University of Miskolc
Faculty of Mechanical Engineering and Informatics

Java Web Application Development Technology

N13020008

JSP – Java Servlet Pages
TomCat
JavaBeans

Tamás Tompa, PhD
assistant professor
Department of Information Technology
University of Miskolc

2024.

What is JSP?

- Java Server Pages (JSP) is **a server-side programming technology** that enables the **creation of dynamic**, platform-independent method for building **Web-based applications**
  - technology for developing Webpages that supports dynamic content
- JSP have access to the entire family of Java APIs, including the JDBC API to access enterprise databases
- JavaServer Pages component is **a type of Java servlet that is designed to fulfill the role of a user interface** for a Java web application
- Using JSP, you can **collect input from users through Webpage forms**, present records from a database or another source, and create Webpages dynamically
- JavaServer Pages **often serve the same purpose as programs implemented using the Common Gateway Interface** (CGI)

Environment Setup

- **Java Software Development Kit (SDK)**
  - setting up the PATH environment variable appropriately

```
set PATH = C:\jdk1.5.0_20\bin;%PATH%
set JAVA_HOME = C:\jdk1.5.0_20
```

- **Setting up Web Server: Tomcat**

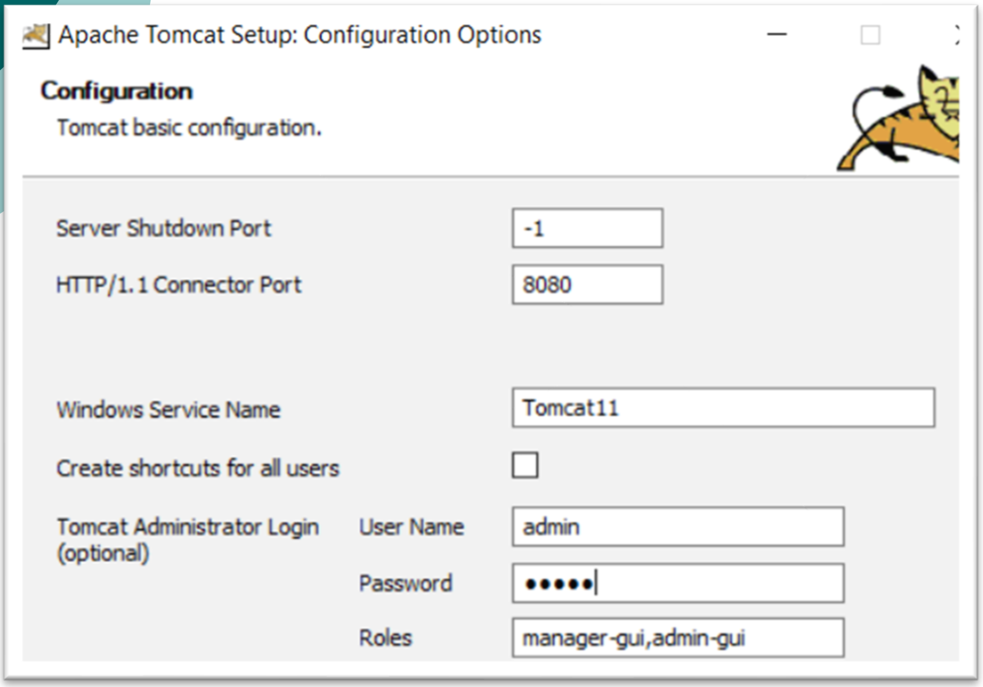
  - Apache **Tomcat is an open source software implementation of the JavaServer Pages and Servlet technologies (**https://tomcat.apache.org/)
  - Since servlets are not part of the Java Platform, Standard Edition, you must identify the servlet classes to the compiler
    - set CATALINA = C:\apache-tomcat-5.5.29
  - After a **successful startup**, the default web-applications included with **Tomcat will be available by visiting** http://localhost:8080/
  - **Start the server: C:\apache-tomcat-5.5.29\bin\startup.bat**

Apache Tomcat

- **Open-source web server and servlet container** developed by the Apache Software Foundation
- **Completely open-source** and free to use under the Apache License
- Widely used for **running Java-based web applications** and supports technologies such as Java Servlets, JavaServer Pages (JSP), WebSockets, and more
- Designed to be lightweight and **easy to use**
- **Can be configured through XML files**, making it flexible for different development and production environments. For example, the **server.xml** file is used to configure
- **Servlet and JSP Support**
  - implements the Java Servlet and JavaServer Pages (JSP) specifications, allowing developers to run dynamic web applications written in Java
- Compontents: Catalina (servlet container), Coyote (HTTP connector), Jasper (JSP engine), Cluster (load balancing)

Tomcat install

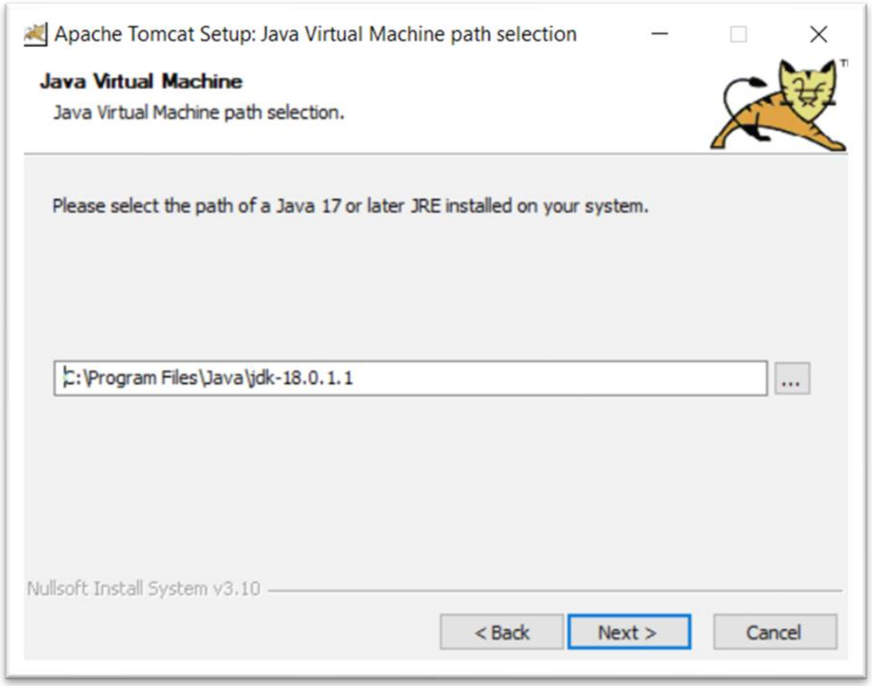
- **Zip or Windows Service Installer** (preferred)
  - https://tomcat.apache.org/download-11.cgi



Apache Tomcat Setup: Configuration Options

Configuration
Tomcat basic configuration.

Server Shutdown Port		-1
HTTP/1.1 Connector Port		8080
Windows Service Name		Tomcat11
Create shortcuts for all users		☐
Tomcat Administrator Login (optional)	User Name	admin
	Password	•••••
	Roles	manager-gui,admin-gui



Apache Tomcat Setup: Java Virtual Machine path selection

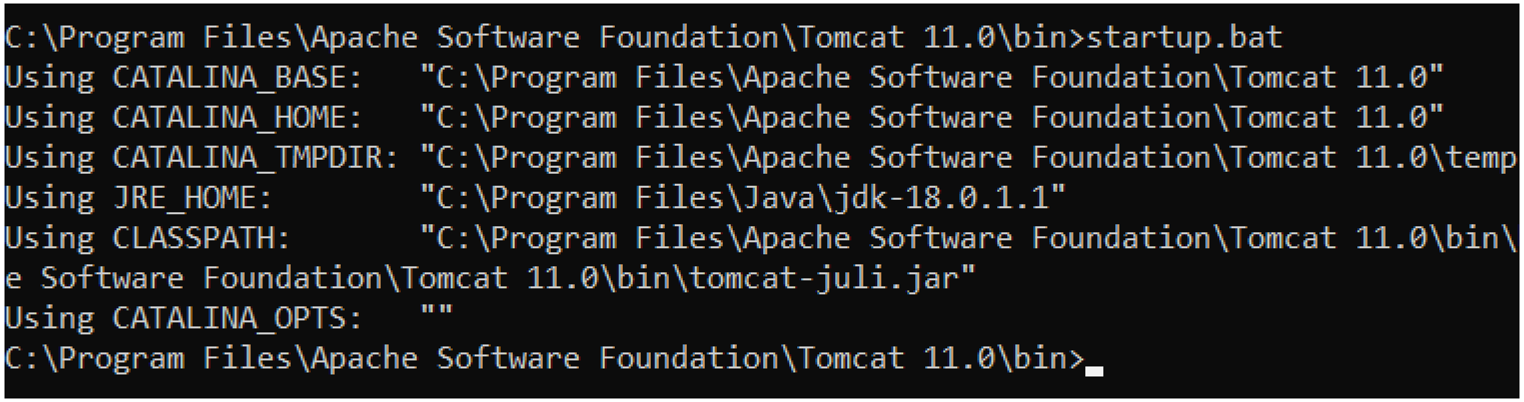
Java Virtual Machine
Java Virtual Machine path selection.

Please select the path of a Java 17 or later JRE installed on your system.

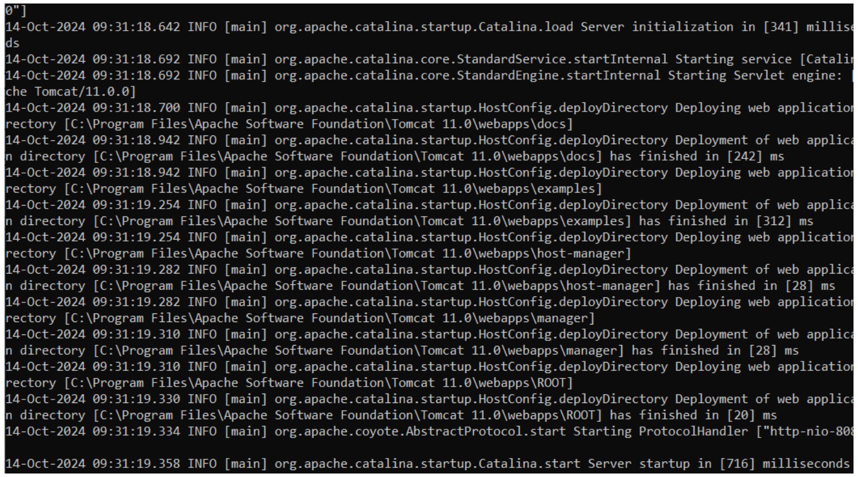
C:\Program Files\Java\jdk-18.0.1.1

Tomcat starting

- **C:\Program Files\Apache Software Foundation\Tomcat 11.0\bin>**
- **startup.bat**



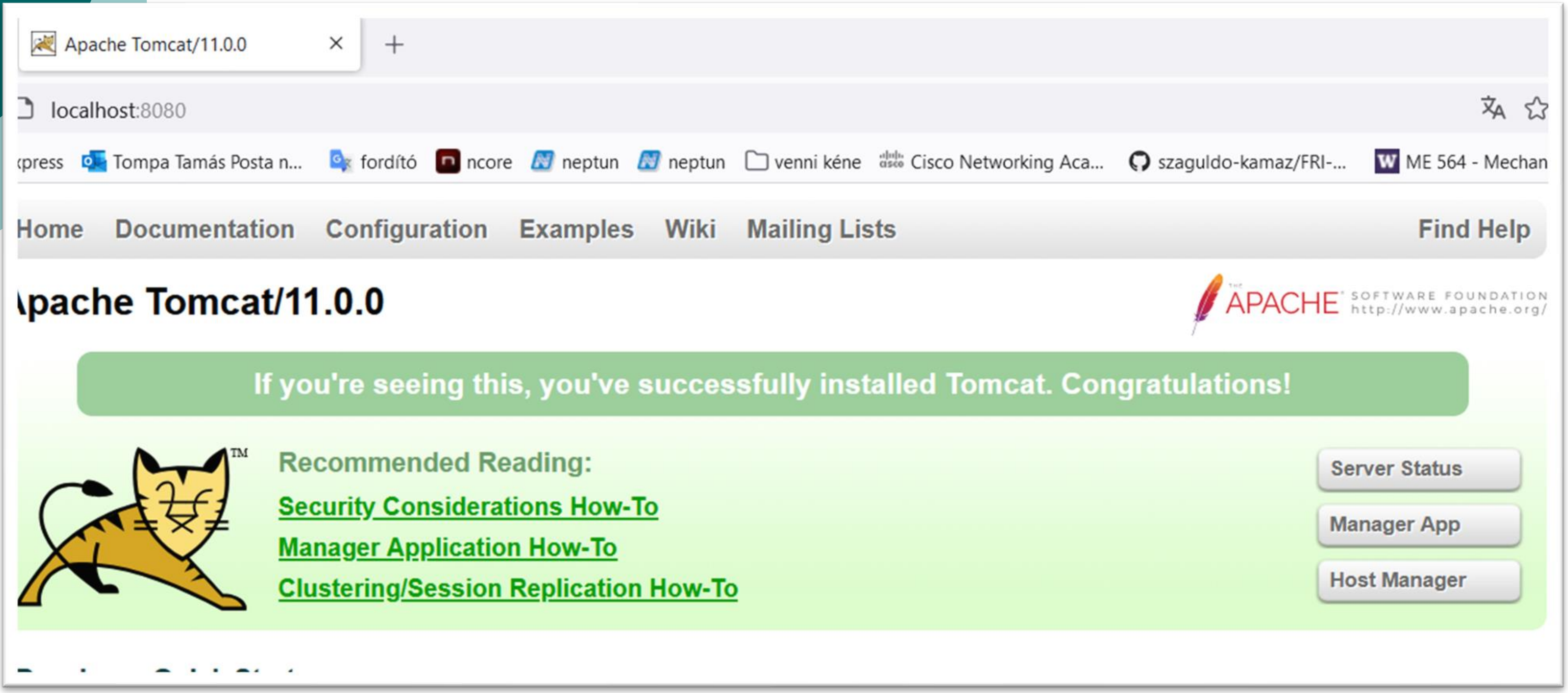
```
C:\Program Files\Apache Software Foundation\Tomcat 11.0\bin>startup.bat
Using CATALINA_BASE:   "C:\Program Files\Apache Software Foundation\Tomcat 11.0"
Using CATALINA_HOME:   "C:\Program Files\Apache Software Foundation\Tomcat 11.0"
Using CATALINA_TMPDIR: "C:\Program Files\Apache Software Foundation\Tomcat 11.0\temp
Using JRE_HOME:        "C:\Program Files\Java\jdk-18.0.1.1"
Using CLASSPATH:       "C:\Program Files\Apache Software Foundation\Tomcat 11.0\bin\
e Software Foundation\Tomcat 11.0\bin\tomcat-juli.jar"
Using CATALINA_OPTS:   ""
C:\Program Files\Apache Software Foundation\Tomcat 11.0\bin>
```



```
0"]
14-Oct-2024 09:31:18.642 INFO [main] org.apache.catalina.startup.Catalina.load Server initialization in [341] millise
ds
14-Oct-2024 09:31:18.692 INFO [main] org.apache.catalina.core.StandardService.startInternal Starting service [Catalin
14-Oct-2024 09:31:18.692 INFO [main] org.apache.catalina.core.StandardEngine.startInternal Starting Servlet engine: [
che Tomcat/11.0.0]
14-Oct-2024 09:31:18.700 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web applicatio
rectory [C:\Program Files\Apache Software Foundation\Tomcat 11.0\webapps\docs]
14-Oct-2024 09:31:18.942 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web applica
n directory [C:\Program Files\Apache Software Foundation\Tomcat 11.0\webapps\docs] has finished in [242] ms
14-Oct-2024 09:31:18.942 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web applicatio
rectory [C:\Program Files\Apache Software Foundation\Tomcat 11.0\webapps\examples]
14-Oct-2024 09:31:19.254 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web applica
n directory [C:\Program Files\Apache Software Foundation\Tomcat 11.0\webapps\examples] has finished in [312] ms
14-Oct-2024 09:31:19.254 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web applicatio
rectory [C:\Program Files\Apache Software Foundation\Tomcat 11.0\webapps\host-manager]
14-Oct-2024 09:31:19.282 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web applica
n directory [C:\Program Files\Apache Software Foundation\Tomcat 11.0\webapps\host-manager] has finished in [28] ms
14-Oct-2024 09:31:19.282 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web applicatio
rectory [C:\Program Files\Apache Software Foundation\Tomcat 11.0\webapps\manager]
14-Oct-2024 09:31:19.310 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web applica
n directory [C:\Program Files\Apache Software Foundation\Tomcat 11.0\webapps\manager] has finished in [28] ms
14-Oct-2024 09:31:19.310 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web applicatio
rectory [C:\Program Files\Apache Software Foundation\Tomcat 11.0\webapps\ROOT]
14-Oct-2024 09:31:19.330 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web applica
n directory [C:\Program Files\Apache Software Foundation\Tomcat 11.0\webapps\ROOT] has finished in [20] ms
14-Oct-2024 09:31:19.334 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-nio-808

14-Oct-2024 09:31:19.358 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in [716] milliseconds
```

Tomcat after starting

- http://localhost:8080/



Apache Tomcat/11.0.0

localhost:8080

Apache Tomcat/11.0.0

If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:

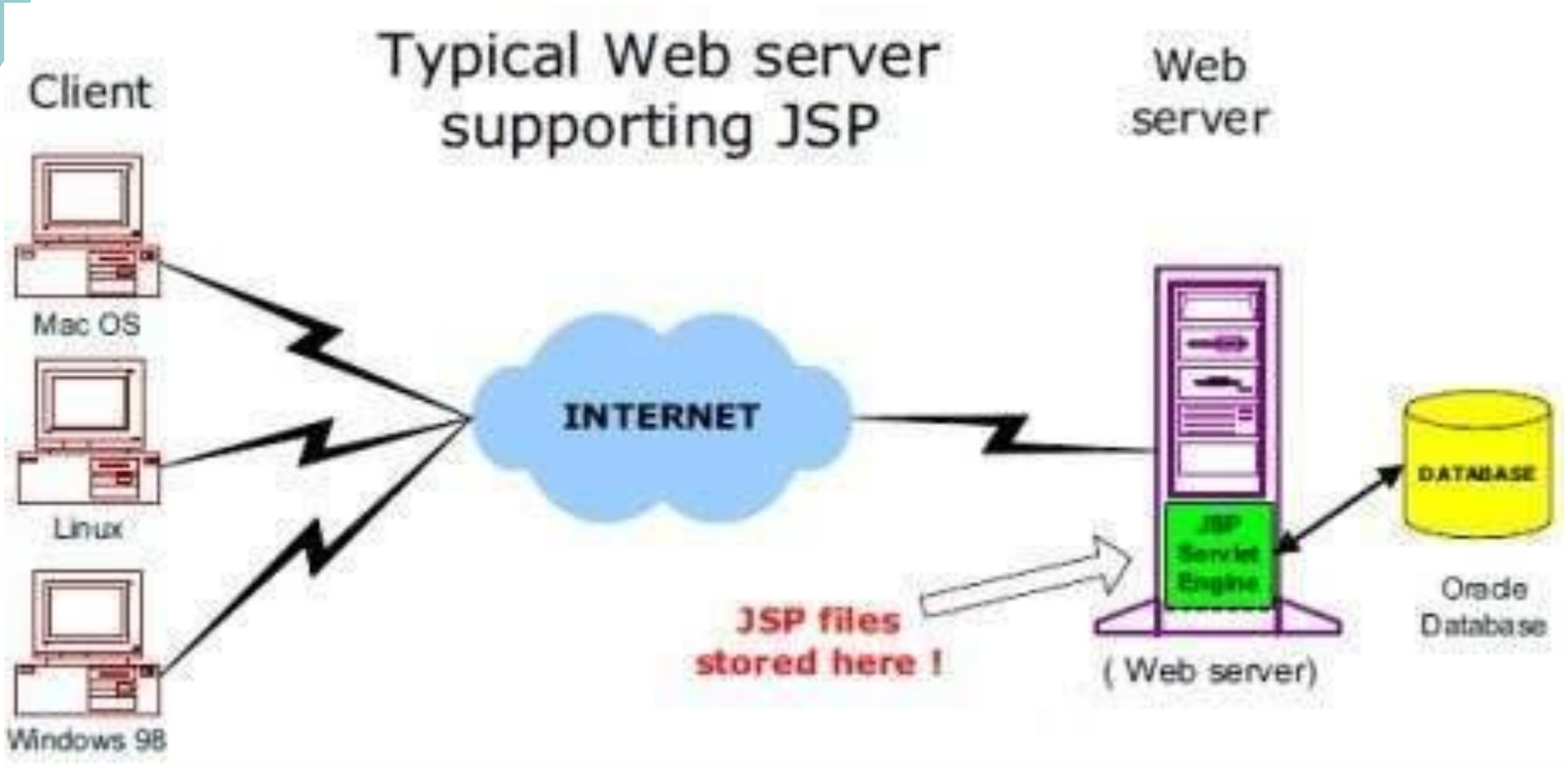
Security Considerations How-To

Manager Application How-To

Clustering/Session Replication How-To

JSP architecture

- **The web server needs a JSP engine**, i.e, a **container** to process JSP pages
- The **JSP container is responsible for intercepting requests for JSP pages**



JSP processing

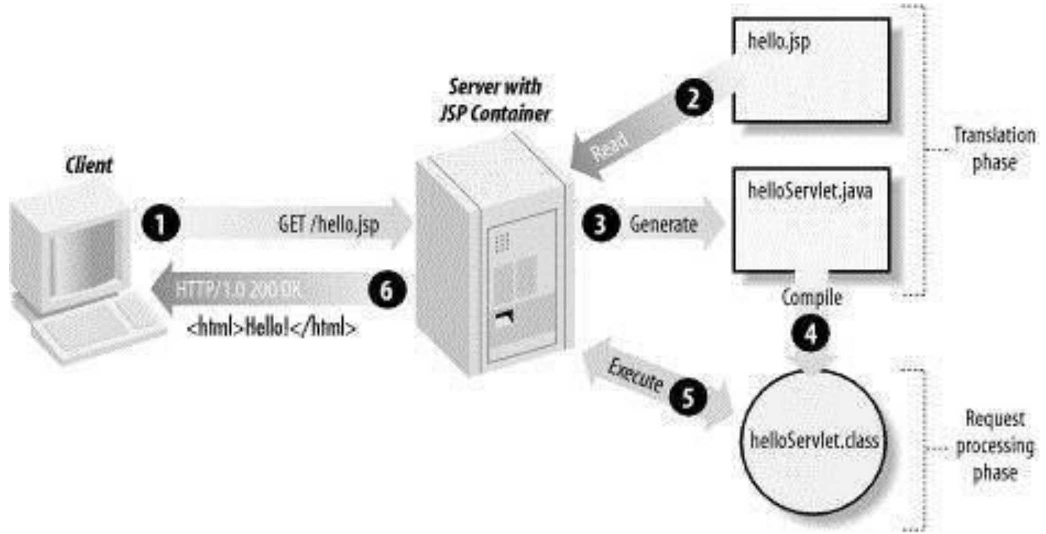
1. As with a normal page, **your browser sends an HTTP request to the web server**
2. The web **server recognizes that the HTTP request is for a JSP page and forwards it to a JSP engine**. This is done by using the URL or JSP page which ends with .jsp instead of .html
3. The JSP **engine loads the JSP page from disk and converts it into a servlet content**. This conversion is very simple in which all template text is converted to println( ) statements and all JSP elements are converted to Java code. This code implements the corresponding dynamic behavior of the page
4. The JSP **engine compiles the servlet into an executable class and forwards the original request to a servlet engine**

Processing

5. A part of the **web server called the servlet engine loads the Servlet class and executes it**. During execution, the servlet produces an output in HTML format. The output is furthur passed on to the web server by the servlet engine inside an HTTP response

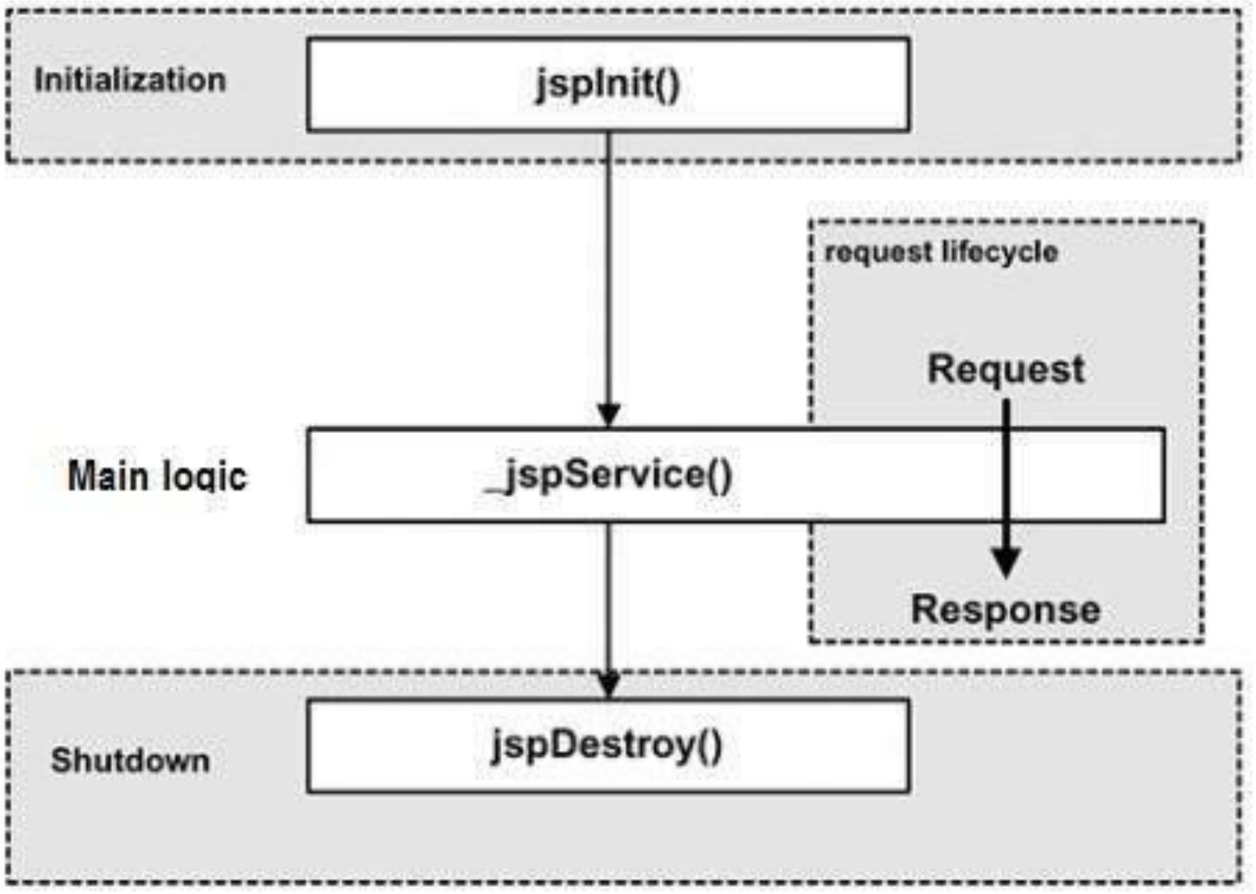
6. **The web server forwards the HTTP response to your browser in terms of static HTML content**

7. **Finally, the web browser handles the dynamically-generated HTML page** inside the HTTP response exactly as if it were a static page



Lifecycle

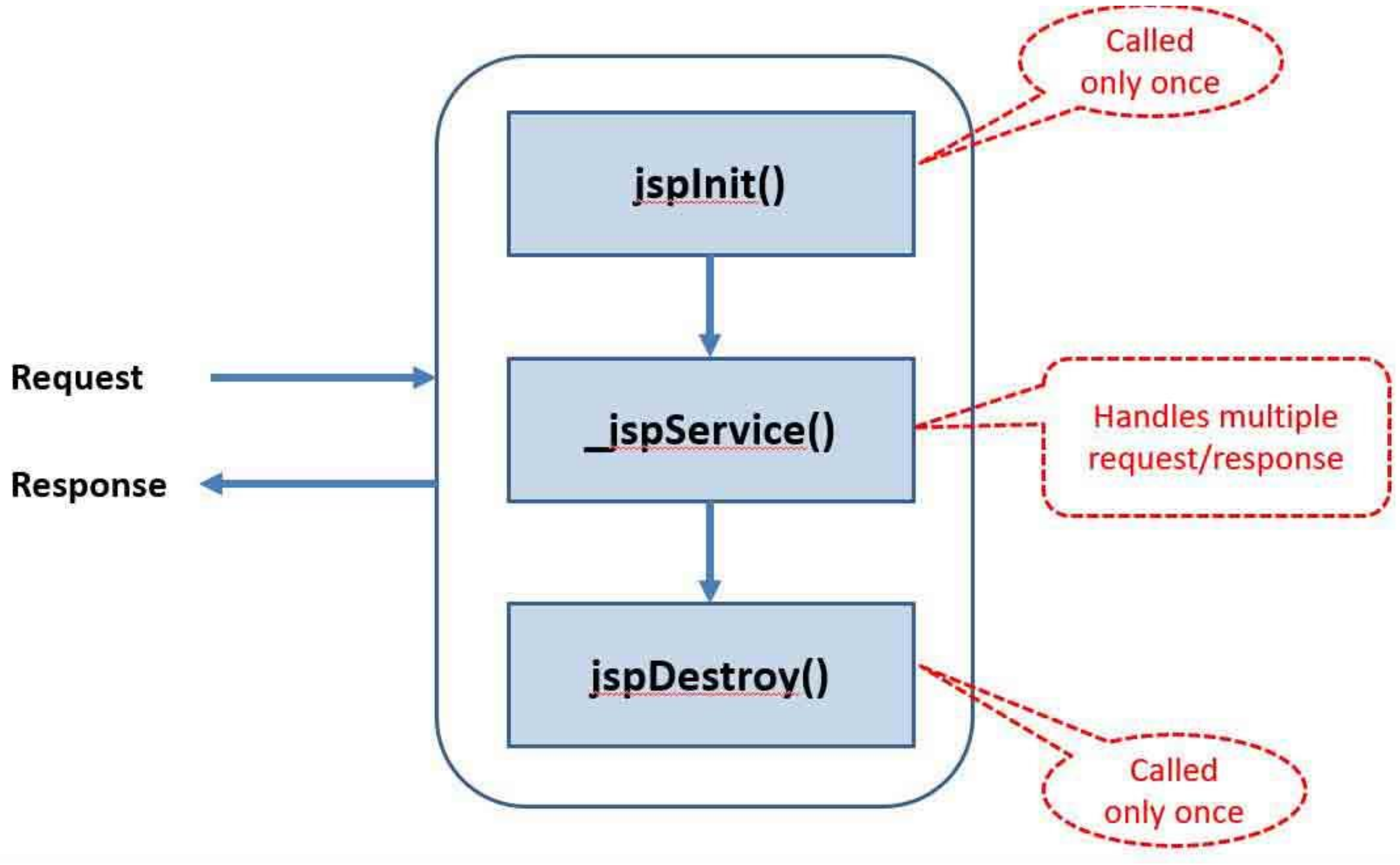
1. Compilation
2. Initialization
3. Execution
4. Cleanup



Lifecycle

- **Compilation**
  - JSP engine first checks to see whether it needs to compile the page. If the page has never been compiled, or if the JSP has been modified since it was last compiled, the JSP engine compiles the page
- **Initialization**
  - When a container loads a JSP it invokes the **jspInit()** method before servicing any requests
- **Execution**
  - represents all interactions with requests until the JSP is destroyed
- **Cleanup**
  - represents when a JSP is being removed from use by a container
  - **jspDestroy()** method

Lifecycle


Called only once
jspInit()
Request
Response
_jspService()
Handles multiple request/response
jspDestroy()
Called only once

Syntax

- **The Scriptlet**
  - a scriptlet can contain any number of JAVA language statements

<% code fragment %>
<%= expression %>

hello.jsp and put this file in C:\apache-tomcat7.0.2\webapps\ROOT
-> http://localhost:8080/hello.jsp

```
<html>
    <head>
    <title>
    Hello World
    </title>
    </head>

<body>
    Hello World!<br/>
    <% out.println("Your IP address is " + request.getRemoteAddr()); %>
</body>
</html>
```

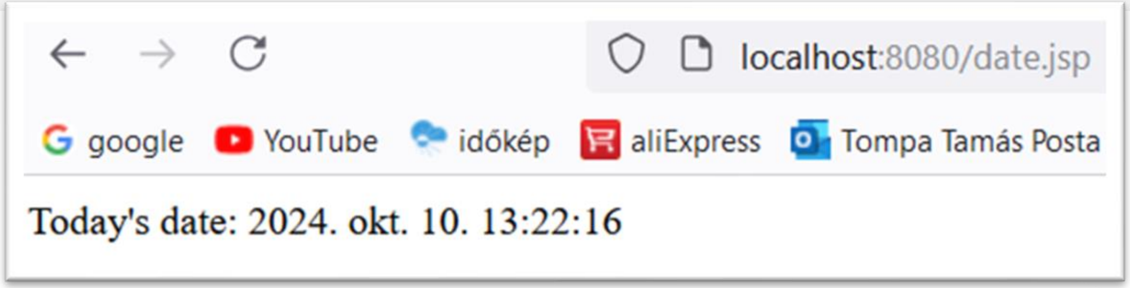
Syntax

- **The Scriptlet**
  - a scriptlet can contain any number of JAVA language statements

<% code fragment %>
<%= expression %>

hello.jsp and put this file in C:\apache-tomcat7.0.2\webapps\ROOT -> http://localhost:8080/date.jsp

```
<html>
   <head><title>A Comment Test</title></head>

   <body>
      <p>Today's date: <%= (new java.util.Date()).toLocaleString()%></p>
   </body>
</html>
```



Today's date: 2024. okt. 10. 13:22:16

Comments

- JSP comment marks text or statements that the JSP container should ignore

<%-- This is JSP comment --%>

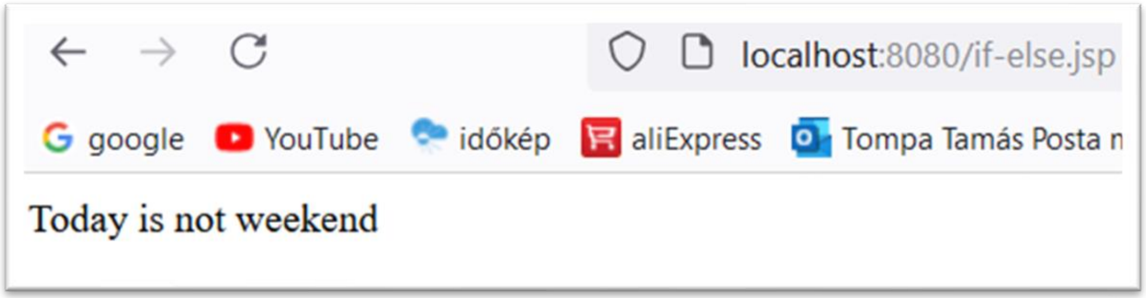
```
<html>
   <head><title>A Comment Test</title></head>

   <body>
      <h2>A Test of Comments</h2>
      <%-- This comment will not be visible in the page source --%>
   </body>
</html>
```

Syntax

S.No.	Syntax & Purpose
1	**<%-- comment --%>** A JSP comment. Ignored by the JSP engine.
2	**<!-- comment -->** An HTML comment. Ignored by the browser.
3	**<\%** Represents static <% literal.
4	**%\>** Represents static %> literal.
5	**\'** A single quote in an attribute that uses single quotes.
6	**\"** A double quote in an attribute that uses double quotes.

Decision-Making Statements

```
<%! int day = 3; %>
<html>
   <head><title>IF...ELSE Example</title></head>

   <body>
      <% if (day == 1 || day == 7) { %>
         <p> Today is weekend</p>
      <% } else { %>
         <p> Today is not weekend</p>
      <% } %>
   </body>
</html>
```



```
Today is not weekend
```

Actions

- The **actions** use constructs in XML syntax **to control the behavior of the servlet engine**
- You can dynamically insert a file, reuse JavaBeans components, forward the user to another page, or generate HTML for the Java plugin

```
<jsp:action_name attribute = "value" />
```

- **<jsp:include> Action**

```
<jsp:include page = "relative URL" flush = "true" />
```

- Let us define the following two files date_action.jsp and main_action.jsp as follows. Following is the content of the date_action.jsp file:

```
<p>Today's date: <%= (new java.util.Date()).toLocaleString()%></p>
```

Actions

- main_action.jsp

```
<html>
<head>
       <title>The include Action Example</title>
</head>
<body>
   <center>
       <h2>The include action Example</h2>
       <jsp:include page = "date_action.jsp" flush = "true" />
   </center>
</body>
</html>
```

Task1: date and time

- **Create a basic JSP web page that displays a welcome message and dynamically shows the current date and time**
- Create a new JSP file called welcome.jsp
- Structure the JSP page:
  - Add the following elements to your JSP file:
  - A header (<h1>) that says "Welcome to My JSP Page!"
  - A paragraph that introduces the purpose of the page
  - Use JSP code to dynamically display the current date and time
  - Inside the JSP file, add a scriptlet (<% %>) to get the current date and time using Java's java.util.Date class.
  - Display the result inside your HTML content

Task1: date and time

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head> <meta charset="UTF-8">
        <title>JSP Welcome Page</title>
    </head>

    <body>
        <h1>Welcome to My JSP Page!</h1>
        <p>This page dynamically shows the current date and time using JSP.</p>
        <p>Current date and time:
            <% java.util.Date currentDate = new java.util.Date();
            out.println(currentDate.toString());
            %>
        </p>
    </body>
 </html>
```

Form processing

- The browser uses two methods to pass this information to the web server
  - These methods are **the GET Method and the POST Method**

- **GET method**
  - **sends the encoded user information** appended to the page request. The page and the encoded information are **separated by the ? character**

    http://www.test.com/hello?key1=value1&key2=value2

- **POST method**
  - packages the information in **exactly the same way as the GET** method, **but** instead of sending it as a text string after a ? in the URL it **sends it as a separate message**
    - using **getParameter() method to read simple parameters** and **getInputStream() method to read binary data stream** coming from the client

Form processing

- **GET Method Example Using URL**

  http://localhost:8080/main.jsp?first_name=ZARA&last_name=ALI

main.jsp

```
<html>
<head>
        <title>Using GET Method to Read Form Data</title>
</head>
<body>
    <h1>Using GET Method to Read Form Data</h1>
    <ul>
    <li><p><b>First Name:</b> <%= request.getParameter("first_name")%> </p></li>
     <li><p><b>Last Name:</b> <%= request.getParameter("last_name")%> </p></li>
    </ul>
</body>
</html>
```

http://localhost:8080/main.jsp?first_name=ZARA&last_name=ALI

Form processing

- **GET Method Example Using URL**

http://localhost:8080/main.html

main.html

```
<html>
<body>
    <form action = "main.jsp" method = "GET">
        First Name: <input type = "text" name = "first_name"> <br />
        Last Name: <input type = "text" name = "last_name" />
        <input type = "submit" value = "Submit" />
    </form>
</body>
</html>
```

http://localhost:8080/main.html

Form processing

- **POST Method Example Using URL**

http://localhost:8080/main2.jsp

main2.jsp

```
<html>
<head>
         <title>Using GET and POST Method to Read Form Data</title>
</head>
     <body>
     <center>
     <h1>Using POST Method to Read Form Data</h1>
     <ul>
         <li><p><b>First Name:</b> <%= request.getParameter("first_name")%> </p></li>
         <li><p><b>Last Name:</b> <%= request.getParameter("last_name")%> </p></li>
     </ul>
</body>
</html>
```

Form processing

- **POST Method Example Using URL**

http://localhost:8080/hello.html

hello.html

```
<html>
<body>
    <form action = "main.jsp" method = "POST">
        First Name: <input type = "text" name = "first_name"> <br/>
        Last Name: <input type = "text" name = "last_name"/>
        <input type = "submit" value = "Submit"/>
    </form>
</body>
</html>
```

Task2: input form

- **Create a JSP web page that takes a user's name through an input form and displays a personalized greeting message based on the input**

- Create two JSP files:
  - input.jsp and greet.jsp

- Create the input form (input.jsp):
  - In the input.jsp file, create a simple form that asks for the user's name
  - When the user submits the form, the input will be sent to greet.jsp.

Task2: input form

input.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
 <!DOCTYPE html>
<html>
    <head> <meta charset="UTF-8">
        <title>User Input Form</title>
    </head>

    <body>
        <h1>Welcome! Please Enter Your Name</h1>
         <form action="greet.jsp" method="post">
            <label for="name">Name:</label>
             <input type="text" id="name" name="userName">
            <input type="submit" value="Submit">
        </form>
    </body>
</html>
```

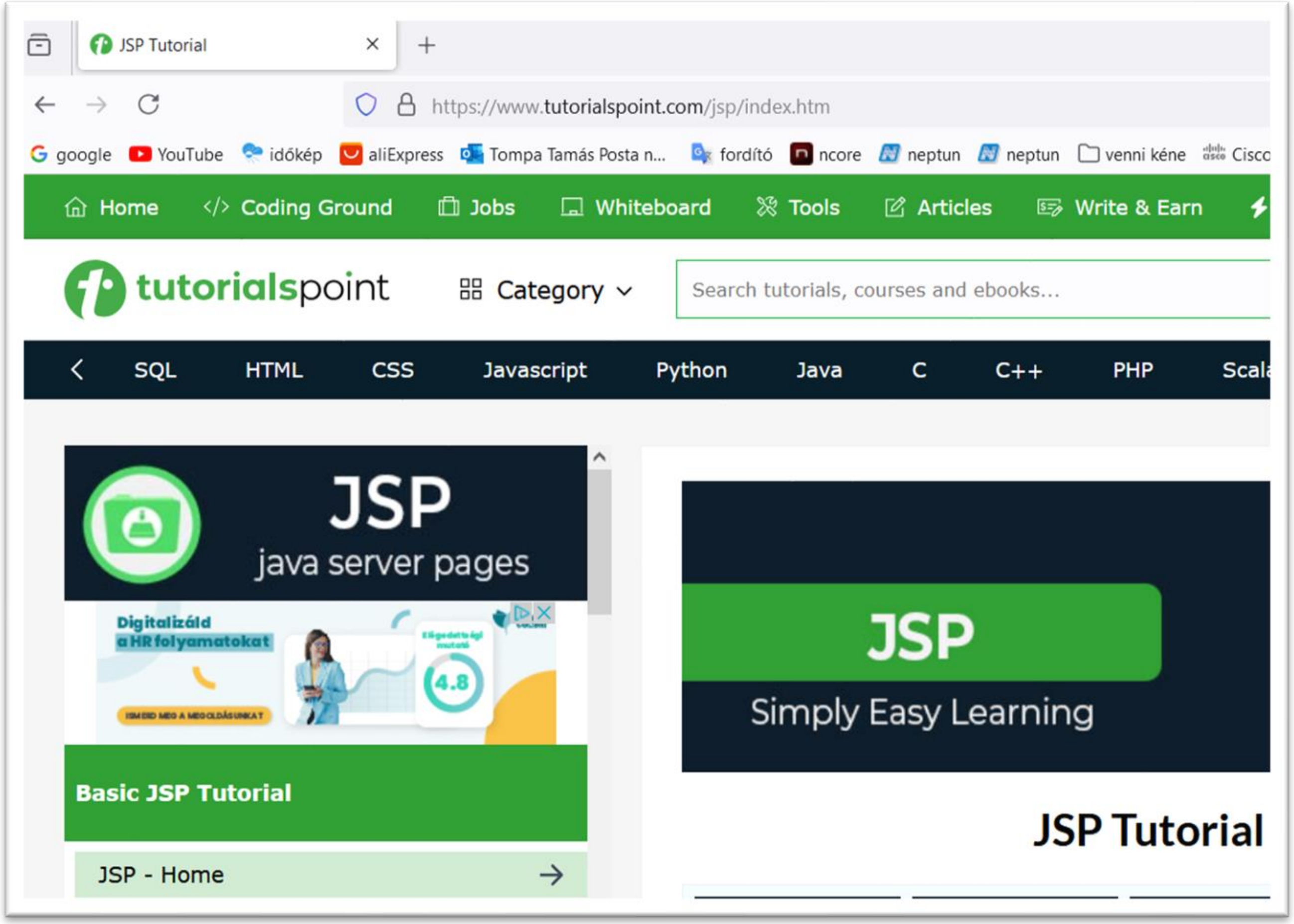
Task2: input form

greet.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
	<head>
		<meta charset="UTF-8">
		<title>Greeting Page</title>
	</head>
	<body>
		<h1>Greeting Page</h1>

		<%
		String userName = request.getParameter("userName");

		if (userName != null && !userName.trim().isEmpty()) {
				out.println("<h2>Hello, " + userName + "!</h2>"); }
		else {
				out.println("<h2>Hello, Guest!</h2>");
		}%>

	</body>
</html>
```

JSP cont…

- More examples on the https://www.w3schools.com/html/default.asp web page...


JSP Tutorial
https://www.tutorialspoint.com/jsp/index.htm
google YouTube időkép aliExpress Tompa Tamás Posta n... fordító ncore neptun neptun venni kéne Cisco
Home Coding Ground Jobs Whiteboard Tools Articles Write & Earn
tutorialspoint
Category
Search tutorials, courses and ebooks...
SQL HTML CSS Javascript Python Java C C++ PHP Scala
JSP
java server pages
Digitalizáld a HR folyamatokat
Basic JSP Tutorial
JSP - Home
JSP
Simply Easy Learning
JSP Tutorial

JSP - JavaBeans

- A JavaBean is a **specially constructed Java class** written in the Java and coded **according to the JavaBeans API** specifications

- Following are the unique characteristics that distinguish a JavaBean from other Java classes:
  - It provides a default, **no-argument constructor**
  - It should be serializable and that which can implement the **Serializable** interface
  - It may **have a number of properties** which can be read or written
  - It may have a number of "**getter**" and "**setter**" methods for the properties

JavaBeans Properties

- JavaBean property is a named attribute that can be accessed by the user of the object
- The attribute can be of any Java data type, including the classes that you define
- Property may be **read, write, read only**, or **write only**

- Accessed through two methods
  - get**PropertyName**()
    - if property name is *firstName*, your method name would be **getFirstName()** to read that property. This method is called accessor
    - read-only

  - set**PropertyName**()
    - if property name is *firstName*, your method name would be **setFirstName()** to write that property. This method is called mutator
    - write-only

JavaBeans example

```java
public class StudentsBean implements java.io.Serializable {
      private String firstName = null;
      private String lastName = null;
      private int age = 0;

       public StudentsBean() {
      }

      public String getFirstName(){
            return firstName;
      }
      public String getLastName(){
            return lastName;
      }
      public int getAge(){
            return age;
      }
      public void setFirstName(String firstName){
            this.firstName = firstName;
      }
      public void setLastName(String lastName){
            this.lastName = lastName;
      }
      public void setAge(Integer age){
            this.age = age;
      }
}
```

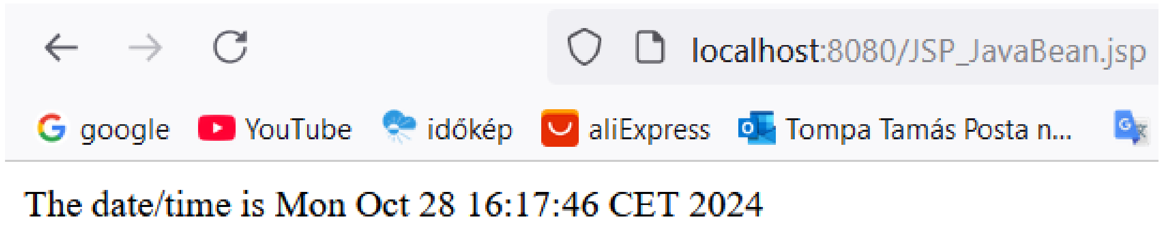
Accessing JavaBeans

- **useBean** action declares a JavaBean for use in a JSP

```
<jsp:useBean id = "bean's name" scope = "bean's scope" typeSpec/>
```

- Example:

```
<html>
   <head>
      <title>useBean Example</title>
   </head>

   <body>
      <jsp:useBean id = "date" class = "java.util.Date" />
      <p>The date/time is <%= date %>
   </body>
</html>
```



The date/time is Mon Oct 28 16:17:46 CET 2024

Accessing JavaBeans Properties

- **<jsp:getProperty/>** action to access the get methods
- **<jsp:setProperty/>** action to access the set methods

```
<jsp:useBean id = "id" class = "bean's class" scope = "bean's scope">
    <jsp:setProperty name = "bean's id" property = "property name"  value = "value"/>
    <jsp:getProperty name = "bean's id" property = "property name"/>
      ……………………………. . . . . . . . . . . . .
</jsp:useBean>
```

- The property attribute is the name of the **get** or the **set** methods that should be invoked

Accessing JavaBeans Properties

- Example:

```
<html>
<head>
   <title>get and set properties Example</title>
</head>

<body>
   <jsp:useBean id = "students" class = "StudentsBean">
      <jsp:setProperty name = "students" property = "firstName" value = "Zara"/>
      <jsp:setProperty name = "students" property = "lastName" value = "Ali"/>
      <jsp:setProperty name = "students" property = "age" value = "10"/>
 </jsp:useBean>

  <p>Student First Name: <jsp:getProperty name = "students" property = "firstName"/> </p>
  <p>Student Last Name: <jsp:getProperty name = "students" property = "lastName"/> </p>
  <p>Student Age: <jsp:getProperty name = "students" property = "age"/>  </p>
</body>
</html>
```

- Output:
  - Student First Name: Zara
  - Student Last Name: Ali
  - Student Age: 10

Thank you for your attention!

thank you