FPT UNIVERSITY
DREAM OF INNOVATION

JavaServer Pages (JSP)

Objectives

- JavaServer Pages (JSP)
  - What JSP Technology is and how you can use it.
  - JSP Page
  - Translation Time
  - How JSP Works?
  - JSP Elements
  - JSP Predefined Variable – Implicit Objects
  - JSP Lifecycle

Objectives

- Dispatching Mechanisms
  - <jsp:forward>
  - <jsp:include>
- JSPs in XML

JavaServer Pages (JSP)

- What JSP Technology is and how you can use it.
  - JavaServer Pages (JSP) technology provides a simplified, fast way to create web pages that display dynamically-generated content.
  - The JSP 1.2 specification is an important part of the Java 2 Platform, Enterprise Edition. Using JSP and Enterprise JavaBeans technologies together is a great way to implement distributed enterprise applications with web-based front ends.
  - The first place to check for information on JSP technology is *http://java.sun.com/products/jsp*/

JSP Page

- A JSP page is a page created by the web developer that includes JSP technology-specific tags, *declarations, and possibly scriptlets*, in combination with other *static HTML or XML tags*.

- A JSP page has the extension *.jsp*; this signals to the web server that the *JSP engine* will process elements on this page.

- Pages built using JSP technology are typically implemented using a translation phase that is performed once, the first time the page is called. *The page is compiled into a Java Servlet class* and remains in server memory, so subsequent calls to the page have very fast response times.

Translation Time

- A JSP application is usually a collection of JSP files, HTML files, graphics and other resources.
- A JSP page is compiled when your user loads it into a Web browser

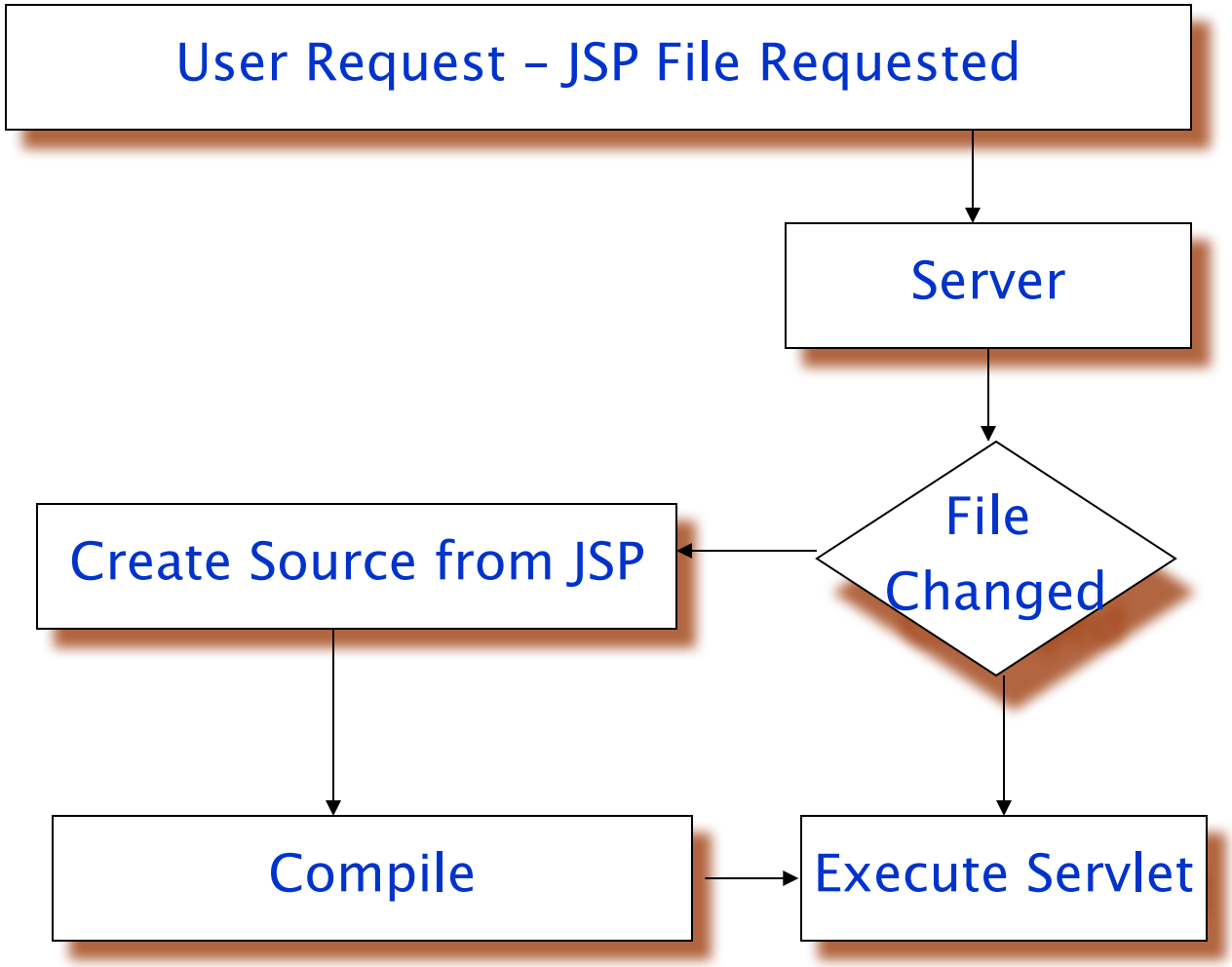
1. *When the user loads the page for the first time, the files that make up the application are all translated together, without any dynamic data, into one Java source file (a .java file)*

2. *The .java file is compiled to a .class file. In most implementations, the .java file is a Java servlet that complies with the Java Servlet API.*

Simple JSP Page

```
<HTML>
    <H1> First JSP Page </H1>
    <BODY>
     <%
          out.println("Welcome to JSP world");
      %>
     </BODY>
</HTML>
```

How JSP Works?



JSP Elements

- Declarations

```
<%! code %>
```

- Expressions

```
<%= expression%>
```

- Scriplets

```
<% code %>
```

HTML Comment

- Generates a comment that is sent to the client.
  - Syntax

```
<!--  comment [ <%= expression %> ] -->
```

  - Example:

```
<!-- This page was loaded on
     <%= (new java.util.Date()).toLocaleString() %>
 -->
```

Declaration

- Declares a variable or method valid in the scripting language used in the JSP page.
  - Syntax

```
<%! declaration; [ declaration; ]+   ... %>
```

  - Examples

```
<%! String destin; %>
<%! public String getDestination()
  {return destin;}%>
<%! Circle a = new Circle(2.0); %>
```

  - **You can declare any number of variables or methods within one declaration**

element, as long as you end each declaration with a semicolon.

  - **The declaration must be valid in the Java programming language.**

Declaration Demo

Demo\JSP1\dec.jsp

Expression

- Contains an expression valid in the scripting language used in the JSP page.
  - Syntax

```
<%= expression %>

<%! String name = new String("JSP World"); %>
<%! public String getName() { return name; } %>

<B><%= getName() %></B>
```

  - Description:

    An expression element contains a scripting language expression that is evaluated, converted to a String, and inserted where the expression appears in the JSP file.
  - Because the value of an expression is converted to a String, you can use an expression within a line of text, whether or not it is tagged with HTML, in a JSPfile. Expressions are evaluated from left to right.

Expression Demo

Demo\JSP1\exp.jsp

Scriptlet

- Contains a code fragment valid in the page scripting language.
  - Syntax

```
<% code fragment %>
```

```
<%
  String var1 = request.getParameter("name");
  out.println(var1);
%>
```

- This code will be placed in the generated servlet **method:**

```
_jspService()
```

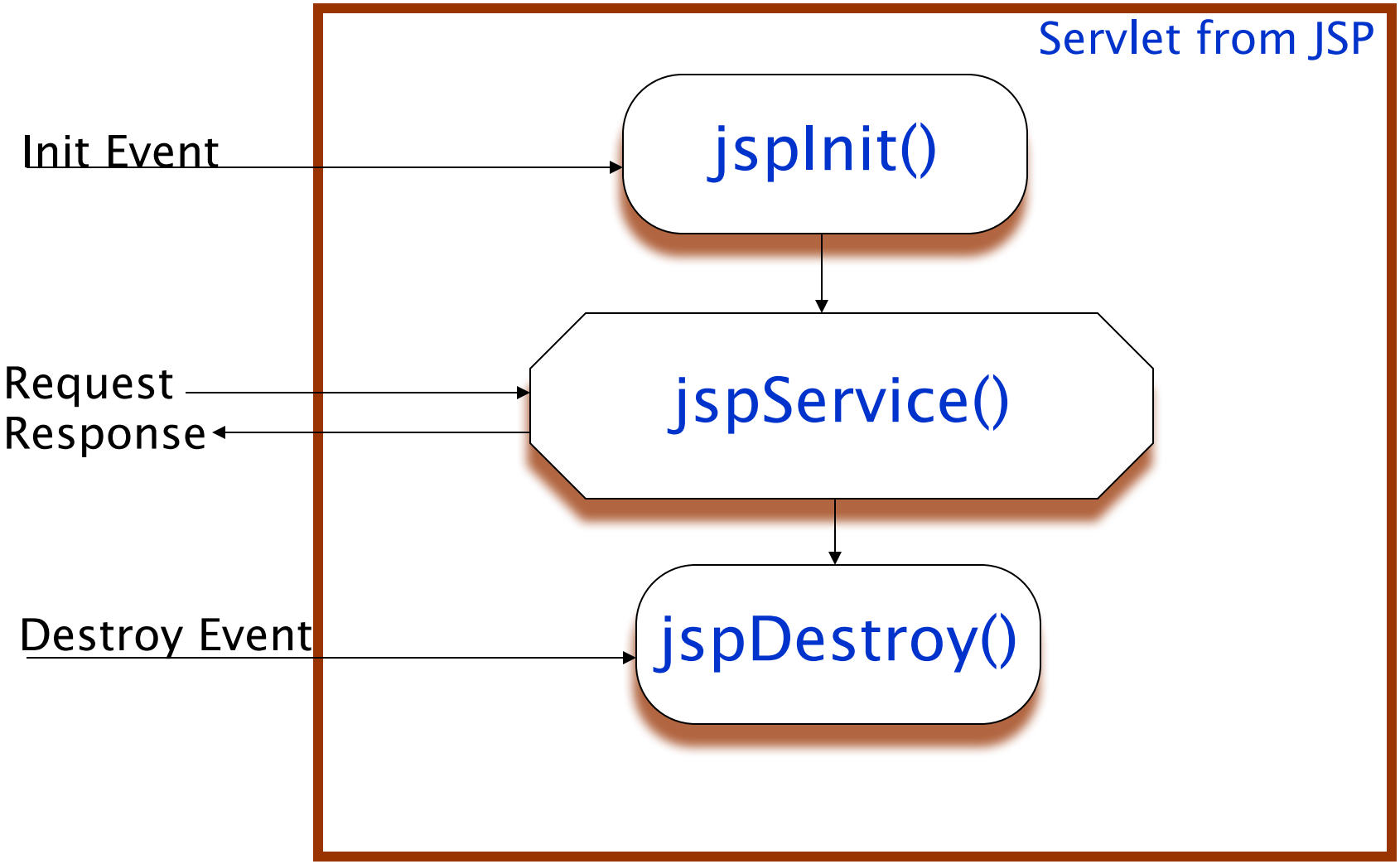
Scriplet Demo

Demo\JSP1\scriptlet.jsp

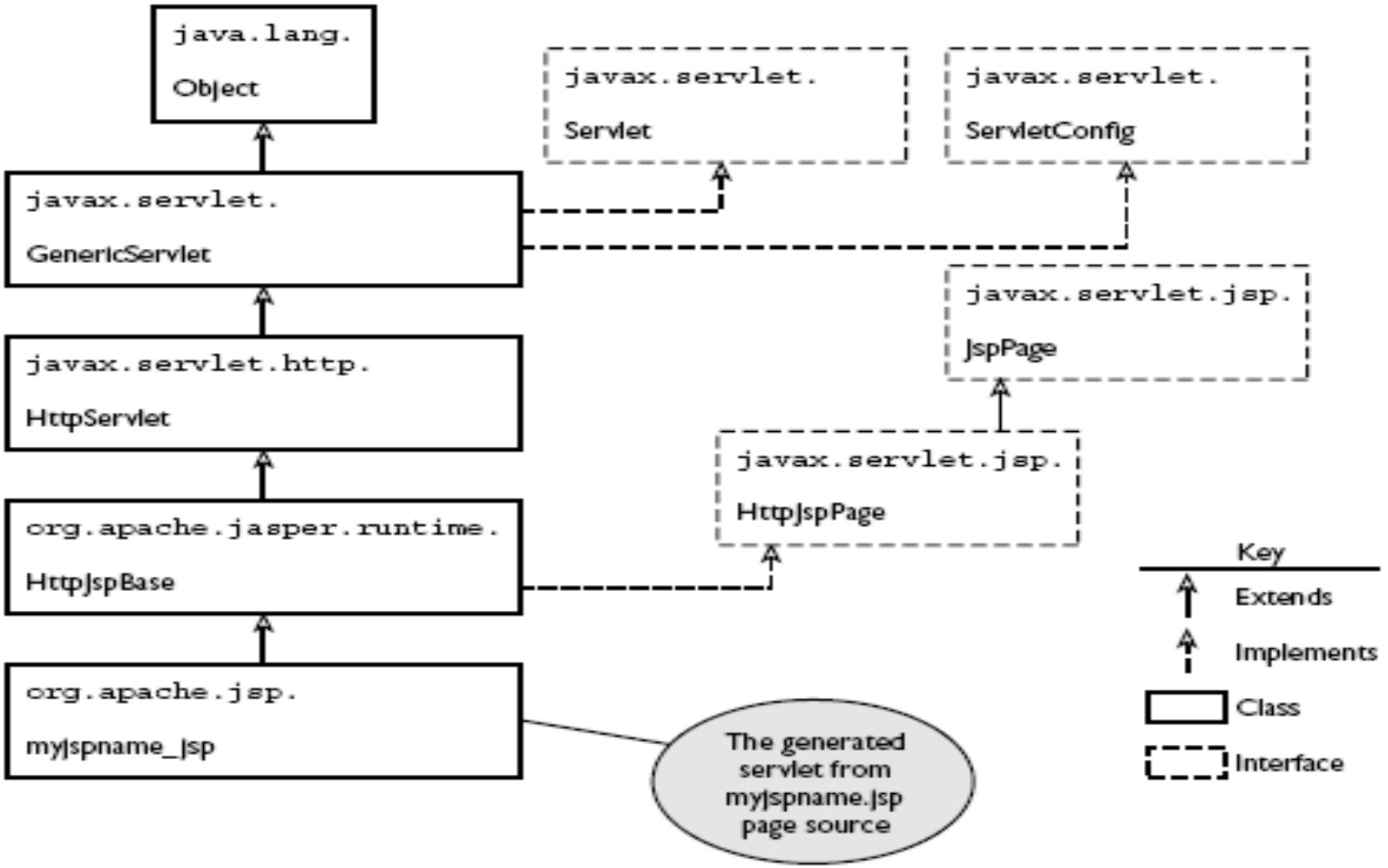
JSP Predefined Variable – Implicit Objects

- *request* – Object of HttpServletRequest (request parameters, HTTP headers, cookies
- *response* – Object of HttpServletResponse
- *out* - Object of PrintWriter buffered version JspWriter
- *session* - Object of HttpSession associated with the request
- *application* - Object of ServletContext shared by all servlets in the engine
- *config* - Object of ServletConfig
- *pageContext* - Object of PageContext in JSP for a single point of access
- *page* – variable synonym for this object

JSP Lifecycle



Class Hierarchy Generated Servlet



JSP Directives

- Directives, which—like scripting elements—are pieces of JSP tag-like syntax. Like an XML or HTML tag, directives have attributes that dictate their meaning and effect. In almost all cases, the effects produced by directives can't be replicated using expressions, scriptlets, or declarations.
- We'll consider the three directives:
  - page,
  - include,
  - and taglib.

The page Directive

- You can include a page directive anywhere in your JSP page source: beginning, middle, or end.
- An example of how one looks:

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

The page Directive-*import attribute*

- **Import** attribute
  - Use the import attribute to create import statements in the generated servlet source produced by the JSP container's translation phase.

  Example:

  <%@ page import="java.util.StringTokenizer" %>
- JSP Container will import 04 packages at translation time:
  - javax.servlet
  - javax.servlet.http
  - javax.servlet.jsp
  - java.lang

The page Directive - *contentType attribute*

- The value of the **contentType** attribute is a String that specifies the MIME type of the response to be sent back.

```
<%@ page contentType="image/gif" %>
<%@ page import="java.io.*" %>
<% /* response.setContentType("image/gif"); */
String path = getServletContext().getRealPath("tomcat.gif");
File imageFile = new File(path);
long length = imageFile.length();
response.setContentLength((int) length);
OutputStream os = response.getOutputStream();
BufferedInputStream bis =
new BufferedInputStream(new FileInputStream(imageFile));
int info;
while ((info = bis.read()) > -1) {
            os.write(info);
}
os.flush();
 %>
```

FPT UNIVERSITY
DREAM OF INNOVATION

Other attributes of the page directive

language	<%@ page language="Java" %> Denotes the scripting language. "Java" is the only value supported by all J2EE-compliant containers. Your container might support something different and specialized, but your page won't in all likelihood be portable to other containers then.
extends	<%@ page extends="com.osborne.webcert.MyBaseJSPServlet" %> If you want to override the base servlet that your container provides when generating servlets, you can—and substitute your own through this directive. Do so at your own peril; your container may not like you for it.
buffer, autoFlush	<%@ page buffer="none" autoFlush="true" %> You can use these attributes to control whether or not you have a buffer (you can specify a size in kilobytes), and how this buffer is flushed. The example above causes the response to be flushed as soon as it is generated.
isThreadSafe	<%@ page isThreadSafe="true" %> Causes the generated JSP servlet to implement the deprecated SingleThreadModel interface: not recommended. The default is, of course, false.
info	<%@ page info="My Clever Hacks Page" %> Use this attribute to publish information about your JSP page, accessible through the `getServletInfo()` method.
errorPage, isErrorPage	<%@ page errorPage="errorPage.jsp" %> <%@ page isErrorPage="true" %> Use errorPage to set a URL pointing to another JSP within the same web application. Should an exception occur, your users will be forwarded to this other JSP. The page you forward to must have "isErrorPage" set to true, and is the only sort of page to have access to the implicit variable exception.

The include Directive

- A JSP source file and the pages it includes through the include directive are collectively referred to as a **"translation unit."**
- The file type you include doesn't have to be JSP page source, nor does it have to have a .jsp extension. Once included, **the file contents must make sense to the JSP translation process**, but this gives scope to include entire HTML or XML documents, or document fragments.

The include Directive…

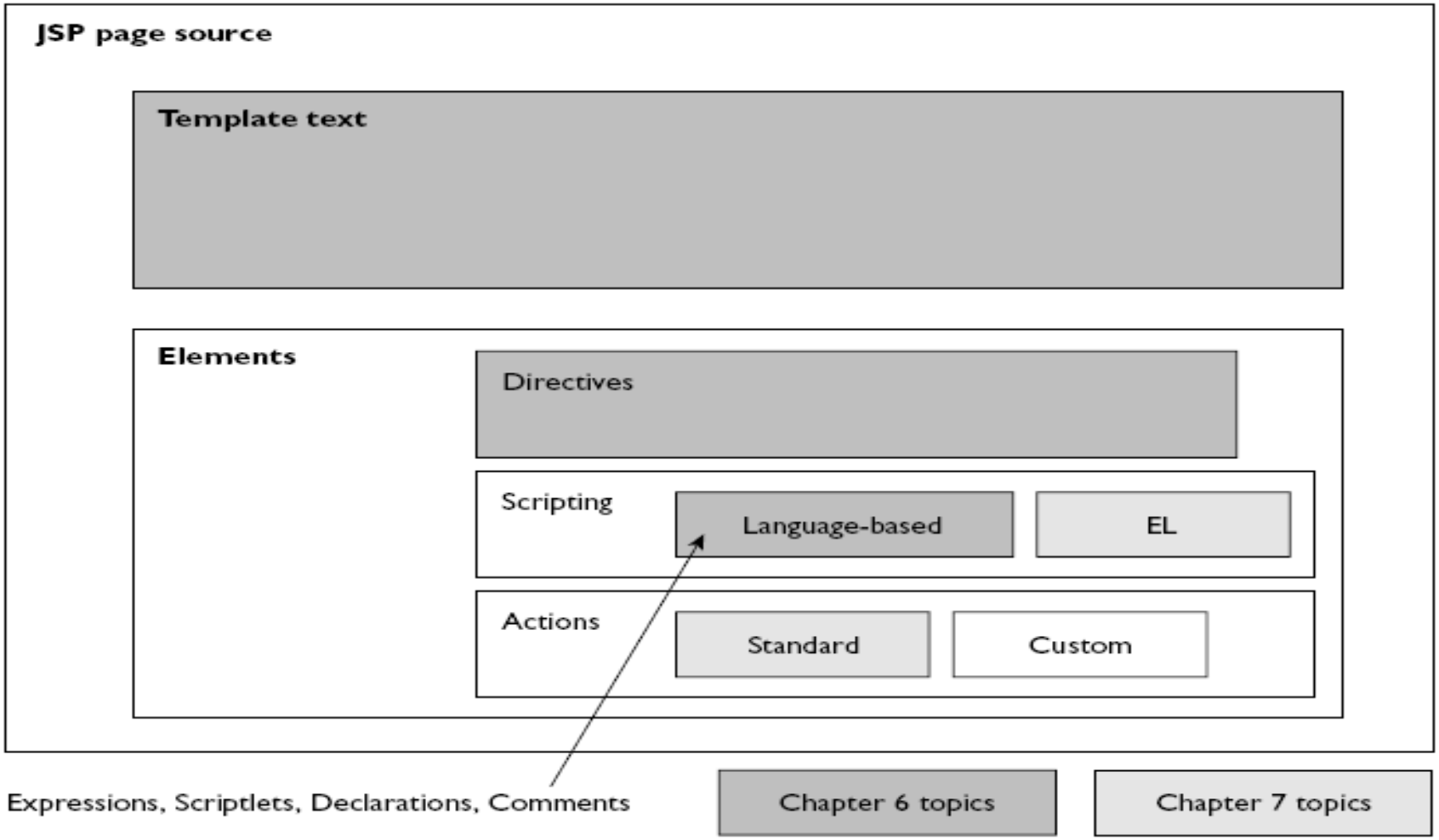
<%@ include file="**relativeURL**" %>

```
<html>
<body>
        <%@ include file="header.jsp" %>
        <h1>Including Page</h1>
        <%@ include file="footer.html" %>
</body>
</html>
```

The taglib Directive

- The *taglib* directive makes **custom actions** available in the JSP page by referencing a tag library (will study in the next session).

JSP Source Elements



Standard Actions

- Standard actions are used for the same purpose as Java language-based scripting: Most if not all the goals that you can achieve with standard actions are achievable with other scripting elements.
- So why use them?
  - The answer is that they get the job done more elegantly. They often provide an alternative to inserting java code into your neatly designed presentation page.

Standard actions general syntax

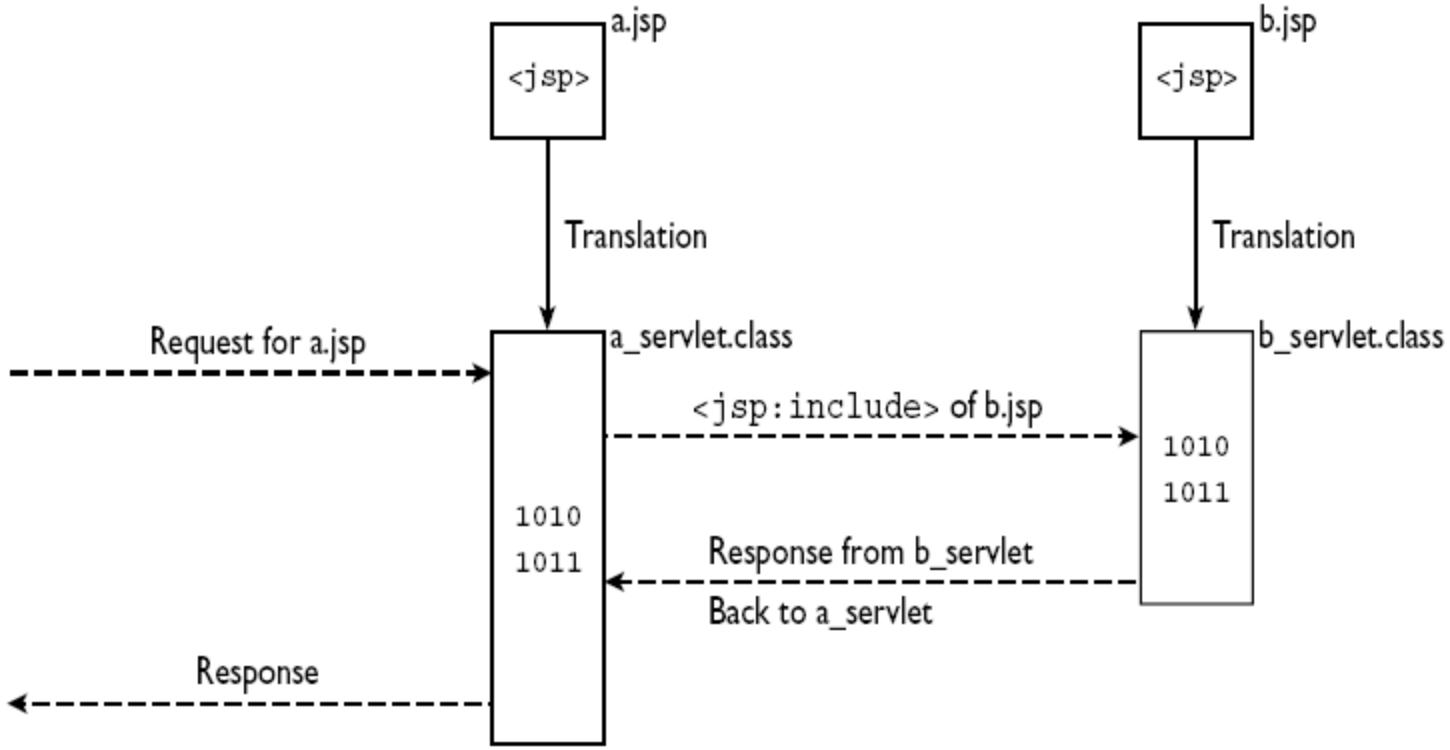
```
<prefix:tagname firstAttribute="value"
   secondAttribute="value">
   ...
</prefix:tagname>
```

Dispatching Mechanism <jsp:include>

- The standard action <jsp:include> can be used to include the response from another file within your JSP page output.
- You specify the file whose response should be included with the page attribute, like this:

  <jsp:include page="pageToInclude.jsp" />
- The file whose response should be included has to reside somewhere in your web application but doesn't have to be present until the page is actually requested.

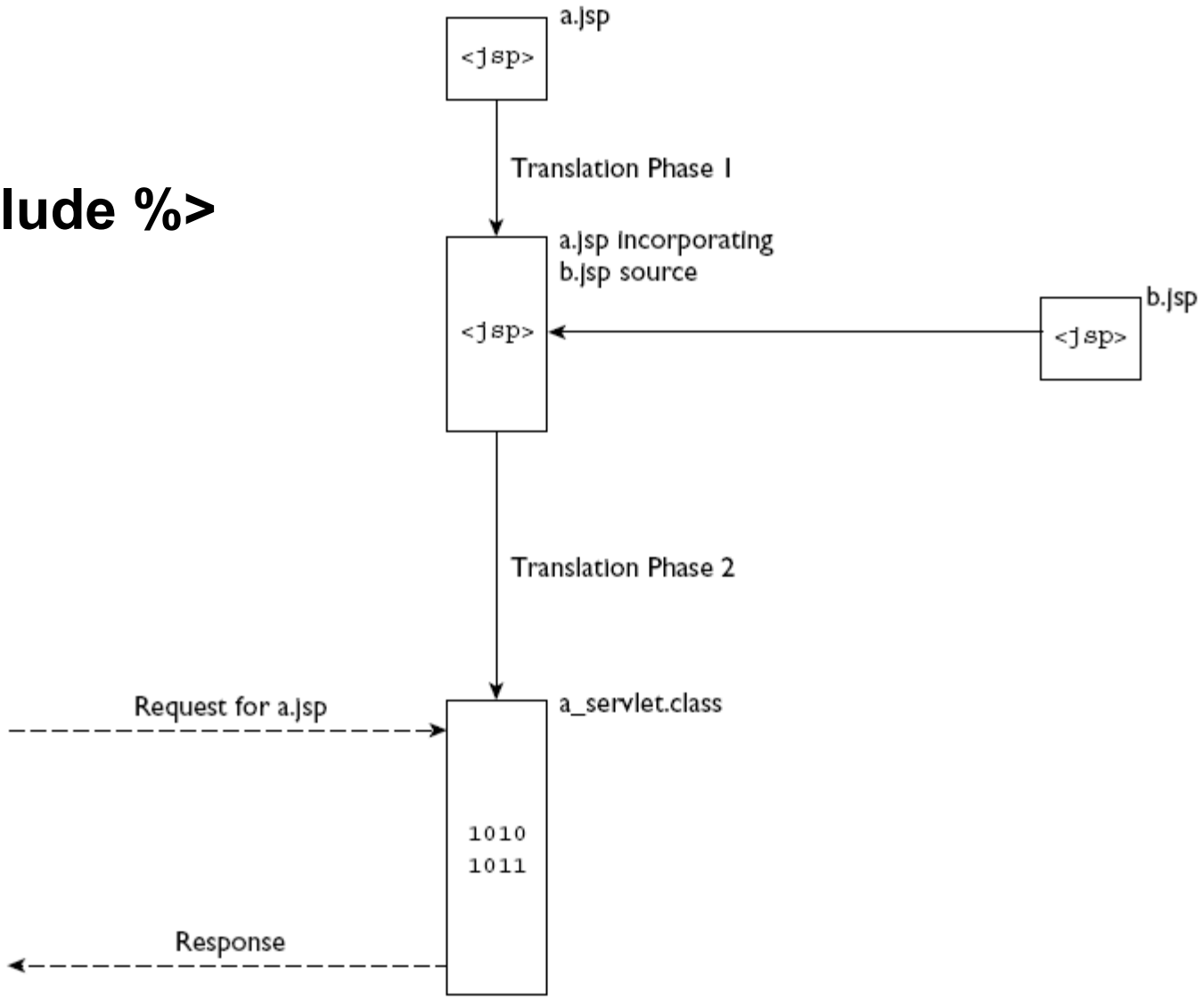
<jsp:include> vs. <%@ include %>



<jsp:include>

<jsp:include> vs. <%@ include %>

<%@ include %>



Dispatching Mechanism <jsp:forward>

- The purpose of this standard action is to forward processing to another resource within the web application.

```
<jsp:forward page="destinationPage">
  <jsp:param name=".." value=".." />
  …
</jsp:forward>
```

<jsp:forward> demo

http://localhost:8080/stdActions/select.html

Summary

- **JavaServer Pages (JSP)**
  - **What JSP Technology is and how you can use it.**
  - **JSP Page**
  - **Translation Time**
  - **How JSP Works?**
  - **JSP Elements**
  - **JSP Predefined Variable – Implicit Objects**
  - **JSP Lifecycle**
- **JSP Directives**
  - **The page Directive**
  - **The include Directive**
  - **The taglib Directive**

Summary

- **Dispatching Mechanism**
  - **<jsp:forward>**
  - **<jsp:include>**

Q&A