

Developing Web Applications Using JSP

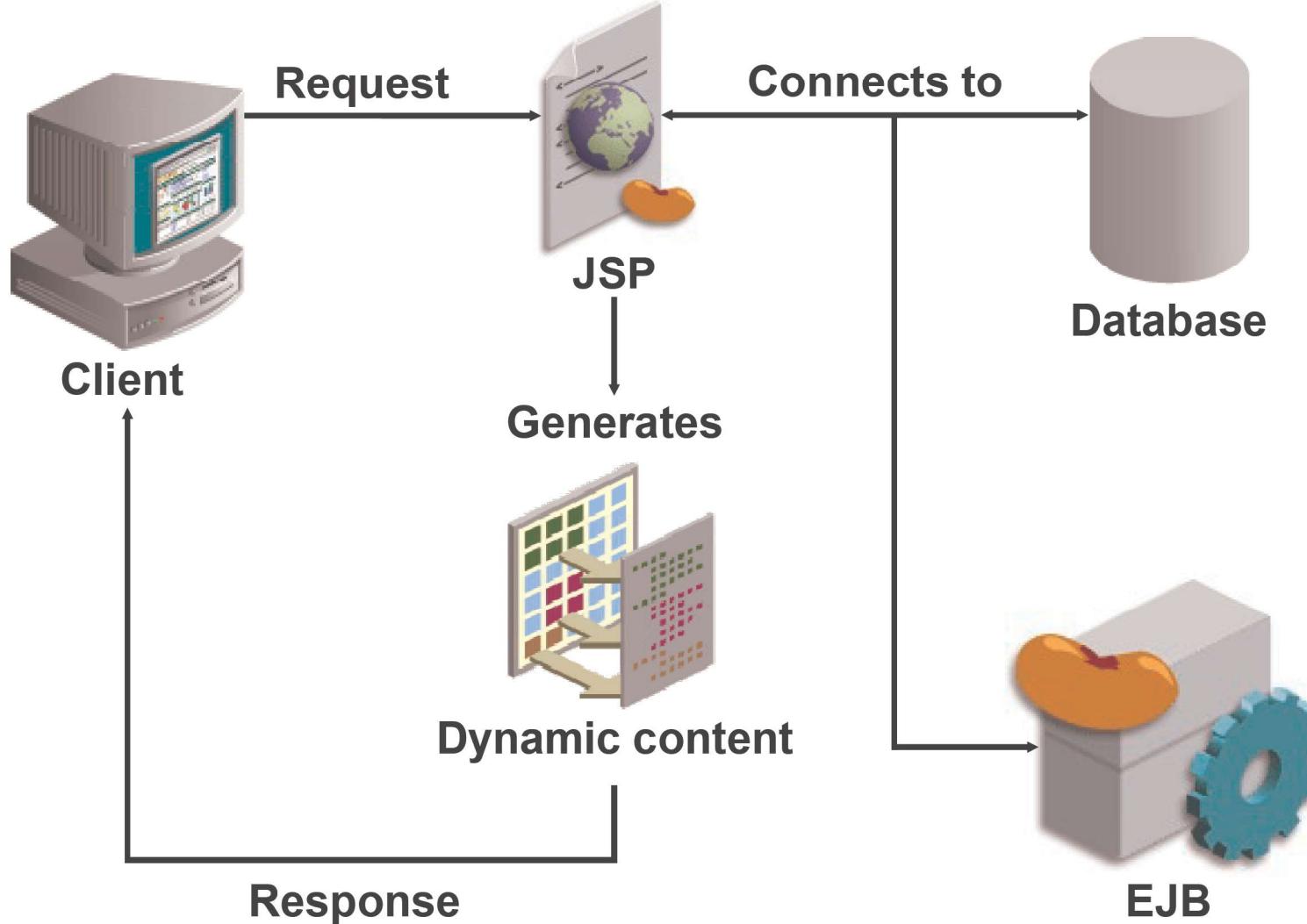
Objectives

After completing this lesson, you should be able to do the following:

- Describe the relationship between JavaServer Pages (JSP) and servlets
- List implicit objects on JSP pages
- Describe the semantics of JSP tags
- Create a JSP segment
- Explain the use of JSP tag files
- Run and debug a JSP-based application



JavaServer Pages



Comparing Servlets and JSPs

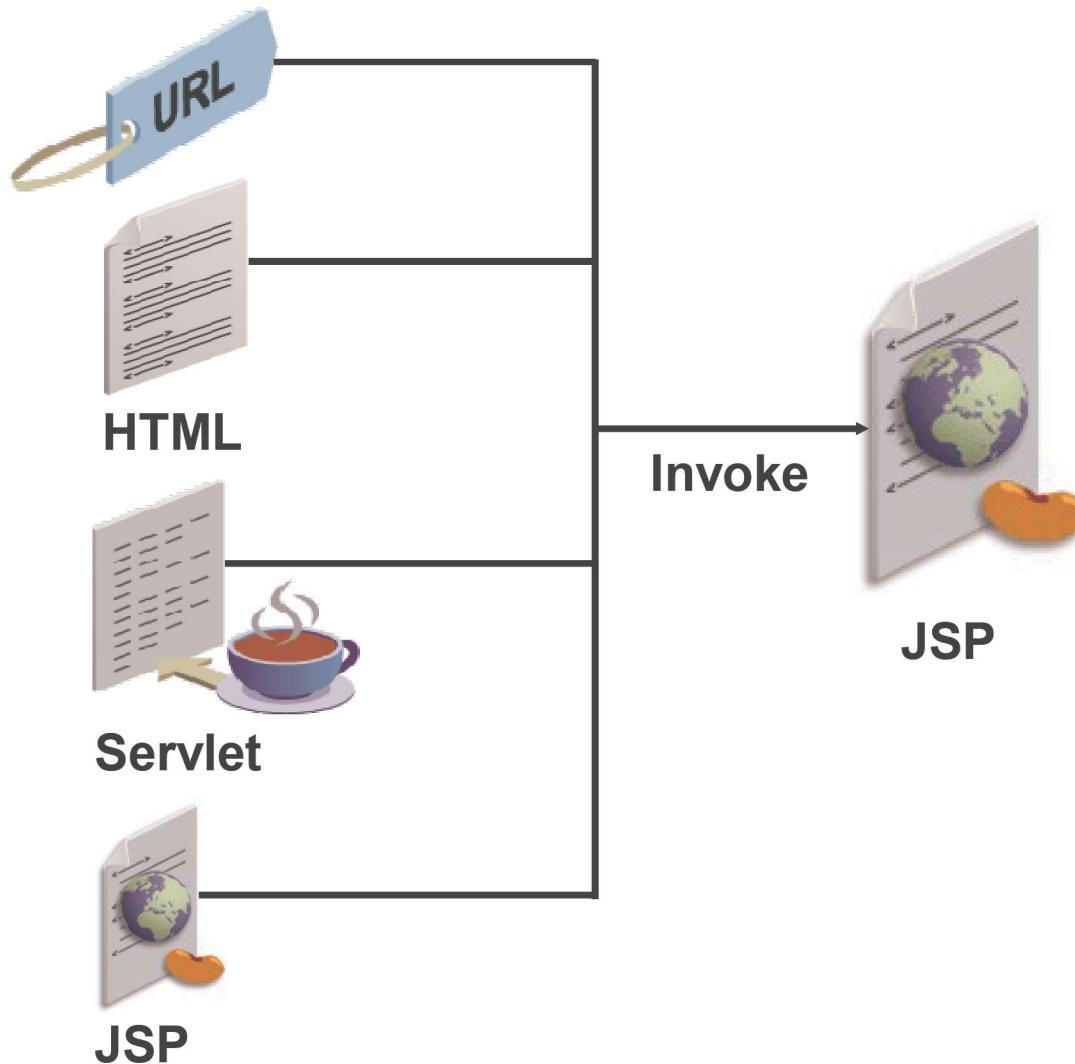
Servlets:

- Are Java programs with embedded HTML code
- Generate dynamic content
- Do not separate static and dynamic content

JSPs:

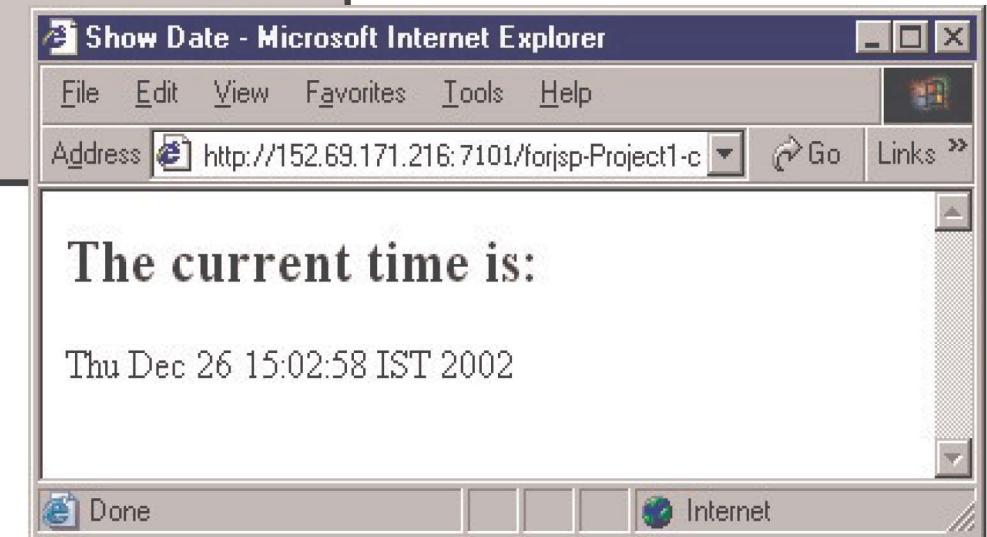
- Are HTML pages with embedded Java code or pure XML
- Generate dynamic content
- Separate static and dynamic content

Invoking JSPs



Date.jsp

```
<%@ page contentType="text/html; charset=WINDOWS-1252"%>
<html> <head>
<meta http-equiv="Content-Type" content="text/html;
charset=WINDOWS-1252">
<title> Show Date </title>
</head>
<body>
<h2> The current time is: </h2>
<p> <%= new java.util.Date() %> </p>
</body>
</html>
```



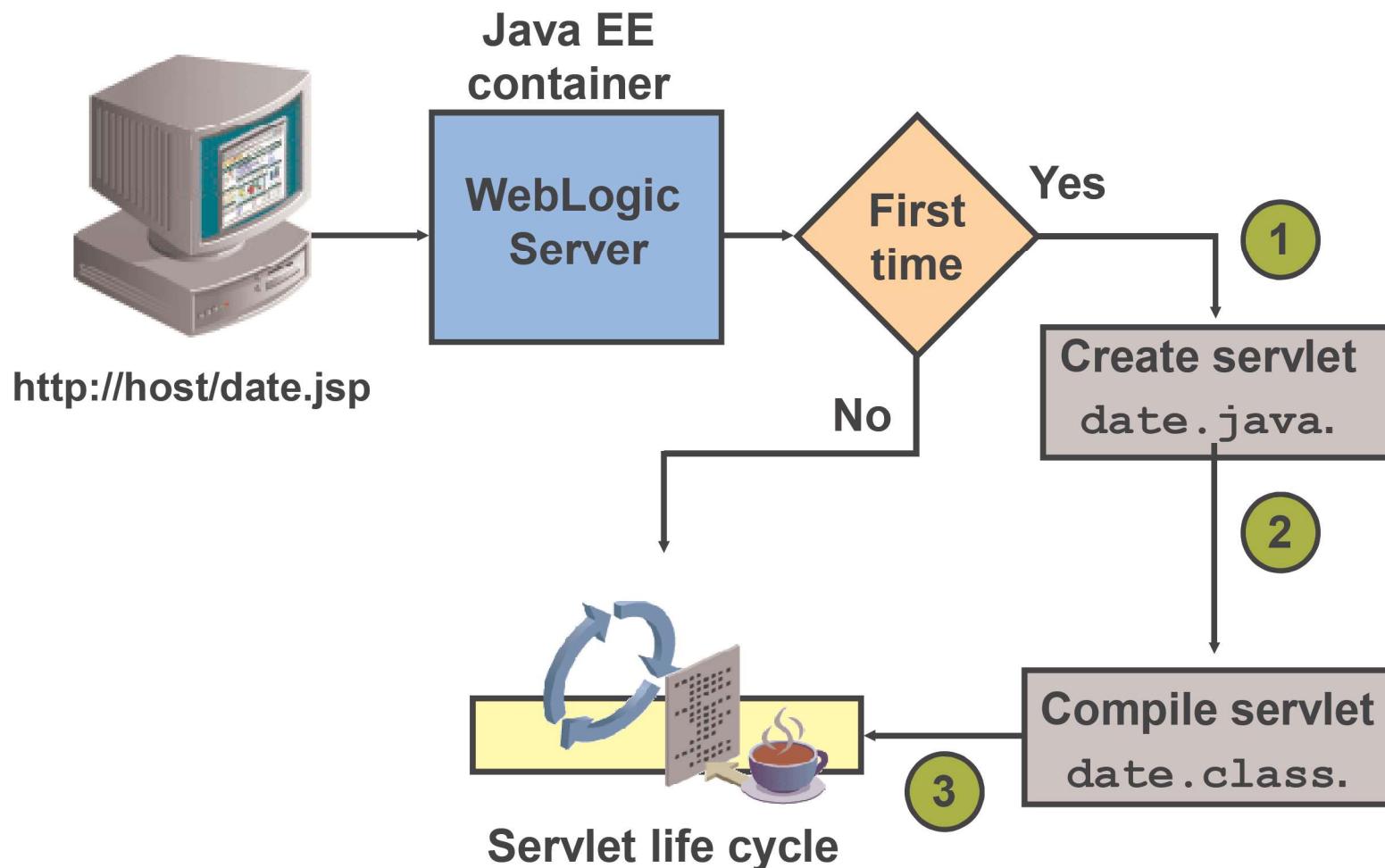
Date Servlet

```
...
public void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException
{
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head><title>Show Date
</title></head><body><h2>The current time
is:</h2><p>\"");
    out.println(new java.util.Date());
    out.println("</body></html>");
    out.close();
}
...
```

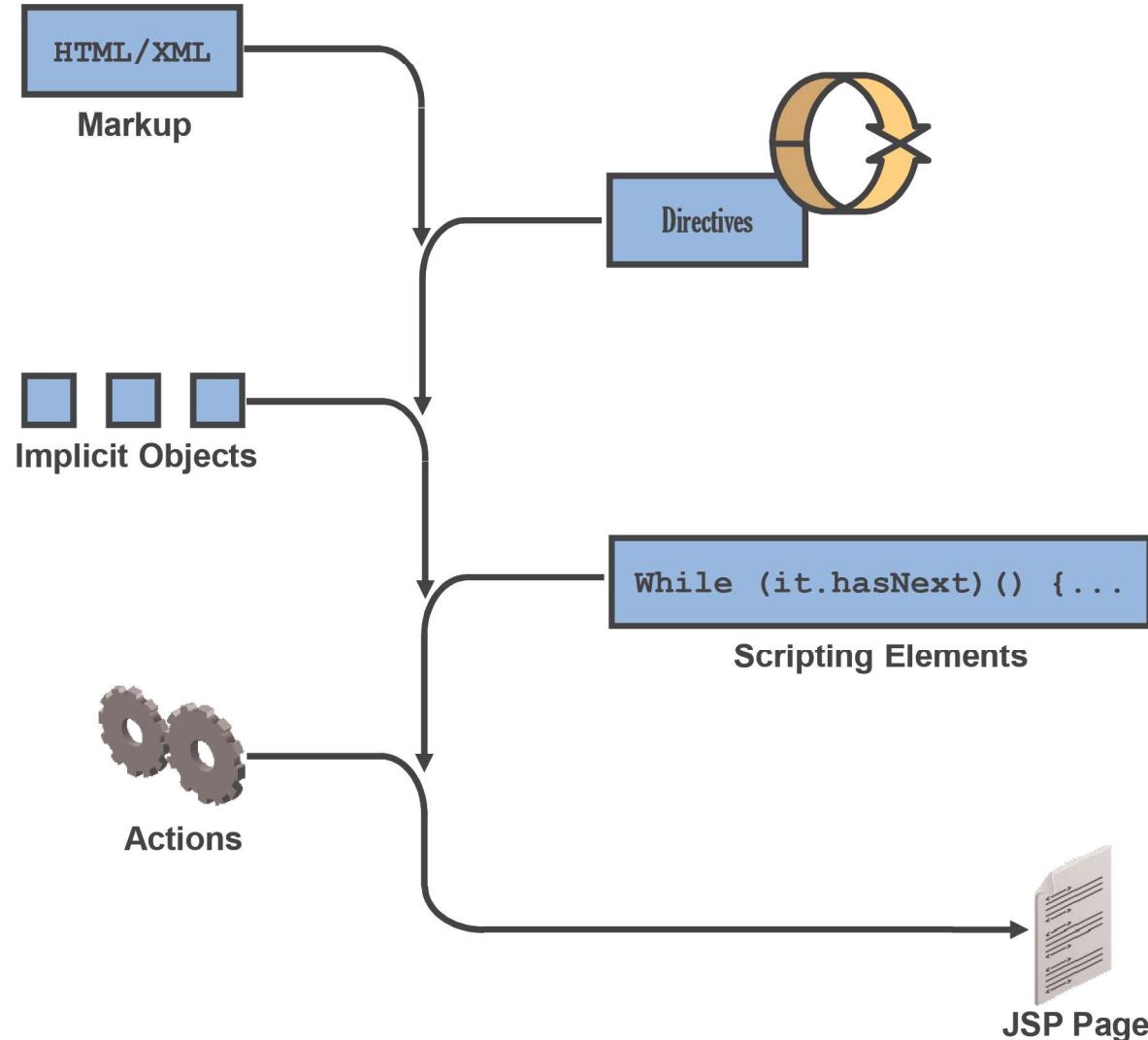
Automated JSP Features

- A JSP is automatically converted into a servlet the first time it is invoked:
 - Java source files are generated.
 - Java class files are generated.
 - The Java Just-In-Time compiler can be used.
- A JSP can contain extensible components:
 - Tags: Libraries such as custom-developed tags
 - JavaBeans (Beans are reused and their properties are automatically introspected.)

JSP Life Cycle



JSP Page Components



Basic JSP Elements

A JSP contains three main elements:

- Text elements
- Directives
- Scripting elements:
 - Declarations
 - Expressions
 - Scriptlets

Syntactic Forms of JSP Tags

Syntactic forms of tags based on JSP tags can be represented in two different ways:

Old Syntax: Similar to other tag-based dynamic presentation technologies	XML Syntax: With beginning and end tags
<%! ... %>	<jsp:declaration> ... </jsp:declaration>
<%= ... %>	<jsp:expression> ... </jsp:expression>
<% ... %>	<jsp:scriptlet> ... </jsp:scriptlet>
<%@ ... %>	<jsp:directive.type ... />

Declarations

- Are used to define methods or variables
- Begin with the sequence <%!
- End with the sequence %>
- Are inserted into the body of the servlet class, not within a method, during translation
- Are used in conjunction with expressions or scriptlets

```
<%! private int i=3; %>
<%! private String a="Hello", b=" World"; %>
```

Expressions

- Begin with the sequence <%=
- Contain Java expressions that are evaluated and inserted into the servlet's output
- End with the sequence %>
- Do not end with a semicolon

```
<%= i+1 %>
<%= a + b %>
<%= new java.util.Date() %>
```

1

2

Scriptlets

- Begin with the sequence <%
- Contain a block of Java code that is executed every time a request is made
- End with the sequence %>

```
<% if (i<3)
    out.print("i<3");
    if (i==3)
        out.print("i==3");
    else
        out.print("i>3");
%>
```

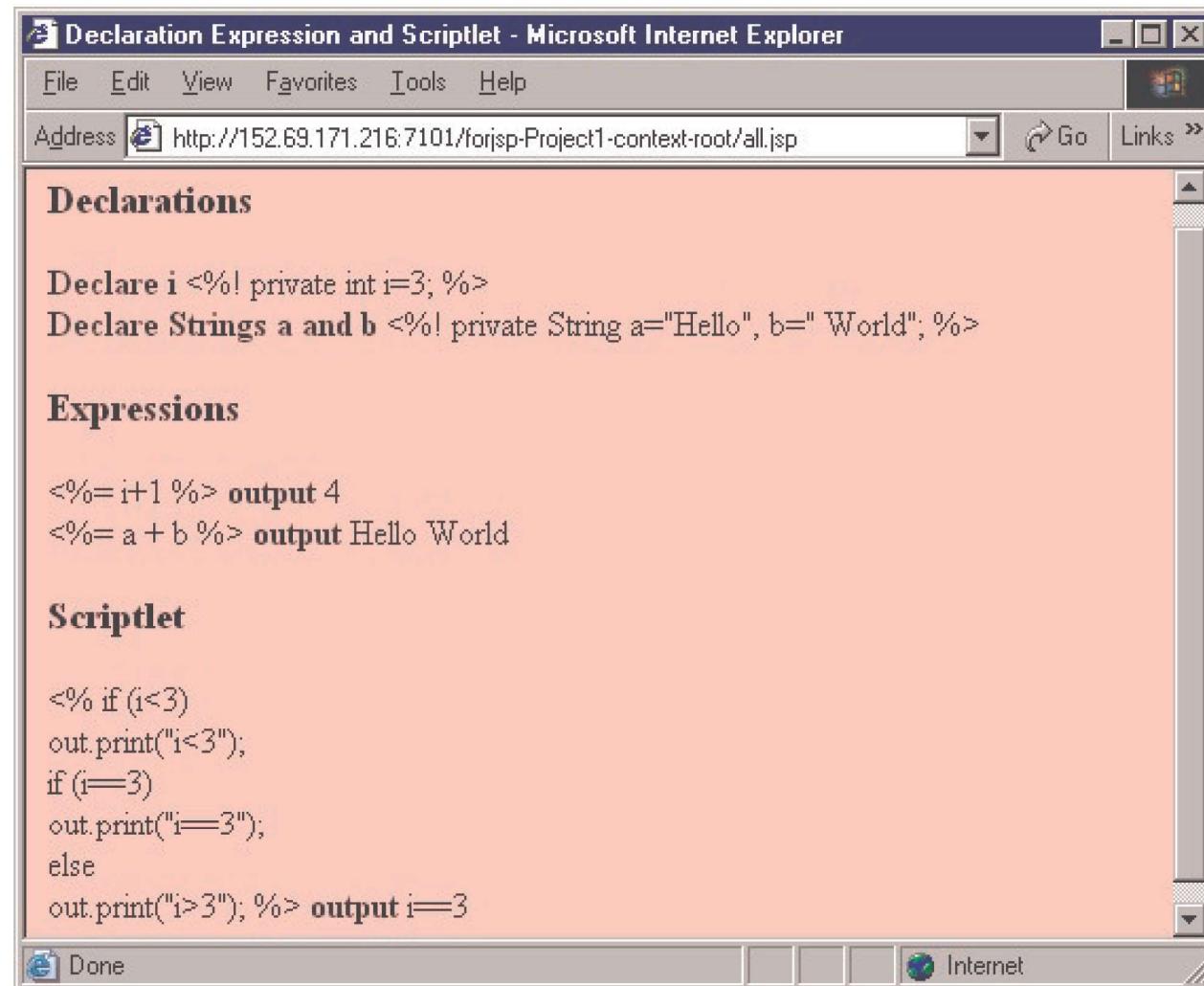
Implicit Objects

There are eight implicit objects, also known as predefined variables, in JSP:

- **request**
- **response**
- **session**
- **out**
- **application**
- **config**
- **pageContext**
- **page**

Note: These objects are created in the generated servlet.

Example



The screenshot shows a Microsoft Internet Explorer window displaying the output of a JSP file. The title bar reads "Declaration Expression and Scriptlet - Microsoft Internet Explorer". The address bar shows the URL "http://152.69.171.216:7101/forjsp-Project1-context-root/all.jsp". The page content is organized into sections:

- Declarations**:
Declare i <%! private int i=3; %>
Declare Strings a and b <%! private String a="Hello", b=" World"; %>
- Expressions**:
<%= i+1 %> **output** 4
<%= a + b %> **output** Hello World
- Scriptlet**:
<% if (i<3)
out.print("i<3");
if (i==3)
out.print("i==3");
else
out.print("i>3"); %> **output** i==3

Quiz

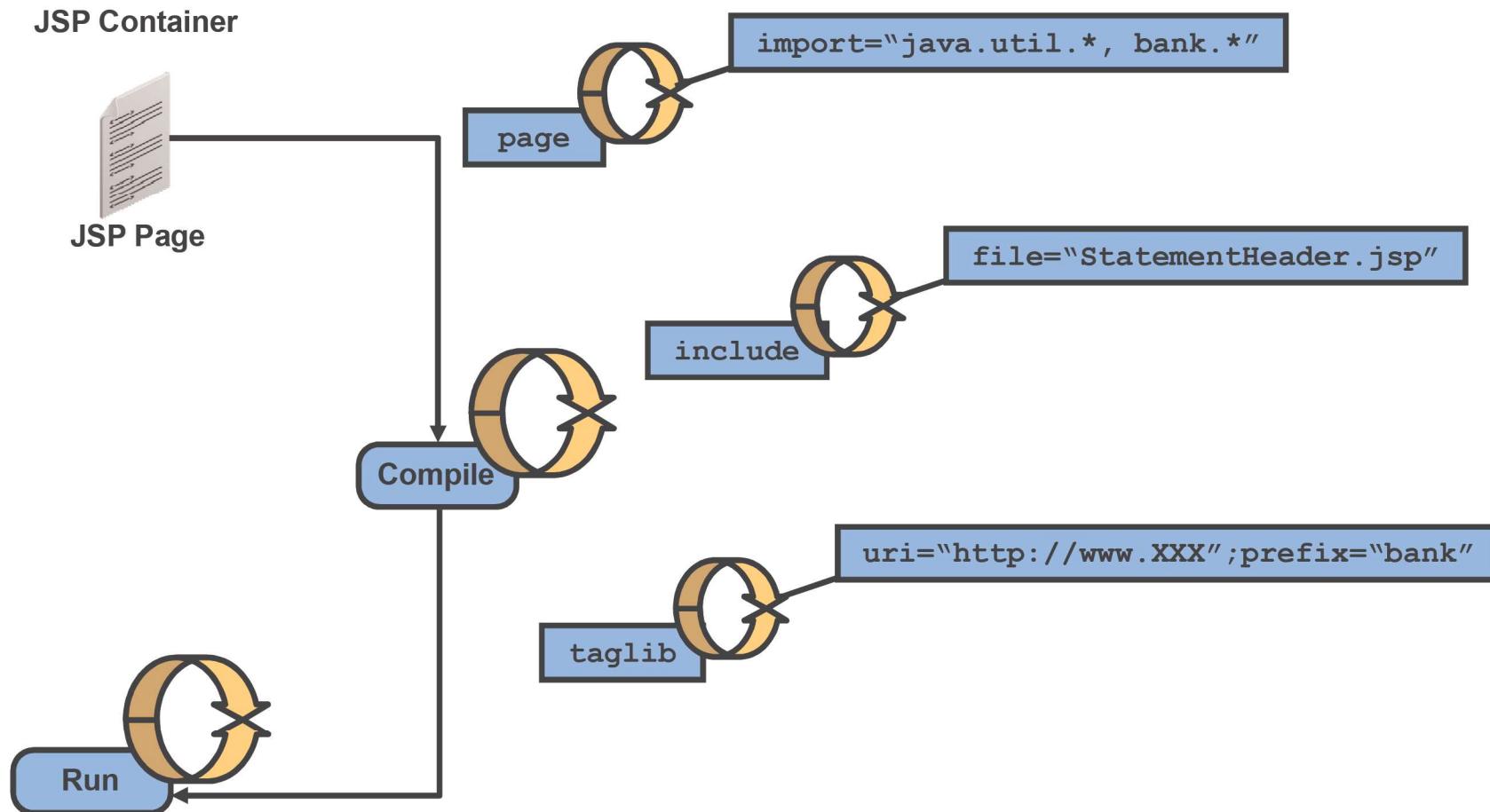
Which of the following pieces can a JSP be broken down into?

1. Static data such as HTML
2. JSP directives
3. JSP scripting elements and variables
4. A servlet class

Directives

- Are used to set global values such as class declarations and method implementations
- Begin with the sequence `<%@`
- End with the sequence `%>`
- Are of the following types:
 - page
 - include
 - taglib

JSP Directives



The page Directive

The page directive defines page-dependent attributes:

- An attribute and value pair cannot be redefined within a translation unit, with the exception of the `include` page directive.
- Redefining a page directive results in a fatal translation error, unless the new and the old definitions are the same.

The page Directive

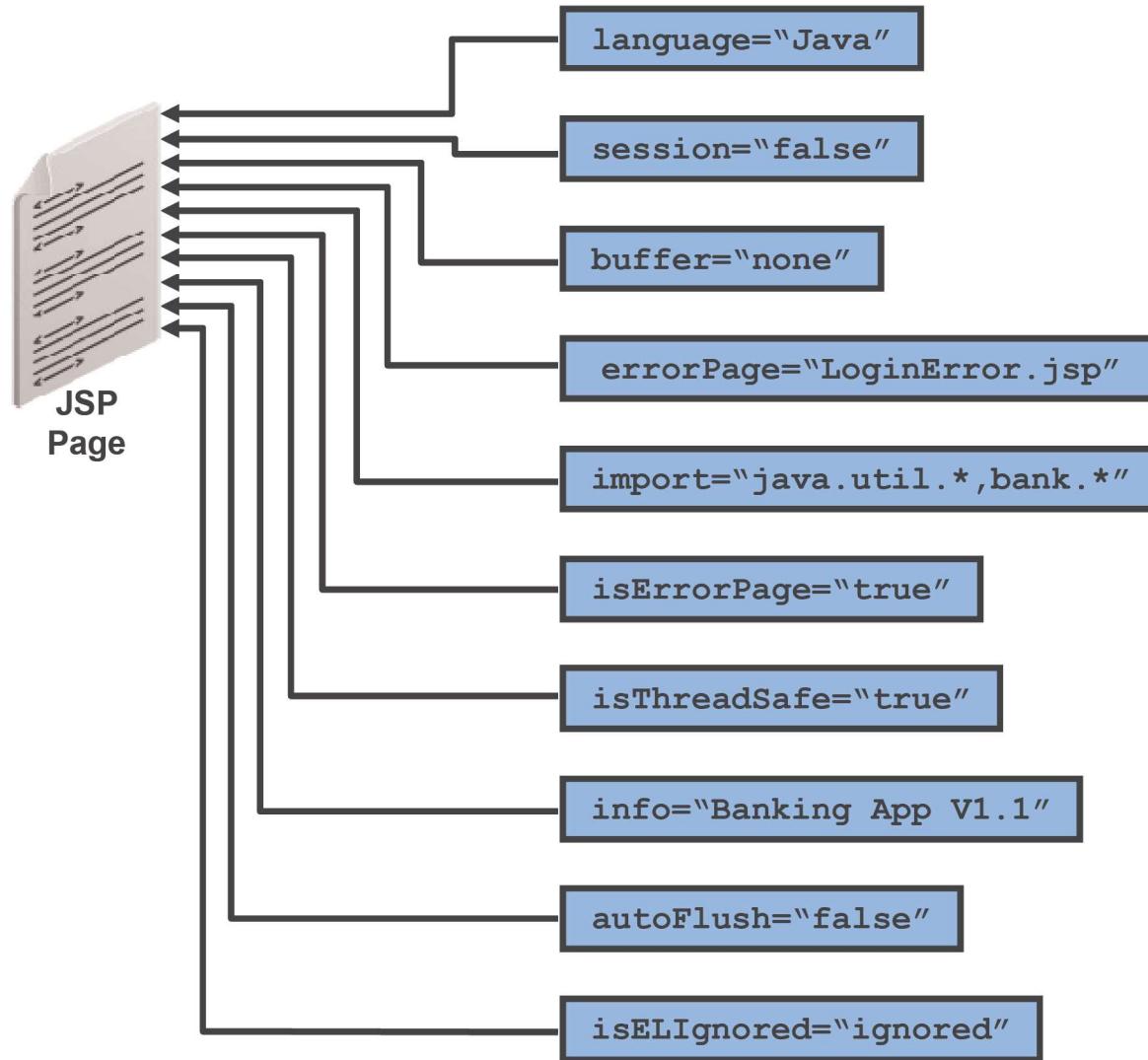
Examples using both styles of syntax:

```
<%@ page import="java.util.*, java.lang.*" %>

<%@ page buffer="5kb" autoFlush="false" %>

<jsp:directive.page errorPage="error.jsp" />
```

JSP page Directives



The include Directive

The `include` directive:

- Inserts the text of the specified resource into the `.jsp` file at page translation time
- Treats resources as static objects
- Can be other HTML files or other JSP pages that contain text, or code, or both

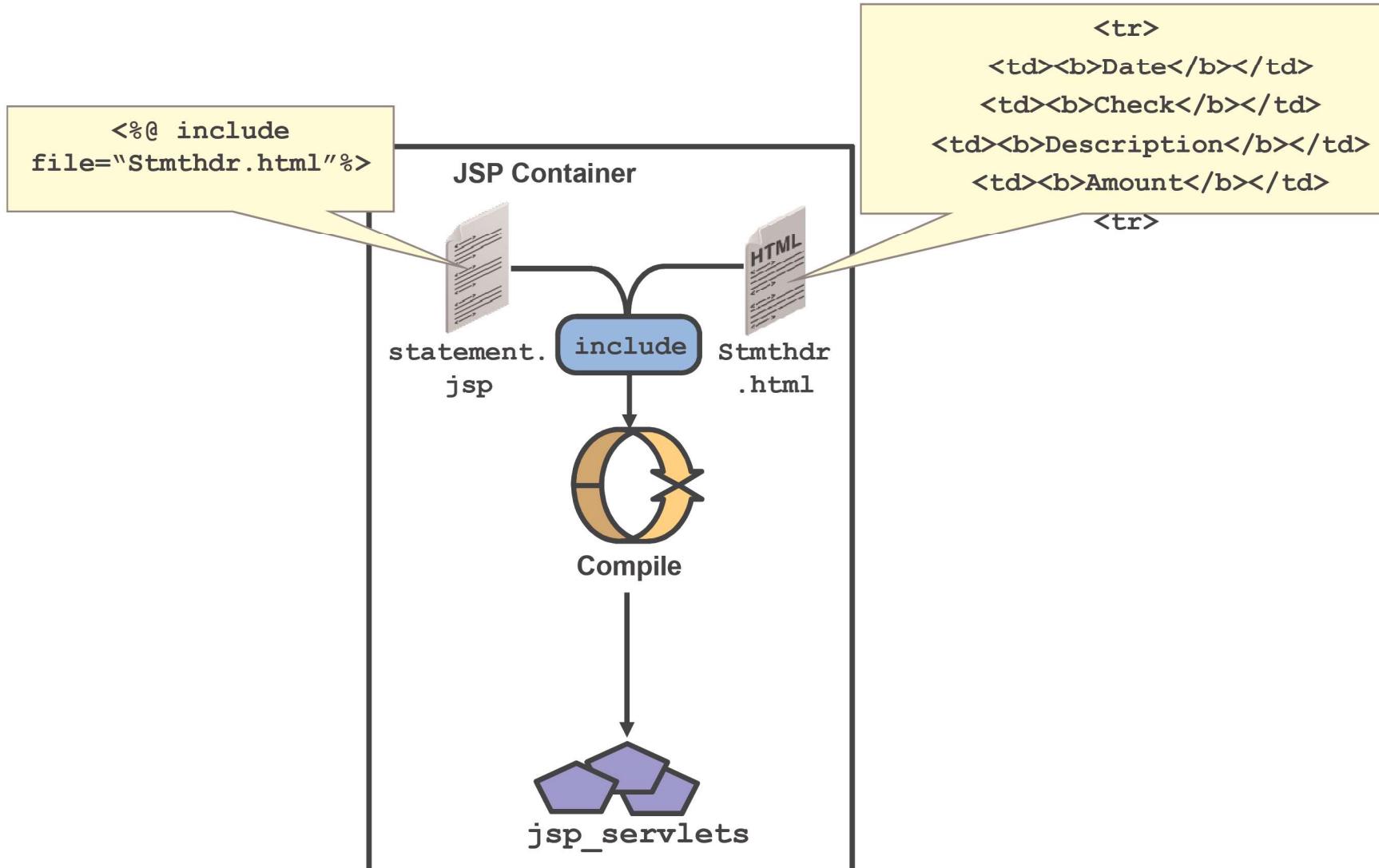
Examples of the `include` directive:

```
<%@ include file="relativeURL" %>
```

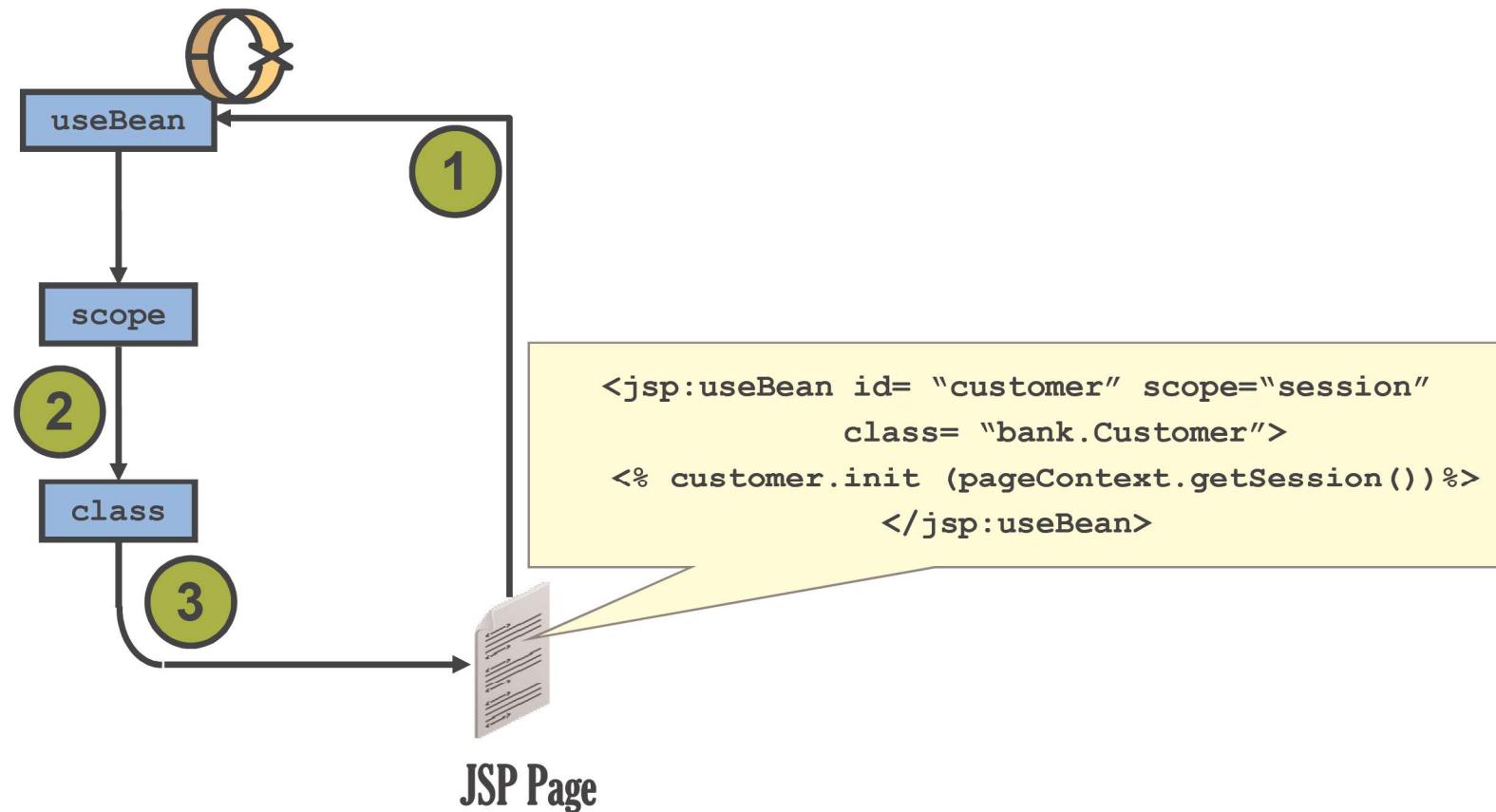
Or

```
<jsp:directive.include file="relativeURL" />
```

The include Directive

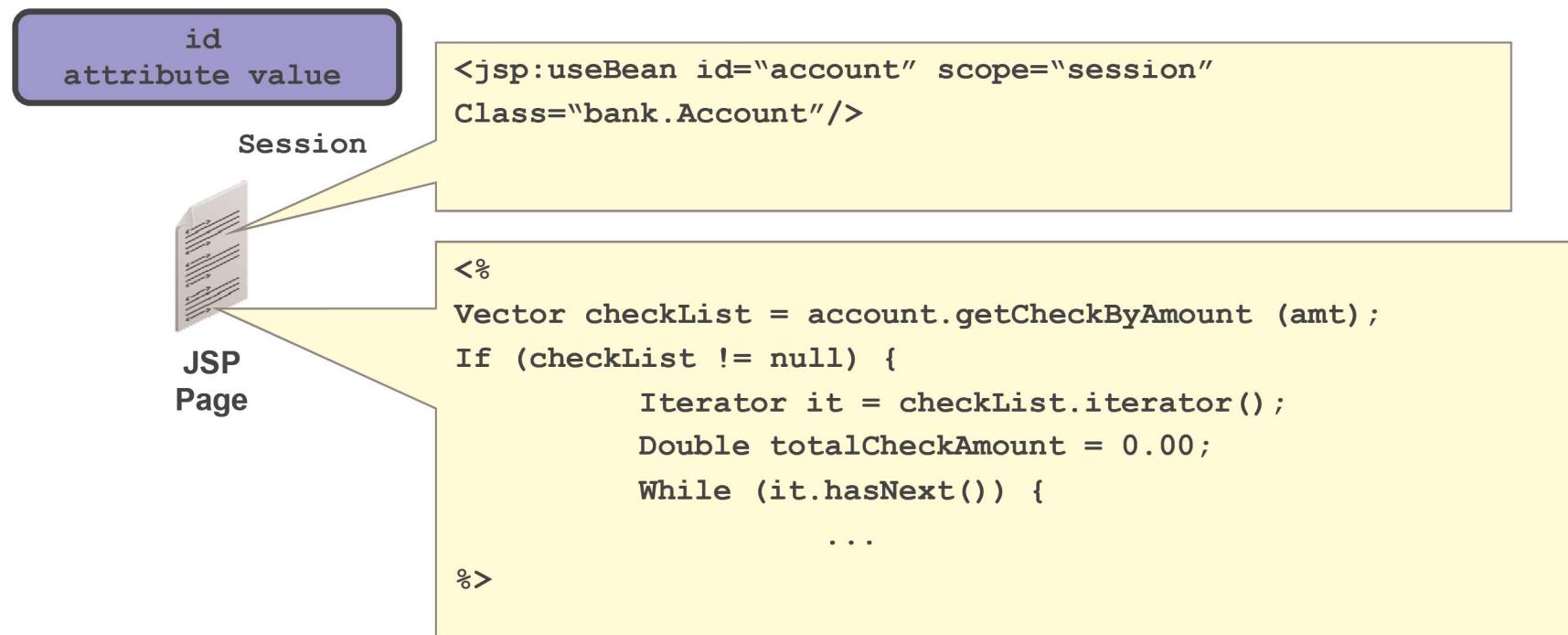


The jsp:useBean Action



The jsp:useBean Action

This graphic shows the id attribute.



JSP and JavaBeans

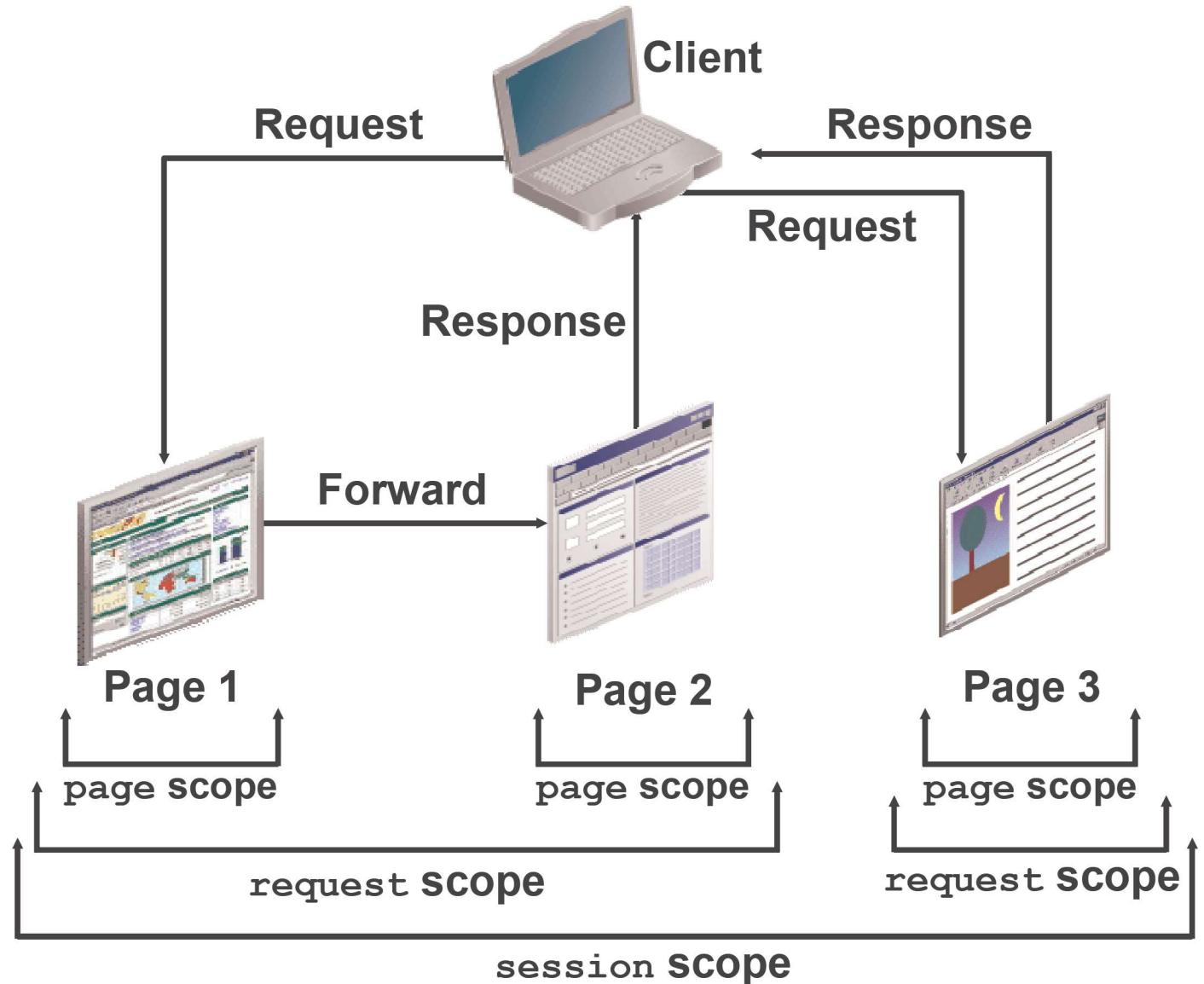
```
package lesson05;
import java.lang.*;
import java.util.*;
public class LuckyNumberBean {
    private int luckyNum;
    public LuckyNumberBean() {
        luckyNum = (int) (1000 * Math.random());
    }
    public int getLuckyNum() {
        return luckyNum;
    }
    public void setLuckyNum(int luckyNum) {
        this.luckyNum = luckyNum;
    }
}
```

Using JavaBeans with JSP

Accessing JavaBeans with the <jsp:useBean> tag:

```
<jsp:useBean id="myBean" scope="session"  
             class="lesson05.LuckyNumberBean" />
```

scope Attribute of <jsp:useBean> Tag



Accessing and Setting Bean Properties

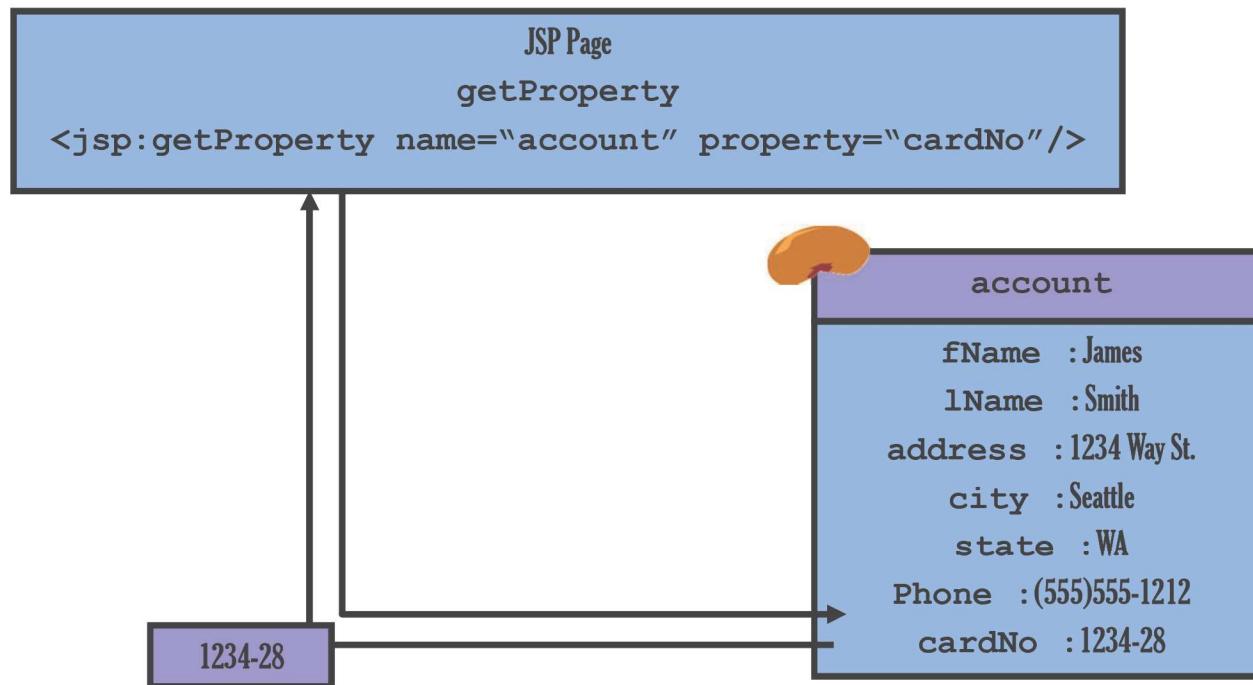
- Accessing bean property:

```
<jsp:getProperty name="myBean"  
property="luckyNum" />
```

- Setting bean property:

```
<jsp:setProperty name="myBean"  
property="luckyNum" value="10" />
```

The jsp:getProperty Action



JSP XML Document

- Contains <jsp:root> as its root element
- Includes only the XML syntax and does not include the traditional JSP tags
- Can be processed directly by the JSP container
- Can be used with XML development tools

Traditional Syntax Versus XML Syntax

Traditional:

- It has no root element.
- page directive
`<%@page %>`
- Declaration tag
`<%! %>`
- Expression tag
`<%= expression %>`
- Scriptlet
`<% %>`

XML:

- `<jsp:root>` is the root element.
- `<jsp:directive.page/>`
- `<jsp:declaration>`
...
`</jsp:declaration>`
- `<jsp:expression>`
...
`</jsp:expression>`
- `<jsp:scriptlet>`
...
`</jsp:scriptlet>`

JSP Segments

- Use a JSP segment for creating a partial JSP page.
- Include one or more segments in a JSP using
`<jsp:include>`.

JSP segment – footer.jspf:

```
Copyright 2020,  
<a href="http://www.oracle.com">  
    Oracle  
</a>
```

JSP:

```
<%@ include file="/footer.jspf"%>
```

Quiz

A JSP XML document contains both XML syntax and the traditional JSP tags.

1. True
2. False

What Is a Tag Library?

- JSPs define six standard actions: useBean, getProperty, setProperty, include, forward and plugin.
- JSPs allow developers to define custom actions that can be invoked by using custom tags.
- A tag library is a collection of custom tags.
- Custom tags can be used to process forms, send mails, access database, control flow, or perform business logic.

Tag Interfaces

- Implement tags using interfaces and classes in `javax.servlet.jsp.tagext.*`.
- Custom tags implement interfaces `Tag`, `BodyTag`, and `SimpleTag`, and are called as tag handlers.

Tag Handlers

- Are objects that a JSP invokes when it encounters a custom tag
- Provide several methods that are called at various stages of a tag's life cycle. For example:

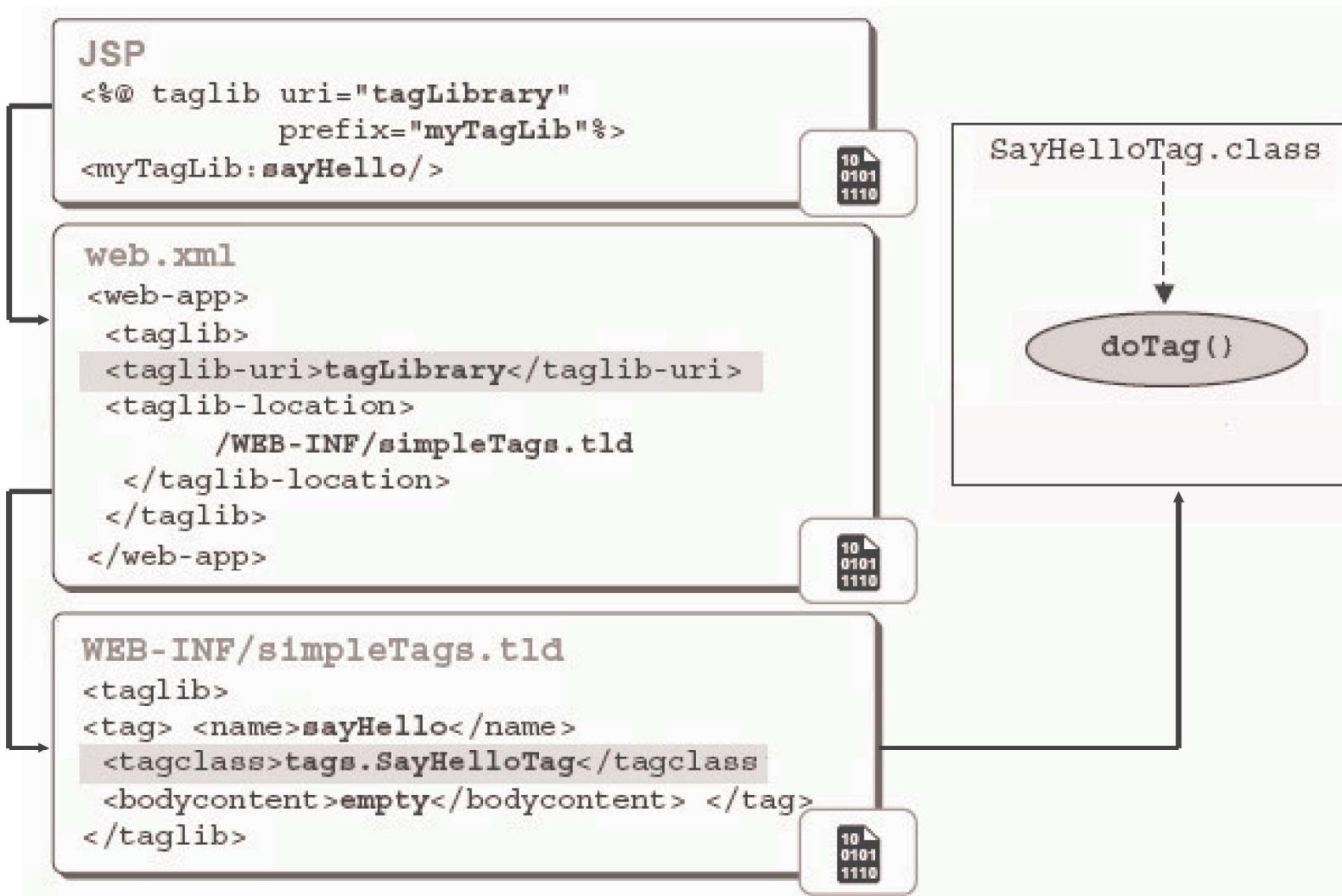
```
import javax.servlet.jsp.tagext.*  
...  
public class SayHelloTag extends SimpleTagSupport {  
    private String var;  
...  
    public void doTag() throws JspException {  
        //contains tag logic, iteration, body evaluations  
        ...  
    }  
...  
    public void setVar( String var ) {  
        ...  
    }  
}
```

Tag Library Descriptor

A tag library descriptor describes a custom tag library.

```
<?xml version = '1.0' ...?>
<taglib ... >
    <tlib-version>1.0</tlib-version>      <!-- Tag lib version -->
    <short-name>...</short-name>          <!-- Preferred name -->
    <uri>...</uri>                      <!-- uniquely identifies the tag library-->
    <tag>
        <description>...</description> <!-- Describes tag library -->
        <name>sayHello</name>              <!-- Defines tag name -->
        <tag-class>tag.SayHelloTag</tag-class> <!-- Specify tag handler class-->
        </tag>
        <body-content>empty</body-content>
        <attribute>                         <!-- Declares an attribute -->
            <name>var</name>
            <required>true</required>
        </attribute>
    </tag>
</taglib>
```

Implementing Simple Tags



JSP Expression Language

- Enables access to application data stored in JavaBeans component
- Is simple to implement as compared to custom tags and scriptlets
- Syntax (JSP and XML)
 - \${expression}
- Examples:
 - \${1.2 + 2.3}
 - \${4.0 >= 3}
 - \${customer.firstName}

Expression Language Implicit Objects

The expression language provides the following implicit objects:

- **pageScope**
- **requestScope**
- **sessionScope**
- **param**
- **header**
- **pageContext**

First.jsp

```
...
    First Name: <input type='text' name='Name' />
    Last Name:  <input type='text' name='Address' />
                <input type='submit' value='Submit' />
...

```

Second.jsp

```
...
    Name is : ${param.Name}
    Address is : ${param.Address}
...

```

JDeveloper and JSPs

- The JSP Wizard in JDeveloper is used to create JSPs containing skeleton code.
- Structure Pane helps to ensure that the JSP and HTML tags are properly formatted.
- Tag Insight automatically inserts end tags after starting a scriptlet.
- The JSP code is automatically created and recompiled.
- JDeveloper increases productivity while debugging JSPs:
 - Automatically includes source Java files such as your JavaBean source
 - Enables you to set breakpoints and watch expressions in JSPs

Summary

In this lesson, you should have learned how to:

- Identify the differences and similarities between JSPs and servlets
 - Use declarations, expressions, scriptlets, and implicit objects on JSP pages
 - Use JavaBeans with JSPs
 - Create a JSP segment
 - Explain the use of JSP tag files
 - Run and debug a JSP-based application



Practice 1: Overview

- This practice covers the following topics:
 - Using SQL Developer
 - Selecting all data from different tables
 - Describing the structure of tables
 - Performing arithmetic calculations and specifying column names

