SWEN90007 - Software Design and Architecture

Semester 2 – 2020 – Additional Learning Resource for Students

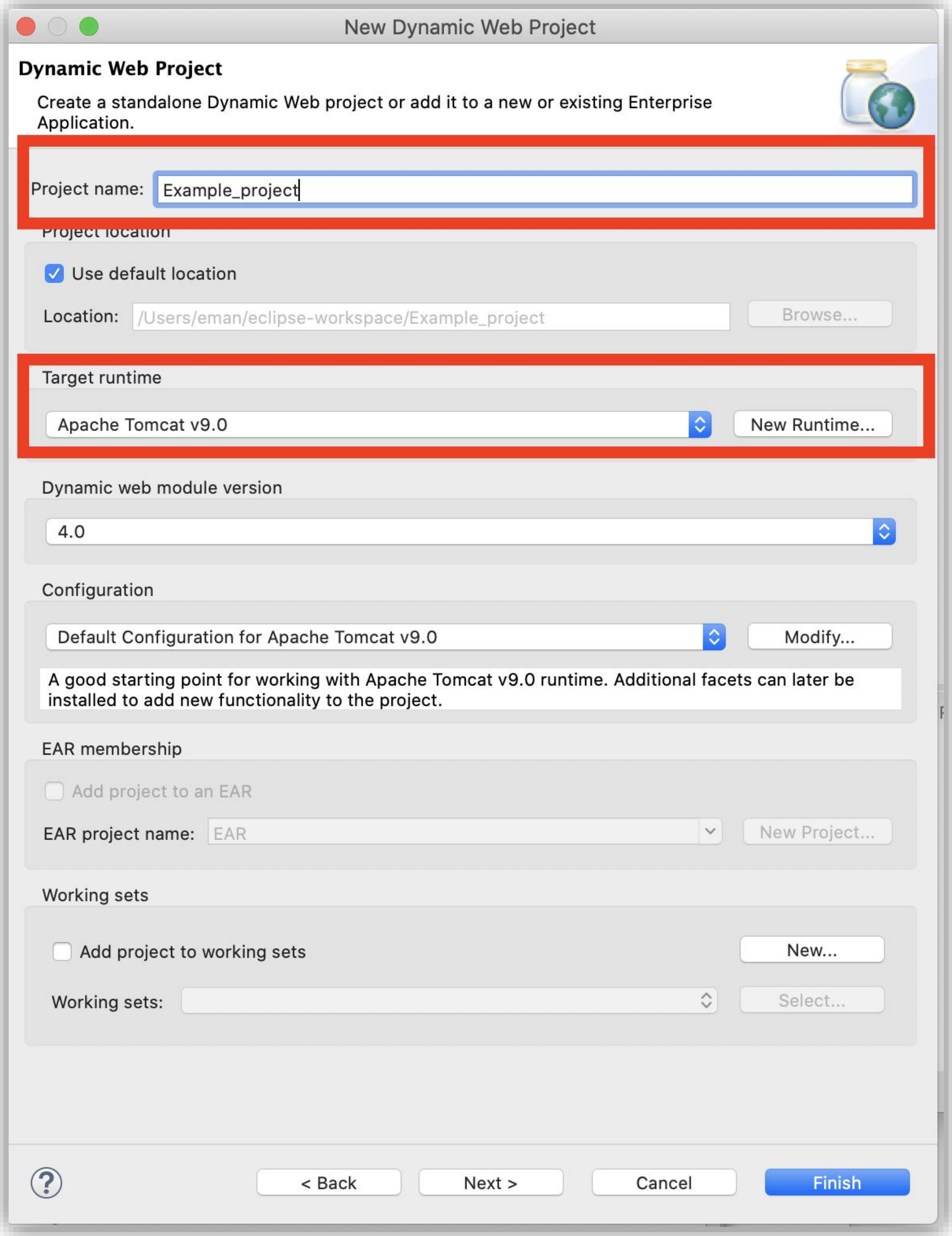
Introduction to Dynamic Web Application Development

We did setup JRE and Apache Tomcat in Eclipse IDE and we have created our first project.

1. Download and install **JDK 14**: https://www.oracle.com/java/technologies/javase-jdk14-downloads.html *(do not download the JRE. Make sure you download the* ***JDK****);*
2. Download and unzip Eclipse IDE for **Enterprise Java Developers [IDE 202006];** https://www.eclipse.org/downloads/packages/
3. Download and unzip Apache Tomcat 9.0: https://tomcat.apache.org/download-90.cgi
   (a directory of your choice. There's no installation for Apache, just unzip the project somewhere and make sure you remember that).

Recap: create a dynamic web application:

- In Eclipse, File => New => Other… =>Web Dynamic Web Project.
- Name it as "Example_project",
- Choose Apache Tomcat 9.0 as the runtime.
- Use default values for the required parameters, then press Next.
- In the last screen "check" the "`Generate web.xml deployment descriptor`" before clicking "Finish"


New Dynamic Web Project

Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name: Example_project

Project location

☑ Use default location

Location: /Users/eman/eclipse-workspace/Example_project

Target runtime

Apache Tomcat v9.0 — New Runtime...

Dynamic web module version

4.0

Configuration

Default Configuration for Apache Tomcat v9.0 — Modify...

A good starting point for working with Apache Tomcat v9.0 runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

☐ Add project to an EAR

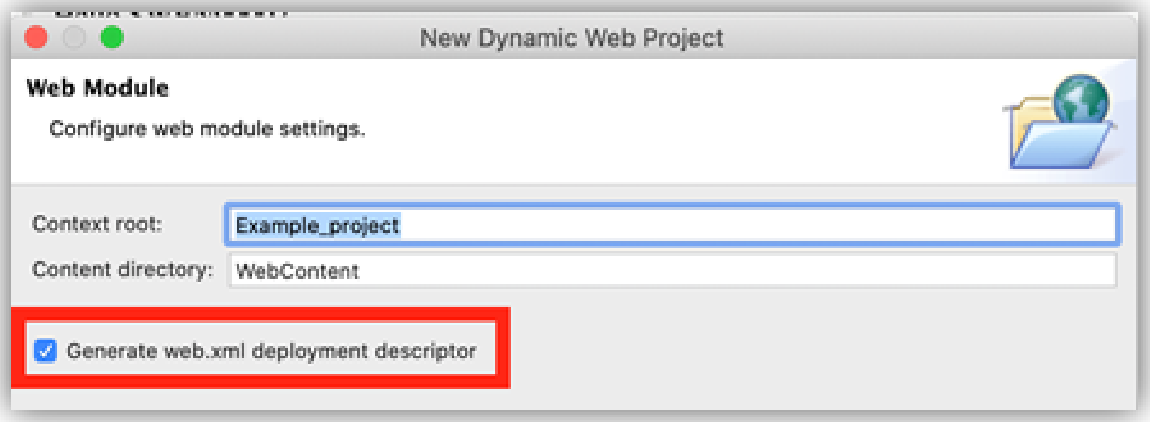
EAR project name: EAR — New Project...

Working sets

☐ Add project to working sets — New...

Working sets: — Select...

< Back | Next > | Cancel | Finish


New Dynamic Web Project

Web Module

Configure web module settings.

Context root: Example_project

Content directory: WebContent

[✓] Generate web.xml deployment descriptor

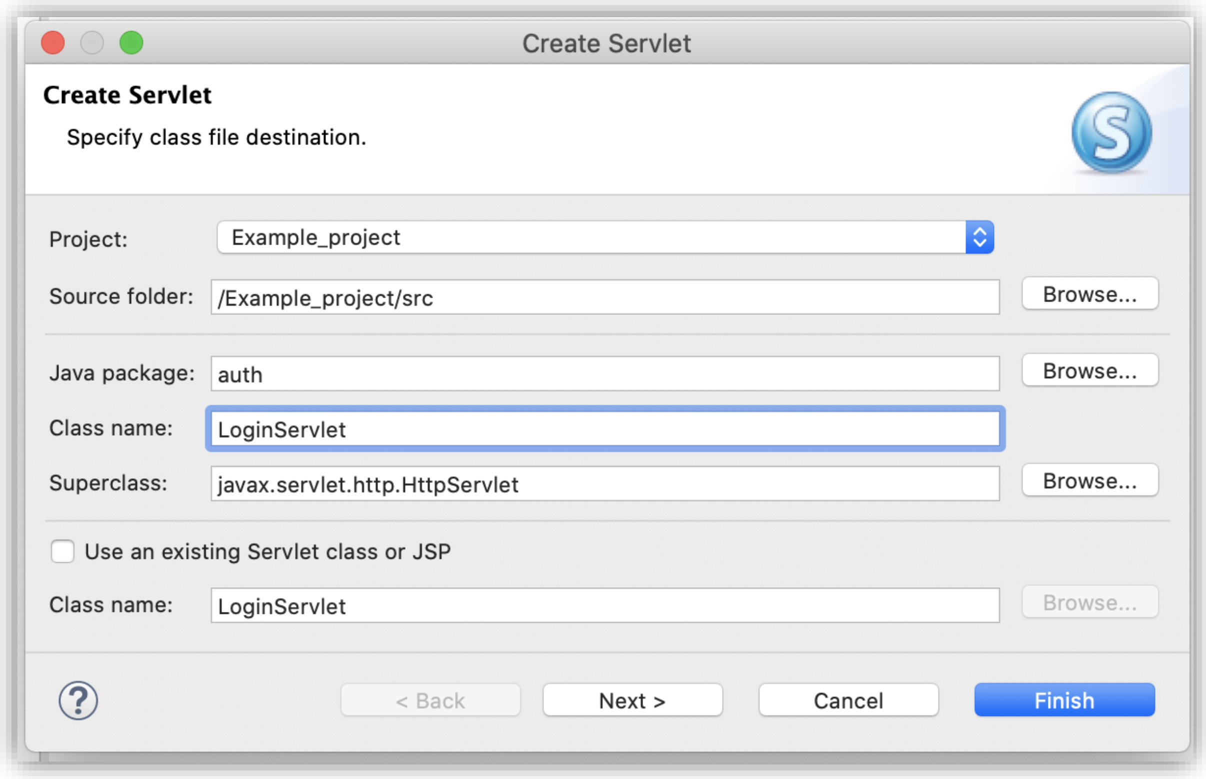
Part 1: Develop a simple Servlet based web application

Servlet:
A Servlet is Java class that runs on the server, to serve requests accessed via a request-response programming model. A Servlet has some methods that get executed when handling a client request. A common use of Servlets is in applications hosted by Web servers.

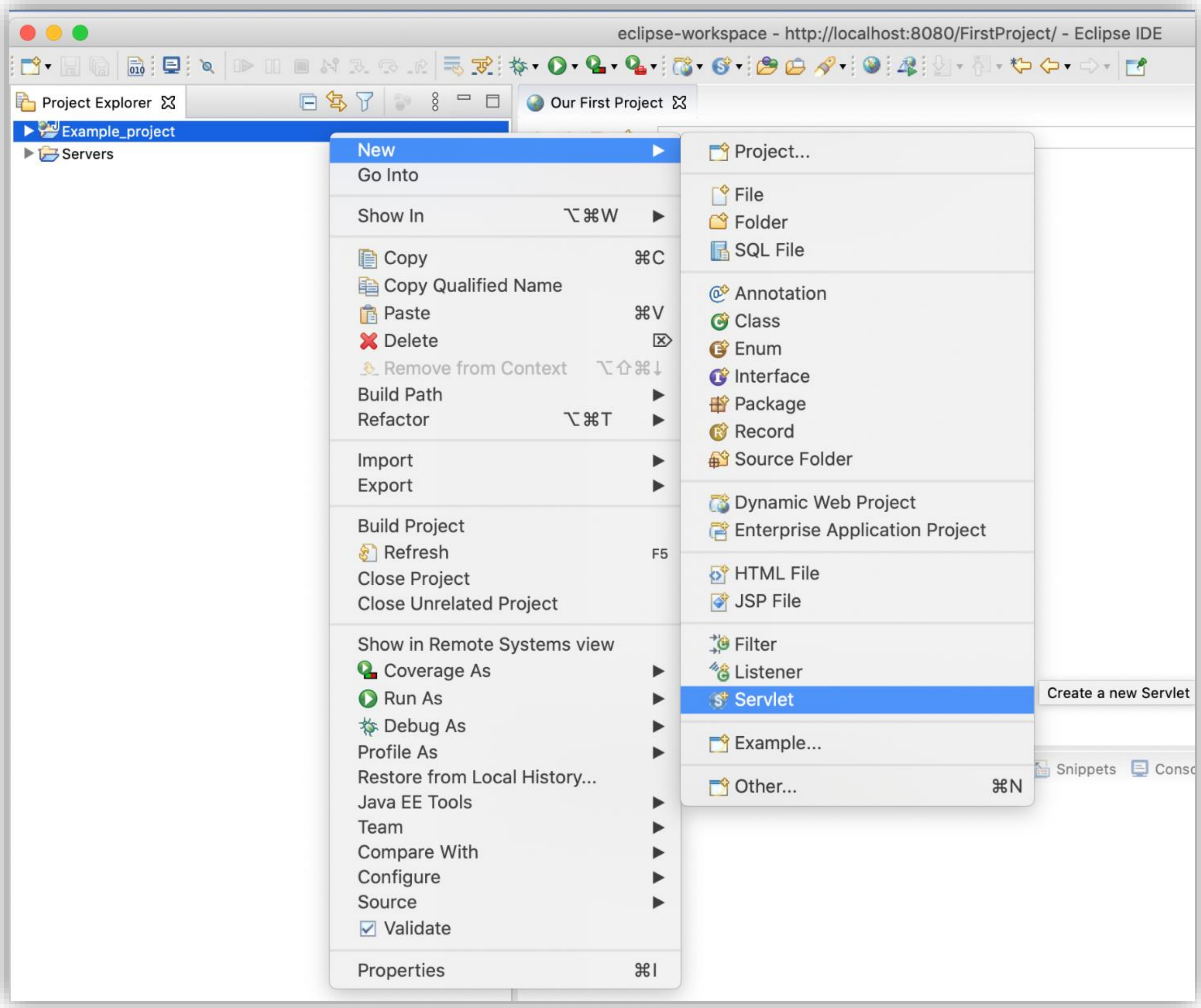
There are two different ways to define a **Servlet**:
(1) Using Java Annotations;
(2) Through a web configuration file defined in the project (web.xml).

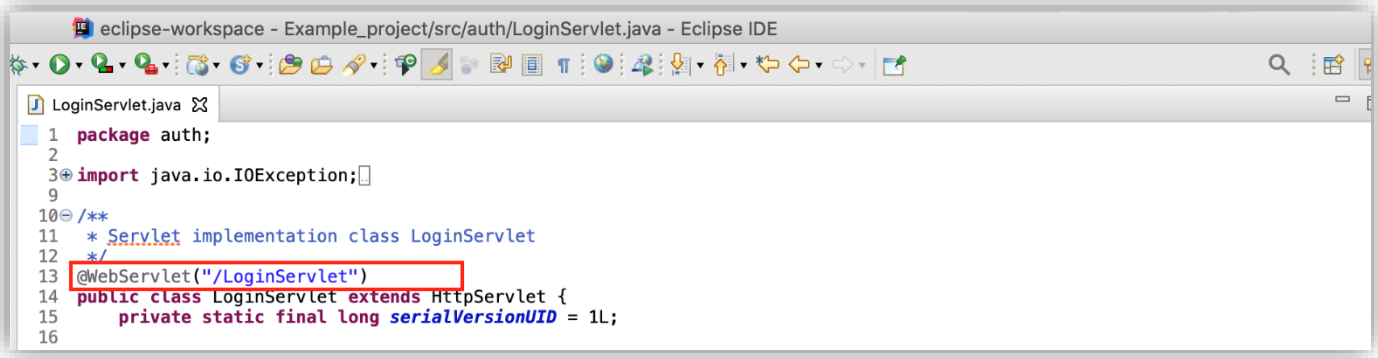
1- Defining a Servlet using Java annotation @WebServlet

- Create a new Servlet (choose the option under the web option in Eclipse), name it `LoginServlet` in package authentication - use the default values.

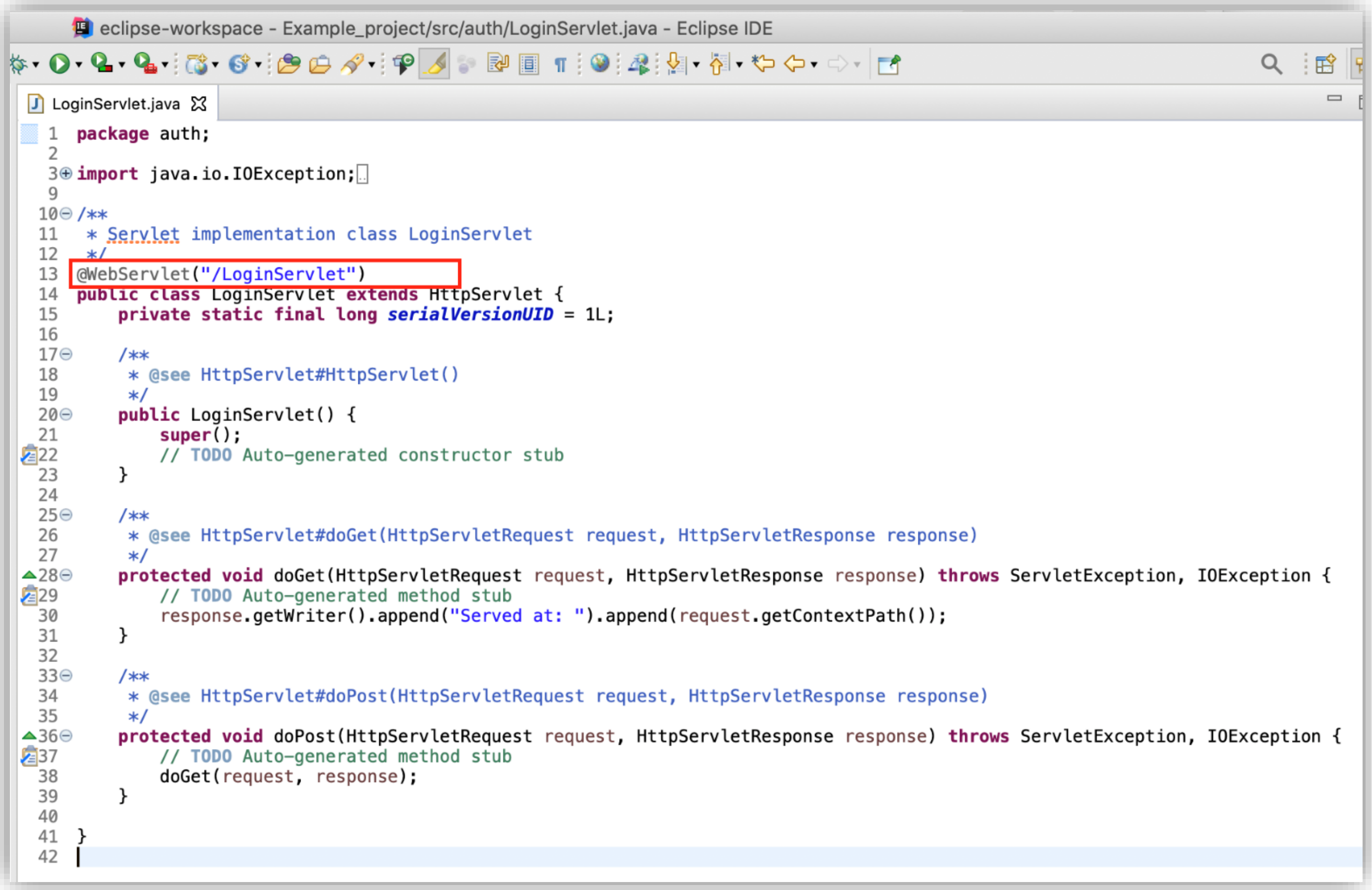


Project: Example_project
Source folder: /Example_project/src
Java package: auth
Class name: LoginServlet
Superclass: javax.servlet.http.HttpServlet
Use an existing Servlet class or JSP
Class name: LoginServlet



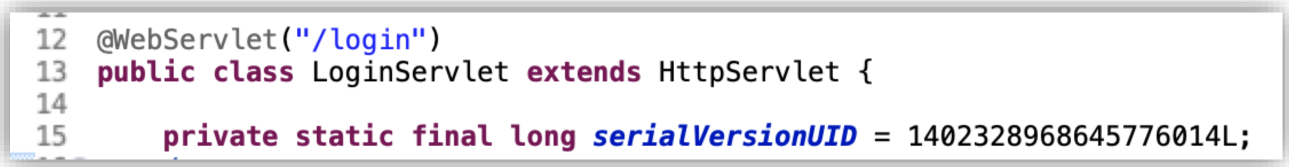

eclipse-workspace - Example_project/src/auth/LoginServlet.java - Eclipse IDE
LoginServlet.java

```
package auth;

import java.io.IOException;

/**
 * Servlet implementation class LoginServlet
 */
@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

```


eclipse-workspace - Example_project/src/auth/LoginServlet.java - Eclipse IDE
LoginServlet.java

```
package auth;

import java.io.IOException;

/**
 * Servlet implementation class LoginServlet
 */
@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public LoginServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        response.getWriter().append("Served at: ").append(request.getContextPath());
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        doGet(request, response);
    }

}
```

- We define the URL pattern of the servlet in annotation `@WebServlet()`, let’s name it “/login”



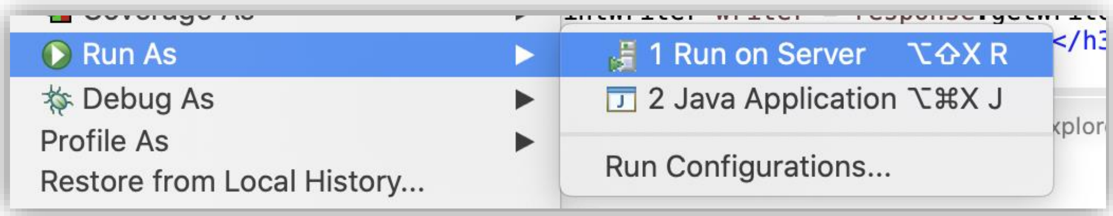
```
@WebServlet("/login")
public class LoginServlet extends HttpServlet {

    private static final long serialVersionUID = 1402328968645776014L;
```

- Add the following to the `doGet` method of the Servlet. Add the imports as needed.

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

        response.setContentType("text/html");
        System.out.println("Hello from Get method");
        PrintWriter writer = response.getWriter();
        writer.println("<h3> Hello in HTML</h3>");
```

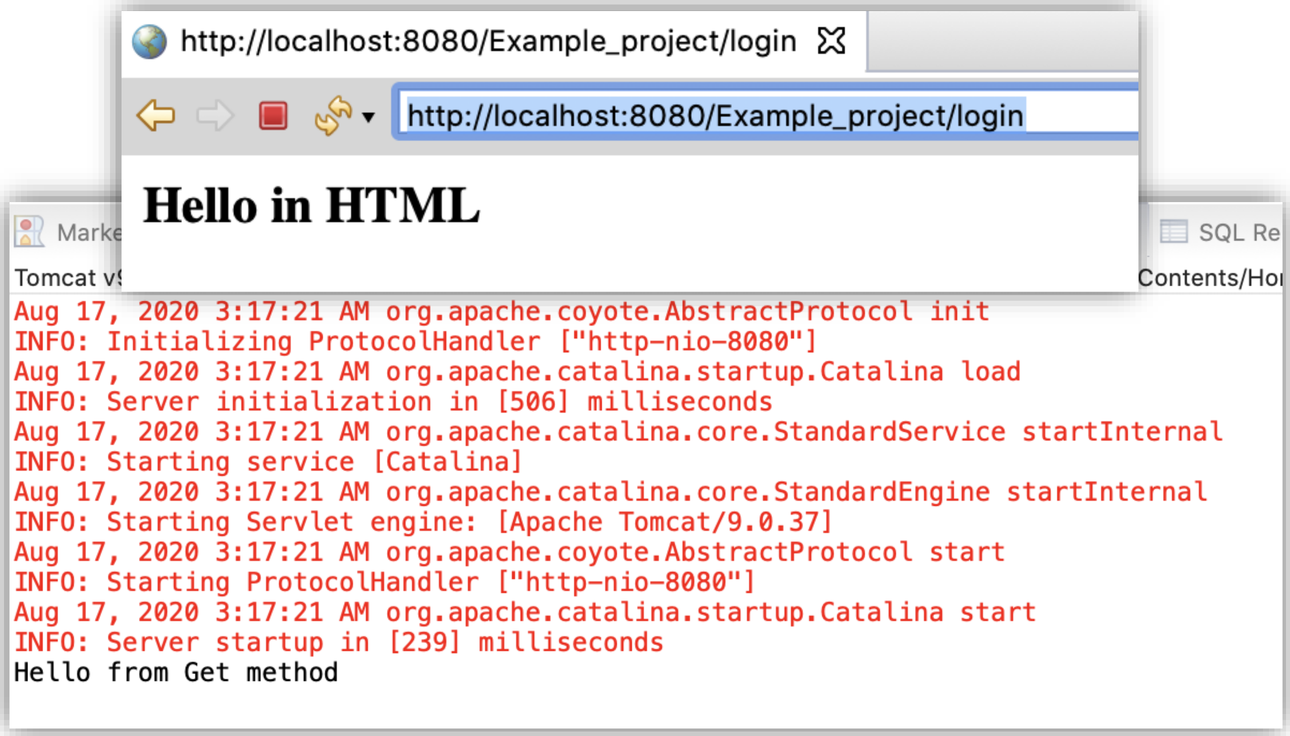
```
    }
```

- Deploy the application ***on the server***.



- The Servlet can be accessed at: [`http://localhost:[APACHE_TOMCAT_PORT]/[ProjectName]/[Servlet URL pattern]`].

  The Servlet Name is specified in the `@WebServlet` annotation in the `login` (URL Pattern).


http://localhost:8080/Example_project/login
http://localhost:8080/Example_project/login

Hello in HTML

```
Aug 17, 2020 3:17:21 AM org.apache.coyote.AbstractProtocol init
INFO: Initializing ProtocolHandler ["http-nio-8080"]
Aug 17, 2020 3:17:21 AM org.apache.catalina.startup.Catalina load
INFO: Server initialization in [506] milliseconds
Aug 17, 2020 3:17:21 AM org.apache.catalina.core.StandardService startInternal
INFO: Starting service [Catalina]
Aug 17, 2020 3:17:21 AM org.apache.catalina.core.StandardEngine startInternal
INFO: Starting Servlet engine: [Apache Tomcat/9.0.37]
Aug 17, 2020 3:17:21 AM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-nio-8080"]
Aug 17, 2020 3:17:21 AM org.apache.catalina.startup.Catalina start
INFO: Server startup in [239] milliseconds
Hello from Get method
```

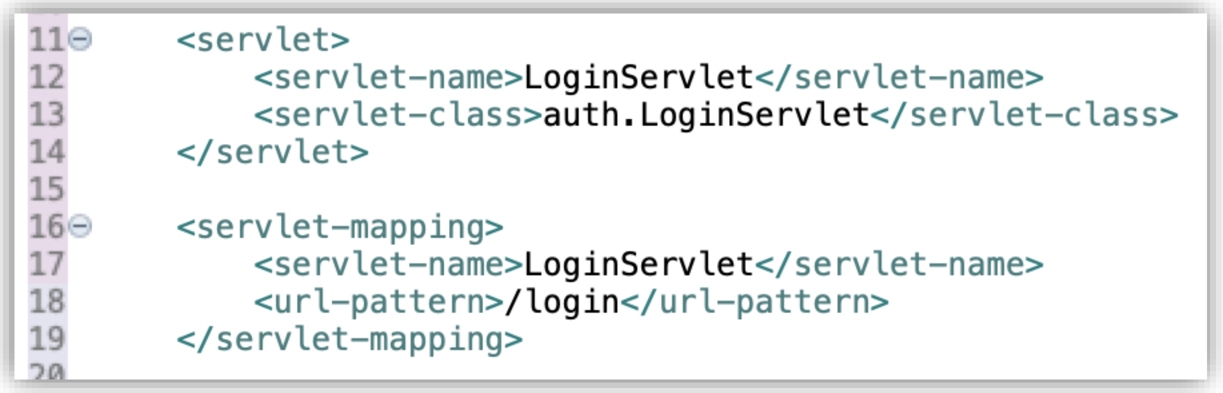
Note: if you change anything in the project make sure that you clear and re build the project.

Note: Each servlet in an application has a unique id, and tomcat causes problem with two servlets having same id.

```
private static final long serialVersionUID = 1L;
```

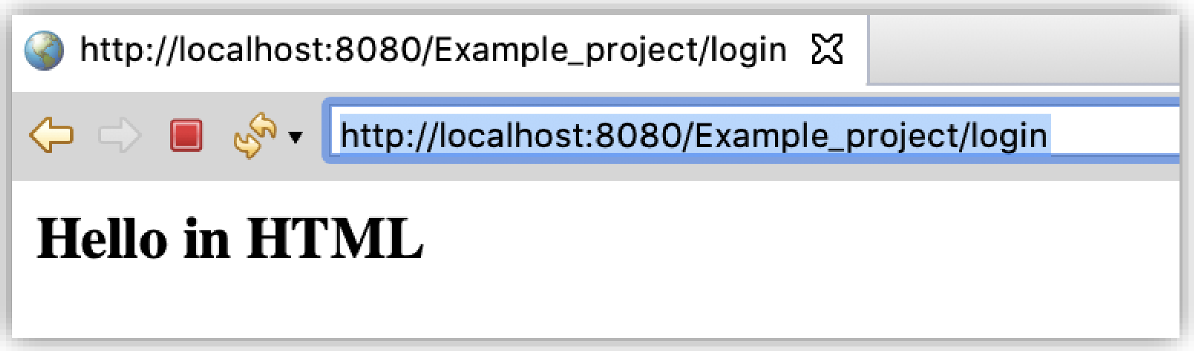
2- Defining a Servlet in the configuration file web.xml

- In `LoginServlet.java` class, remove the line `@WebServlet`.
- Define the Servlet in the web configuration file by adding the following lines of code to the `web.xml` file in the `WebContent/WEB-INF` folder.



```
    <servlet>
        <servlet-name>LoginServlet</servlet-name>
        <servlet-class>auth.LoginServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>LoginServlet</servlet-name>
        <url-pattern>/login</url-pattern>
    </servlet-mapping>
```

- Run and deploy the project:



http://localhost:8080/Example_project/login

http://localhost:8080/Example_project/login

Hello in HTML

Note: you should only use one of these methods to define the servlet.

Part 2: Java Server Pages - JSPs

JSP is an alternative approach for generating dynamic web page. In JSP Java code for generating dynamic content is embedded in the JSP page within appropriate tags. JSPs are compiled into Java servlets on the server-side i.e. on Tomcat .

Passing parameters to a Servlet

There are two different ways to pass parameters (arguments) to a Servlet:
(1) Using doGet() method; and
(2) Using doPost() method.

We will learn how to pass a parameter using both these techniques.

1- Using doGet() method

Parameters passed as URL arguments in the form of:

```
?[parameterName1]=[parameterValue1]&[parameterName2]=[parameterValue2]
```

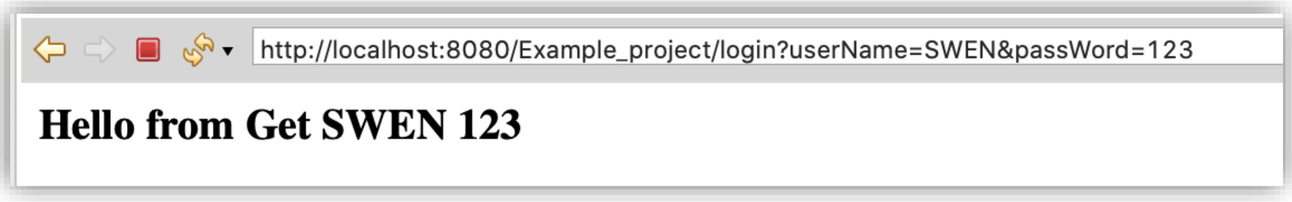
Appended to the URL can be retried using in the **doGet**() method of the Servlet.

- Use `LoginServlet.java`.
- Add the necessary code to the **doGet**() method to retrieve parameters `userName` and `passWord` passed as URL arguments, and output the following html.
- `userName` and `passWord` are the name and the password of the user passed as URL arguments to the Servlet.

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

            response.setContentType("text/html");
            System.out.println("Hello from GET method in LoginServlet");
            String user = request.getParameter("userName");
            String pass = request.getParameter("passWord");

            PrintWriter writer = response.getWriter();
            writer.println("<h3> Hello from Get "+user+  "   " +pass+ "</h3>");


    }
```

- Run the Servlet with `userName` and `passWord` as URL parameters to test your code.

```
http://localhost:8080/Example_project/login?userName=SWEN&passWord=123
```



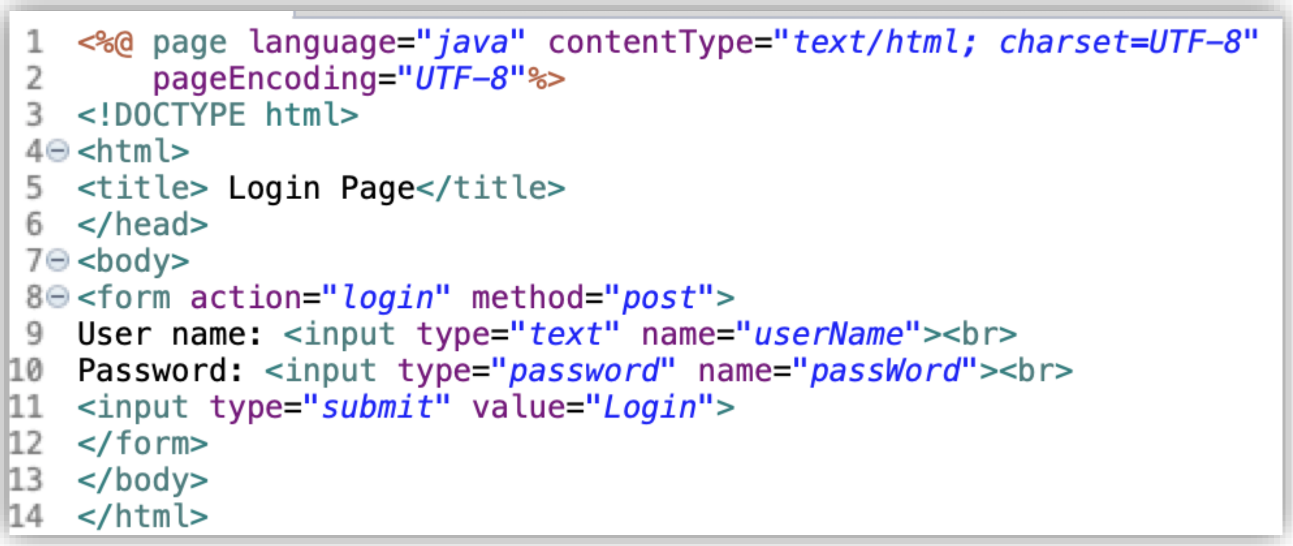
http://localhost:8080/Example_project/login?userName=SWEN&passWord=123

Hello from Get SWEN 123

2- Using doPost() method

Parameters passed using the **post** method in a web form is retrieved using the **doPost**() method.

- Create the `LoginForm.jsp` file in the `WebContent` directory to pass parameters using the post method.



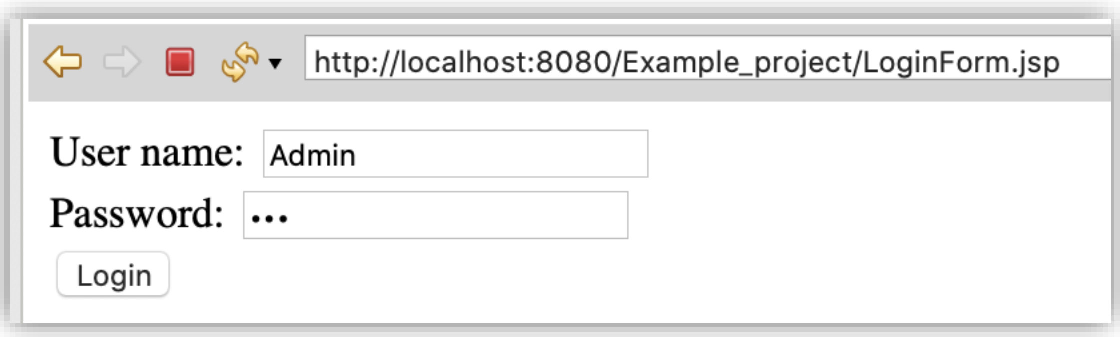
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<title> Login Page</title>
</head>
<body>
<form action="login" method="post">
User name: <input type="text" name="userName"><br>
Password: <input type="password" name="passWord"><br>
<input type="submit" value="Login">
</form>
</body>
</html>
```

- Add the follwing code to the **doPost()** method in the `LoginServlet.java` class to retrieve the parameters and print the following:

  "Hello from Post: Your user name is: [userName], Your password is: [passWord]."

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType("text/html");
        System.out.println("Hello from Post method in LoginServlet");
        String user = request.getParameter("userName");
        String pass = request.getParameter("passWord");

        PrintWriter writer = response.getWriter();
        writer.println("<h3> Hello from Post: Your user name is: "+user+", Your password is: " +pass+
"</h3>");


    }
```

- Access `http://localhost:[APACHE_TOMCAT_PORT]/[ProjectName]/[FormName].`

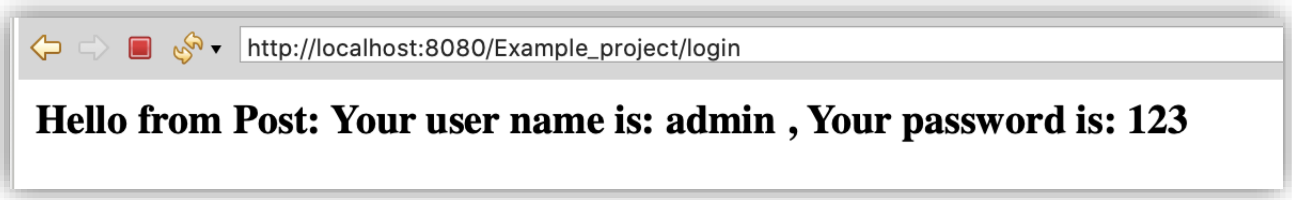

http://localhost:8080/Example_project/LoginForm.jsp

User name: Admin

Password: •••

Login

- When the submit button is pressed the doPost() method of the Servlet gets executed.


http://localhost:8080/Example_project/login

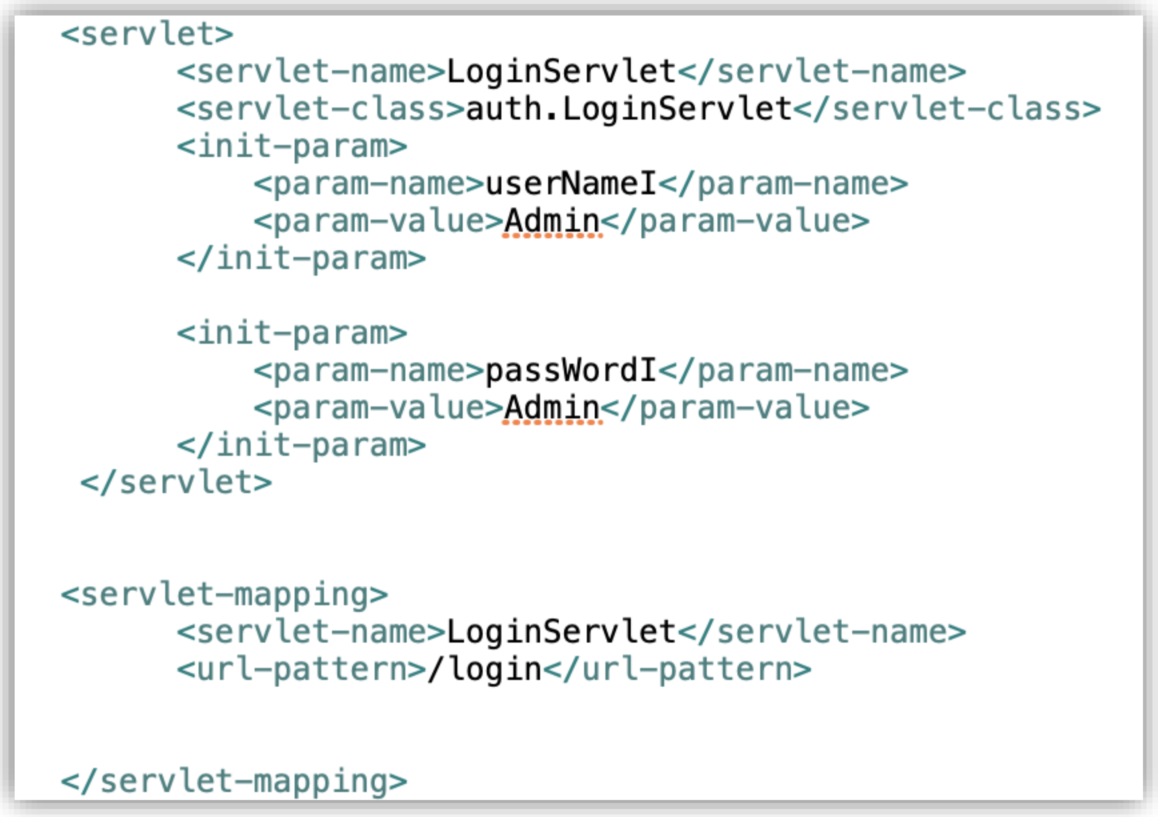
Hello from Post: Your user name is: admin , Your password is: 123

Discussion:
• What is the difference between doGet() and doPost()?

3- Initializing Servlets

There are use cases where Servlets need to get initialized at the time of creation and common behaviour being executed before the user specific actions are performed. The **init()** method gets executed at the time a Servlet object gets created. The **service**() method gets invoked every time the Servlet object gets accessed - before the **doGet**() or **doPost**() methods get executed.

- To define and initialize an initialization parameter for `LoginServlet` servlet, modify `web.xml`



```xml
<servlet>
      <servlet-name>LoginServlet</servlet-name>
      <servlet-class>auth.LoginServlet</servlet-class>
      <init-param>
          <param-name>userNameI</param-name>
          <param-value>Admin</param-value>
      </init-param>

      <init-param>
          <param-name>passWordI</param-name>
          <param-value>Admin</param-value>
      </init-param>
 </servlet>


<servlet-mapping>
      <servlet-name>LoginServlet</servlet-name>
      <url-pattern>/login</url-pattern>


</servlet-mapping>
```

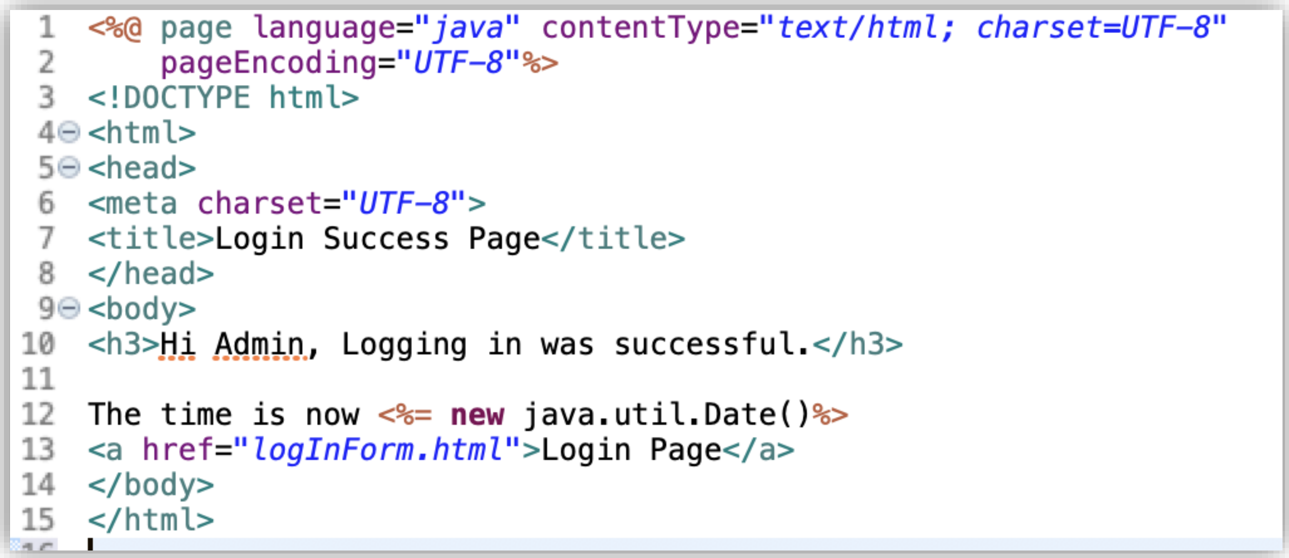
- Use `getServletConfig().getInitParameter()` method to retrieve the values of default `userName` and `passWord` defined in the `LoginServlet` Servlet.

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType("text/html");
        System.out.println("Hello from Post method in LoginServlet");
        String user = request.getParameter("userName");
        String pass = request.getParameter("passWord");

        String correctUser = getServletConfig().getInitParameter("userNameI");
        String correctPass = getServletConfig().getInitParameter("passWordI");
        PrintWriter writer = response.getWriter();

        if(user.equals(correctUser) && pass.equals(correctPass)) {

            response.sendRedirect("success.jsp");
        }else {
            writer.println("<h3> Error </h3>");
        }
    }
```

- Then, modify the code of **doPost**() in previous activity, so the user will receive the hello message if the entered userName and passWord are exactly as the default values defined in the servlet. Otherwise, an error message will be printed.

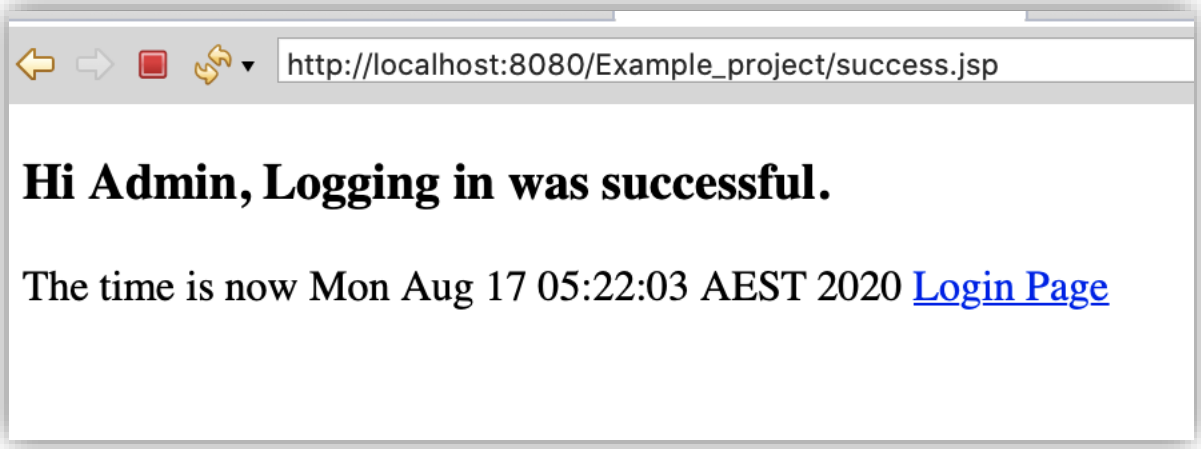
Discussion:
• Research on the **init**() and **service**() methods and understand their behaviour in the context of Servlets.

- Create `success.jsp` file and add the following.



```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Login Success Page</title>
</head>
<body>
<h3>Hi Admin, Logging in was successful.</h3>

The time is now <%= new java.util.Date()%>
<a href="logInForm.html">Login Page</a>
</body>
</html>
```

- Access `http://localhost:[APACHE_TOMCAT_PORT]/[ProjectName]/[FormName]`.
- When the submit button is pressed the doPost() method of the Servlet gets executed



http://localhost:8080/Example_project/success.jsp

Hi Admin, Logging in was successful.

The time is now Mon Aug 17 05:22:03 AEST 2020 Login Page