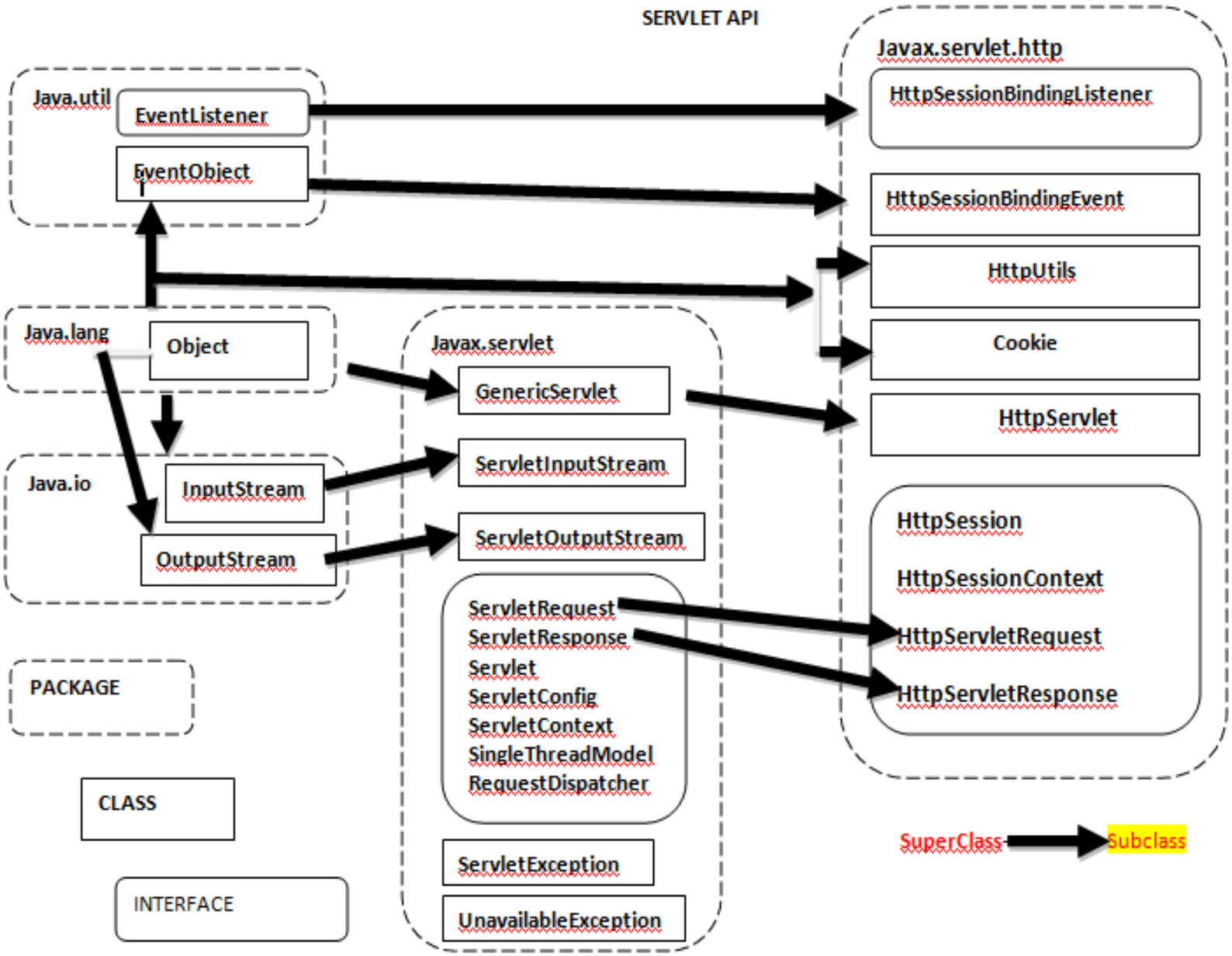
JSP-JAVA SERVER PAGES

Syllabus…

- **Servlets:** Introduction
- Web application Architecture
- Http Protocol & Http Methods
- Web Server & Web Container
- Servlet Interface
- GenericServlet
- HttpServlet
- Servlet Life Cycle
- ServletConfig
- ServletContext
- Servlet Communication
- Session Tracking Mechanisms


SERVLET API
Javax.servlet.http
Java.util
EventListener
EventObject
HttpSessionBindingListener
HttpSessionBindingEvent
HttpUtils
Cookie
Java.lang
Object
Javax.servlet
GenericServlet
HttpServlet
Java.io
InputStream
OutputStream
ServletInputStream
ServletOutputStream
HttpSession
HttpSessionContext
HttpServletRequest
HttpServletResponse
ServletRequest
ServletResponse
Servlet
ServletConfig
ServletContext
SingleThreadModel
RequestDispatcher
PACKAGE
CLASS
INTERFACE
ServletException
UnavailableException
SuperClass
Subclass

jsp

- **JSP:** Introduction
- JSP LifeCycle
- JSP Implicit Objects & Scopes
- JSP Directives
- JSP Scripting Elements
- JSP Actions: Standard actions and customized actions

JSP Overview - History

- Earlier dynamic web pages were developed with CGI
  - Web Applications eventually outgrew CGI because of the following:
    - Datasets became large
    - Increased traffic on sites
    - Trouble maintaining state and session information
    - Performance bottlenecks
    - Can involve proprietary APIs

JSP Overview - Software

- With a foundation in Java, JSP
  - Threads incoming requests
  - Supports the definition of tags that abstract functionality within a page in a library
  - Interacts with a wide array of databases
  - Serves dynamic content in a persistent and efficient manner
    - Initially the server automatically separates HTML and JSP code, compiles the JSP code, and starts the servlet, from then on the servlets are stored in the web server's memory in Java byte code
  - Extends the same portability, large class library, object-oriented code methodology, and robust security measures that have made Java a top choice
  - Does not require a Java programmer

JSP

- JSP is another technology for developing web applications.
- It is a template for web page that uses java code to generate an HTML document dynamically
- It is not meant to replace servlet.
- JSP is an extension of servlet technology, & it is common practice to use both in same web applications.
- JSP run in server-side component known as a JSP container which translates them into equivalent Servlet
- JSP uses the same techniques as those found in servlet programming.

Java Server Pages(JSP)

- JSP pages typically comprise of:
- Static HTML/XML components.
- Special JSP tags
- Optionally, snippets of code written in the Java programming language called "scriptlets."

JSP Advantages

- Separation of static from dynamic content.
- Write Once Run Anywhere
- Dynamic content can be served in a variety of formats.
- Recommended Web access layer for n-tier architecture.

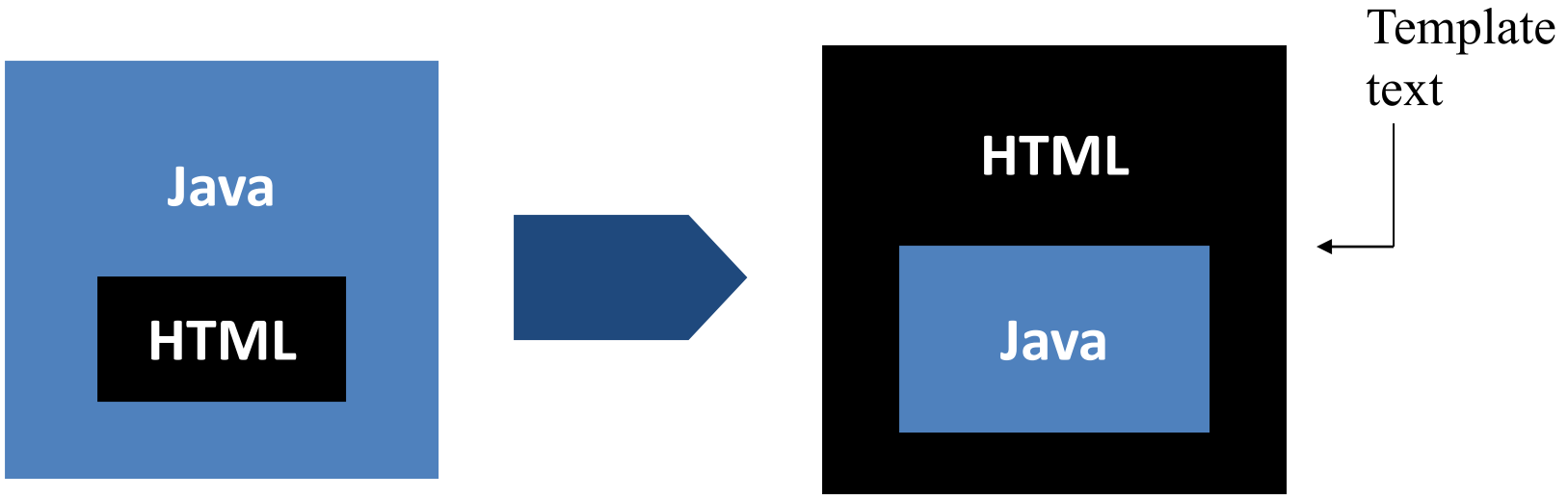
  Completely leverages the Servlet API.
- JSP uses simplified scripting language based syntax for embedding HTML into JSP.
- JSP containers provide easy way for accessing standard objects and actions.
- JSP reaps all the benefits provided by JAVA servlets and web container environment, but they have an added advantage of being simpler and more natural program for web enabling enterprise developer
- JSP use HTTP as default request /response communication paradigm and thus make JSP ideal as Web Enabling Technology.

Why Use JSP

- JSP is built on top of servlets, so it has all the advantages of servlets
- JSP is compiled into its corresponding servlet when it is requested at the first time
- The servlet stays in memory, speeding response times
- Extensive use of Java API (Applications Programming Interface) for networking, database access, distributed objects, and others like it
- Powerful, reusable and portable to other operating systems and Web servers
- servlets is a programmatic technology requiring significant developer expertise,
- JSP can be used not only by developers, but also by page designers
- JSP is the inherent separation of presentation from content facilitated

Relationships

- In servlets,
  - HTML code is printed from java code
- In JSP pages
  - Java code is embadded in HTML code

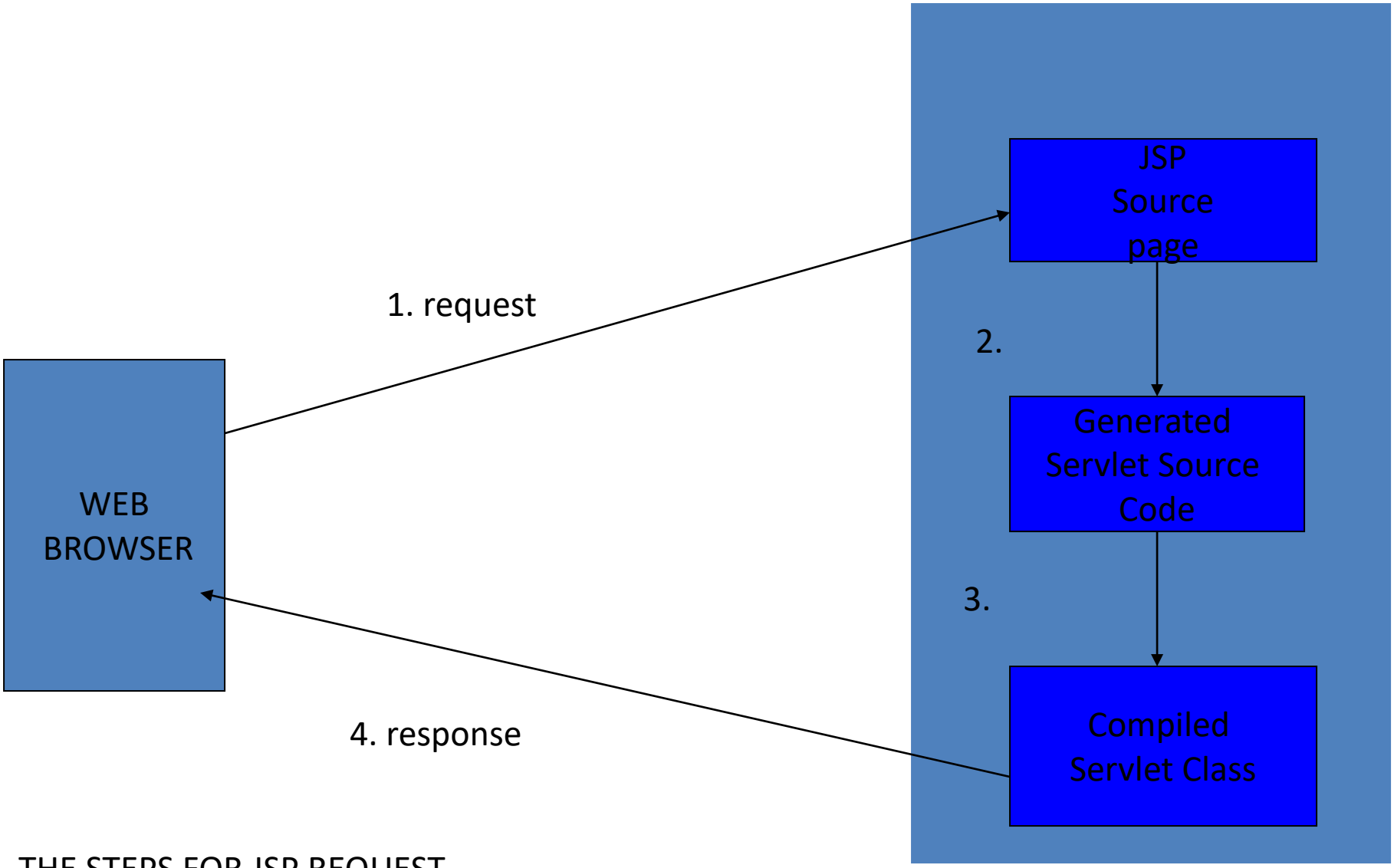


JSP

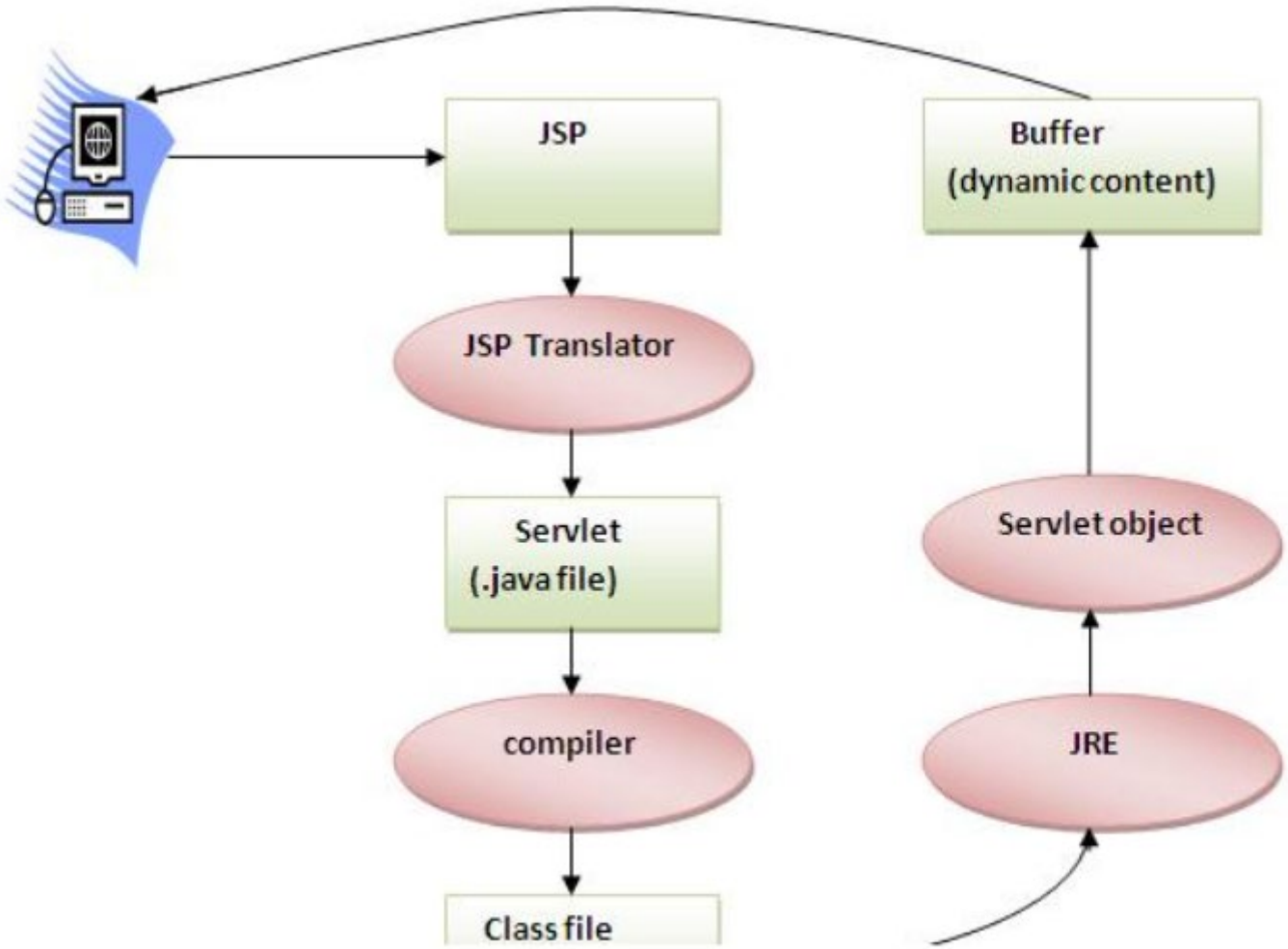
- They are automatically recompiled when necessary
- They exist in web server document space, no special URL mapping is required to address them
- They are like HTML pages ,they have compatibility with web development tools

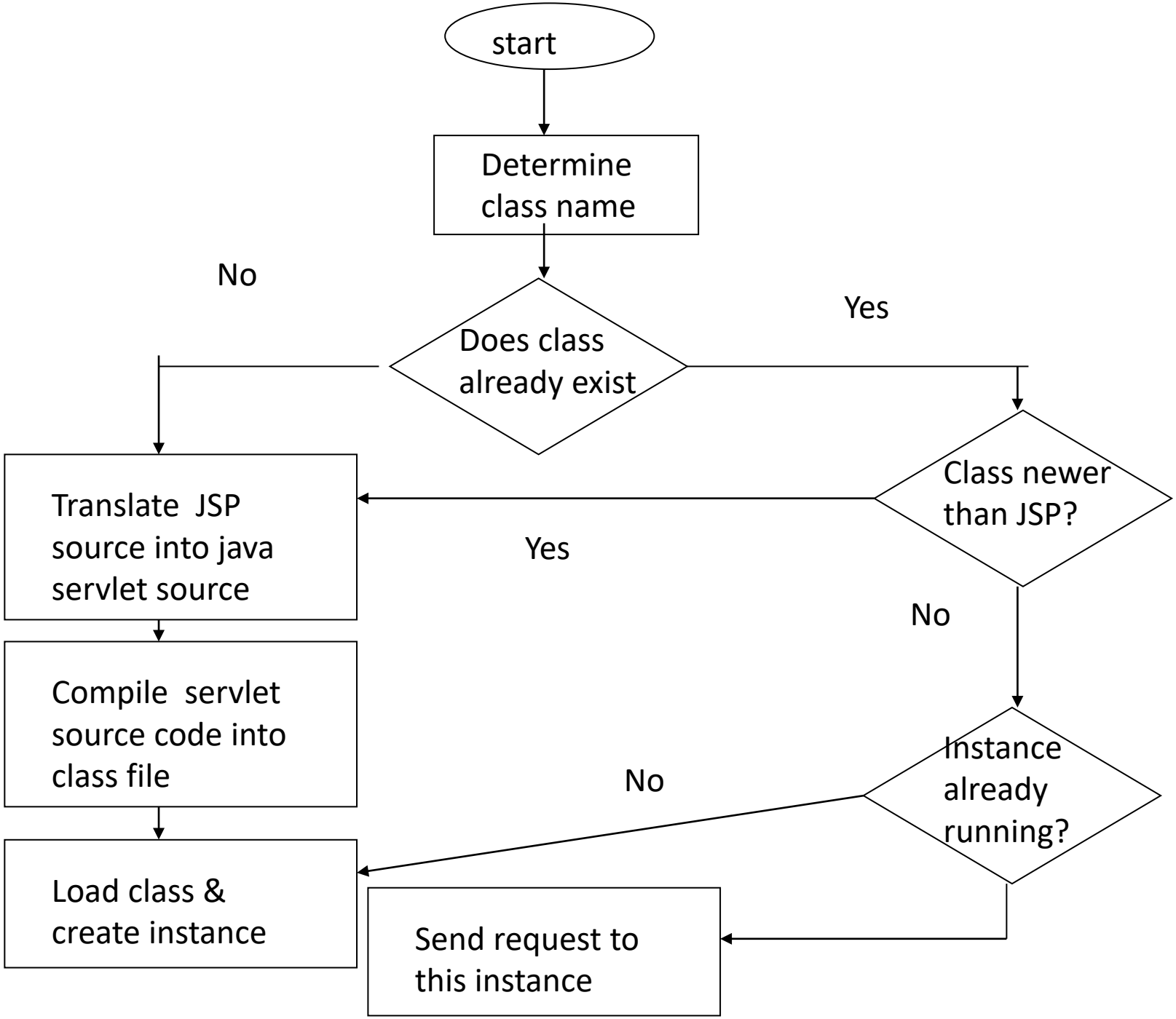
How it works

- Request for JSP , container determines name of class
- If it does not exist or older one then it creates Java Source Code
- Compiles it , loads its instance if not loaded
- If JSP file is modified then container retranslates and recompiles it



THE STEPS FOR JSP REQUEST

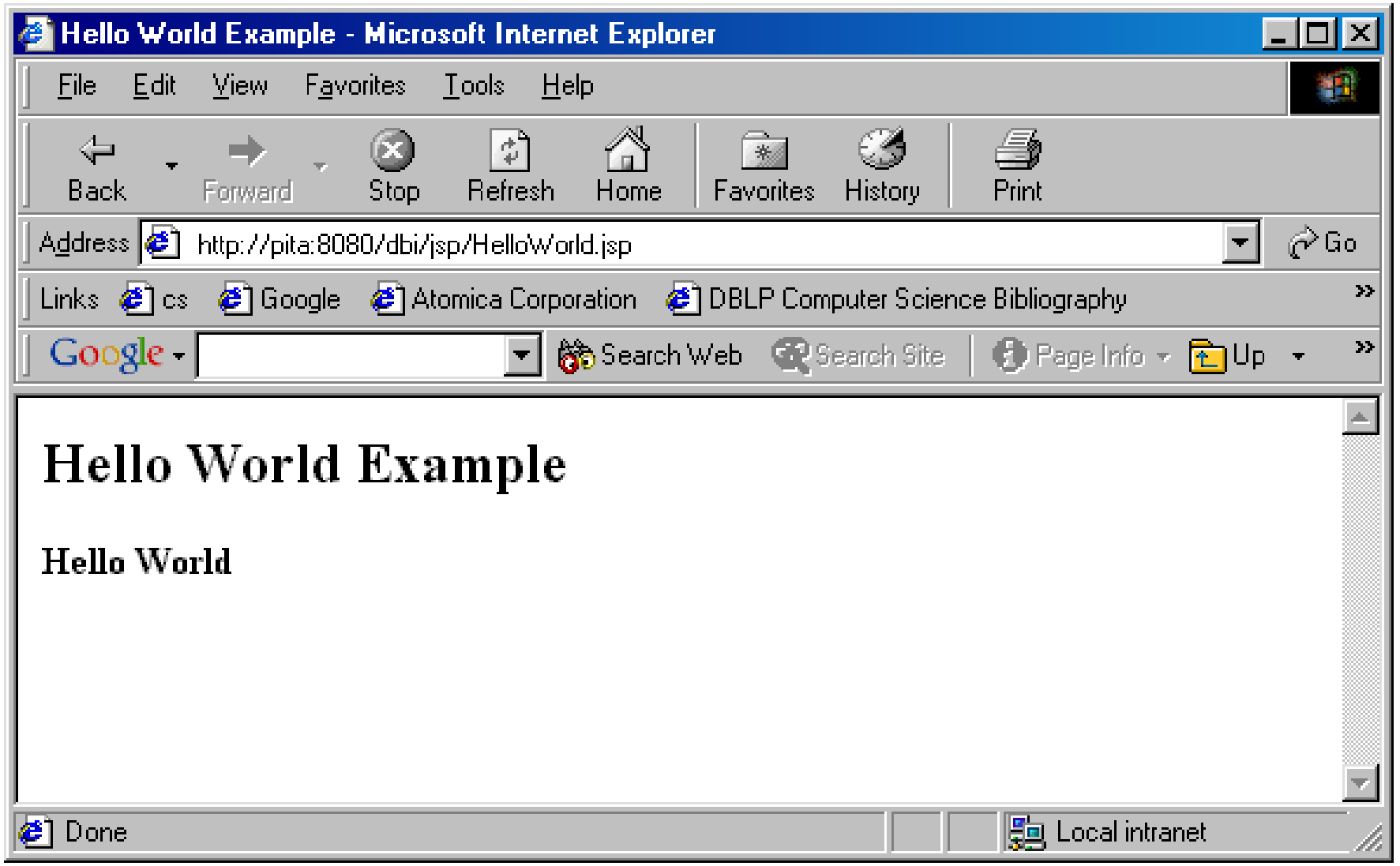

JSP
Buffer
(dynamic content)
JSP Translator
Servlet
(.java file)
Servlet object
compiler
JRE
Class file


start
Determine class name
No
Does class already exist
Yes
Translate JSP source into java servlet source
Class newer than JSP?
Yes
No
Compile servlet source code into class file
Instance already running?
No
Load class & create instance
Send request to this instance

An Example

```
<HTML>
<HEAD>
<TITLE>Hello World Example</TITLE>
</HEAD>

<BODY>
<H2>Hello World Example</H2>
<B><% out.println("Hello World"); %></B>
</BODY>
</HTML>
```


Hello World Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Favorites History Print
Address http://pita:8080/dbi/jsp/HelloWorld.jsp
Go
Links cs Google Atomica Corporation DBLP Computer Science Bibliography
Google Search Web Search Site Page Info Up
Hello World Example
Hello World
Done
Local intranet

JSP Architecture

- The JSP page implementation class file extends HttpJspBase,
- which in turn implements the Servlet interface.
- the service method of this class, _jspService(), essentially inlines the contents of the JSP page.
- Although _jspService() cannot be overridden,
- the developer can describe initialization and
- destroy events by providing implementations for the jspInit() and jspDestroy() methods within their JSP pages.

- Once this class file is loaded within the servlet container,
- the _jspService() method is responsible for replying to a client's request.
- By default, the _jspService() method is dispatched on a separate thread
- by the servlet container in processing concurrent client requests,

JSP -API

The JSP API consists of two packages:

1. javax.servlet.jsp
2. javax.servlet.jsp.tagext

javax.servlet.jsp package

The javax.servlet.jsp package has two interfaces and classes.The two interfaces are as follows:

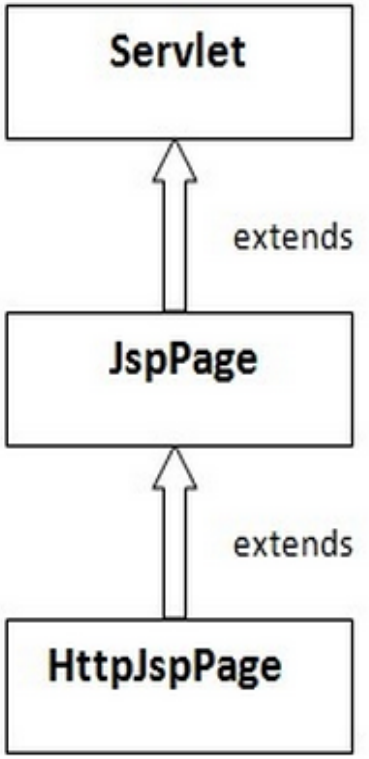
1. JspPage
2. HttpJspPage

The classes are as follows:

- JspWriter
- PageContext
- JspFactory
- JspEngineInfo
- JspException
- JspError

The JspPage interface

According to the JSP specification, all the generated servlet classes must implement the JspPage interface. It extends the Servlet interface. It provides two life cycle methods.



Methods of JspPage interface

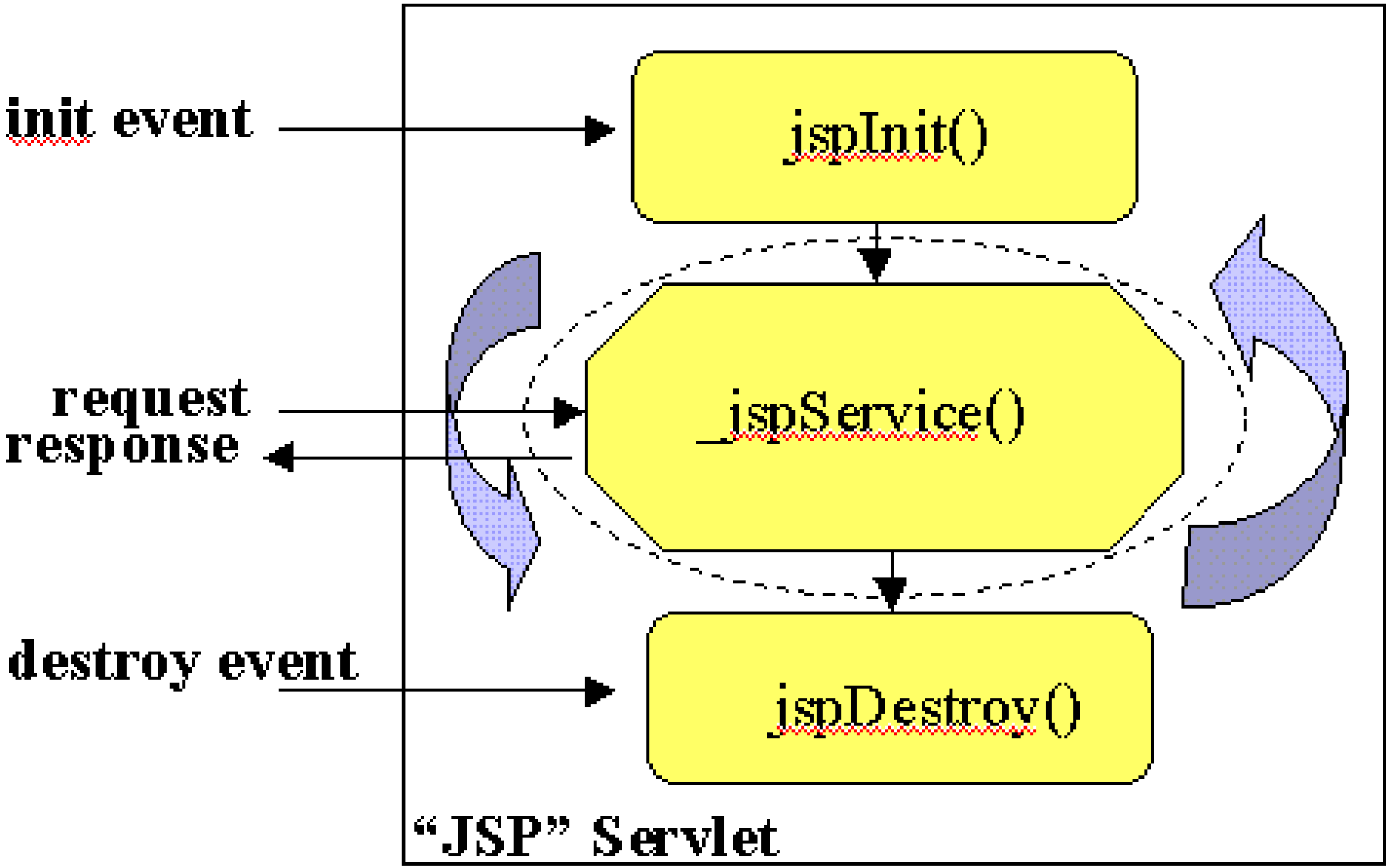
1. **public void jspInit():** It is invoked only once during the life cycle of the JSP when JSP page is requested firstly. It is used to perform initialization. It is same as the init() method of Servlet interface.
2. **public void jspDestroy():** It is invoked only once during the life cycle of the JSP before the JSP page is destroyed. It can be used to perform some clean up operation.

The HttpJspPage interface

The HttpJspPage interface provides the one life cycle method of JSP. It extends the JspPage interface.

Method of HttpJspPage interface:

1. **public void _jspService():** It is invoked each time when request for the JSP page comes to the container. It is used to process the request. The underscore _ signifies that you cannot override this method.


init event
jspInit()
request
response
_jspService()
destroy event
jspDestroy()
"JSP" Servlet

Component of JSP

- JSP file can contain JSP elements, fixed template data or both .
- The elements which are called jsp tags make up the syntax & semantics of jsp.
- Template data includes parts that the jsp container does not understand such as html tags.

Introduction - JSP v/s Servlet

How is JSP different / similar to Servlet?

<table>
<tr><th>Servlets</th><th>JSP</th></tr>
<tr><td colspan="2">Handles dynamic data</td></tr>
<tr><td>Handles business logic</td><td>Handles presentation logic</td></tr>
<tr><td>Lifecylce methods
init() : can be overridden
service() : can be overridden
destroy() : can be overridden</td><td>Lifecylce methods
jspInit() : can be overridden
_jspService() : cannot be overridden
jspDestroy() : can be overridden</td></tr>
<tr><td>Html within java
out.println(“<htrml><body>”);
out.println(“Time is” + new Date());
out.println(“</body></html>”);</td><td>Java within html
<html><body>
Time is <%=new Date()%>
</body></html></td></tr>
<tr><td colspan="2">Runs within a Web Container</td></tr>
</table>

Introduction - Comments

- **JSP Comments**
  - <%-- {CODE HERE} --%>
  - Does not show the comments on the page
  - Does not show the comments in page source
  - Can only be used outside JSP Elements
- **HTML Comments**
  - <!- {CODE HERE} -->
  - Does not show the comments on the page
  - Shows the comments in page source
  - Can only be used outside JSP Elements
- **Single line Comments**
  - // {CODE HERE}
  - When put outside JSP Elements, shows comments on the page & source
  - When put inside Scriptlets/Declarations, does not show on page & source
  - Can be used inside scriptlets/declarations and outside JSP Elements

JSP Elements - Intro

- Need to write some Java in your HTML?
- Want to make your HTML more dynamic?

- **JSP Declarations** :: Code that goes outside the service method
- **JSP Scriptlets** :: Code that goes within the service method
- **JSP Expressions** :: Code inside expressions is evaluated
- **JSP Directives** :: Commands given to the JSP engine

JSP Scriptlet tag (Scripting elements)

In JSP, java code can be written inside the jsp page using the scriptlet tag. Let's see what are the scripting elements first.

Scripting elements

The scripting elements provides the ability to insert java code inside the jsp. scripting elements:

- scriptlet tag
- expression tag
- declaration tag

JSP scriptlet tag

A scriptlet tag is used to execute java source code in JSP. Syntax is as follows:

```
<%  java source code %>
```

✓*Simple Example of JSP scriptlet tag*

In this example, we are displaying a welcome message.

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

JSP expression tag

The code placed within expression tag is written to the output stream of the response. So you need not write out.print() to write data. It is mainly used to print the values of variable or method.

Syntax of JSP expression tag

```
<%=  statement %>
```

✓Example of JSP expression tag

In this example of jsp expression tag, we are simply displaying a welcome message.

```
<html>
<body>
<%= "welcome to jsp" %>
</body>

</html>
```

JSP declaration tag

The JSP declaration tag is used to declare fields and methods. The code written inside the jsp declaration tag is placed outside the service() method of auto generated servlet. So it doesn't get memory at each request.

Syntax of JSP declaration tag

The syntax of the declaration tag is as follows:

```
<%!  statement %>
```

```
<html>
<body>

<%! int data=50; %>
<%= "Value of the variable is:"+data %>

</body>
</html>
```

```
<html>
<body>

<%!
int cube(int n){
return n*n*n*;
}
%>

<%= "Cube of 3 is:"+cube(3) %>

</body>
</html>
```

JSP Elements - Example

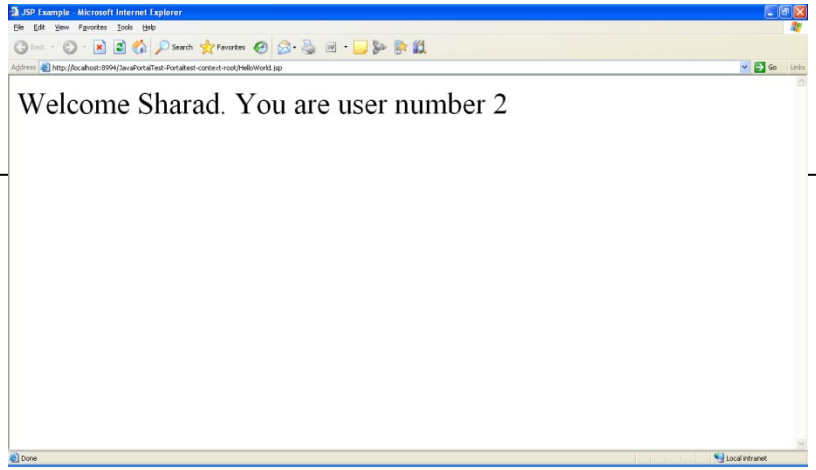
```
<html><head><title>JSP Elements Example</title>
 </head>
 <body>
  <%! int userCnt = 0; %>

  <%
    String name = "Sharad";
    userCnt++;
  %>
   <table>
    <tr><td>
      Welcome <%=name%>. You are user number <%=userCnt%>
    </td></tr>
   </table>
 </body>
</html>
```

Declarations

Scriptlets

Expressions



JSP Implicit Objects

There are 9 implicit objects available for the JSP page. The Auto Generated Servlet contains many objects like out, request, config, session, application etc. The 9 implicit objects are as follows:

JSP Implicit Object	Super Class	Annotation
request	HttpServletRequest	Provides HTTP request information.
response	HttpServletResponse	Send data back to the client
out	JspWriter	Write the data to the response stream
session	HttpSession	Track information about a user from one request to another
application	ServletContext	Data shared by all JSPs and servlets in the application.
pageContext	PageContext	Contains data associated with the whole page
config	ServletConfig	Provides servlet configuration data.
page	Object	Similar with “this” in Java
exception	Throwable	Exceptions not caught by application code.

1) out implicit object

For writing any data to the buffer, JSP provides an implicit object named out. It is the object of PrintWriter. In case of servlet you need to write:

```
PrintWriter out=response.getWriter();
```

But in JSP, you don't need to write this code.

✓*Example of out implicit object*

In this example we are simply displaying date and time.

index.jsp

```
<html>
<body>
<% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>
</body>
```

2) request implicit object:

In JSP, request is an implicit object of type HttpServletRequest.

Example of request implicit object:

index.html

```
<html>
<body>
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
</body>
</html>
```

welcome.jsp

```
<html>
<body>
<%
String name=request.getParameter("uname");
out.print("welcome "+name);
%>
</body>
</html>
```

3) response implicit object:

In JSP, response is an implicit object of type HttpServletResponse.

Example of response implicit object:

index.html

```
<html>
<body>
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
</body>
</html>
```

welcome.jsp

```
<html>
<body>
<%

response.sendRedirect("http://www.google.com");

%>
</body>
</html>
```

Remaining implicit objects

4) config implicit object:

In JSP, config is an implicit object of type ServletConfig. This object can be used to get configuration information for a particular JSP page.This variable information can be used for one jsp page only.

welcome.jsp

```
<html>
<body>
<%

out.print("Welcome "+request.getParameter("uname"));

String driver=config.getParameter("dname");
out.print("driver name is="+driver);


%>
</body>
</html>
```

5) application implicit object:

In JSP, application is an implicit object of type ServletContext.This object can be used to get configuration information from configuaration file(web.xml). This variable information can be used for all jsp pages.

Example of application implicit object:

index.html

```
<html>
<body>
<form action="welcome">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
</body>
</html>
```

welcome.jsp

```
<html>
<body>
<%

out.print("Welcome "+request.getParameter("uname"));

String driver=application.getParameter("dname");
out.print("driver name is="+driver);

%>
</body>
</html>
```

6) session implicit object:

In JSP, session is an implicit object of type HttpSession.The Java developer can use this object to set,get or remove attribute or to get session information.

Example of session implicit object:

index.html

```
<html>
<body>
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
</body>
</html>
```

welcome.jsp

```
<html>
<body>
<%

String name=request.getParameter("uname");
out.print("Welcome "+name);

session.setAttribute("user",name);

<a href="second.jsp">second jsp page</a>

%>
</body>
</html>
```

second.jsp

```
<html>
<body>
<%

String name=session.getAttribute("user");
out.print("Hello "+name);

%>
</body>
</html>
```

7) pageContext implicit object:

In JSP, pageContext is an implicit object of type PageContext class.The pageContext object can be used to set,get or remove attribute from one of the following scopes:

- page
- request
- session
- application

In JSP, page scope is the default scope.

index.html

```
<html>
<body>
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
</body>
</html>
```

welcome.jsp

```
<html>
<body>
<%

String name=request.getParameter("uname");
out.print("Welcome "+name);

pageContext.setAttribute("user",name,PageContext.SESSION_SCOPE);

<a href="second.jsp">second jsp page</a>

%>
</body>
</html>
```

second.jsp

```
<html>
<body>
<%

String name=pageContext.getAttribute("user",PageContext.SESSION_SCOPE);
out.print("Hello "+name);

%>
</body>
</html>
```

8) page implicit object:

In JSP, page is an implicit object of type Object class.This object is assigned to the reference of auto generated servlet class. It is written as:

Object page=this;

For using this object it must be cast to Servlet type.For example:

<% (HttpServlet)page.log("message"); %>

Since, it is of type Object it is less used because you can use this object directly in jsp.For example:

<% this.log("message"); %>