Programare avansata pe obiecte - laborator 13 (231)

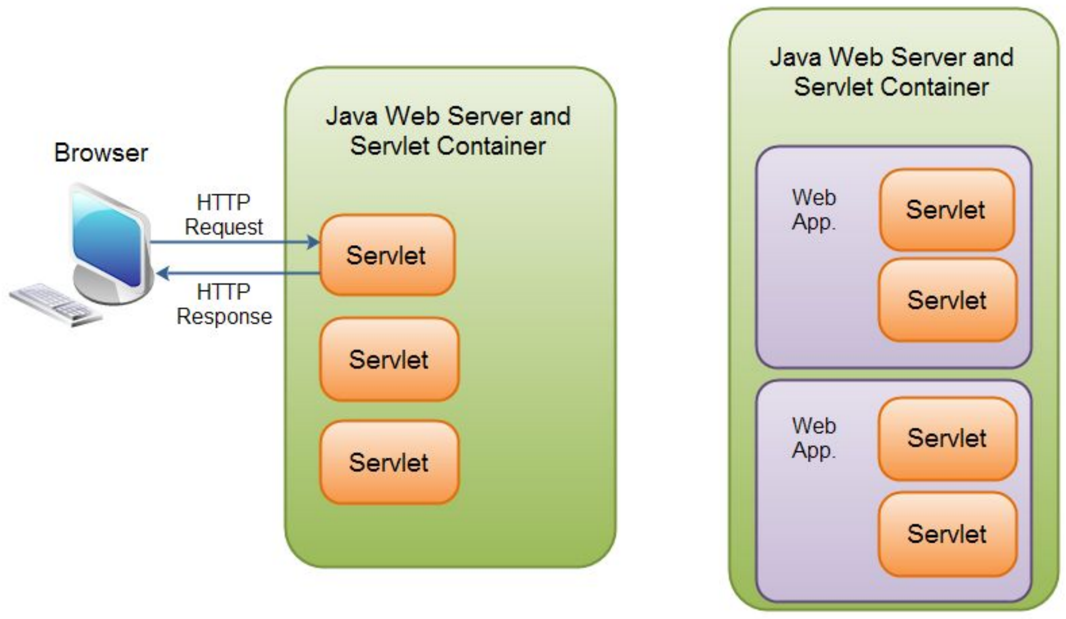
Butan Silvia
silvia.butan@endava.com
butan.silvia@gmail.com

Object oriented programming - lab 13

Java Servlets

A Servlet is a class that handles requests, processes them and replies back with a response. A Java Servlet is a Java object that responds to HTTP requests. It runs inside a Servlet container.

A Servlet is part of a Java web application. A Servlet container may run multiple web applications at the same time, each having multiple servlets running inside.



A Java web application can contain other components than servlets. It can also contain Java Server Pages (JSP) and Web Services.

HTTP Request and Response

The browser sends an HTTP request to the Java web server. The web server checks if the request is for a servlet. If it is, the servlet container is passed the request. The servlet container will then find out which servlet the request is for, and activate that servlet. The servlet is activated by calling the Servlet.service() method. Once the servlet has been activated via the service() method, the servlet processes the request, and generates a response. The response is then sent back to the browser.

Servlet Containers

Java servlet containers are usually running inside a Java web server such as Jetty, Tomcat, etc. For our lab we will use a Tomcat web server. Please setup a Tomcat web server by:

1. Download Tomcat from https://tomcat.apache.org/ - Tomcat 9.0 zip file
2. Unzip and add the folder in a location of your preference
3. In your IDE create a project: **Java Enterprise** and set up the application server as the downloaded tomcat web server.

A servlet follows a certain life cycle. The servlet life cycle is managed by the servlet container. The life cycle contains the following steps:

1. Load Servlet Class.
2. Create Instance of Servlet.
3. Call the servlets init() method.
4. Call the servlets service() method.
5. Call the servlets destroy() method.

A Java Servlet is just an ordinary Java class which implements the interface.

```
javax.servlet.Servlet;
```

The easiest way to implement this interface is to extend the class HttpServlet.

The HttpServlet class reads the HTTP request, and determines if the request is an HTTP GET, POST, PUT, DELETE, HEAD etc. and calls the corresponding method. To respond to HTTP GET requests you will extend the HttpServlet class, and override the doGet() method.

```
package servlet;

import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;

/**
* Simple servlet that implements doPost and doGet
```

```java
* The doPost takes two parameters (login and password) and displays them.
*/
@WebServlet(name = "Servlet", urlPatterns={"/Servlet"})
public class SimpleServlet extends HttpServlet {

   protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws IOException {
       response.setContentType("text/html");
       PrintWriter writer = response.getWriter();
       writer.println("This resource is not available directly.");
   }

   protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws IOException {
       PrintWriter out = response.getWriter();
       response.setContentType("text/html");
       out.println("<html><head></head><body>");
       String login = request.getParameter("login");
       String password = request.getParameter("password");
       out.println("<h1>Super Secret Login Information</h1>");
       out.println("<p>Login: " + login + "</p>");
       out.println("<p>Password: " + password + "</p>");
       out.println("</body></html>");
   }
}
```

```html
<!DOCTYPE html>
<html lang="en">
   <head>
       <meta charset="UTF-8">
       <title>Title</title>
   </head>
   <body>
       <form action="Servlet" method="post">
           <p>Login<input name="login" type="text"/></p>
           <p>Password<input name="password" type="password"/></p>
           <p><input type="submit" value="Login"/></p>
       </form>
   </body>
</html>
```

Servlet, HttpServlet and JSP

It's important to understand that the Servlet technology is not limited to the HTTP protocol. Servlet is a generic interface and the HttpServlet is an extension of that interface – adding HTTP specific support – such as doGet and doPost.

The Servlet technology is also the main driver of a number of other web technologies such as JSP- JavaServer Pages, Spring MVC, etc.

JavaServer Pages (JSP) allows dynamic content injection into static contents using Java and Java Servlets. We can make requests to a Java Servlet, perform relevant logic, and render a specific view server-side to be consumed client-side.

JavaServer Pages (JSP) enabled Java-specific data to be passed into or placed within a .jsp view and consumed client-side.

JSP files are essentially .html files with some extra syntax, and a couple of minor initial differences:

- the .html suffix is replaced with .jsp (it's considered a .jsp file type)
- the following tag is added to the top of the .html markup elements:

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
```

JSP Syntax

There are two ways to add Java code to a .jsp.

1. We can use basic **Java Scriptlet** syntax which involves placing Java code blocks within two Scriptlet tags:

```
<% Java code here %>
```

2. The second method is specific to XML:

```
<jsp:scriptlet>
    Java code here
</jsp:scriptlet>
```

IWe can use conditional logic client side with JSP by using if, then, and else clauses and then wrapping the relevant blocks of markup with those brackets.

```
<% if (condition) {%>
    <div>Hei!</div>
<% } else { %>
    <p>Hello!</p>
<% } %>
```

Resources:

Check the following tutorial for more details: https://www.baeldung.com/jsp

Examples:

index.jsp which will be displayed when we access the URL context in Tomcat:

```
<%-- Created by IntelliJ IDEA. --%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
   <head>
       <title>$Title$</title>
   </head>
   <body>
       Hello World
       <p>To invoke the java servlet click <a
href="${pageContext.request.contextPath}/Servlet">here</a></p>
   </body>
</html>
```

Java in Static Page Example:

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
   <head>
       <title>Java in Static Page Example</title>
   </head>
   <body>
       <h1>Java in Static Page Example</h1>
       <%
           String[] arr = {"What's up?", "Hello", "It's a nice day
today!"};
```

```
            String greetings = arr[(int)(Math.random() * arr.length)];
        %>
        <p><%= greetings %></p>
    </body>
</html>
```

JSP With Forwarding Example:

```
package servlet;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet(name = "SecondServlet", description = "JSP Servlet With
Annotations", urlPatterns = {"/SecondServlet"})
public class SecondServlet extends HttpServlet {

   @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
       String message = request.getParameter("message");
       request.setAttribute("text", message);
       request.getRequestDispatcher("/secondpage.jsp").forward(request,
response);
   }
}
```

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
   <head>
       <title>Java Binding Example</title>
   </head>
   <body>
       <h1>Bound Value</h1>
       <p>You said: ${text}</p>
   </body>
```

```
</html>
```

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
   <head>
       <title>$Title$</title>
   </head>
   <body>
       Hello World
       <p>To invoke the java servlet click <a
href="${pageContext.request.contextPath}/Servlet">here</a></p>
       <p>Java in static page: <a href="firstpage.jsp"
target="_blank">here</a></p>
       <p>Java injected by Servlet: <a href="SecondServlet?message=hello!"
target="_blank">here</a></p>
   </body>
</html>
```