COMP 3800SEF/3820SEF/S380F/S380W Web Applications: Design and Development
Lab 2: Servlet

In this lab session, we will use ***IntelliJ IDEA Ultimate*** to create a simple web application containing a ***Servlet*** and deploy the web application to the web container ***Apache Tomcat 10.1***. Answers to Task 1 and Task 2 are available in the branches `lab02-task1` and `lab02-task2`, respectively, in the GitHub repository https://github.com/FloraZHANGJingyu/wadd2026.git.

Task 1: Coding Servlets with IntelliJ IDEA Ultimate & deploying to Apache Tomcat 10.1

A Servlet is a Java program that runs within an application server (i.e., Jakarta EE server) or a web container. Servlets receive and respond to requests from Web clients, usually across HTTP. In this task, we will use IntelliJ IDEA Ultimate to create a simple servlet called "myServlet". We will deploy the Servlet to the web container Apache Tomcat 10.1 and then use IntelliJ to debug the Servlet.

System requirements: Please refer to the "Software Installation Guide" on the OLE to install *BellSoft Liberica JDK 17*, *Apache Tomcat 10.1*, and *IntelliJ IDEA Ultimate*.

Your task: Follow the following steps to create a **Web Application** containing a **Servlet** "myServlet". Upon receiving an HTTP GET request, this servlet will return the string "Hello, World!" in the HTTP response.

1. In IntelliJ's welcome screen, create a Maven web app project with the following properties:
   - Generator: **Jakarta EE**
   - Name: **Hello-World**
   - Template: **Web application**
   - Application Server: **Tomcat 10.1**
   - Build system: **Maven**
   - Group: **hkmu.wadd**
   - Jakarta EE Version: **Jakarta EE 10**
2. In the "Project" panel, select **pom.xml**. (POM stands for "Project Object Model" and pom.xml is a one-stop-shop for all things concerning the Maven project.).

   In pom.xml, look for the XML tag ***<dependencies>***, which is the project's dependency list. Many projects depend upon other libraries to build and run correctly. Maven downloads and links the dependencies on the compilation, including the dependencies of those dependencies (transitive dependencies). This allows the dependency list to focus solely on the dependencies the project requires.

   Update the <dependencies> tag, as follows. Then, right-click on the project name and build the project to let Maven download the required libraries from the central repository.

```
<dependencies>
    <dependency>
        <groupId>jakarta.servlet</groupId>
        <artifactId>jakarta.servlet-api</artifactId>
        <version>6.0.0</version>
        <scope>provided</scope>
    </dependency>
</dependencies>
```

   Useful hotkey: Ctrl + Alt + L = Reformat file

3. In the "Project" panel, expand **src > main > java**. In the package "hkmu.wadd.helloworld", a template servlet "HelloServlet" has been created by IntelliJ.

4. Right-click on the directory "java", and select **New > Java Class** to create a Servlet with the fully qualified class name: **hkmu.wadd.MyServlet**

5. In the Servlet `MyServlet.java`, modify the class, as shown below.
   Useful hotkey & setting:
   - Ctrl + Shift + Enter = Add semi-colon ; to the end of line
   - Go to **File** > **Settings** > **Editor** > **General** > **Auto Import**. Check the two options "Add unambiguous imports on the fly" and "Optimize imports on the fly".

```
public class MyServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws IOException {
        response.getWriter().println("Hello, World!");
    }
}
```

6. In the deployment descriptor `/src/main/webapp/WEB-INF/web.xml`, inside the tag `<web-app>`, create the URL mappings for "myServlet", and add an ***<display-name>*** element (which will be shown in the Apache Tomcat manager interface), as follows:

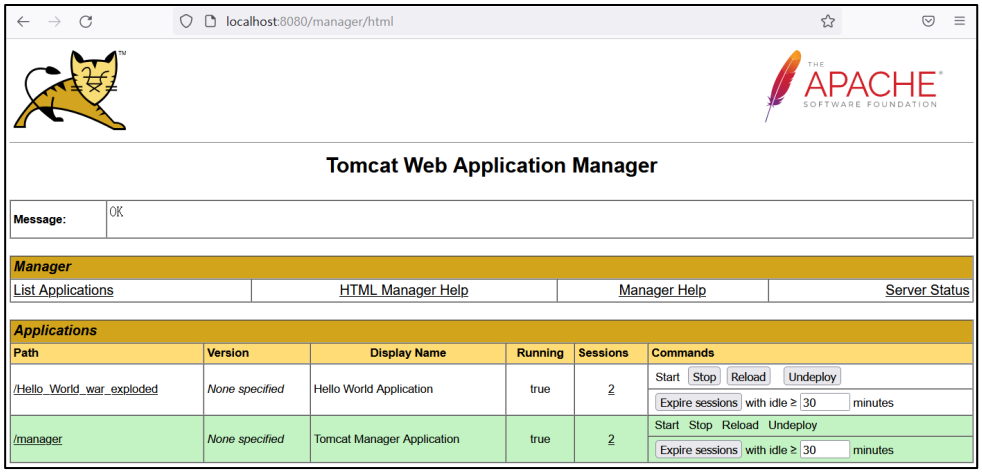
```
<display-name>Hello World Application</display-name>

<servlet>
    <servlet-name>myServlet</servlet-name>
    <servlet-class>hkmu.wadd.MyServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>myServlet</servlet-name>
    <url-pattern>/greeting</url-pattern>
    <url-pattern>/salutation</url-pattern>
    <url-pattern>/wazzup</url-pattern>
</servlet-mapping>
```

7. When we run the web application on Apache Tomcat, we say that the web application is ***deployed*** to Tomcat. Whenever you update some Java class in your web application, you need to ***build and redeploy*** the web application again. A web browser will be opened and display the default index page of the project at http://localhost:8080/Hello_World_war_exploded/. Note that this is ***not*** the output returned by any servlet.

8. Apache Tomcat provides a GUI for managing the deployed web applications at
   http://localhost:8080/manager/html
   To use it, you need to go to the **Tomcat directory** and edit the file `/conf/tomcat-users.xml` by adding the following `admin` account (e.g., with password `tcuser`) inside the tag `<tomcat-users>`:

```
<user username="admin" password="tcuser" roles="manager-gui"/>
```

You need to **restart** Tomcat to load the new configurations. This GUI is particularly useful when the execution of Tomcat is not managed by IntelliJ IDEA.



9. The path "Hello_World_war_exploded" is the **context root** of the web application. We can update the context root to "Hello-World", as follows:
   - Click on the name of the Apache Tomcat container on the top-right menu bar.
   - Select "Edit Configurations…".
   - In the tab "Deployment", change "Application context" to "/Hello-World".
   - **Redeploy** the project again.

10. To see the output from helloServlet, you can try any one of the following URLs:
    - http://localhost:8080/Hello-World/greeting
    - http://localhost:8080/Hello-World/salutation
    - http://localhost:8080/Hello-World/wazzup

11. To **understand the life cycle of a Servlet**, add the ***init()*** and ***destroy()*** methods, and run again.

```
@Override
public void init() {
    System.out.println("Servlet " + this.getServletName() + " has started.");
}

@Override
public void destroy() {
    System.out.println("Servlet " + this.getServletName() + " has stopped.");
}
```

To undeploy the web application:

1. In the "Services" panel on the bottom, expand **Tomcat 10.1.x**.
2. Right-click on "Hello-World:war exploded" and select "Undeploy".
3. You can click on the **Tomcat 10.1.x** to check the Tomcat's console output.

To debug the Servlet, follow the following steps:

1. Create breakpoints: In `MyServlet.java`, click on the right-hand side of the line number of each of the Java statements inside the three functions doGet(), init() and destroy() to create breakpoints (with a red circle); alternatively, you can move the cursor to the line and use the hotkey Ctrl + F8.

2. Debug the project  (Shift + F9) and perform the following steps. Notice the output and how the debugger interrupts the execution of myServlet (You can press F9 to resume the execution).
   i. Browse http://localhost:8080/Hello-World/greeting .
   ii. Browse http://localhost:8080/Hello-World/salutation .
   iii. Browse http://localhost:8080/Hello-World/wazzup .
   iv. Undeploy the web application.

Task 2 *(Take-home)*: Replacing Maven with Gradle

Gradle is another build automation tool, which supports multiple programming languages, e.g., Java, Kotlin, Groovy, Scala, C/C++, and JavaScript. When the number of dependencies in a Maven project increases, the `pom.xml` will become very bulky. On the contrary, the Gradle build configuration script `build.gradle` replaces the POM file in Maven and is very succinct. Let's redo Task 1 with Gradle:

- Generator: *Jakarta EE*
- Name: **Hello-World2**
- Template: *Web application*
- Application Server: *Tomcat 10.1*
- Build system: **Gradle**
- Group: *hkmu.wadd*
- Jakarta EE Version: *Jakarta EE 10*

Task 3 (*Take-home*): Cloning a project from GitHub repository

To get the lab answers from GitHub, you need to download the project of our course GitHub repository https://github.com/FloraZHANGJingyu/wadd2026.git, as follows:

1. On the main page of IntelliJ, click "Clone Repository" with the following properties:
   - Version control: **Git**
   - URL: https://github.com/FloraZHANGJingyu/wadd2026.git.
2. After opening the cloned project, click on the menu bar  : Git > Branches… . Select the suitable **remote branch**, e.g., "origin/lab02-task2" > Checkout, to check out the Lab 2 answer project.
3. You can update the project name and module name to "**Hello-World**" by clicking on the menu bar: File > Project Structure, and set up the new name in "**Project**" and "**Modules**".
4. Close and re-open the project.
5. The downloaded project does not have the Tomcat setting. We add it, as follows:
   - Click on the name of the "Current File" on the top-right menu bar.
   - Select "Edit Configurations…".
   - Click the **"+"** button (or **Add New**) and add **Tomcat Server > local** .
   - A warning "No artifacts marked for deployment" will be displayed at the below of the Tomcat 10.1.x page. Click its "**Fix**" button, and select an artifact, e.g., "**Gradle: … (exploded)**".
   - Update "Application context" to "/Hello-World".
   - **Redeploy** the project again.