

Lab1: Code Explanation

This document gives step-by-step guide to finish Lab1.

Lab1 covers:

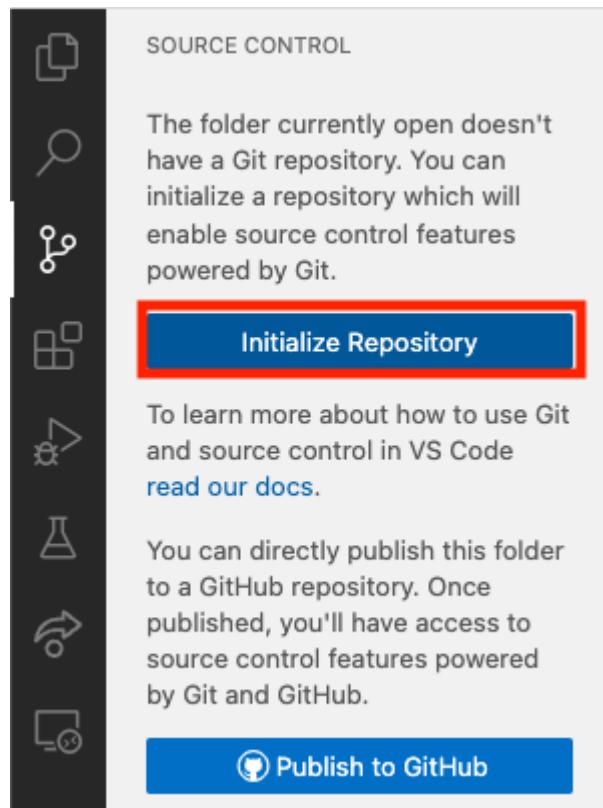
- Download recommended code assets.
- Explore the example `modresorts` application
- Explain `modresorts` application.

1. Code Asset Download

Download and unzip the `modresorts-twaz-j8.zip` file from the [box shared folder](#).

Open the `modresorts-twaz-j8` folder in VScode.

Note: We recommend to '*Initialize Repository*' to enable source control features and keep track of your changes. Commit all the new changes but you don't need to publish the branch.



2. Build Application Project

- Open a terminal, and go to your project folder, and navigate to `was_dependency` folder.

```
cd <your-path>/modresorts-twaz-j8/was_dependency
```

- Once inside the folder, run the following command to build project:

```
mvn install:install-file -Dfile=was_public.jar -DpomFile=was_public-9.0.0.pom
```

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
zsh - was_dependency + v □ ... ^ ×
nanshi1@Nans-MBP ~ ~/Documents/1_Clients_and_Projects_APAC/HK_WCA_Incubation_Nov/Lab cd modresorts-twaz-j8/was_dependency
mvn install:install-file -Dfile=was_public.jar -DpomFile=was_public-9.0.0.pom
[INFO] Scanning for projects...
[INFO]
[INFO] -----< org.apache.maven:standalone-pom >-----
[INFO] Building Maven Stub Project (No POM) 1
[INFO] [ pom ]
[INFO]
[INFO] --- install:3.1.2:install-file (default-cli) @ standalone-pom ---
[INFO] Installing /Users/nanshi1/Documents/1_Clients_and_Projects_APAC/HK_WCA_Incubation_Nov/Lab/modresorts-twaz-j8/was_dependency/was_public.jar to /Users/nanshi1/.m2/repository/com/ibm/websphere/appserver/was_public/9.0.0/was_public-9.0.0.jar
[INFO] Installing /Users/nanshi1/Documents/1_Clients_and_Projects_APAC/HK_WCA_Incubation_Nov/Lab/modresorts-twaz-j8/was_dependency/was_public-9.0.0.pom to /Users/nanshi1/.m2/repository/com/ibm/websphere/appserver/was_public/9.0.0/was_public-9.0.0.pom
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time:  0.185 s
[INFO] Finished at: 2024-11-13T11:10:01+08:00
[INFO]

```

For Windows users, you might need to give full path for the build files `was_public.jar` and `was_public-9.0.0.pom`.

```

mvn install:install-file -Dfile=C:\Users\Administrator.
CE-APAC-WIND202\Documents\wca4ej-workshop\modresorts-twaz-j8\was_dependency\was_public.jar
-DpomFile=C:\Users\Administrator.
CE-APAC-WIND202\Documents\wca4ej-workshop\modresorts-twaz-j8\was_dependency\was_public-9.0.0.pom

```

3. View Liberty App

After you installed LibertyTools from VSCode marketplace, there should be a Liberty Dashboard section in your explorer. Click `Add project to Liberty Dashboard` and put the path to the `modresort-twaz-j8` folder (this would be automatic if you open in this project).

The screenshot shows a sidebar with three items: OUTLINE, TIMELINE, and LIBERTY DASHBOARD. The LIBERTY DASHBOARD item is highlighted with a red border. Below the sidebar, there are two numbered steps:

1. If no projects are open in the Explorer, open or create a Liberty project using the File menu.
Open Project
2. If one or more existing Maven or Gradle projects are open in the Explorer, try one of the following actions:
 - a. Configure the Liberty build plugin in the build file of an existing [Maven](#) or [Gradle](#) project.
 - b. Add a `server.xml` file to an existing Maven or Gradle project at "src/main/liberty/config".
 - c. Manually add an existing Maven or Gradle project to the Liberty Dashboard through the Command Palette action "Liberty: Add project to Liberty Dashboard".

A red box highlights the "Add project to Liberty Dashboard" button.

Once you selected the right project, a `modresrots` app will show up. Right click on the app to start.

The screenshot shows the VSCode interface with a context menu open over a project named "modresrots". The menu includes the following options:

- Start
- Start...
- Attach debugger
- Stop
- Run tests
- View integration test report
- View unit test report

VSCode will go through downloading required packages, which you can see in terminal.

PROBLEMS 28 OUTPUT DEBUG CONSOLE TERMINAL PORTS

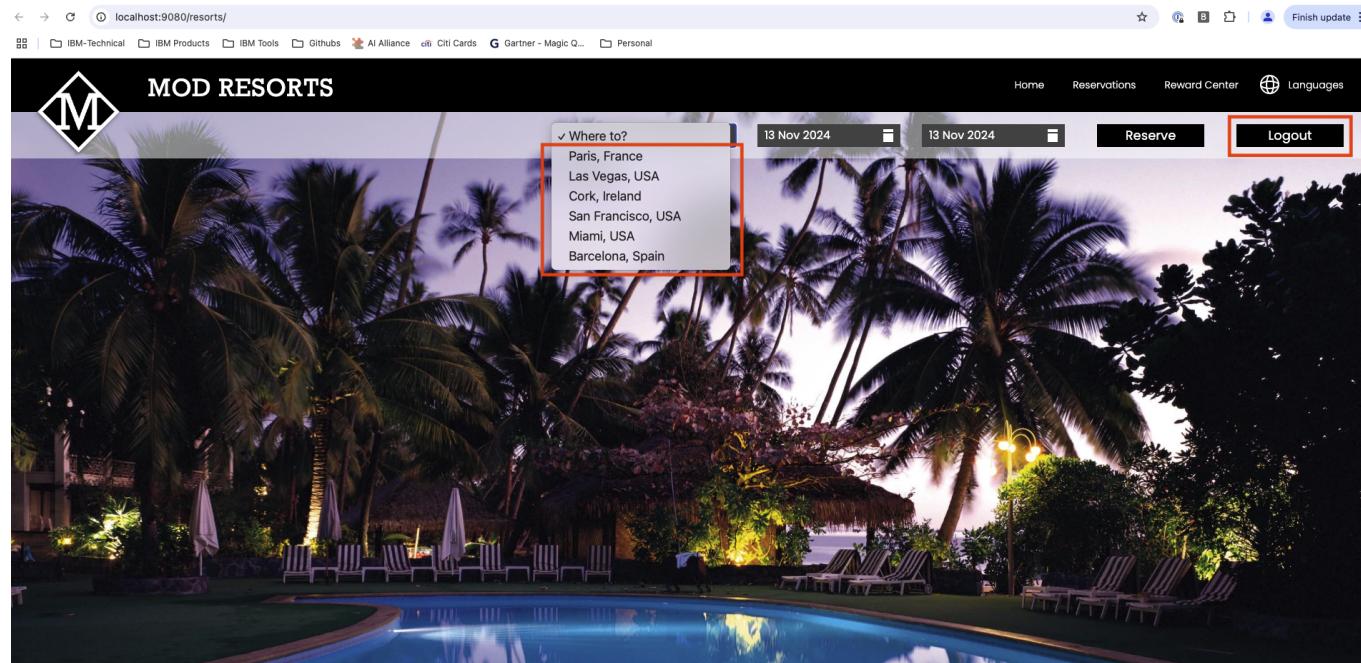
```
[INFO] ****
[INFO] * Liberty is running in dev mode.
[INFO] *           Automatic generation of features: [ Off ]
[INFO] *           h - see the help menu for available actions, type 'h' and press Enter.
[INFO] *           q - stop the server and quit dev mode, press Ctrl-C or type 'q' and press Enter.
[INFO] *
[INFO] *   Liberty server port information:
[INFO] *       Liberty server HTTP port: [ 9080 ]
[INFO] *       Liberty debug port: [ 7777 ]
[INFO] ****
```

Note: Some input files use or override a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Note: Some input files use or override a deprecated API that is marked for removal.
Note: Recompile with -Xlint:removal for details.
Note: /Users/nanshi1/Documents/1_Clients_and_Projects_APAC/HK_WCA_Incubation_Nov/Lab/modresorts-twars-j8/src/main/java/com/acme/modres/WeatherServlet.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

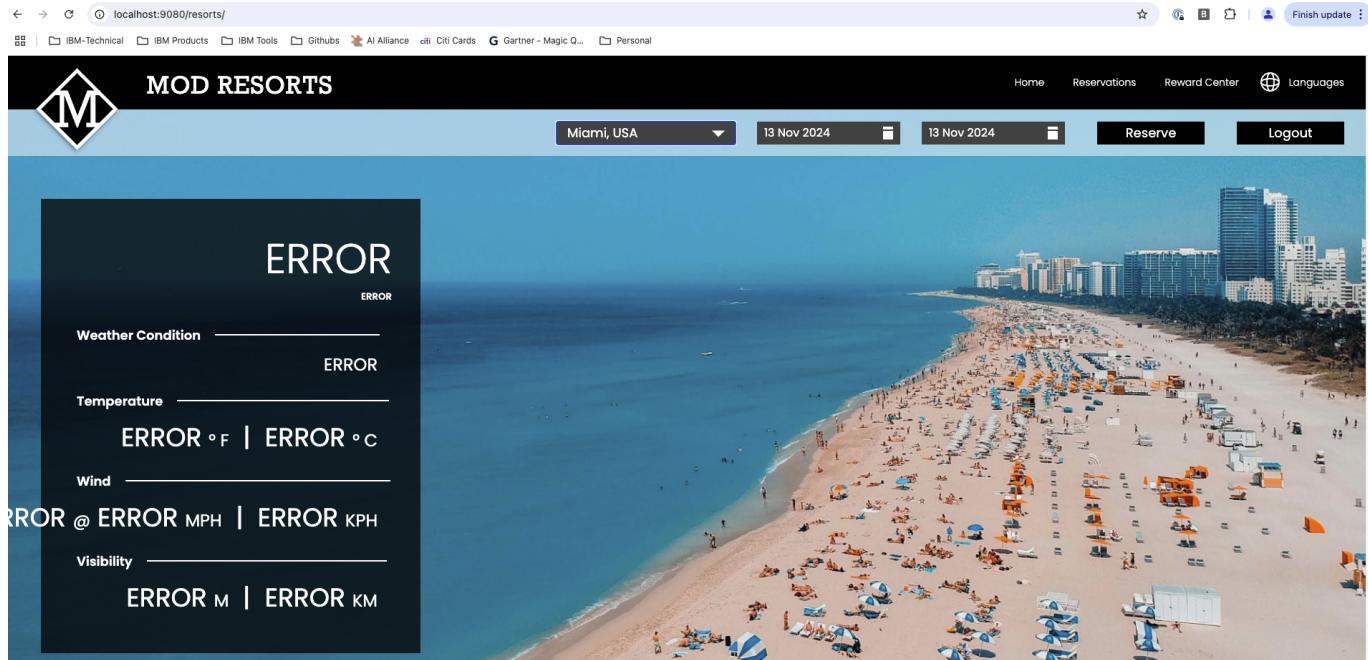
```
[INFO] Source compilation was successful.
[INFO] [AUDIT  ] CWWKT0017I: Web application removed (default_host): http://localhost:9080/resorts/
[INFO] [AUDIT  ] CWWKZ0009I: The application modresorts-2.0.0 has stopped successfully.
[INFO] [AUDIT  ] CWWKT0016I: Web application available (default_host): http://localhost:9080/resorts/
[INFO] [AUDIT  ] CWWKZ0003I: The application modresorts-2.0.0 updated in 0.062 seconds.
```

Once the app started, you can get the url (in my case <http://localhost:9080/resorts/>) and open in your browser to checkout the web app.

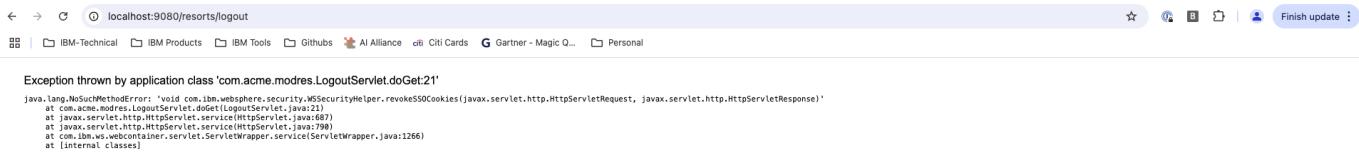
[IMPORTANT] Because we are running this application using Liberty in Java21, while this application is built for WebSphere in Java8, even though the application started successfully, **there are 2 places that have error because of this migration + upgrade.**



The first one, if you click the **Where to?** dropdown and select any location, you will find the location information module showing errors.



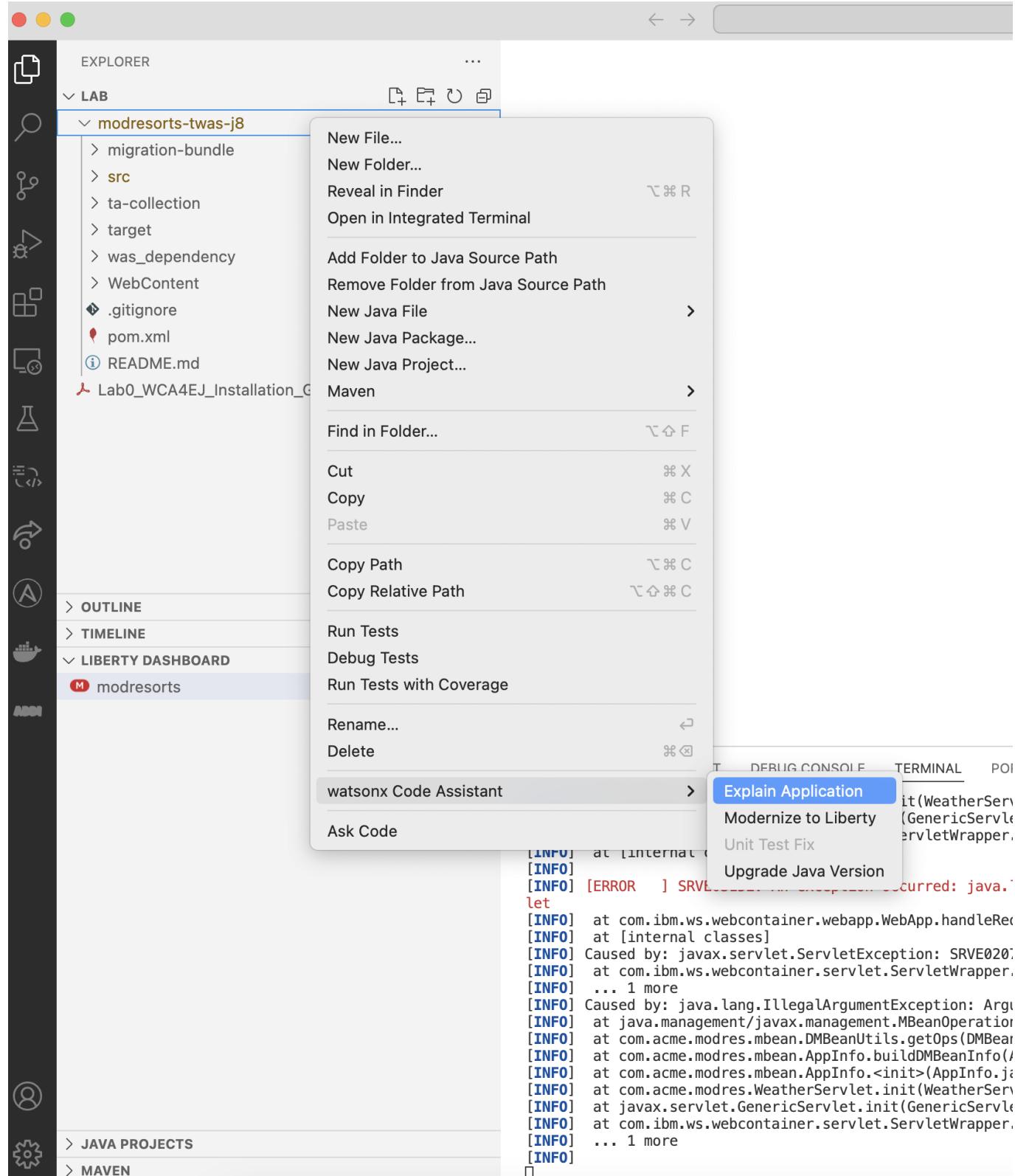
The Second one, the **Logout** button does not work if you click on it.



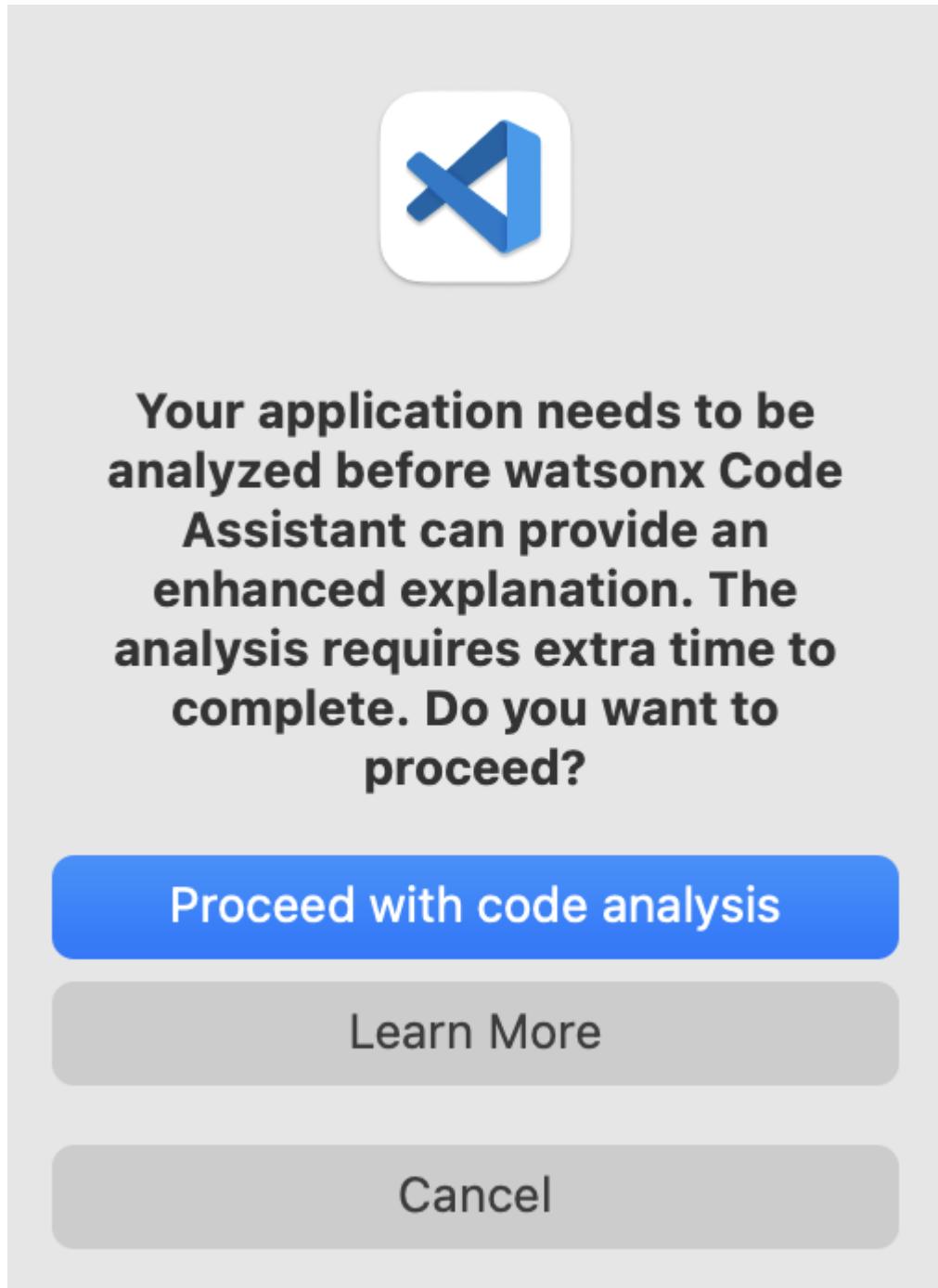
We will **fix these errors** in the later labs.

4. Explain Project Code

To understand the whole project, right click on the **modresorts-twaz-j8** folder and select **watsonx Code Assistant - Explain Application**.



VSCode will prompt you that the process takes extra time. Click [Proceed with code analysis](#).



The analysis might take 1-2 minutes to finish and a prompt will show up in the bottom right corner.

A screenshot of the VS Code terminal window. It shows the following log output:

```
PROBLEMS 28 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Java Applications activated
2024-11-13 11:30:46 - [wca] - [INFO]: - Authenticating...
2024-11-13 11:32:28 - [wca4ej] - [INFO]: - Using the Java developer kit that is defined in the JAVA_HOME environment variable to run watsonx Code Assistant components. The path is: /Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home
java version "21.0.4" 2024-07-16 LTS
Java(TM) SE Runtime Environment (build 21.0.4+8-LTS-274)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.4+8-LTS-274)
```

A red box highlights a message in the terminal: "The explanation of Lab has been completed." with a small circular icon. Below it, a smaller box says "Source: watsonx Code Assistant for Enterprise Java..." and "Open explanation".

Now we can open the report and read through the details.

The screenshot shows the Watsonx Code Assistant interface. At the top, there's a header bar with a logo, the title "Application Explanation - Lab", and a close button. Below the header is a section titled "Application explanation - Lab" with a note about the explanation being generated by Watsonx Code Assistant. A "Application: Lab" section follows. The main content area is titled "Executive summary of important application functionalities". It describes a Java application with various services and their functionality. Below this is a "Summary of service with entry method: doFilter in class: com.acme.modres.SecondFilter" section, which details how the filter intercepts and modifies requests and responses. The bottom part of the screenshot shows a terminal window titled "Java Applications activated" with log entries from November 13, 2024, at 11:30:46 and 11:32:28. These logs show the Java developer kit being used, the path to Java Virtual Machines, and the Java version.

This explanation was generated by Watsonx Code Assistant for Enterprise Java Applications on November 13, 2024 at 11:33:28.

Application: Lab

Executive summary of important application functionalities

Project is a Java application that provides a range of functionalities related to online reservation and management. The application has a homepage that welcomes users, an availability checker that allows users to check the availability of a reservation for a specific date, a weather service that provides real-time weather information for a given city, a logout servlet that logs users out of their WSO2 Identity Server sessions, and a filter that intercepts and modifies requests and responses before they are sent to the target resource.

After analysis we have found following services: 1. class: com.acme.modres.SecondFilter method: doFilter 2. class: com.acme.modres.FirstFilter method: doFilter 3. class: com.acme.modres.LogoutServlet method: doGet 4. class: com.acme.modres.AvailabilityCheckerServlet method: doGet 5. class: com.acme.modres.WelcomeServlet method: doGet 6. class: com.acme.modres.WeatherServlet method: doGet 7. class: com.acme.modres.UpperServlet method: doGet. We will summarize the functionality of each of them below.

Summary of service with entry method: doFilter in class: com.acme.modres.SecondFilter

The purpose of the SecondFilter app is to intercept and modify the request and response objects before they are sent to the target resource. The filter reads the request body and appends the string " to our site!" to it, before sending it on to the target resource. The filter also sets the content type of the response to text/plain. The input to the SecondFilter app is the servlet request and response objects, as well as the filter chain. The output is the modified request and response objects, as well as any other actions performed by the filter chain. The doFilter method utilizes the BufferedReader and PrintWriter classes to read and write the request body and set the response content type, respectively. The filter then appends the string " to our site!" to the request body and sends it on to the filter chain. Finally, the filter chain processes the request and response, and the modified request and response objects are returned to the client.

Summary of service with entry method: doFilter in class: com.acme.modres.FirstFilter

PROBLEMS 28 OUTPUT DEBUG CONSOLE TERMINAL PORTS WCA

```
Java Applications activated
2024-11-13 11:30:46 - [wca] - [INFO]: - Authenticating...
2024-11-13 11:32:28 - [wca4ej] - [INFO]: - Using the Java developer kit that is defined
in the JAVA_HOME environment variable to run Watsonx Code Assistant components. The
path is: /Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home
java version "21.0.4" 2024-07-16 LTS
Java(TM) SE Runtime Environment (build 21.0.4+8-LTS-274)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.4+8-LTS-274, mixed mode, sharing)
```