# JSP Basics

Content By: Bairon Vasquez

Week 10 - Day 1

## Questions?

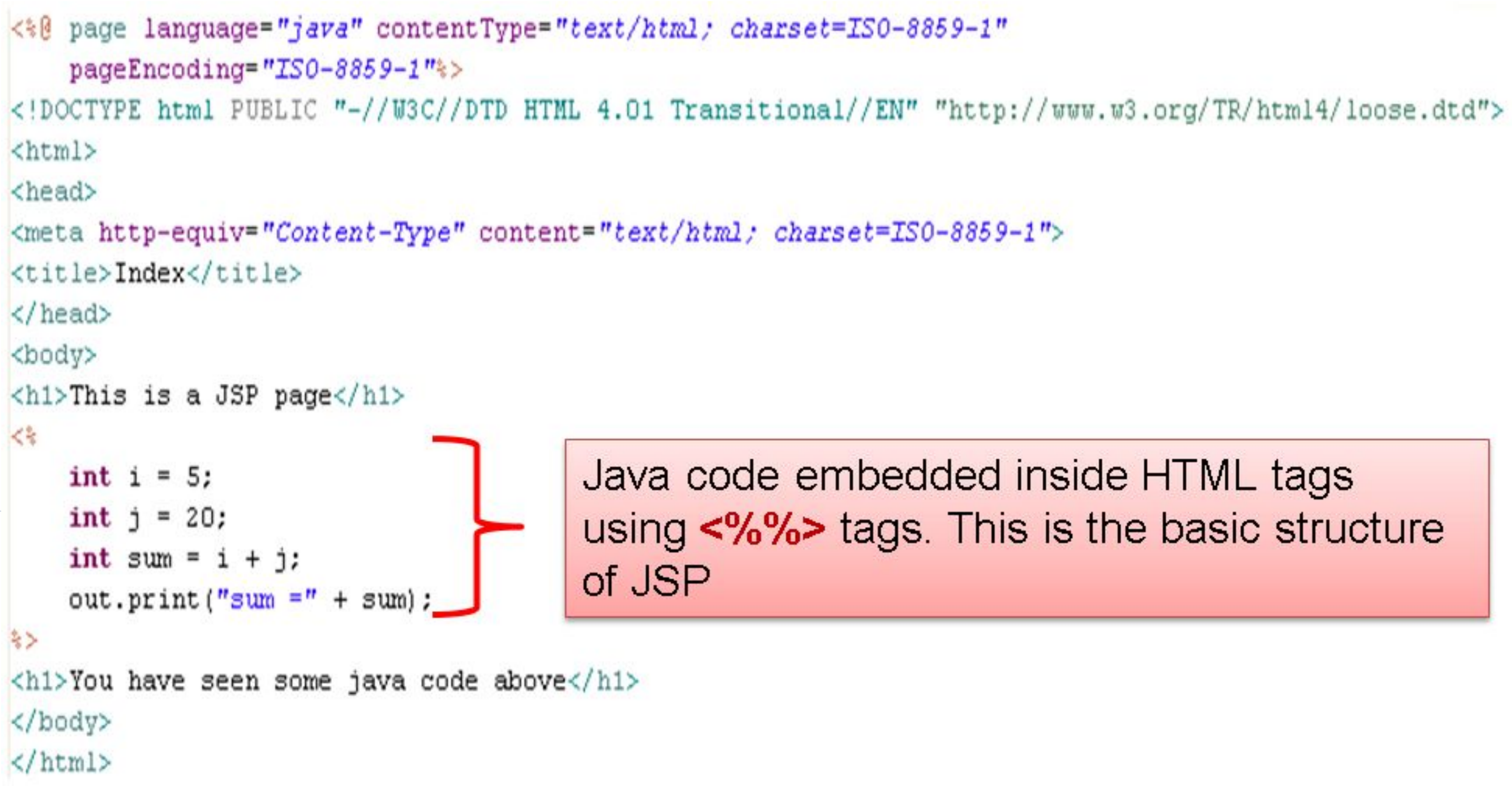
## Agenda

- ❑ Describing what a JSP is
- ❑ Describing the Life Cycle of JSPs
- ❑ The Differences between JSP and Servlet files
- ❑ Introducing JSP Elements

## What is a JSP?

- JSP files, at their core, are HTML files that allow for special tags to enable the use of Java code
  - This allows for dynamic content to be generated on otherwise static HTML pages
- This concept of dynamic webpages allows for the ability to use the same HTML file, but render varying results
  - Example: The way Google works is that the “search engine” will display the same HTML template, but will have different results appear based on it’s input field
  - Different data, but same page

## Sample JSP Page



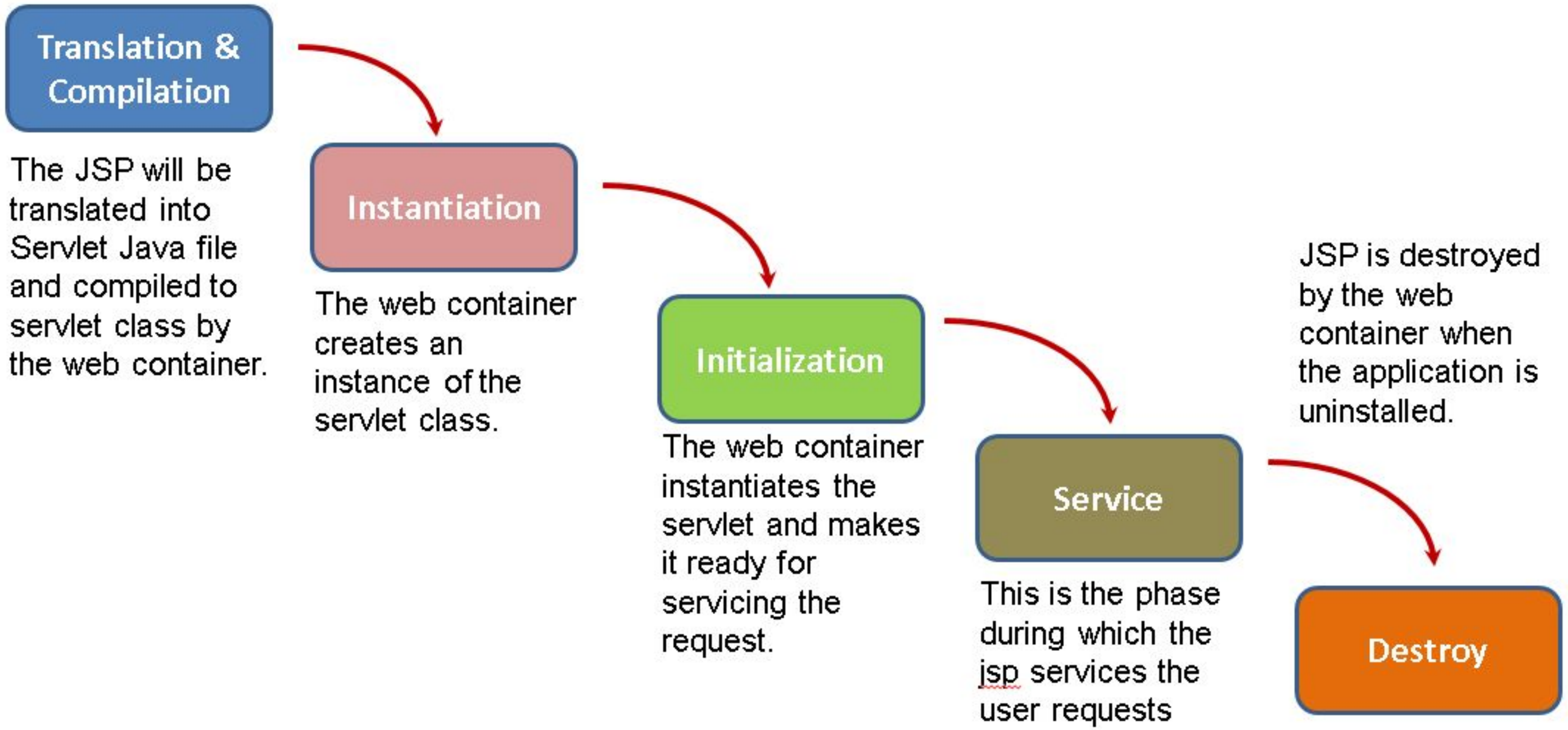
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Index</title>
</head>
<body>
<h1>This is a JSP page</h1>
<%
    int i = 5;
    int j = 20;
    int sum = i + j;
    out.print("sum =" + sum);
%>
<h1>You have seen some java code above</h1>
</body>
</html>
```

Java code embedded inside HTML tags using **<%%>** tags. This is the basic structure of JSP

# Servlet Vs JSP

- In Servlets, HTML code is written inside java code using print statements. In JSP, java code is embedded inside HTML code
- In reality, JSPs are converted into Servlets by the web container
  - So it actually does the same thing as a Java Servlet
- JSPs are easier for creating HTML content, but Servlets are easier for writing Java code
- A combination of JSPs and Servlets can be used to separate the presentation (HTML) and logic (Java code) in a web application

## JSP Life Cycle Phases



The JSP will be translated into Servlet Java file and compiled to servlet class by the web container.

The web container creates an instance of the servlet class.

The web container instantiates the servlet and makes it ready for servicing the request.

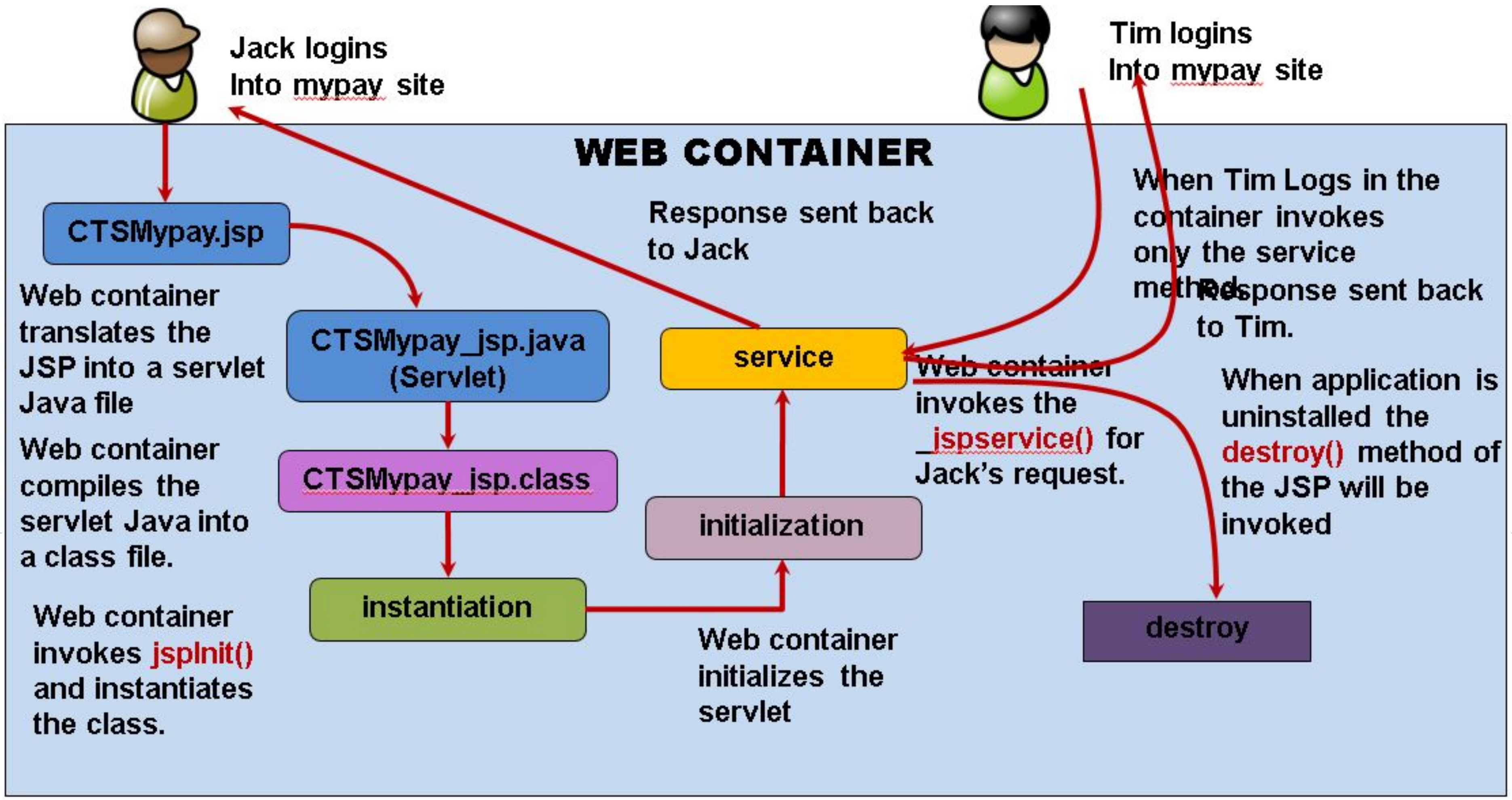
This is the phase during which the jsp services the user requests

JSP is destroyed by the web container when the application is uninstalled.

## JSP Life Cycle methods

- The following methods will be generated by the web container when translating the JSP into a Servlet Java file
  - **jspInit()** – The web container calls the jspInit() method to initialize the Servlet instance generated. It is invoked before servicing the client request and invoked only once for the Servlet instance
  - **_jspService()** – The container calls the jspService() method for each user request; the request and response objects are passed to this method
  - **jspDestroy()** – The container calls this method when it decides to take the instance out of service. It is the last method called in the Servlet instance

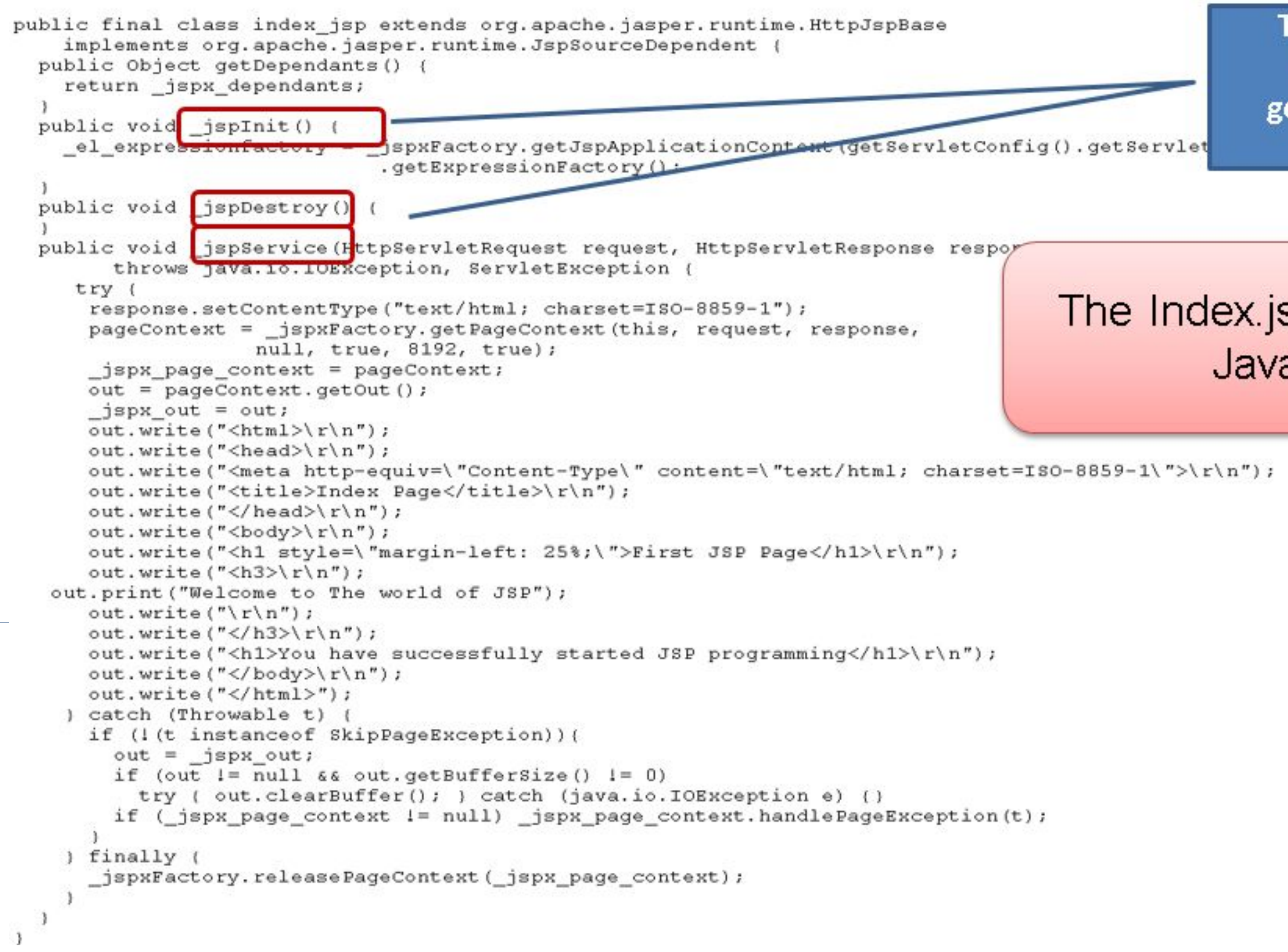
# JSP Life Cycle Phases (Demo)


Jack logins
Into mypay site
Tim logins
Into mypay site
WEB CONTAINER
CTSMypay.jsp
Web container translates the JSP into a servlet Java file
CTSMypay_jsp.java (Servlet)
Web container compiles the servlet Java into a class file.
CTSMypay_jsp.class
instantiation
Web container invokes jspInit() and instantiates the class.
initialization
Web container initializes the servlet
service
Response sent back to Jack
Web container invokes the _jspservice() for Jack's request.
When Tim Logs in the container invokes only the service method
Response sent back to Tim.
When application is uninstalled the destroy() method of the JSP will be invoked
destroy

## What happens to a JSP file?

- Whenever you compile a JSP file, it gets converted to a Servlet file and compiled as a .class file for servicing a user’s request
- You can find this file in the web server folder where the applications is deployed. The web server creates a temporary folder for extracting these files.
    - Note: The folder path varies between web servers (e.g. Tomcat, Glassfish, etc.)

# What happens to a JSP file? (Demo)



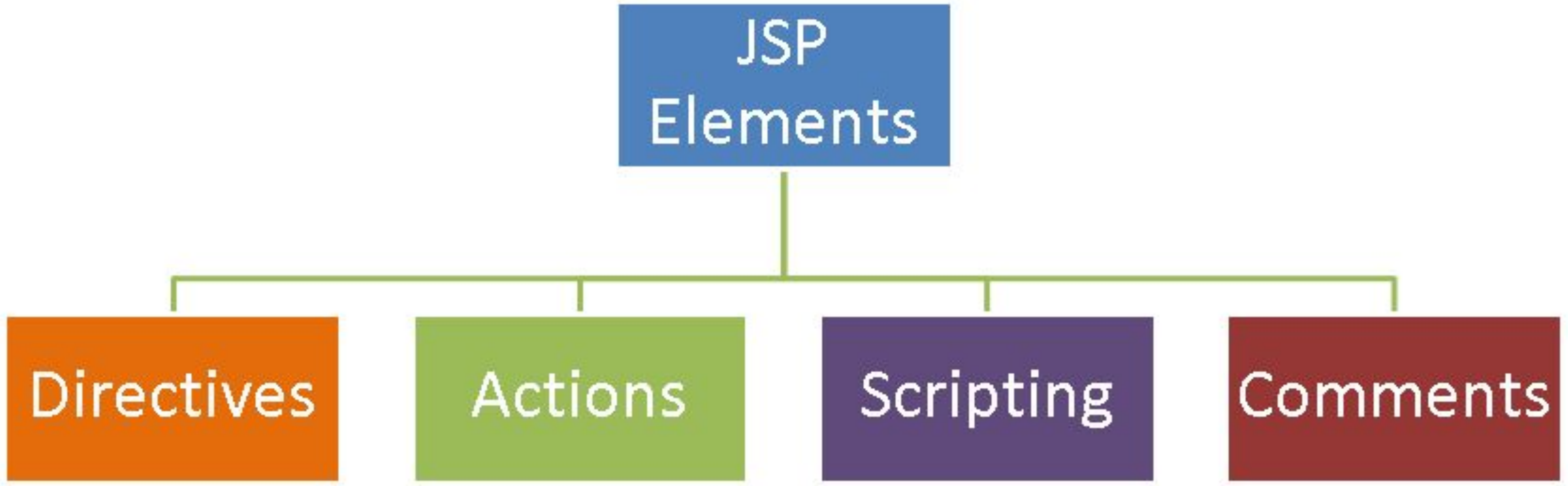
```
public final class index_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent {
  public Object getDependants() {
    return _jspx_dependants;
  }
  public void _jspInit() {
    _el_expressionFactory = _jspxFactory.getJspApplicationContext(getServletConfig().getServlet
                                .getExpressionFactory();
  }
  public void _jspDestroy() {
  }
  public void _jspService(HttpServletRequest request, HttpServletResponse respo
        throws java.io.IOException, ServletException {
     try {
      response.setContentType("text/html; charset=ISO-8859-1");
      pageContext = _jspxFactory.getPageContext(this, request, response,
                    null, true, 8192, true);
      _jspx_page_context = pageContext;
      out = pageContext.getOut();
      _jspx_out = out;
      out.write("<html>\r\n");
      out.write("<head>\r\n");
      out.write("<meta http-equiv=\"Content-Type\" content=\"text/html; charset=ISO-8859-1\">\r\n");
      out.write("<title>Index Page</title>\r\n");
      out.write("</head>\r\n");
      out.write("<body>\r\n");
      out.write("<h1 style=\"margin-left: 25%;\">First JSP Page</h1>\r\n");
      out.write("<h3>\r\n");
   out.print("Welcome to The world of JSP");
      out.write("\r\n");
      out.write("</h3>\r\n");
      out.write("<h1>You have successfully started JSP programming</h1>\r\n");
      out.write("</body>\r\n");
      out.write("</html>");
    } catch (Throwable t) {
      if (!(t instanceof SkipPageException)){
        out = _jspx_out;
        if (out != null && out.getBufferSize() != 0)
          try { out.clearBuffer(); } catch (java.io.IOException e) {}
        if (_jspx_page_context != null) _jspx_page_context.handlePageException(t);
      }
    } finally {
      _jspxFactory.releasePageContext(_jspx_page_context);
    }
  }
}
```

**The service, init and destroy methods generated by the web container.**

The Index.jsp translated to Java code.

# JSP's Element Types

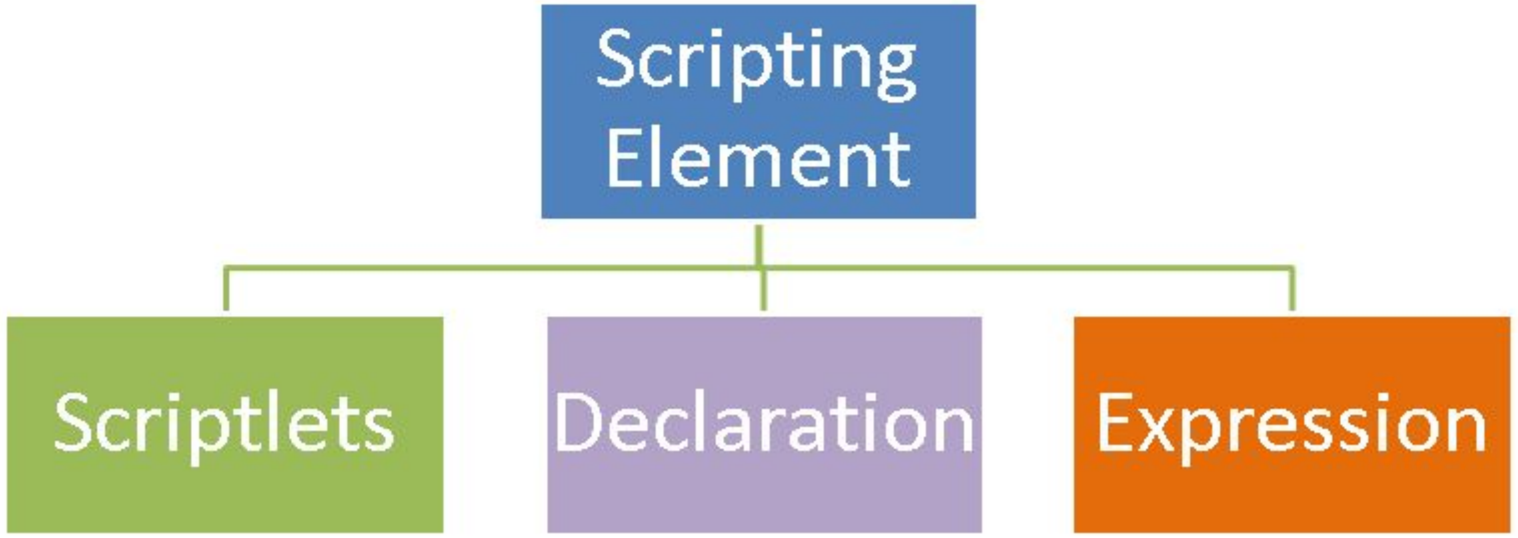
- There are four types of elements in JSP:



- We will be focusing on Scripting Elements for the rest of this slide

# Scripting Elements

- **Scripting Elements** are used to embed java code in JSP files.
- There are three types of Scripting Elements:



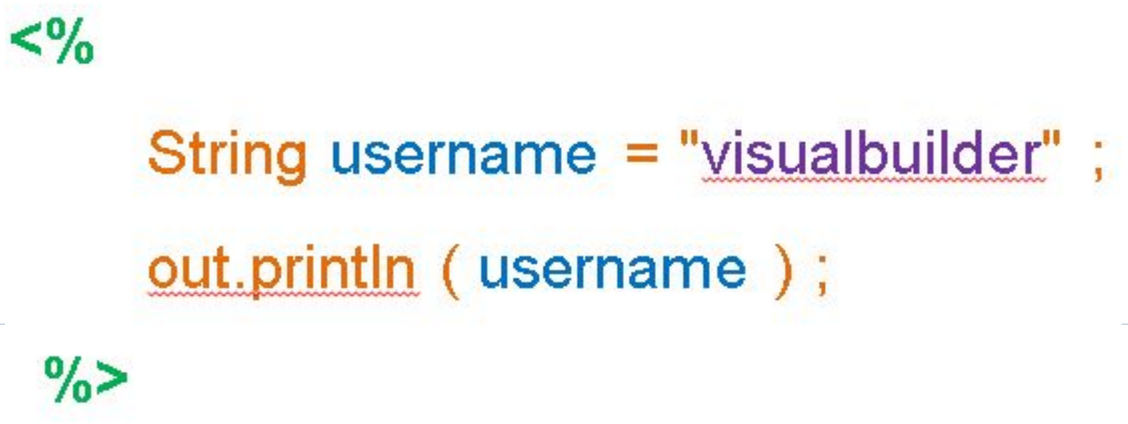
## Scriplets

- Used to embed java code in JSP pages
- The contents within a JSP Scriplet goes into the **_jspService()** method during the translation phase
- Code within a Scriplet should comply with the syntactical and semantic constructs found in normal Java code
- The Java code is embedded between <% and %> delimiters

platform
By Per Scholas

# How to Create a Scriplet

- **Scriplets** are embedded between <% and %> delimiters
- Syntax - *<% //Java code goes here %>*
- Example - To print a variable value:



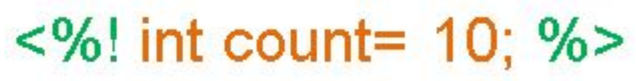
```
<%
    String username = "visualbuilder" ;
    out.println ( username ) ;
%>
```

## Declarations Element

- **Declarations** are used to declare global/instance variables and define methods
- Declaration tags do not produce any output; it is used for global reference outside of the **_jspService()** method
- The methods and classes declared will be translated as class level variables and methods during translation

# How to Create a Declaration

- Methods or variables are declared using <%! And %> delimiters
- Syntax – <%! variable = value; %>
- Example – This declares a global variable *count* as an int and sets it's value to 10:



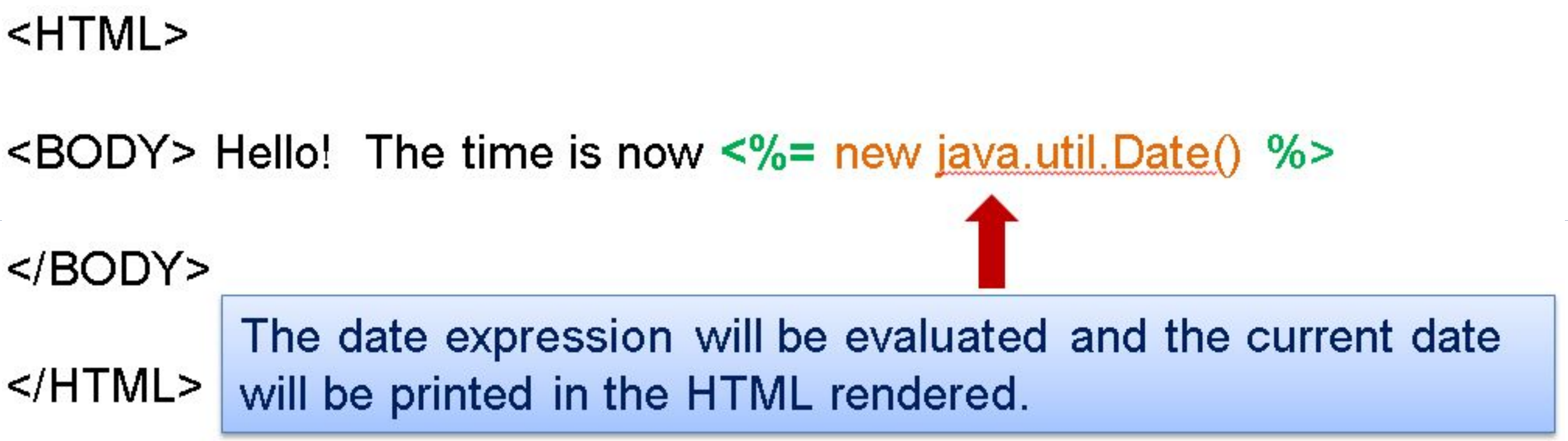
```
<%! int count= 10; %>
```

## Expression Element

- Used to write dynamic content back to the client browser
- Used in place of the **out.print()** method
- During translation the return type of Expression elements go as the argument in the out.print()method
- Expression elements should not be ended with a semicolon (;), since the semicolon is automatically added during translation to the out.print() method

## How to Create an Expression

- Expressions are Embedded in <%= and %> delimiters
- Syntax – <%= expression %>
- Example – To print the date dynamically for each client request:



```
<HTML>

<BODY> Hello!  The time is now <%= new java.util.Date()  %>

</BODY>

</HTML>
```

The date expression will be evaluated and the current date will be printed in the HTML rendered.

## Comments

- There are two type of comments supported in JSP:
  - Standard HTML comment

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

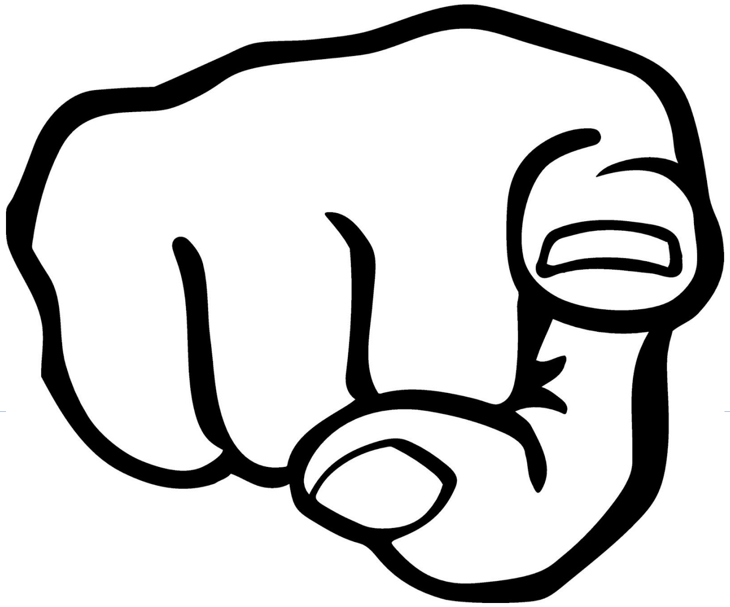
  - JSP comment

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

- HTML comments are passed during the translation phase in the Servlet, and hence can be viewed in the browser's page source
- JSP comments are converted to normal java comments during the translation process and will not appear in the browser's page source

## Questions?

## Your turn

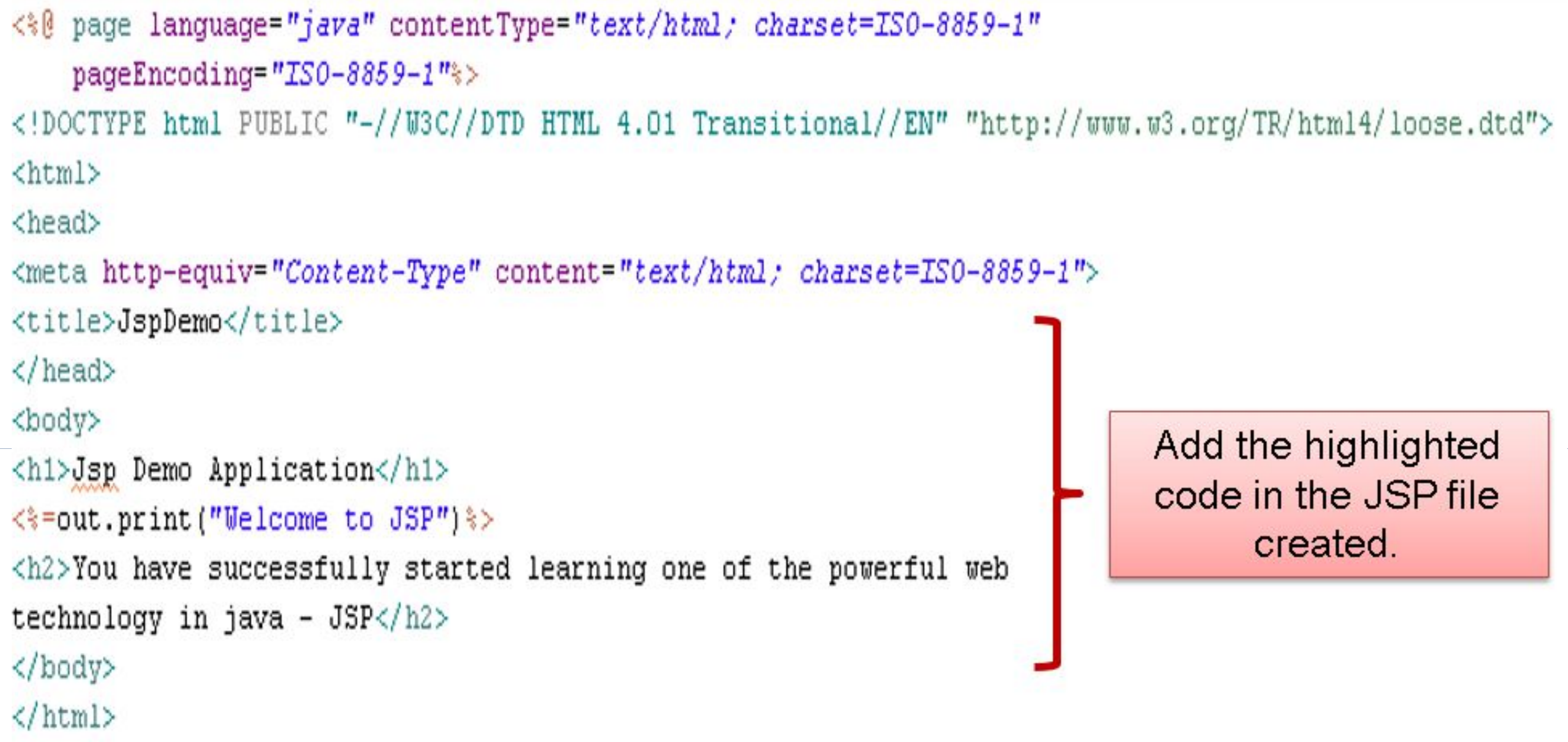


## Your Turn – First JSP

- Here we will create a web application with a simple jsp page which prints a message "Welcome to JSP"
- This Demo is meant for understanding the following things:
  - How to deploy a JSP application?
  - What happens to the JSP when it is deployed?
  - How to call the jsp page from the browser?

# Your Turn (Code Assistance)

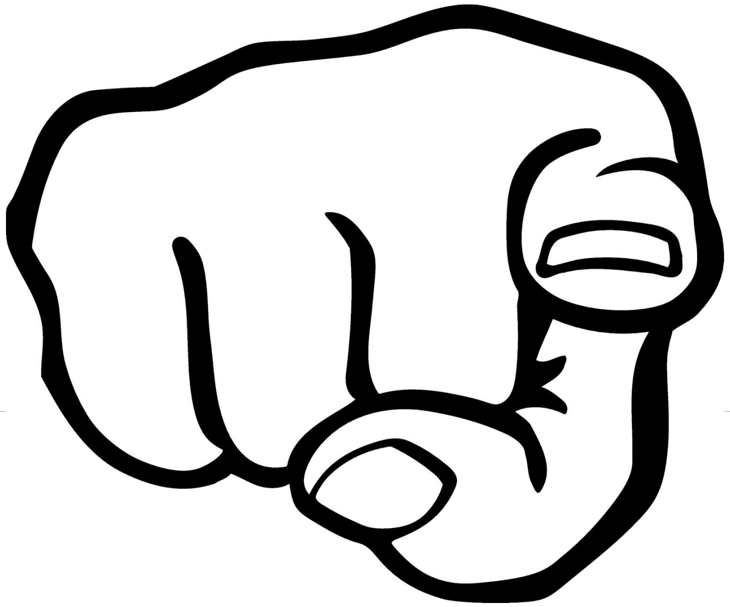
- Sample Code (index.jsp):



```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>JspDemo</title>
</head>
<body>
<h1>Jsp Demo Application</h1>
<%=out.print("Welcome to JSP")%>
<h2>You have successfully started learning one of the powerful web
technology in java - JSP</h2>
</body>
</html>
```

Add the highlighted code in the JSP file created.

## Your turn

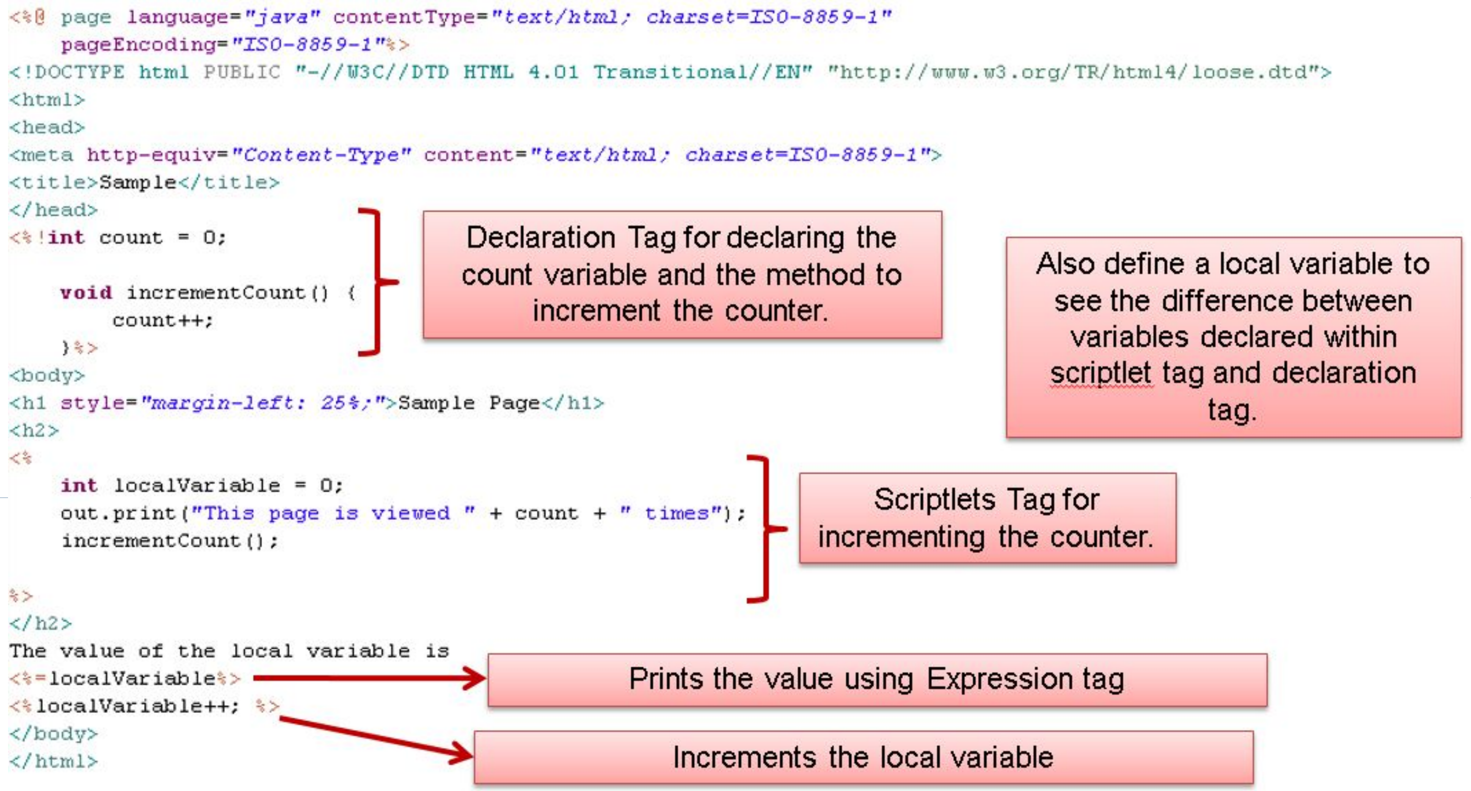


# Your Turn – Scriplet Elements

- This is a demo to familiarize the Scripting elements that are used in JSP
- We will create a JSP page called *sample.jsp*
- The page should calculate the number of times a user visits the page and should print the value on the screen
- The JSP page should use the following Scriplets elements:
  - Declaration tags – for declaring methods and country variable
  - Scriplet tag – Logic for incrementing counters
  - Expression Tags – For printing the counter values

# Your Turn (Code Assistance)

- Sample Code (sample.jsp):



```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Sample</title>
</head>
<%!int count = 0;

    void incrementCount() {
        count++;
    }%>
<body>
<h1 style="margin-left: 25%;">Sample Page</h1>
<h2>
<%
    int localVariable = 0;
    out.print("This page is viewed " + count + " times");
    incrementCount();

%>
</h2>
The value of the local variable is
<%=localVariable%>
<%localVariable++; %>
</body>
</html>
```

Declaration Tag for declaring the count variable and the method to increment the counter.

Also define a local variable to see the difference between variables declared within scriptlet tag and declaration tag.

Scriptlets Tag for incrementing the counter.

Prints the value using Expression tag

Increments the local variable

# Let’s Take A Break…

Time for
a Break

## Summary

- Describing A JSP:
  - JSPs are HTML files that can have Java embedded code
  - Used for dynamic webpages
- The JSP Lifecycle:
  - Translation/Compilation > Instantiation > Initialization > Service > Destroy
- JSP Vs Servlet:
  - Servlets are a Java files, JSP are HTML translated into Java files
- JSP Elements:
  - Scriplets, Declarations, Expressions

platform By Per Scholas

## End of Section

# End of Module