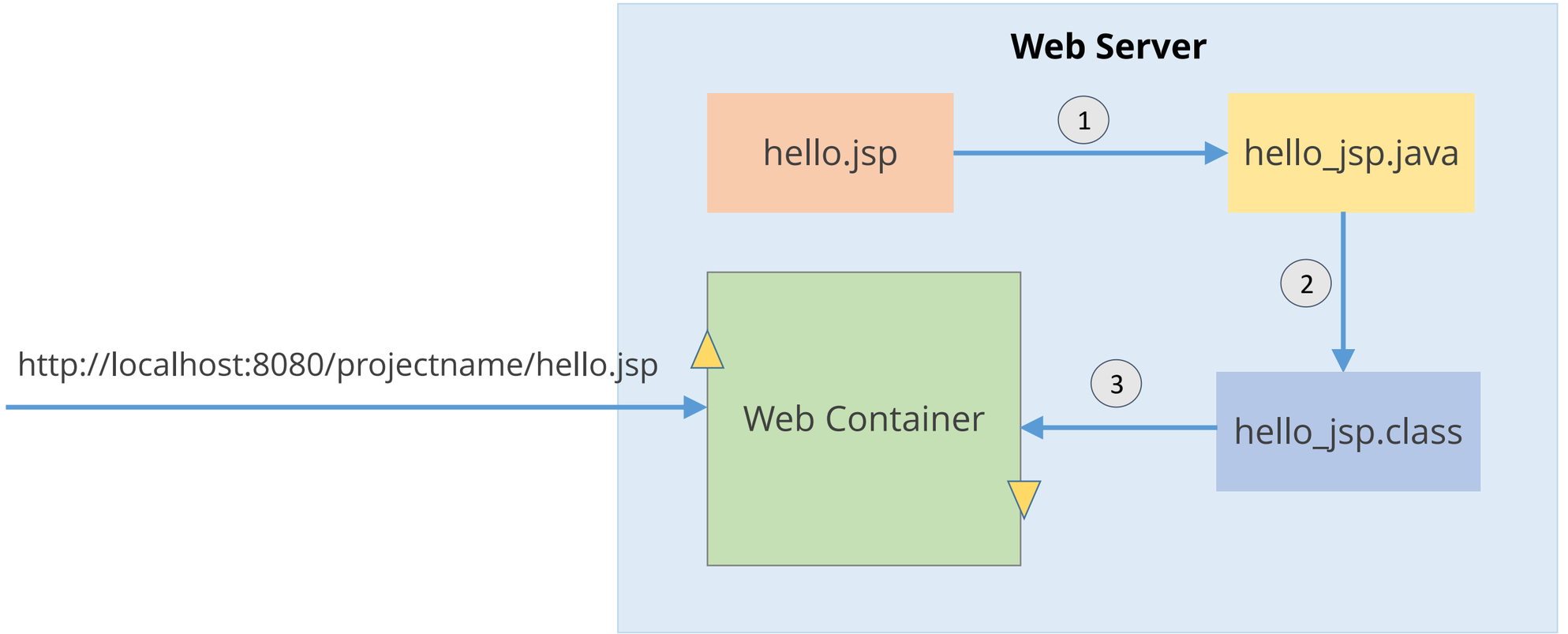


JSP Lifecycle

- Translation of JSP to Servlet code
- Compilation of Servlet to bytecode
- Loading Servlet class
- Creating Servlet instance
- Initialization by calling _jspInit() method
- Request processing by calling _jspService() method
- Destroying object by calling _jspDestroy() method

LOADING SERVLET CLASS

In the third step, the servlet class bytecode is loaded into the web container's JVM software using a class loader.

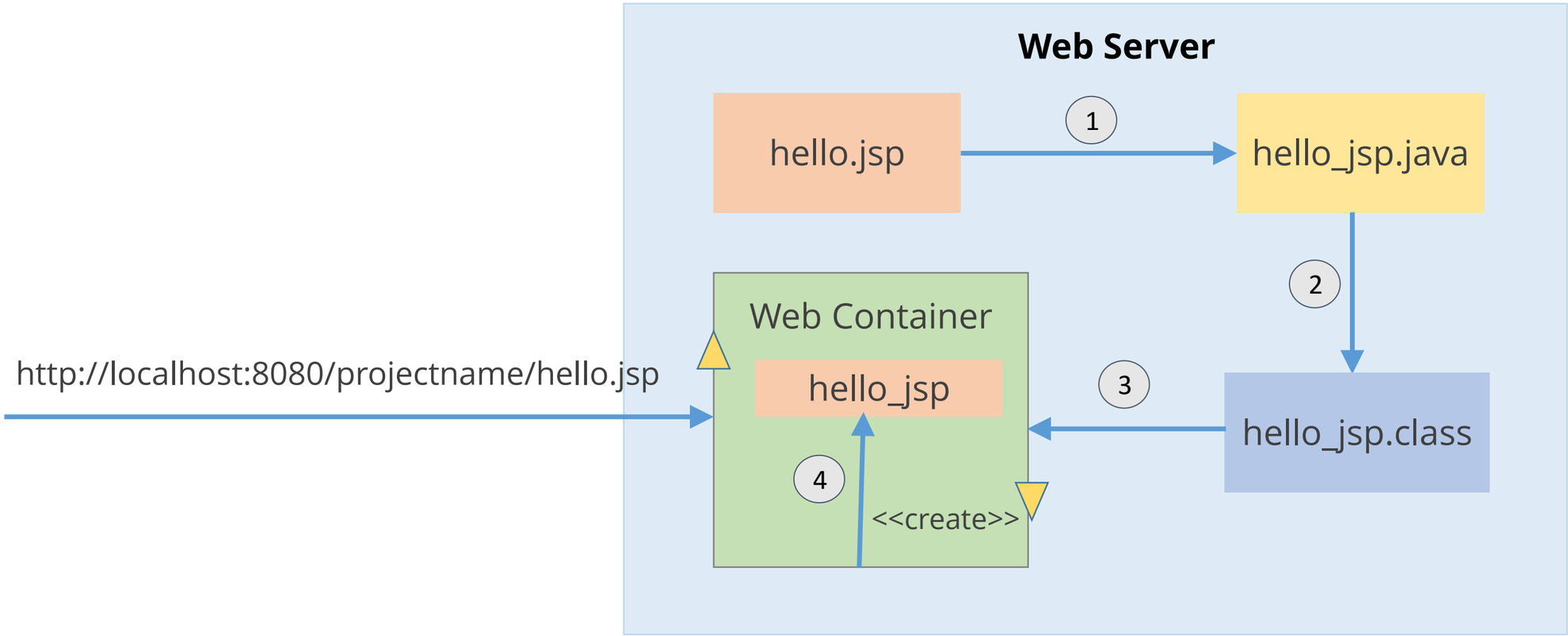


JSP Lifecycle

- Translation of JSP to Servlet code
- Compilation of Servlet to bytecode
- Loading Servlet class
- **Creating Servlet instance**
- Initialization by calling _jspInit() method
- Request processing by calling _jspService() method
- Destroying object by calling _jspDestroy() method

CREATING SERVLET INSTANCE

In the fourth step, the web container creates an instance of the servlet class.

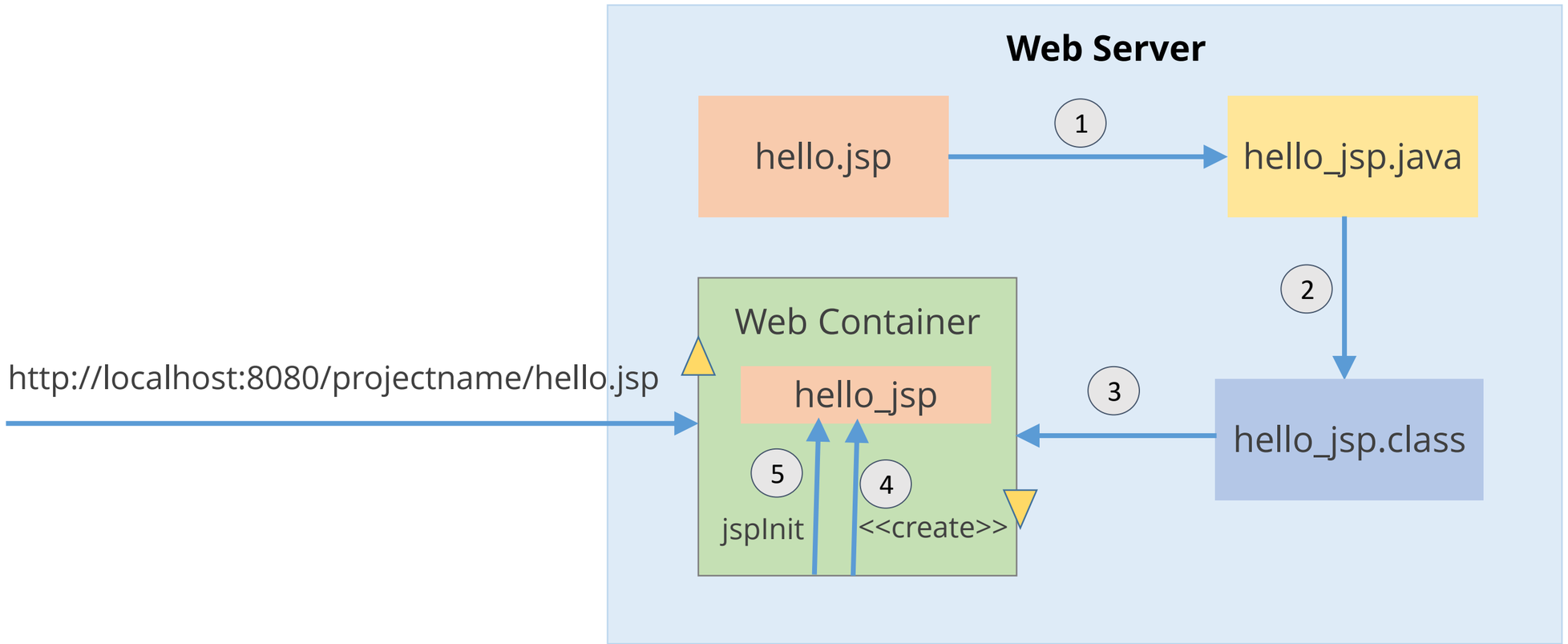


JSP Lifecycle

- Translation of JSP to Servlet code
- Compilation of Servlet to bytecode
- Loading Servlet class
- Creating Servlet instance
- Initialization by calling _jspInit() method
- Request processing by calling _jspService() method
- Destroying object by calling _jspDestroy() method

INITIALIZATION BY CALLING _jspInit() METHOD

In fifth step, the web container initializes the servlet by calling the jspInit method.

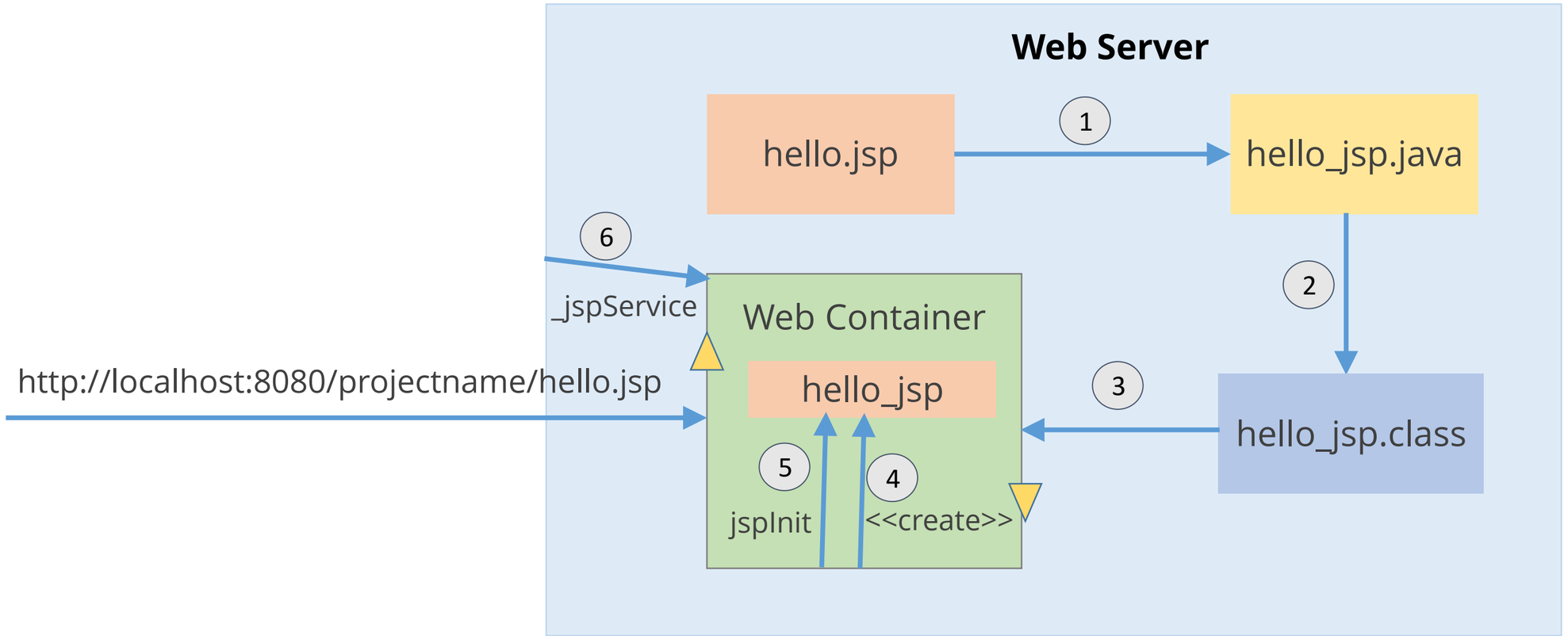


JSP Lifecycle

- Translation of JSP to Servlet code
- Compilation of Servlet to bytecode
- Loading Servlet class
- Creating Servlet instance
- Initialization by calling _jspInit() method
- Request processing by calling _jspService() method
- Destroying object by calling _jspDestroy() method

REQUEST PROCESSING BY CALLING _jspService() METHOD

The initialized servlet can now service requests. With each request, the web container can call the _jspService methods for the converted JSP page.

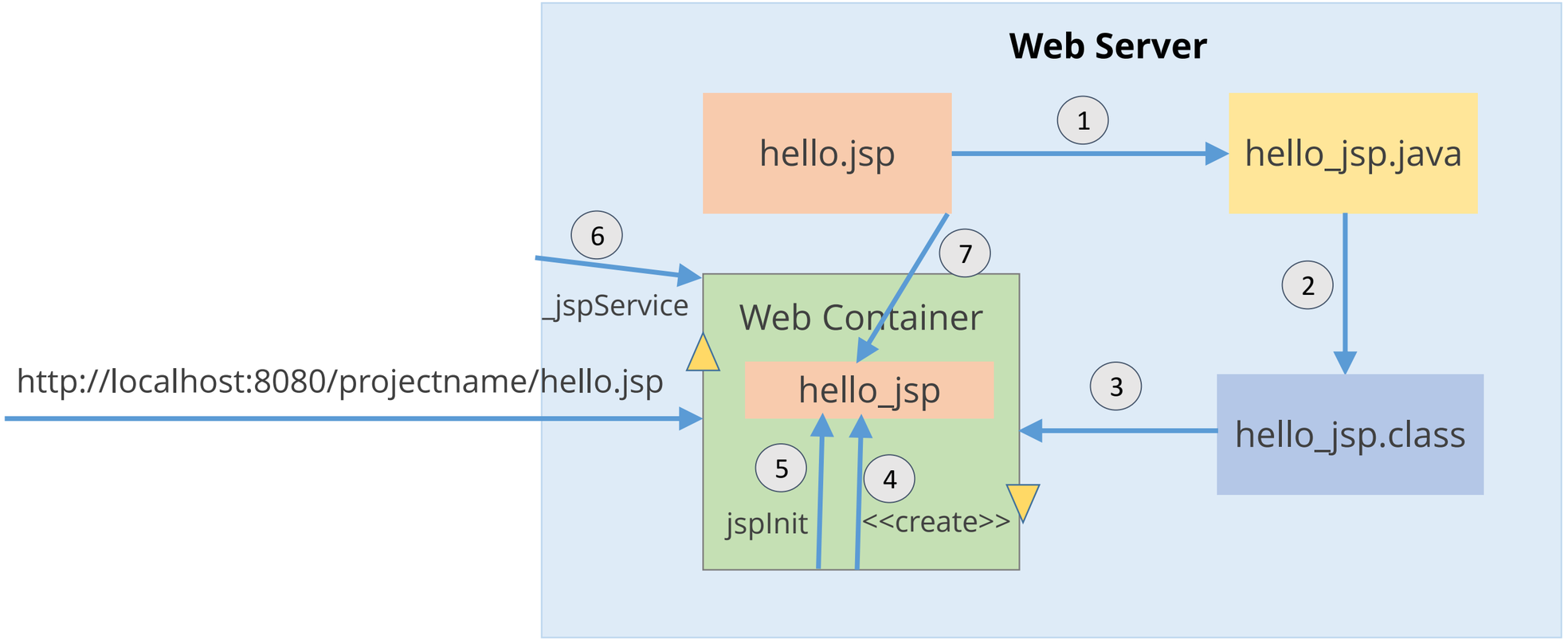


JSP Lifecycle

- Translation of JSP to Servlet code
- Compilation of Servlet to bytecode
- Loading Servlet class
- Creating Servlet instance
- Initialization by calling _jspInit() method
- Request processing by calling _jspService() method
- Destroying object by calling _jspDestroy() method

DESTROYING OBJECT BY CALLING _jspDestroy() METHOD

When the web container removes the JSP servlet instance from services, it first calls the jspDestroy method to allow the JSP page to perform any requirement clean up.



What are Implicit Objects?

Implicit objects are the Java objects that the JSP Container makes available to the developers in each page; the developers can call them directly without explicitly declaring them.

- JSP technology has 9 implicit variables. They represent commonly used objects for servlets that JSP page developers might need to use.
- You can retrieve HTML from parameter data by using the request variable, which represents the HttpServletRequest object.

Implicit Variables

Variable Name	Class Name	Description
request	javax.servlet.http.HttpServletRequest	The HttpServletRequest object associated with the request
response	javax.servlet.http.HttpServletResponse	The HttpResponse object associated with the response that is sent back to the browser
out	javax.servlet.jsp.JspWriter	The JspWriter object associated with the output stream of the response
session	javax.servlet.http.HttpSession	The HttpSession object associated with the session for the given user of the request—only meaningful if the JSP page is participating in an HTTP session
application	javax.servlet.ServletContext	The ServletContext object for the web application
config	javax.servlet.ServletConfig	The ServletConfig object for the web application
pageContext	javax.servlet.jsp.PageContext	The pageContext object that encapsulates the environment of a single request for this JSP page
exception	java.lang.Object.Throwable	The Throwable object that was thrown by some other JSP page—only available in a JSP error page
page	java.lang.Object	Represents current JSP page

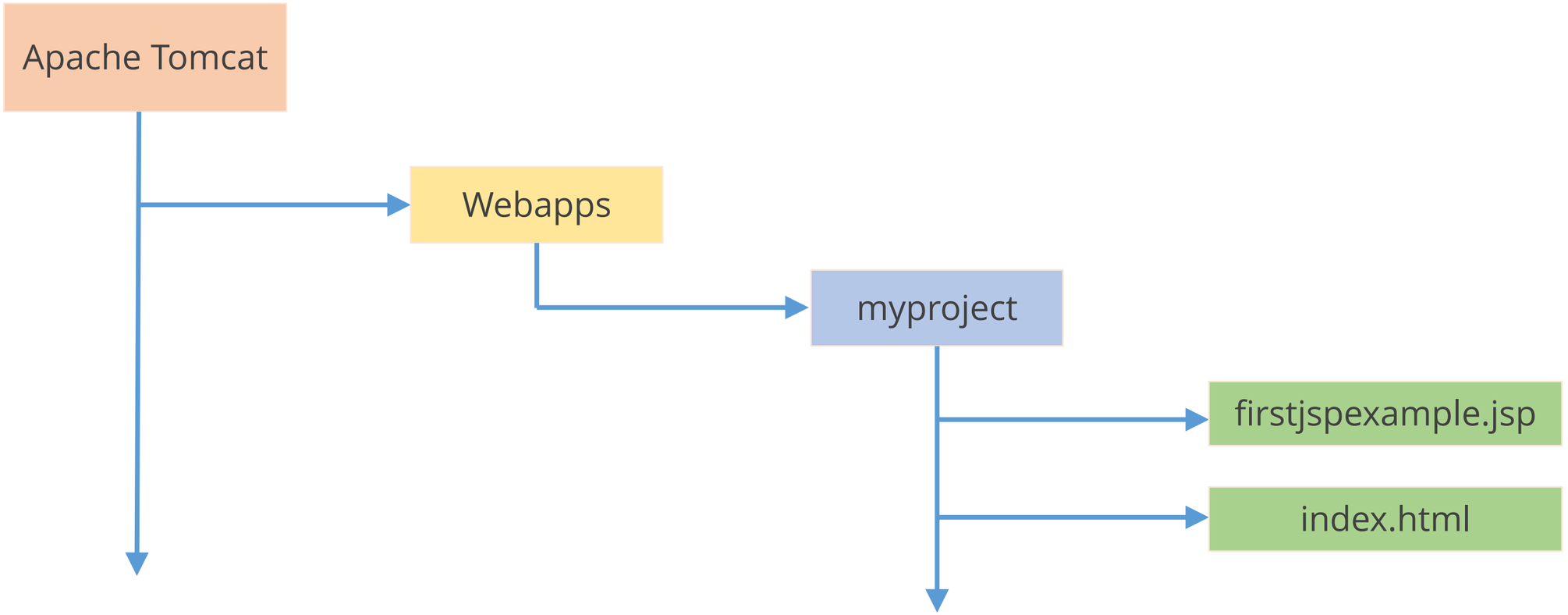
Advanced Java

Topic 3—Creating a JSP and running it in a Web Application

Creating a JSP and Running It In a Web Application

Unlike servlets, deploying JSP pages is as easy as deploying static pages. JSP pages can be placed in the same directory hierarchy as HTML pages.

In the development environment, JSP pages are placed in the web directory. In the deployment environment, JSP pages are placed in the top-level directory of the web application.



Creating a JSP and Running It In a Web Application

1. Configure Server
2. Create Dynamic Web Project
3. Create JSP file in WebContent Folder
4. Create html file in WebContent folder

Advanced Java

DEMO—Writing JSP Program with Implicit Objects

Advanced Java

Topic 4—Working with JSP Elements

JSP Elements

JSP elements in a JSP page can be expressed in two types of syntax:

1. standard
2. XML

Syntax Elements	Standard Syntax	XML Syntax
1. Comments	<% -- -- %>	<!-- -->
2. Declaration	<% ! %>	<jsp: declaration> </jsp: declaration>
3. Directives	<%@ include %> <% @page %> <%@taglib %>	<jsp : directive include../> <jsp:directive page ../> <xmlns :prefix = “tag library url”>
4. Expression	<% = %>	<jsp:expression> </jsp:expression>
5. Scriptlets	<% %>	<jsp:scriptlet> </jsp:scriptlet>

JSP Comment Tag

- JSP comment is used when you are creating a JSP page and want to put in comments about what you are doing.
- JSP comments are only seen in the JSP page.
- These comments are not included in servlet source code during translation phase; they do not appear in the HTTP response.

Comment Tag	Declaration Tag	Directive Tag	Expression Tag	Scriptlets Tag

JSP Declaration Tag

- We can declare a variable or method in JSP inside declaration tag.
- Declaration is made inside the Servlet class but outside the service (or any other method).
- We can declare static member, instance variable, and method inside declaration tag.

Comment Tag	Declaration Tag	Directive Tag	Expression Tag	Scriptlets Tag

JSP Directive Tag

This tag is used for special instruction to web container. It includes three tags:

- <% @ page %> defines page dependent properties such as language session error page.
- <%@ taglib %> declares tag library used in the page.
- <%@ include %> defines file to be included.

Comment Tag	Declaration Tag	Directive Tag	Expression Tag	Scriptlets Tag

JSP Directive Tag

<% @ page %>

<% @ page %>
<%@ taglib %>
<%@ include %>

- <% @ page %> defines page-dependent properties such as language session error page.
- It defines a number of page-dependent properties that communicate with the web container.

```
Syntax:-
    <% @ page attribute =”value” %>
```

Comment Tag	Declaration Tag	Directive Tag	Expression Tag	Scriptlets Tag

JSP Directive Tag

<% @ page %> ATTRIBUTES

<% @ page %>
<%@ taglib %>
<%@ include %>

- import
- language
- extends
- session
- isThreadSafe
- isErrorPage
- errorPage
- contentType
- autoFlush
- buffer

Comment Tag	Declaration Tag	Directive Tag	Expression Tag	Scriptlets Tag

JSP Directive Tag

<%@ taglib %>

<% @ page %>
<%@ taglib %>
<%@ include %>

<%@ taglib %> declares the tag library used in the page. JSP allows you to define custom JSP tags that look like HTML or XML tags:

- A tag library is a set of user-defined tags that implements custom behavior.
- A taglib directive is used to define the tag library that the current JSP page uses.
- A JSP page might include several tag libraries.

Comment Tag	Declaration Tag	Directive Tag	Expression Tag	Scriptlets Tag

JSP Directive Tag

<%@ include %>

<% @ page %>
<%@ taglib %>
<%@ include %>

- <%@ include %> defines the file to be included and the source code.
- It has an attribute for file.

Comment Tag	Declaration Tag	Directive Tag	Expression Tag	Scriptlets Tag

JSP Expression Tag

- Expressions are evaluated when a JSP page is requested. The results are converted into String and fed to the print method of the implicit object.
- If the result cannot be converted into string, an error will be raised at translation time.
- If this is not detected at translation time, a classCastException will be raised at request processing time.

Comment Tag	Declaration Tag	Directive Tag	Expression Tag	Scriptlets Tag

JSP Scriptlet Tag

- Scriptlet tag allows you to write java code inside JSP page.
- Scriptlet tag implements the _jspService method functionality by writing script/Java code.
- Everything written inside the scriptlet tag is compiled as Java code.

Comment Tag	Declaration Tag	Directive Tag	Expression Tag	Scriptlets Tag

Caution

WHEN TO USE DECLARATION TAG OVER SCRIPLET TAG

- If you want to include any method in your JSP file, use declaration tag.
- During translation phase of JSP, methods and variables inside the declaration tag become instance methods and instance variables and are also assigned default values.
- Anything we add in scriplet tag goes inside the _jspService() method. We cannot add any function inside the scriplet tag as it creates a function inside the service method during compilation, which is not allowed in a Java method.

Advanced Java

DEMO—Writing JSP Program with Tags

Advanced Java

Topic 5—Working with JSP Standard Action

JSP Standard Action Elements

- Standard action elements are basically the tags that can be embedded into a JSP page.
- During compilation, they are also replaced by the Java code that corresponds to the pre-defined task.
- Action tags can be written only in XML syntax and can be used for communication.

List of JSP Standard Action Elements

- <jsp : forward >
- <jsp : include >
- <jsp : plugin >
- <jsp : param>
- <jsp : params >
- <jsp : fallback >
- <jsp:usebean>
- <jsp:setProperty>
- <jsp:getProperty>

UseBean Tag

This tag is used to interact with a JavaBean component using the standard tags in a JSP page.

```
Syntax :

<jsp:useBean id=”beanName” scope=”page | request | session | application “
 class =”className” / >
```

- id attribute specifies the attribute name of the bean.
- Location of the bean is specified by the cope attribute.
- The class attribute specifies the fully qualified classname.

setProperty Tag

The setProperty tag is used to store data in the JavaBeans instance.

```
Syntax:

<jsp:setProperty name=”beanName” property_expresion / >
```

- Name attribute specifies the name of the JavaBeans instance.
- This must match the id attribute used in useBean tag.
- The property_expression can be represented as follows:

```
property=”*” | property =”propertyName” | property=”propertyName”
param=”parameterName” | property=”propertyName” value=”propertyValue”
```

Property attribute specifies the property within the bean that will be set.

The getProperty Tag

The getProperty tag is used to retrieve a property from a JavaBeans instance and display it in the output stream.

```
Syntax:

<jsp:getProperty name=”beanName” property_expresion / >
```

- Name attribute specifies the name of the JavaBeans instance.
- This must match the id attribute used in useBean tag, where property attribute specifies the property used for the get method.

Action Tag

- The action tag is used to insert the output of another JSP page into the current JSP page.
- The syntax for the jsp:include action has two forms.

```
jsp:include element that does not have a parameter name / value pair.
<jsp:include page = “relative URL” flush”true”/>
```

```
This syntax is used to pass information to the included resources.
<jsp:include page =”relative url” flush =”true”>
<jsp:param… />
</jsp:include>
```

Difference between Include Directive and Include Action

include directive	include action
include directive includes the source code, so it is static	include action includes the output so it is dynamic.
include directive has an attribute for file.	include action has an attribute for page.
It can be written in HTML or XML syntax.	It can be written in XML syntax only.
It can include source code of a JSP and HTML, but not Servlet.	It can include the output of Servlet JSP and HTML.
We can’t pass the expression as a value to a file.	We can pass the expression as a value to the page attribute.

JSP–Expression Language (EL)

JSP EL allows you to create arithmetic expression and logical expression. It uses integers, floating point numbers, strings, the built-in constants (true and false for Boolean values), and null.

It allows you to access application data stored in JavaBeans components easily.

```
Syntax:

It allows to specify an expression for any of attribute values as follows:

${expr}

Where expr specifies the expression

Example: <jsp:setProperty name = “rectangle” property=“area” value=
“${length*width}”/>
```

EL Implicit Object

pageScope	Provides access to variable stored in page scope level
requestScope	Provides access to variable stored in page request scope
sessionScope	Provides access to variable stored in page session scope
applicatrionScope	Provides access to variable stored in page application scope
param	Gives you access to the parameter values available through the request.getParameter
paramValues	Gives you access to the parameter values available through the request.getParameterValue
header	Gives you HTTP request headers as strings
headerValues	Gives you HTTP request headers as collections of strings
initParam	Used as Context-initialization parameter
cookie	Maps the given cookie name to the cookie value
pageContext	Provides access to object request, session, etc.

Advanced Java

DEMO—Writing JSP Program to demonstrate JSP Standard Action Elements

Advanced Java

Topic 6—JSTL and Custom Tag Libraries

JSTL

- JSTL 1.2 is a part of java EE 5 platform
- JSP standard tag library (JSTL) represents a set of tags used to simplify development of JSP
- We can use the JSTL tags in various pages
- It avoids the use of scriptlet tag

JSTL Jar files can be downloaded from: http://www.oracle.com/technetwork/java/index-137889.html

JSTL Tags

Tag	Uses	URL	Prefix
Core Tags	Provide: • Variable Support • URL Management • Flow Control	http://java.sun.com/jsp/jstl/core	c
Sql tags	Provide SQL support	httlp://java.sun.com/jsp/jstl/sql	sql
XML tags	Provide flow control and transformation	httlp://java.sun.com/jsp/jstl/xml	x
Internationalization tags	Provide support for message formatting and number and date formatting.	http://java.sun.com/jsp/jstl/fmt	fmt
Function tags	Provide support for String manipulation & String length and flow control	httlp://java.sun.com/jsp/jstl/functions	fn

Core JSTL Tag: Example

<c:out>

Core JSTL tag is used to write expression and render data to page.

```
Welcome.html

<form  action =“ process.jsp” method = “post”>
First name: <input type = “text” name =“fname”/> </br>
Last name <input type = “text” name = “fname”/> </br/>
<input type = “Submit” value = “submit”/>
  </form>
```

```
Process.jsp

<%@taglib uri = “http://java.sun.com/jsp/jstl/core” prefix =
“c”%>
First name: <c: out value = “${ param.fname}”> </c: out> <br/>
Second name: <c:out value = “${param.fname}”> </c:out> <br/>
```

Core JSTL Tag: Example

<c:if>

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
<c:set var="income" scope="session" value="${3000*4}"/>
<c:if test="${income > 8000}">
   <p>My income is: <c:out value="${income}"/><p>
</c:if>
</body>
</html>
```

Attribute	Description
Test	Condition to evaluate
Var	Name of the variable to store the condition result
Scope	Scope of the variable to store the conditions result

Core JSTL Tag: Example

<c:catch>

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>

<c:catch var ="catchtheException">
  <% int x = 2/0;%>
</c:catch>

<c:if test = "${catchtheException != null}">
  <p>The type of exception is : ${catchtheException} <br />
  There is an exception: ${catchtheException.message}</p>
</c:if>

</body>
</html>
```

Attribute	Description
Var	The name of the variable to hold the java.lang. Throwable if thrown by element in the body

Custom Tag Syntax Rules

JSP allows you to create your own tags. They are known as custom tags. They uses XML syntax. There are four fundamental XML rules that all custom tags must follow:

1. Standard tag syntax must conform to the following structure:

```
<prefix:name { attribute ={“value” | ‘value’ } } * >
body
</prefix:name>
```

2. Empty tag syntax must conform to the following structure:

```
<prefix :name { attribute ={ “value” | ‘value’ } } * />
```

3. Tag name, attributes and prefix are case sensitive

4. Tag must follow nesting rule:

```
<tag1>
<tag2>
</tag2>
</tag1>
```

Advanced Java

DEMO—Writing JSP Program to demonstrate JSP Tags

Key Takeaways

- JSP (Java Server Pages) is a text document consisting of Hyper Text Markup Language(HTML), Extensible Markup language(XML), and JSP elements that can be expressed in standard and XML syntax.
- Implicit variables are the Java objects that the JSP Container makes available to the developers in each page; the developers can call them directly without explicitly declaring them.
- JSP elements in a JSP page can be expressed in two types of syntax: standard and XML
- JSP EL allows you to create arithmetic and logical expressions. It uses integers, floating point numbers, strings, the built-in constants true and false for Boolean values and null.

Quiz

Thank You