JSP
(Java Server Pages)

What do you mean by Static & Dynamic Contents?

- **Static contents**
  - Typically static HTML page
  - Same display for everyone
- **Dynamic contents**
  - Content is dynamically generated based on conditions
  - Conditions could be
    - User identity
    - Time of the day
    - User entered values through forms and selections

What is JSP Page?

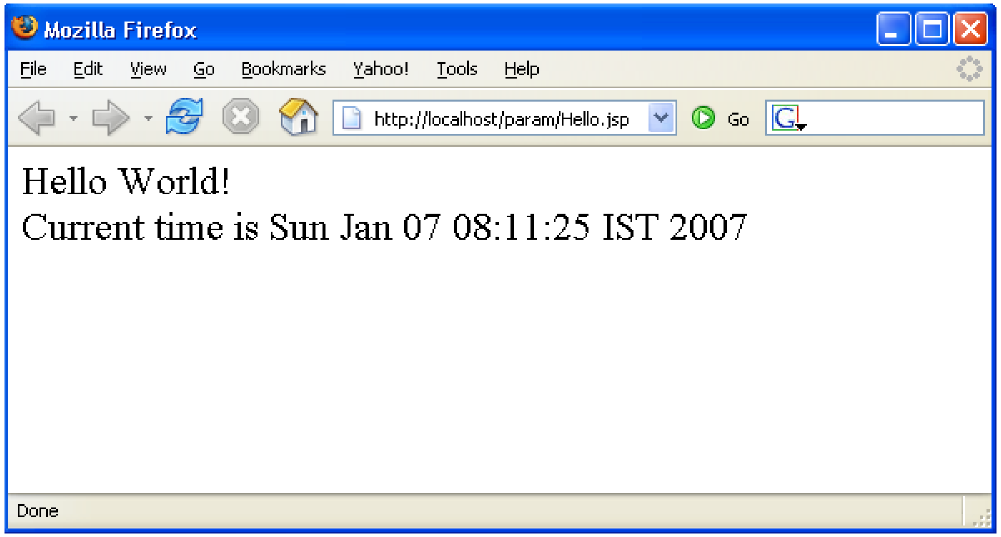
- **A text-based document capable of returning both static and dynamic content to a client browser**
- **Static content and dynamic content can be intermixed**
- **Static content**
  - HTML, XML, Text
- **Dynamic content**
  - Java code
  - Displaying properties of JavaBeans
  - Invoking business logic defined in Custom tags
- **The Java code is enclosed between the construct <% and %>, called a jsp “tag”**

```
<% out.println(“This is a jsp demo”);%>
```

A Simple JSP Page

(Blue: static, Red: Dynamic contents)

```
<html>
<body>
Hello World!
<br>
Current time is <%= new java.util.Date() %>
</body>
</html>
```


Mozilla Firefox
File Edit View Go Bookmarks Yahoo! Tools Help
http://localhost/param/Hello.jsp
Go
Done

```
Hello World!
Current time is Sun Jan 07 08:11:25 IST 2007
```

JSP Benefits

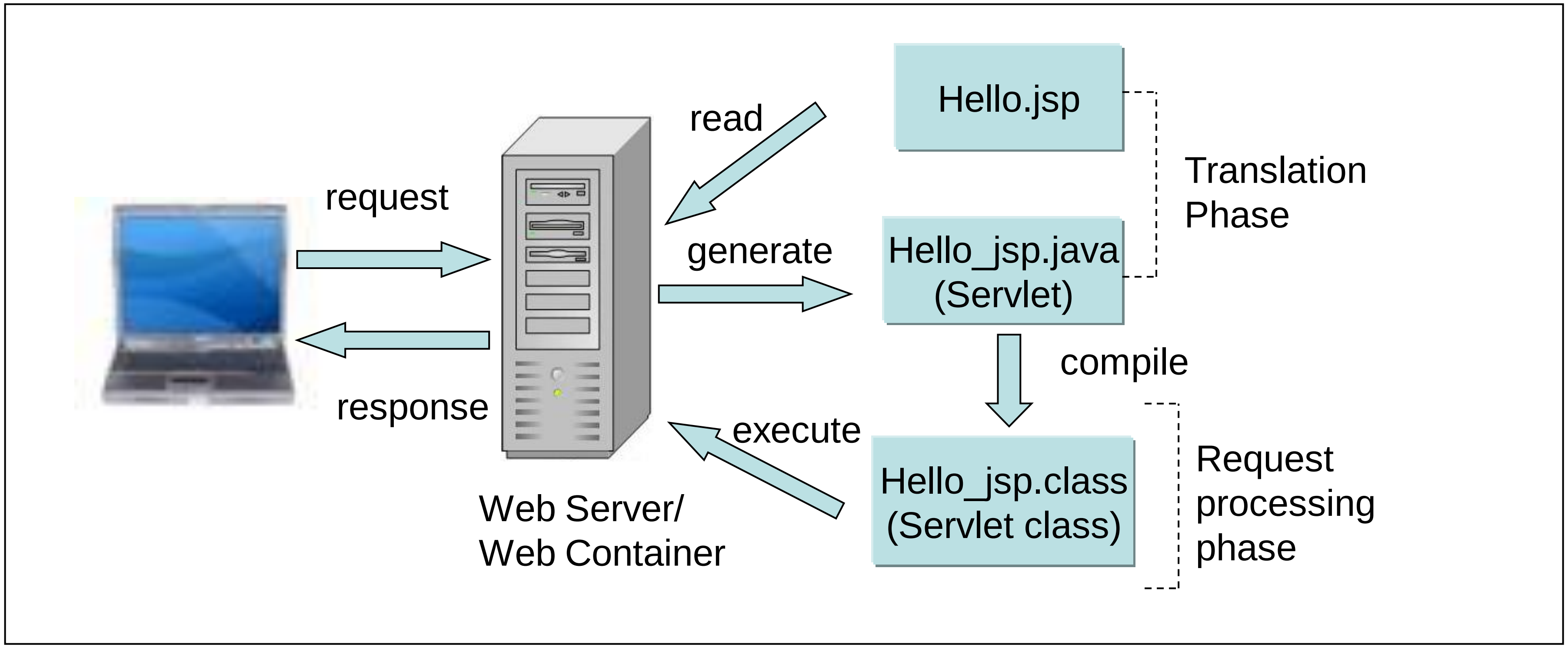
- **Content and display logic are separated**
- **Simplify web application development with JSP, JavaBeans and custom tags**
- **Supports software reuse through the use of components (JavaBeans, Custom tags)**
- **Automatic deployment**
  - Recompile automatically when changes are made to JSP pages
- **Easier to author web pages**
- **Platform-independent**

Why JSP over Servlet?

- **Servlets can do a lot of things, but it is pain to:**
  - Use those println() statements to generate HTML page
  - Maintain that HTML page
- **No need for compiling, packaging, CLASSPATH setting**

JSP Architecture

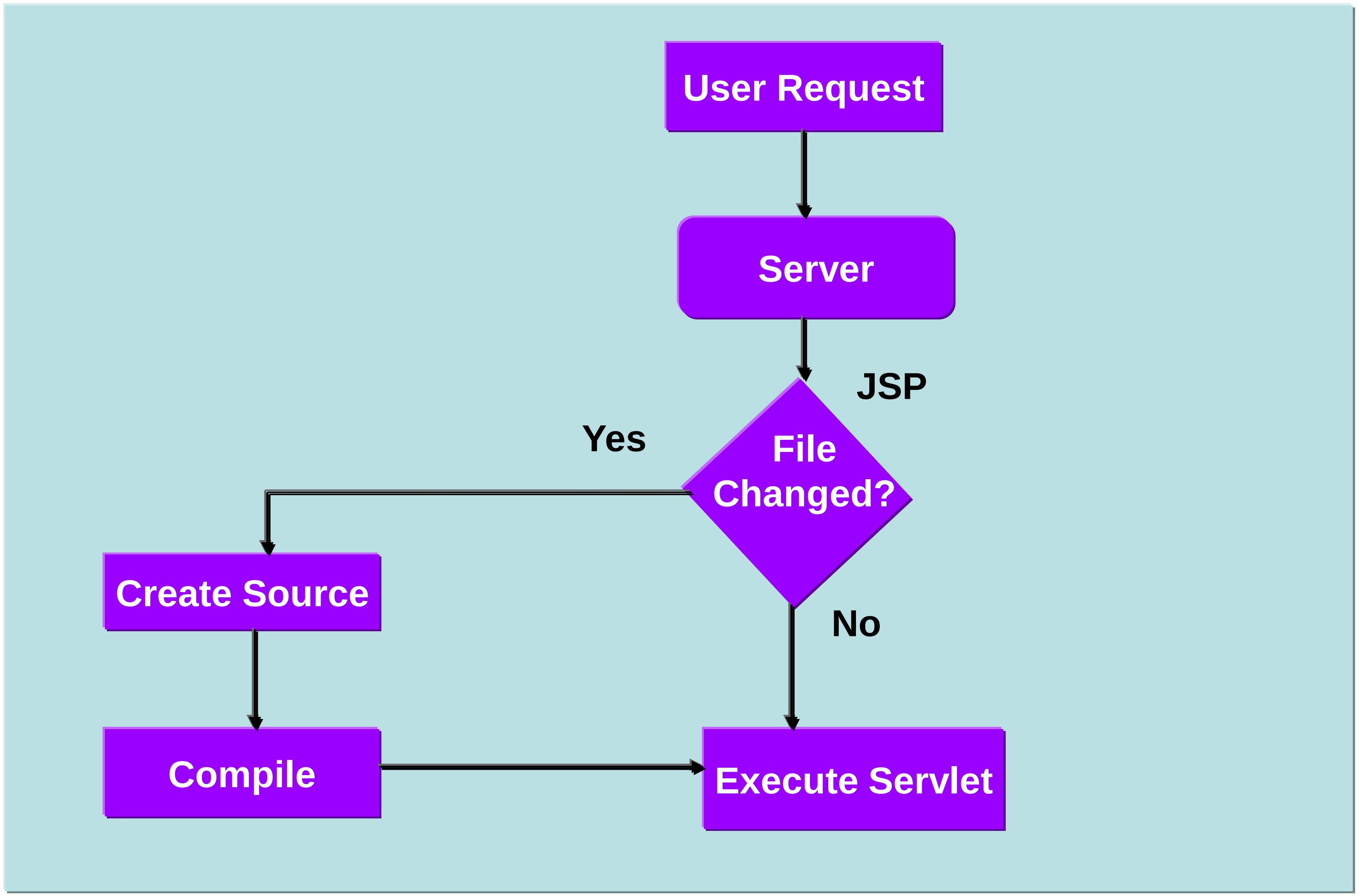
- **Each Java server page is compiled into a servlet before it can be used**
- **The translation phase is typically carried out by the JSP engine itself, when it receives an incoming request for the JSP page for the first time**



JSP Architecture

- **When a request is mapped to a JSP page, the web container first checks whether the JSP page's servlet is older than the JSP page.**
- **If the servlet is older, the web container translates the JSP page into a servlet class and compiles the class.**
- **During development, one of the advantages of JSP pages over servlets is that the build process is performed automatically.**

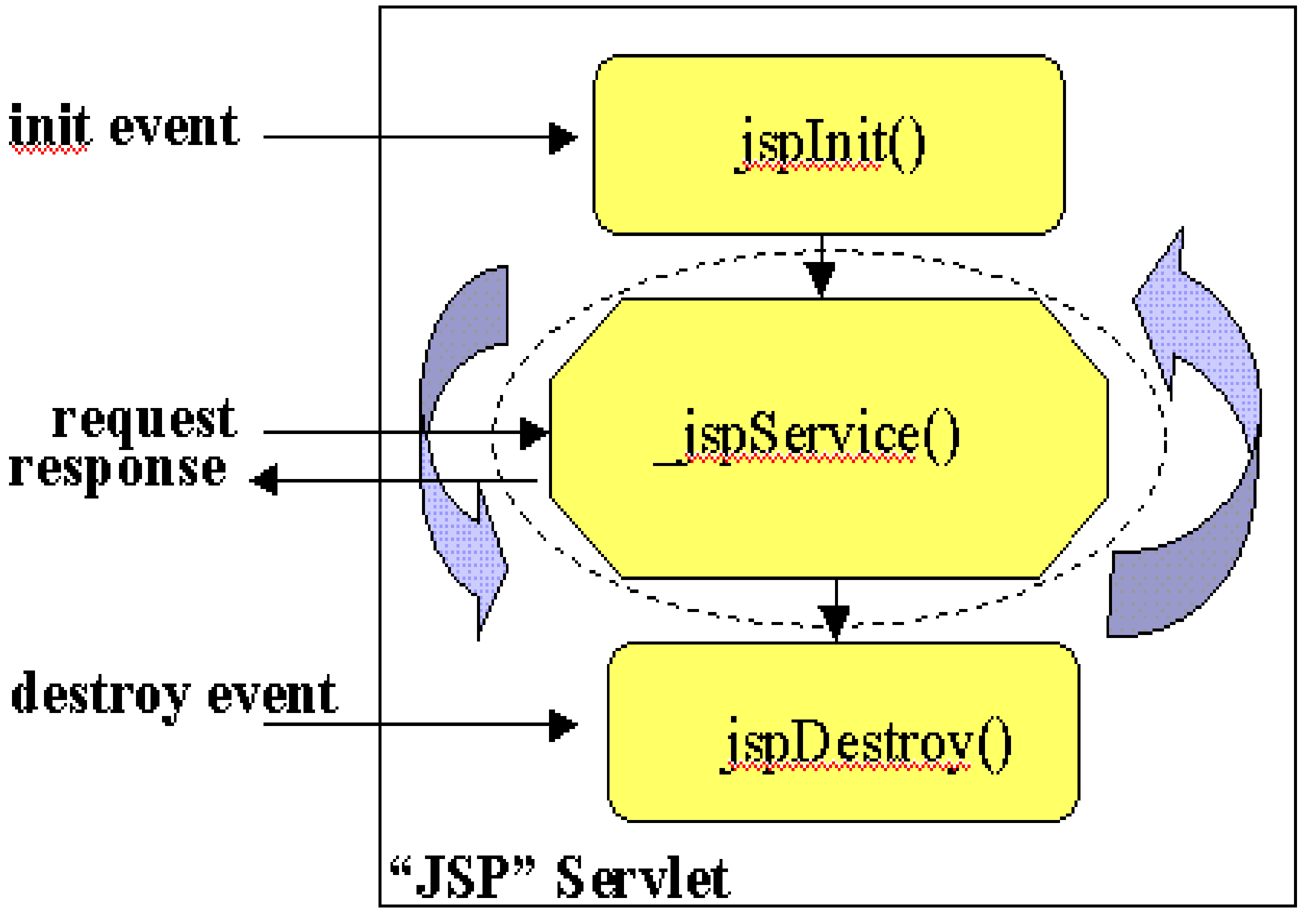
How Does JSP Work



Life Cycle of JSP

- **After the page has been translated and compiled, the JSP page's servlet (for the most part) follows the servlet life cycle**
- **If an instance of the JSP page's servlet does not exist, the container**
  - Loads the JSP page's servlet class
  - Instantiates an instance of the servlet class
  - Initializes the servlet instance by calling the **`jspInit`** method
- **The container invokes the `_jspService` method, passing request and response objects.**
- **If the container needs to remove the JSP page's servlet, it calls the `jspDestroy` method.**

Life Cycle of JSP



JSP Syntax

- JSP syntax is fairly straightforward, and can be classified into
- **Scripting elements**
  - allow us to write inline java code in a JSP
- **Directives**
  - JSP directives are messages for the JSP engine. They do not directly produce any visible output, but tell the engine what to do with the rest of the JSP page.
  - JSP directives are always enclosed within the <%@ ... %> tag.
- **Standard actions**
  - affect the runtime behavior of the JSP Page

Scripting Elements

- **Scripting elements allow us to embed java code in the JSP page**
- **These scripting elements will allow us to insert java code in different parts of the translated java servlet**
- **Scripting elements**
  - JSP Comments `<%-- ... --%>`
  - Declarations of the form `<%! code %>`
  - Expressions of the form `<%= expr %>`
  - Scriptlets of the form `<% code %>`

JSP Comment tag

- **Comments in a JSP page is written using JSP comment tag**
- **Sysntax**

```
<%-- This is a comment --%>
```

alternatively comments can be written in HTML comment syntax also

```
<!-- This is a comment -->
```

- The **statement written in a comment tag will not be included in the translated java code**

JSP Declaration tag

- **Declaration tag is used to declare instance variables and methods in the JSP page implementation class (translated servlet class)**
- **Declarations are found within the <%! … %> tag.**
- **Always end variable declarations with a semicolon, as any content must be valid Java statements:**

```
<%! int i=0; %>
```

- **Methods can be declared using declaration tag**
  - For example, you can override the initialization event in the JSP life cycle by declaring:

```
<%! public void jspInit() {
//some initialization code
} %>
```

JSP Declaration tag

- **For a declaration** `<%! declarations %>` **the Java statements are placed in the class outside the** `_jspService` **method.**
- **Typical declarations can be Java instance variable declarations or Java methods**

```
// declarations would go here
public void _jspService(...)
{
    ...
}
```

Declaration examples

- **Declaring instance variables**

```
<%! private int count = 0; %>
...
The count is <%= count++ %>.
```

- **Declaring methods**

```
<%!
private int toInt(String s)
{
    return Integer.parseInt(s);
}
%>
```

JSP Expression tag

- **JSP expressions begin within <%= ... %> tags and do not include semicolons**
- **For an expression scripting element like `<%= expr %>`, `expr` is evaluated and the result is converted to a string and placed into the JSP's servlet output stream. In a Java servlet this would be equivalent to**

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

Expression examples

- **Displaying request parameters (request is an implicit object available in a JSP)**

```
Your name is <%= request.getParameter("name") %>
and your age is <%= request.getParameter("age") %>
```

- **Doing calculations**

```
The value of pi is <%= Math.PI %> and the square root
of two is <%= Math.sqrt(2.0) %> and today's date is
<%= new java.util.Date() %>.
```

JSP Scriplets

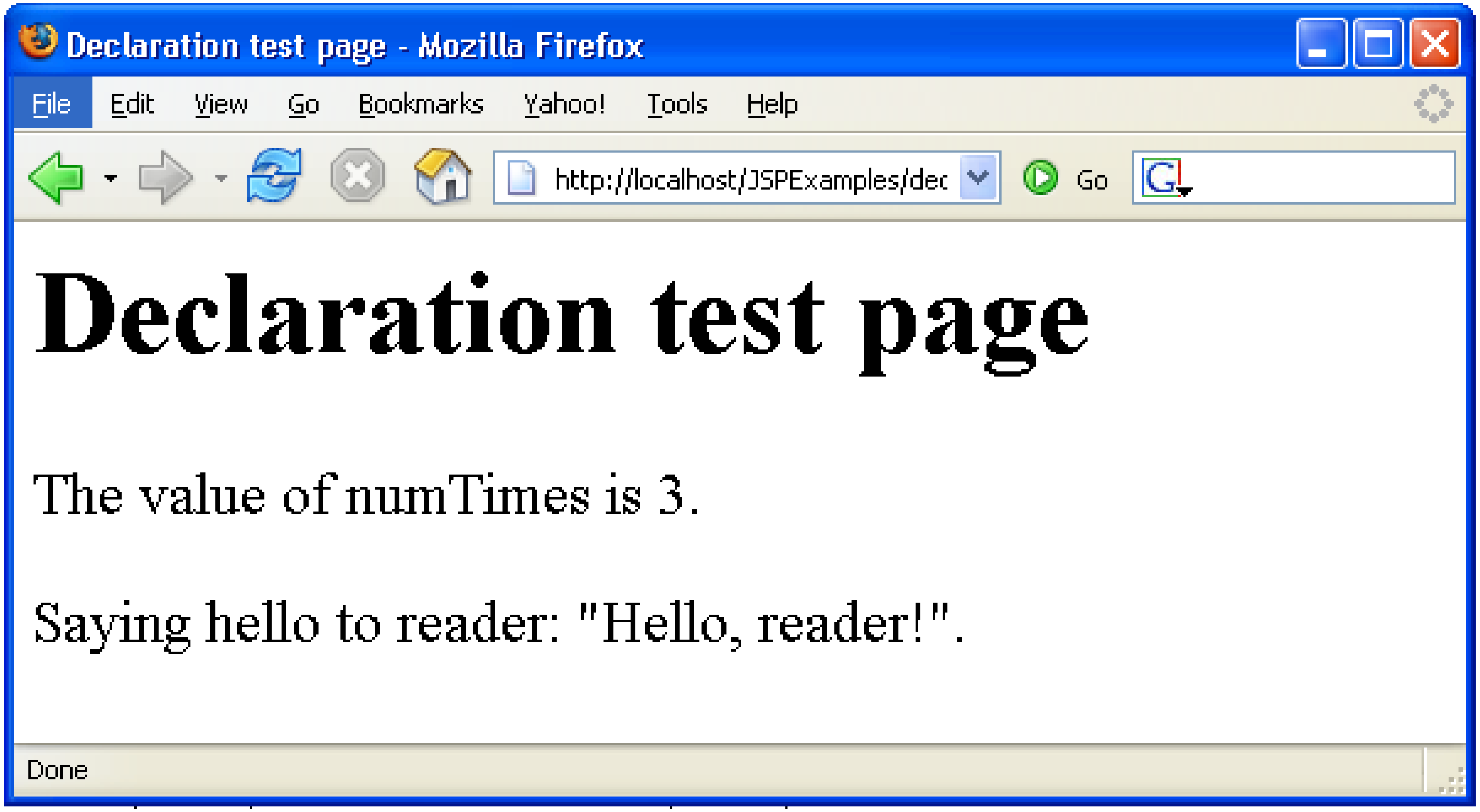
- **JSP code fragments or scriptlets are embedded within <% ... %> tags. This Java code is run when the request is serviced by the JSP page**
- **For a scriplet** `<% statements %>` **the Java statements are placed in the translated servlet's** `_jspService` **method body**
- **Example**

```
<% String name = request.getParameter("name");
   if (name == null)
   { %>
      <h3>Please supply a name</h3>
<% }
   else
   { %>
      <h3>Hello <%= name %></h3>
<% } %>
```

JSP Declaration Example

```
<%!
  int numTimes = 3;

  public String sayHello(String name) {
    return "Hello, " + name + "!";
  }
%>

<html>
  <head>
    <title>Declaration test page</title>
  </head>
  <body>
    <h1>Declaration test page</h1>

    <p>The value of numTimes is <%= numTimes %>.</p>
    <p>Saying hello to reader: "<%= sayHello("reader") %>".</p>
  </body>
</html>
```

Output


Declaration test page - Mozilla Firefox
File Edit View Go Bookmarks Yahoo! Tools Help
http://localhost/JSPExamples/dec
Go
Done

Declaration test page

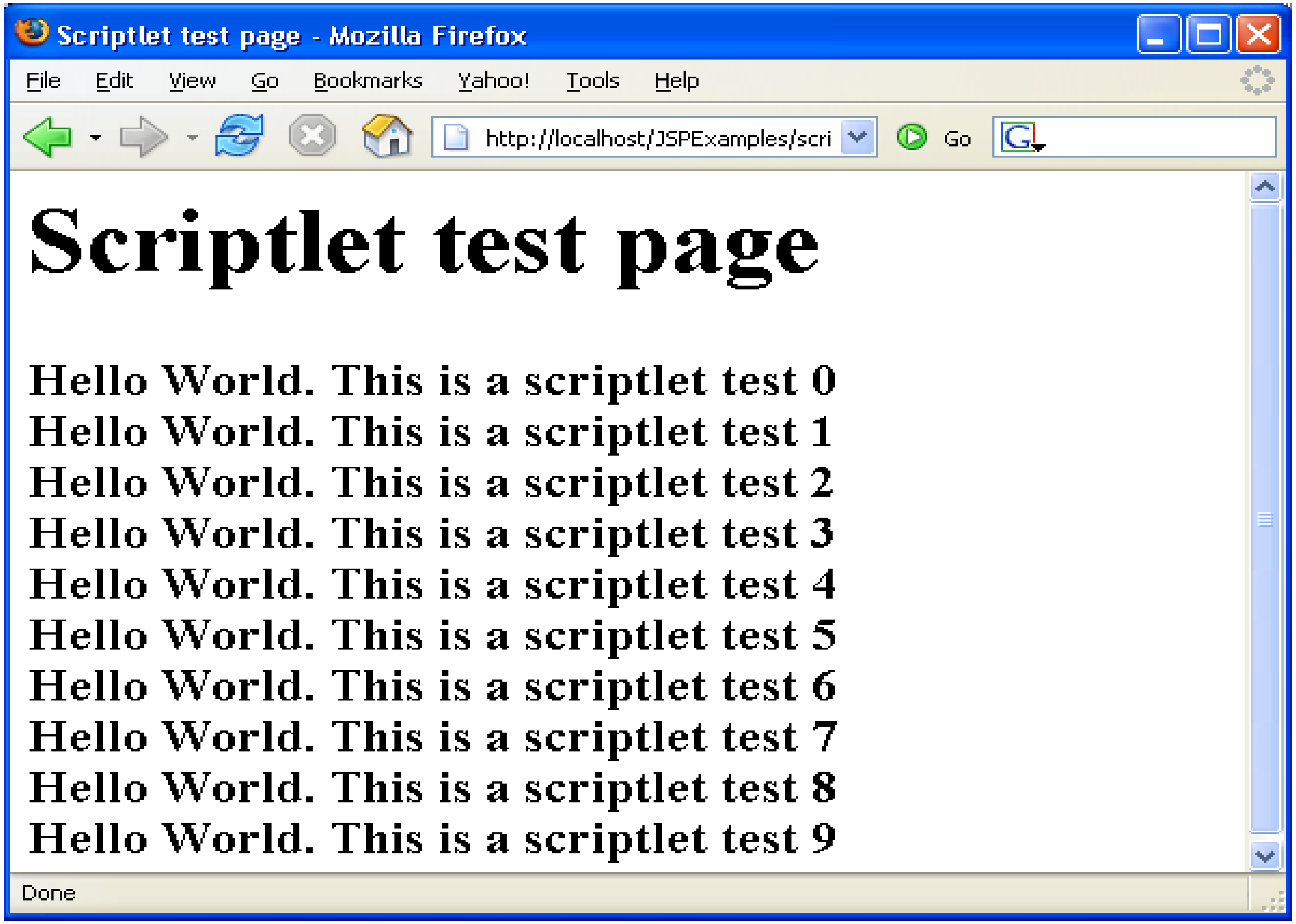
The value of numTimes is 3.

Saying hello to reader: "Hello, reader!".

JSP Scriptlet Example

```
<html>
  <head>
    <title>Scriptlet test page</title>
  </head>
  <body>
    <h1>Scriptlet test page</h1>

    <%
      for(int i=0;i< 10;i++) {
        out.println("<b>Hello World. This is a scriptlet test " + i +
                    "</b><br>");
        System.out.println("This goes to the System.out stream " + i);
      }
    %>

  </body>
</html>
```

Output

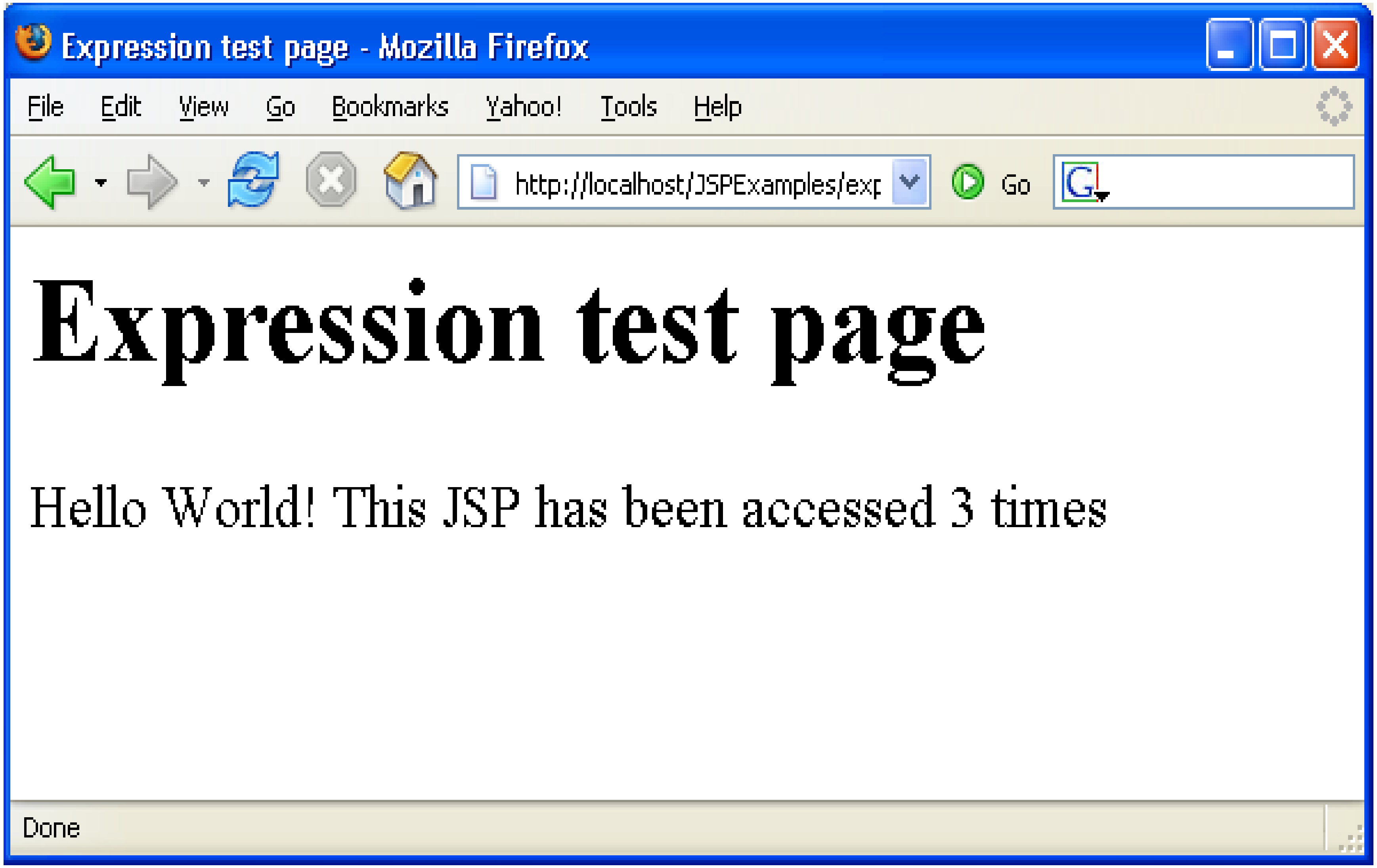

Scriptlet test page - Mozilla Firefox
File Edit View Go Bookmarks Yahoo! Tools Help
http://localhost/JSPExamples/scri
Go
Done

```
Scriptlet test page

Hello World. This is a scriptlet test 0
Hello World. This is a scriptlet test 1
Hello World. This is a scriptlet test 2
Hello World. This is a scriptlet test 3
Hello World. This is a scriptlet test 4
Hello World. This is a scriptlet test 5
Hello World. This is a scriptlet test 6
Hello World. This is a scriptlet test 7
Hello World. This is a scriptlet test 8
Hello World. This is a scriptlet test 9
```

JSP Expression Example

```
<html>
  <head>
    <title>Expression test page</title>
  </head>
  <body>
    <h1>Expression test page</h1>

    <%! int i=0 ; %>

    <%
      i++;
    %>

    Hello World!
    <%= "This JSP has been accessed " + i + " times" %>

  </body>
</html>
```

Output


Expression test page - Mozilla Firefox
File Edit View Go Bookmarks Yahoo! Tools Help
http://localhost/JSPExamples/exp
Go
Done

Expression test page

Hello World! This JSP has been accessed 3 times

Directives

- **The directive elements, specify information about the page itself that remains the same between requests**
  - for example if session tracking is required or not, buffering requirements, and the name of a page that should be used to report errors, if any, etc.
- **Syntax**

  <%@ directivename attribute=“value” %>
- **The three main directives**

  1. `page` directive
  2. `include` directive
  3. `taglib` directive

Directives

- ***page*** **directive**
  - Defines page-dependent attributes, such as session tracking, error page, and buffering requirements
- **Syntax**

  <%@ page attribute="value" %>

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

Attributes of Page directive			
Name	Default value	Name	Default Value
language	**Java**	**extends**	**……**
import	**………**	**session**	**true**
buffer		**auto flush**	**true**
isThreadSafe	**true**	**info**	**…………**
errorPage	**………**	**isErrorPage**	**false**
contentType	**text/html**		

Directives

- ***`include`* directive**
  - Includes a file during the translation phase in place of the directive
- **Syntax**

  <%@ `include` file=“filename” %>

  - `<%@ include file="header.html" %>`
- ***`taglib`* directive**
  - Declares a tag library, containing custom actions, that is used in the page (e.g. custom tags)
- **Syntax**

  <%@ `taglib uri`=“location of tag library” %>

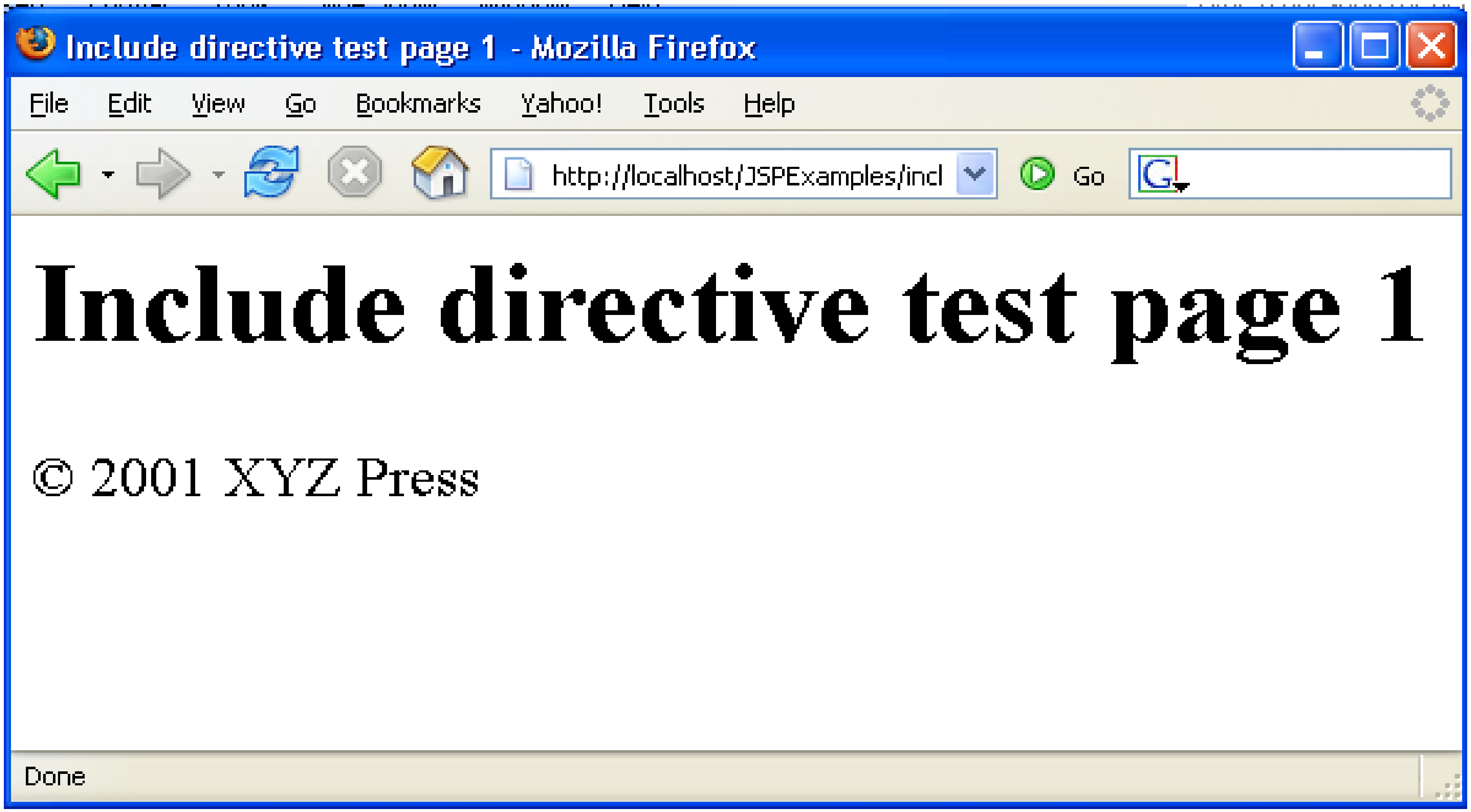
Page Directive Example

```
<%@ page language="Java" import="java.rmi.*,java.util.*"
    session="true" buffer="12kb" autoFlush="true"
    info="my page directive jsp" errorPage="error.jsp"
    isErrorPage="false" isThreadSafe="true" %>

<html>
  <head>
    <title>Page directive test page</title>
  </head>
  <body>
    <h1>Page directive test page</h1>
    This is a JSP to test the page directive.
  </body>
</html>
```

include Directive Example

```
<html>
  <head>
    <title>Include directive test page 1</title>
  </head>
  <body>
    <h1>Include directive test page 1</h1>

    <%@ include file="/copyright.html" %>
  </body>
</html>
```

Output


Include directive test page 1 - Mozilla Firefox
File Edit View Go Bookmarks Yahoo! Tools Help
http://localhost/JSPExamples/incl
Go

Include directive test page 1

© 2001 XYZ Press

Done

Include Directive Example(2)

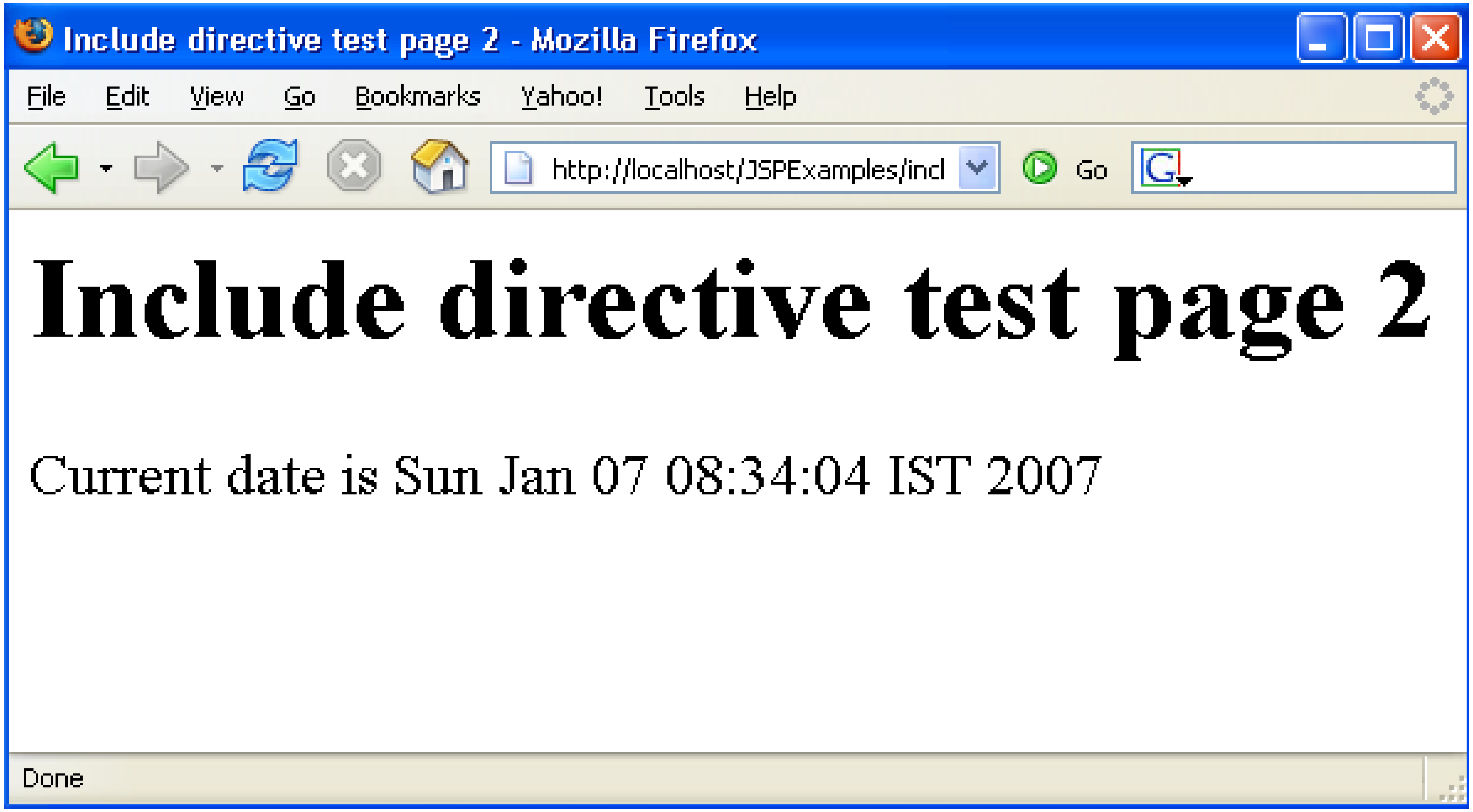
```
<html>
  <head>
    <title>Include directive test page 2</title>
  </head>
  <body>
    <h1>Include directive test page 2</h1>

    <%@ include file="included.jsp" %>
  </body>
</html>
```

Include Directive Example(2)

included.jsp

```
<%@ page import="java.util.Date" %>
<%= "Current date is " + new Date() %>
```

Output


Include directive test page 2 - Mozilla Firefox
File Edit View Go Bookmarks Yahoo! Tools Help
http://localhost/JSPExamples/incl
Go
Done

Include directive test page 2

Current date is Sun Jan 07 08:34:04 IST 2007

Implicit Objects

- **A JSP page has access to certain implicit objects that are always available, without being declared first**
- **Created by container**
- **Corresponds to classes defined in Servlet**
  - These objects act as wrappers around underlying Java classes or interfaces typically defined within the Servlet API.

Implicit Objects

1. request: **represents the HttpServletRequest**
2. response: **represents HttpServletResponse to the request.**
3. pageContext: **encapsulates implementation-dependent features in PageContext.**
4. application: **represents the ServletContext obtained from servlet configuration object.**
5. out: **a JspWriter object that writes into the output stream.**
6. config: **represents the ServletConfig for the JSP. Page scope.**
7. page: **synonym for the "this" operator, as an HttpJspPage.**
8. session: **An HttpSession. Session scope. More on sessions shortly.**
9. exception: **the uncaught Throwable object that resulted in the error page being invoked. Page scope.**

Implicit Objects

- **Iimplicit objects are only visible within the system generated `_jspService()` method.**
- **Theyare not visible within methods you define yourself in declarations.**

JSP Actions

- **Action elements perform some action based on information that is required at the exact time the JSP page is requested by a browser.**

- **An action can access parameters sent with the request to do a database lookup.**

- **It can also dynamically generate HTML, such as a table filled with information retrieved from an external system.**

JSP Actions

- **The JSP specification defines a few standard action elements**

Action element	Description
<jsp:useBean>	Makes a JavaBeans component available in a page
<jsp:getProperty>	Gets a property value from a JavaBeans component and adds it to the response
<jsp:setProperty>	Sets a JavaBeans component property value
<jsp:include>	Includes the response from a servlet or JSP page during the request processing phase
<jsp:forward>	Forwards the processing of a request to servlet or JSP page
<jsp:param>	Adds a parameter value to a request handed off to another servlet or JSP page using <jsp:include> or <jsp:forward>
<jsp:plugin>	Generates HTML that contains the appropriate browser-dependent elements (OBJECT or EMBED) needed to execute an applet with the Java Plug-in software

include Action

- Includes the response from a servlet or JSP page during the request processing phase

include Action Example

```
<html>
  <head>
    <title>Include Action test page</title>
  </head>
  <body>
    <h1>Include Action test page</h1>

      <h2>Using the include directive</h2>

      <%@ include file="included2.html" %>
      <%@ include file="included2.jsp" %>

      <h2>Using the include action</h2>

      <jsp:include page="included2.html" flush="true" />
      <jsp:include page="included2.jsp" flush="true" />

  </body>
</html>
```

forward Action

Forwards the processing of a request to servlet or JSP page

jsp:forward example(1)

forward.html

```
<html>
  <head>
    <title>Forward action test page</title>
  </head>
  <body>
    <h1>Forward action test page</h1>

    <form method="post" action="forward.jsp">
      <p>Please enter your username:
      <input type="text" name="userName">
      <br>and password:
      <input type="password" name="password">
      </p>

      <p><input type="submit" value="Log in">
    </form>

  </body>
</html>
```

jsp:forward example(2)

forward.jsp

```
<%
  if ((request.getParameter("userName").equals("java")) &&
      (request.getParameter("password").equals("java"))) {
%>

<jsp:forward page="forward2.jsp" />

<% } else { %>

<%@ include file="forward.html" %>

<% } %>
```

jsp:forward example(3)

forward2.jsp

```
<html>
  <head>
    <title>Forward action test: Login successful!</title>
  </head>
  <body>
    <h1>Forward action test: Login successful</h1>

    <p>Welcome, <%= request.getParameter("userName") %>

  </body>
</html>
```

What are JavaBeans?

- **Java classes that can be easily reused and composed together into an application**
- **Any Java class that follows certain design conventions can be a JavaBeans component**
  - properties of the class
  - public methods to get and set properties
- **Within a JSP page, you can create and initialize beans and get and set the values of their properties**
- **JavaBeans can contain business logic or data base access logic**
- **JavaBeans components are Java classes that c**

JavaBeans Design Conventions

- **JavaBeans maintain internal properties**
- **A property can be**
  - Read/write, read-only, or write-only
  - Simple or indexed
- **Properties should be accessed and set via getXxx and setXxx methods**
  - PropertyClass getProperty() { ... }
  - setProperty(PropertyClass pc) { ... }
- **JavaBeans must have a zero-argument (empty) constructor**

Why Use JavaBeans in JSP Page?

- **A JSP page can create and use any type of Java programming language object within a declaration or scriptlet like following:**

```
<%
ShoppingCart cart =
    (ShoppingCart)session.getAttribute("cart");
// If the user has no cart, create a new one
if (cart == null) {
cart = new ShoppingCart();
session.setAttribute("cart", cart);
}
%>
```

Why Use JavaBeans in JSP Page?

- **JSP pages can use JSP elements to create and access the object that conforms to JavaBeans conventions**

```
<jsp:useBean id="cart" class="cart.ShoppingCart“
scope="session"/>
```

- **Create an instance of “ShoppingCart” if none exists, stores it as an attribute of the session scope object, and makes the bean available throughout the session by the identifier “cart”**

Compare the Two

```
<%
ShoppingCart cart = (ShoppingCart)session.getAttribute("cart");
// If the user has no cart object as an attribute in Session scope
// object, then create a new one. Otherwise, use the existing
// instance.
if (cart == null) {
cart = new ShoppingCart();
session.setAttribute("cart", cart);
}
%>
```

versus

```
<jsp:useBean id="cart" class="cart.ShoppingCart"
scope="session"/>
```

Why Use JavaBeans in JSP Page?

- No need to learn Java programming language for page designers
- Stronger separation between content and presentation
- Higher reusability of code
- Simpler object sharing through built-in sharing mechanism
- Convenient matching between request parameters and object properties

Creating a JavaBeans

- **Declare that the page will use a bean that is stored within and accessible from the**
- **specified scope by jsp:useBean element**

```
<jsp:useBean id="beanName"
class="fully_qualified_classname" scope="scope"/>
```

or

```
<jsp:useBean id="beanName"
class="fully_qualified_classname" scope="scope">
<jsp:setProperty .../>
</jsp:useBean>
```

setting properties

- **To set a property of a bean use**

```
<jsp:setProperty name="..."
   property="..." value="..." />
```

- **To set a property using the value of a request parameter use**

```
<jsp:setProperty name="..."
   property="..." param="..." />
```

getting properties

- **To get a property of a bean use**

```
<jsp:getProperty name="..."
   property="..." />
```