

Lab 1 [Explain App, Modernize Runtime and Explain Code Changes]

Lab 1 covers:

- Use Case 1 - [Explain Application](#)
- Use Case 2 - [Modernize app runtime \(traditional WebSphere to Liberty\)](#)
- Use Case 3 - [Explain Code Changes](#)

Git clone the project from the [GitHub repository](#)

Build Project

1. Open a terminal, and go to your project folder, and navigate to **was_dependency** folder.

```
cd wca4ej-workshop/modresorts-twas-j8/was_dependency
```

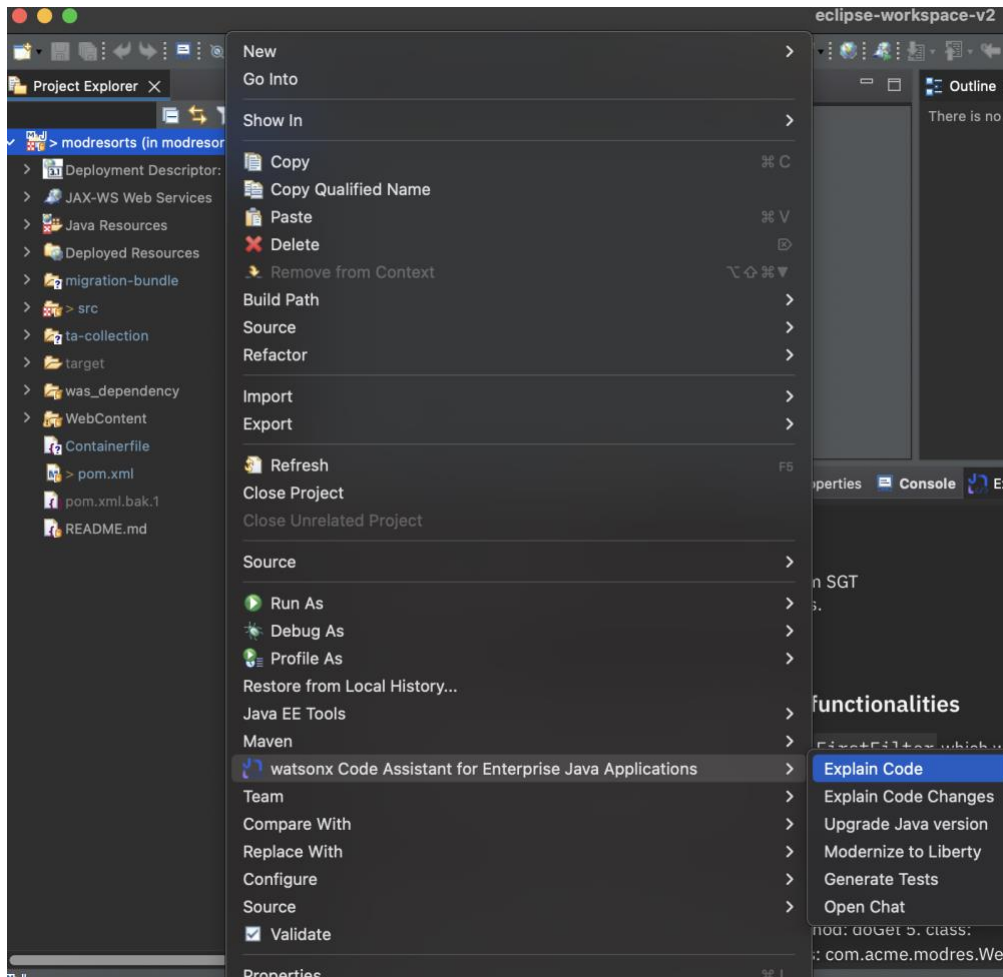
2. Once inside the folder, run the following command to build project:

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

USE CASE 1: Explain Application

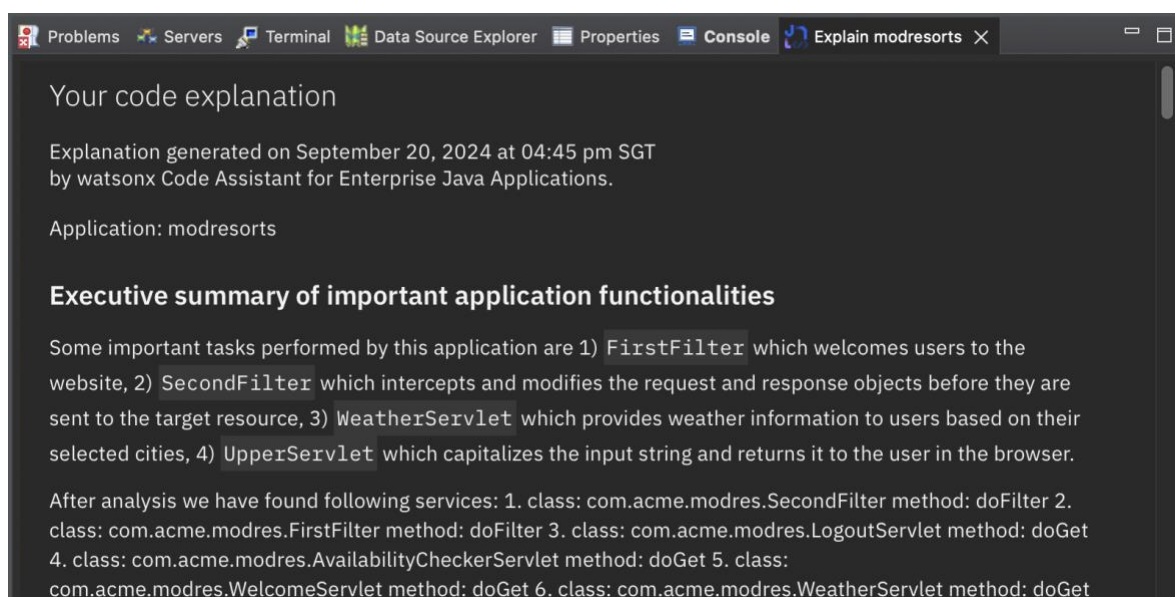
1. Import the project into Eclipse:
 - In the **Package explorer**, select **Import projects**, and select **Maven** followed by selecting **Existing Maven projects**
 - Browse to the project which you cloned from the git:
path_to_cloned_repo/wca4ej-workshop/modresorts-twas-j8
 - Click open and finish.
2. Once your IDE is setup, we will get an explanation of the entire project to understand what the app is doing:

Right Click on **modresorts** project -> select **watsonx Code Assistant for Enterprise Java Applications**, then select **Explain Code**



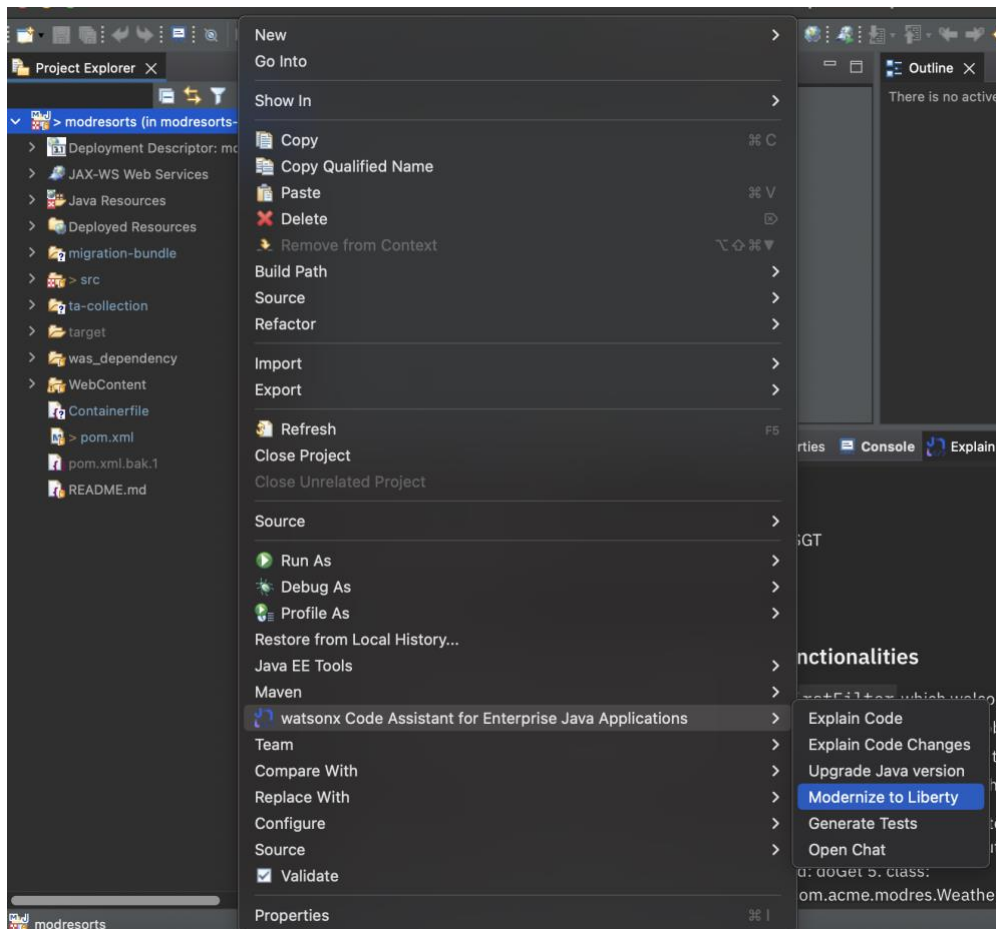
3. Once you click Explain Code, a new tab would open up and it will show the explanation of the entire application.

(Note: Getting explanation might take some time at the moment)

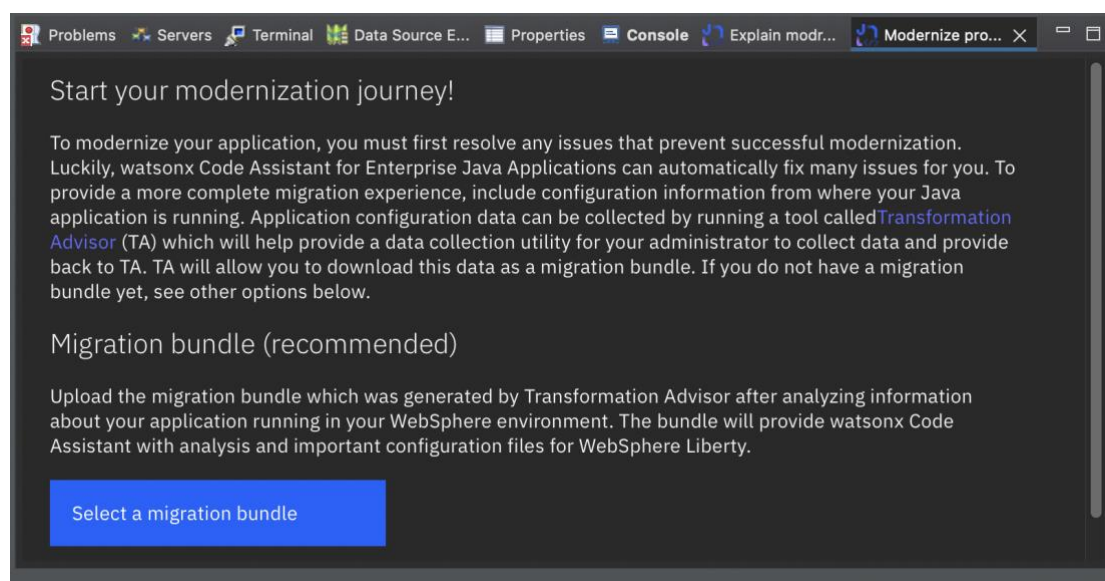


USE CASE 2: Modernize app runtime (traditional WebSphere to Liberty)

1. Right click on the project, select **watsonx Code Assistant for Enterprise Java Applications**, then select **Modernize to Liberty**:



2. After selecting modernize to liberty, following box should appear:



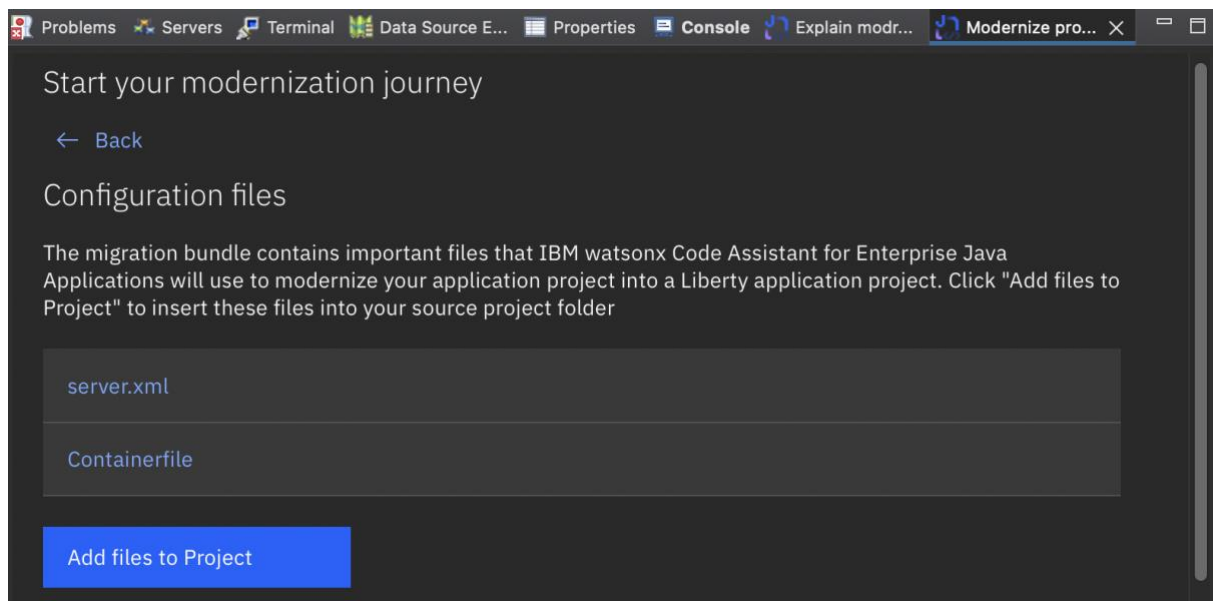
3. Click **Select a migration bundle**, and browse to your cloned project and select the file: **migration-bundle/modresorts.ear_migrationBundle.zip**

- That **migration bundle** has been generated by Transformation Advisor as a result of scanning the **ModResorts** application that was deployed to the traditional **WebSphere** Application Server environment.

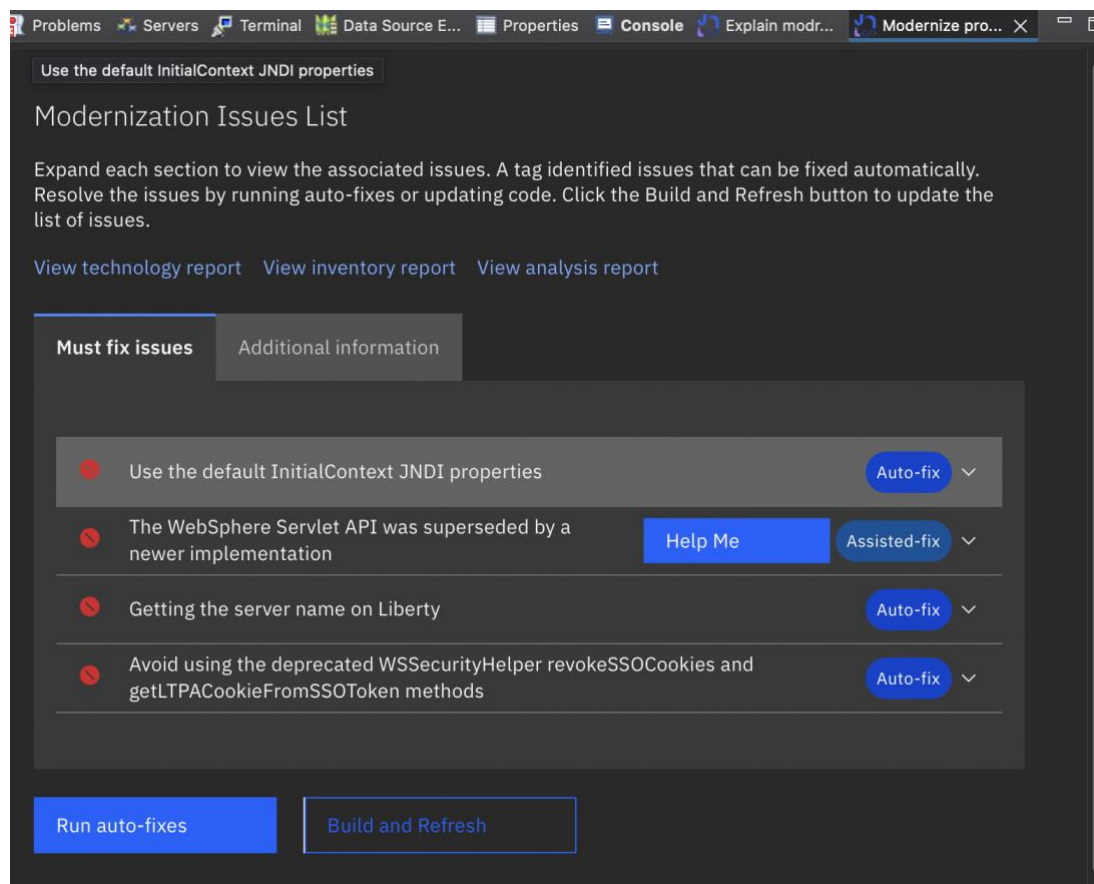
It contains configuration information for the application (the server.xml), and analysis that describes the issues that need to be addressed (the code changes that need to be made) in order for the application to run successfully on **Liberty**.

- The **server.xml** contains the configuration for the application. The **Containerfile** can be used to build a Liberty image of your application (outside the scope of this scenario)

4. You should see the following screen after choosing the migration bundle:

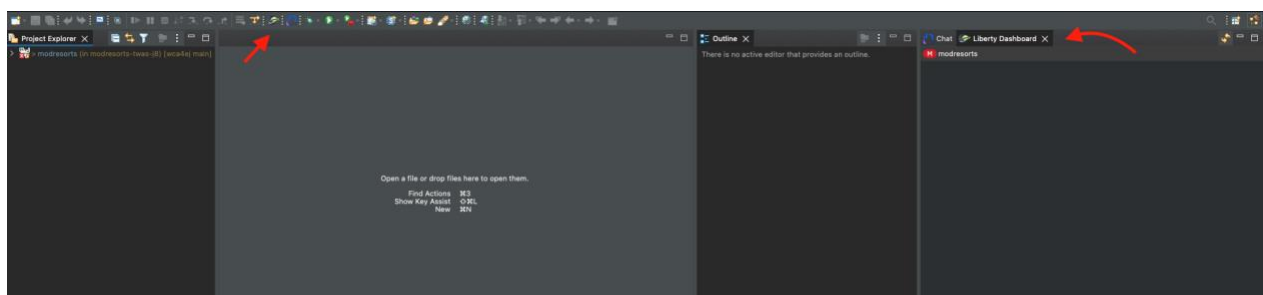


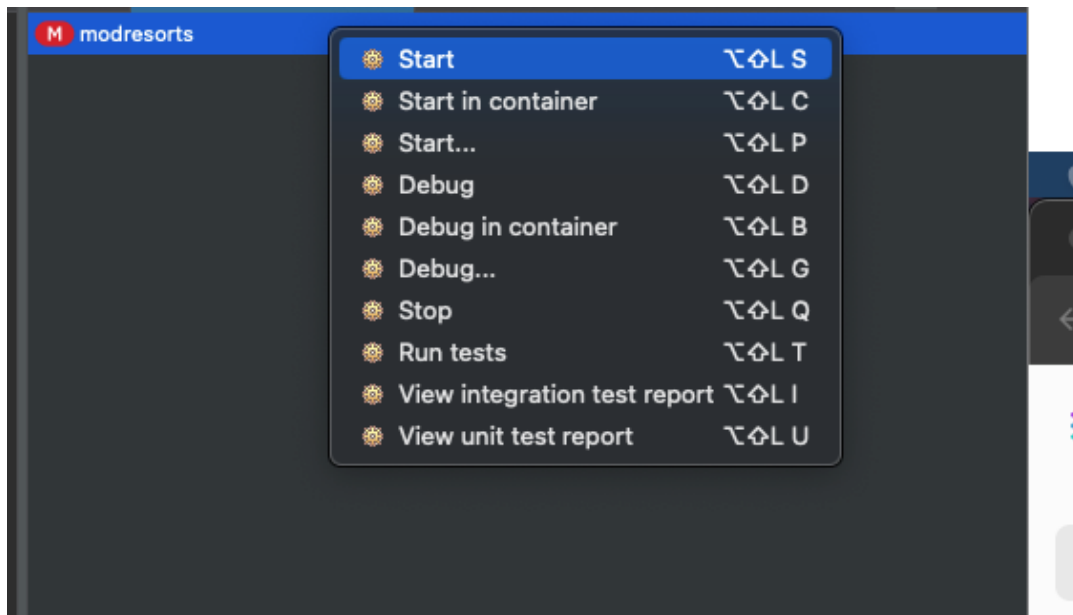
5. Click the button to **Add files to Project**. You should see the following panel:



You will see 4 issues that must be fixed before the application can run successfully in Liberty. Three of the issues have auto fixes, and one issue has an assisted fix, using the watsonx Code Assistant.

6. At this point BEFORE we actually fix the issues, it is interesting to take a look at the ModResorts application. Because we have the server.xml in place, the application can be run on Liberty, although we will expect that at least some functionality is broken because we have not yet addressed the issues.
- Launch **ModResorts** from the **Liberty Dashboard**:
 - NOTE: If you don't see **modresorts** in Liberty Dashboard. Click **Build and Refresh** under Modernize Project tab.





7. Go to <http://localhost:9080/resorts>

You should see the ModResorts UI.

- Let's take a look at a part of the functionality that is broken. Click the **"Logout"** button. You should see an error:

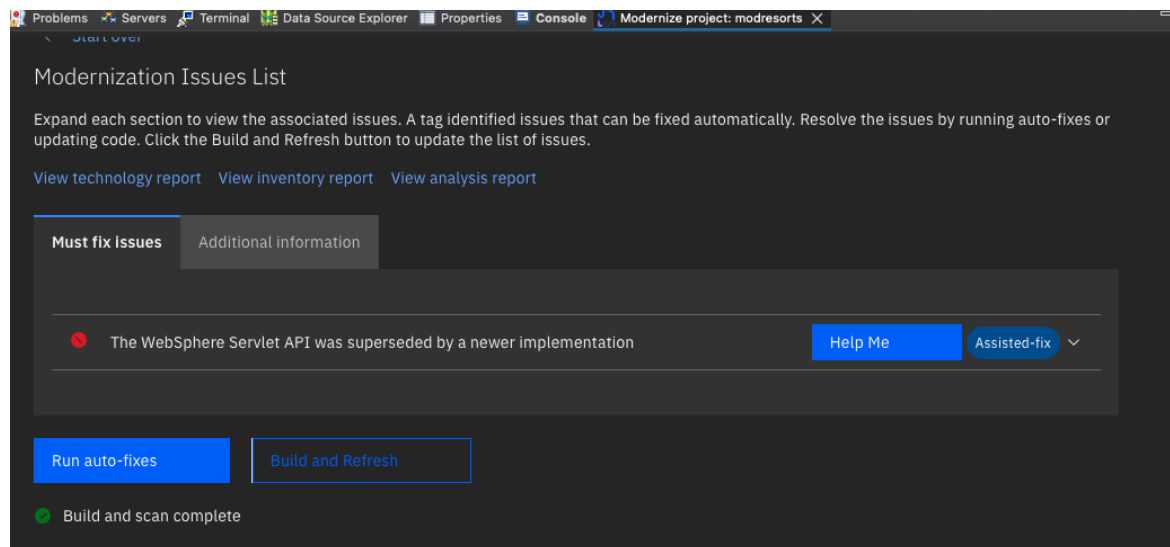
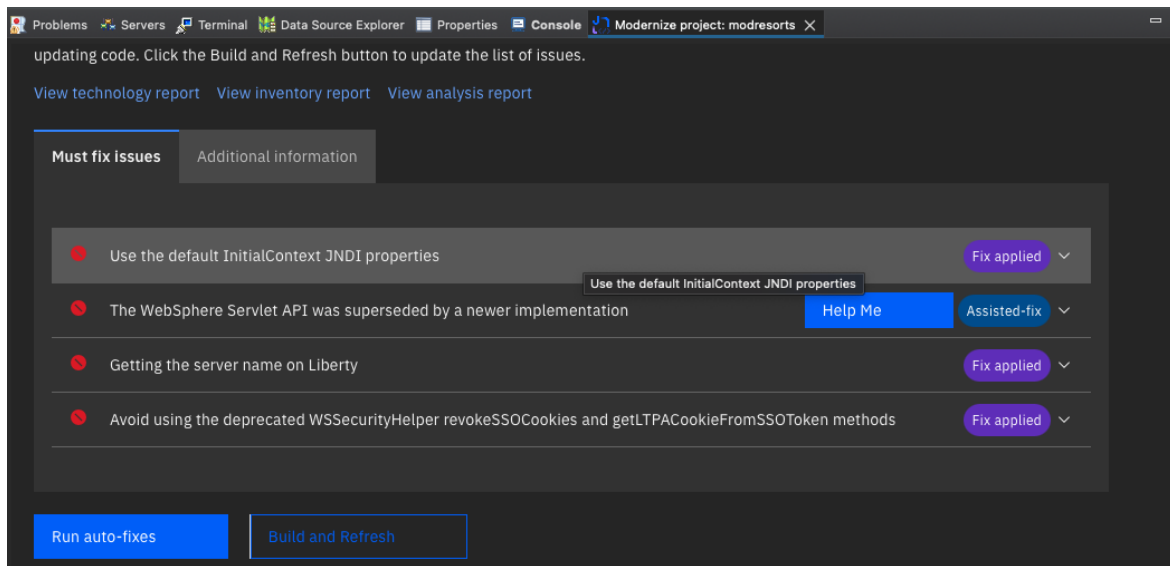
```
localhost:9080/resorts/logout

Exception thrown by application class 'com.acme.modres.LoginServlet.doGet:21'
java.lang.NoSuchMethodError: 'void com.ibm.websphere.security.WSSecurityHelper.revokeSSOCookies(javax.servlet.http.HttpServletRequest, javax.servlet.http.HttpServletResponse)'
    at com.acme.modres.LoginServlet.doGet(LoginServlet.java:21)
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:687)
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:790)
    at com.ibm.ws.webcontainer.servlet.ServletWrapper.service(ServletWrapper.java:1266)
    at [internal classes]
```

- We see this exception because the logout is using a **WebSphere API** that is not supported on **Liberty**. Now we will go back to the list of issues in eclipse to fix that along with other issues.

8. In Eclipse, in the Modernize project panel, click the **Run auto-fixes** button. They should complete in less than a minute.

9. Click the **Build and Refresh** button. You should observe that the 3 issues with auto fixes now go away from the list:



10. Now we will use **watsonx Code Assistant** to fix the remaining issue.

11. Due to limitation in private preview release, it is not easy to know which piece of code the issue relates to. To see this, you need to click on the **View analysis report** link.

Modernization Issues List

Expand each section to view the associated issues. A tag identified issues that can be fixed automatically. Resolve the issues by running auto-fixes or updating code. Click the Build and Refresh button to update the list of issues.

[View technology report](#) [View inventory report](#) [View analysis report](#)




12. In the report, find the issue and click on **Show results** to reveal the file, method and line for the issue.

Detailed Results by Rule

Expand all | Collapse all

Critical Rules

WebSphere traditional to Liberty

 The WebSphere Servlet API was superseded by a newer implementation (1)

Show rule help | Close results

Results

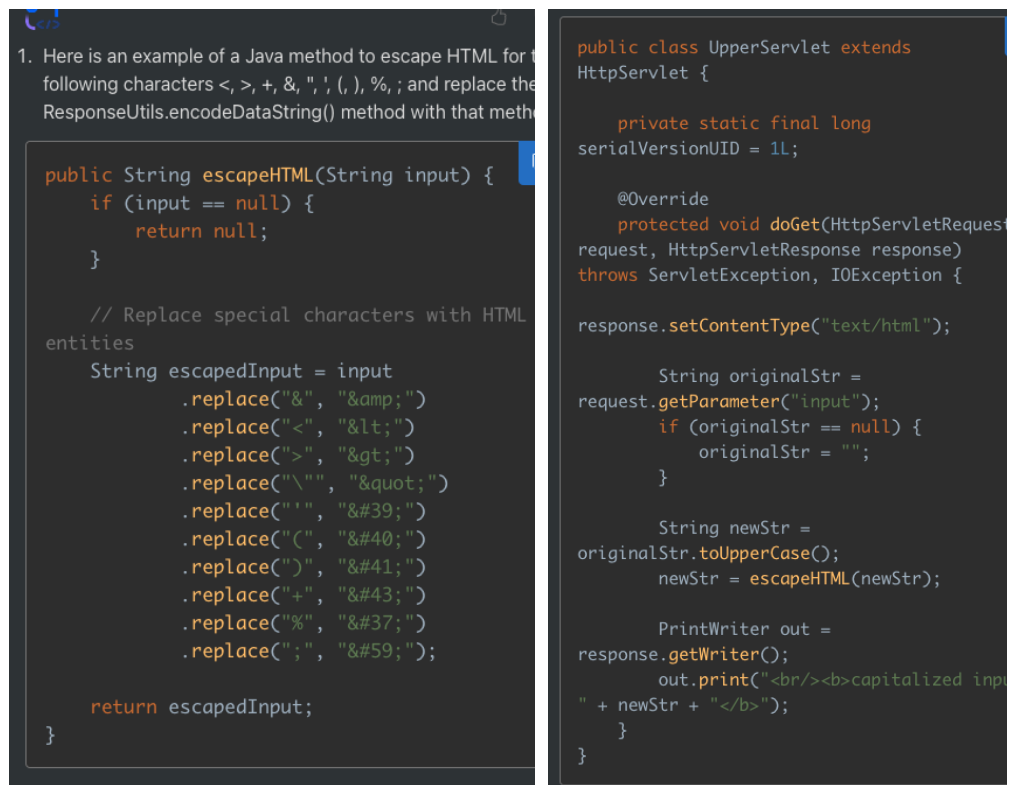
FILE NAME	REFERENCE DETAILS	MATCH CRITERIA	LINE NUMBER
modresorts-2.0.0.war			
WEB-INF/classes/com/acme/modres/UpperServlet.class	Method doGet	com.ibm.websphere.servlet.response.ResponseUtils	29

13. Open the **UpperServlet.java** file in the IDE.

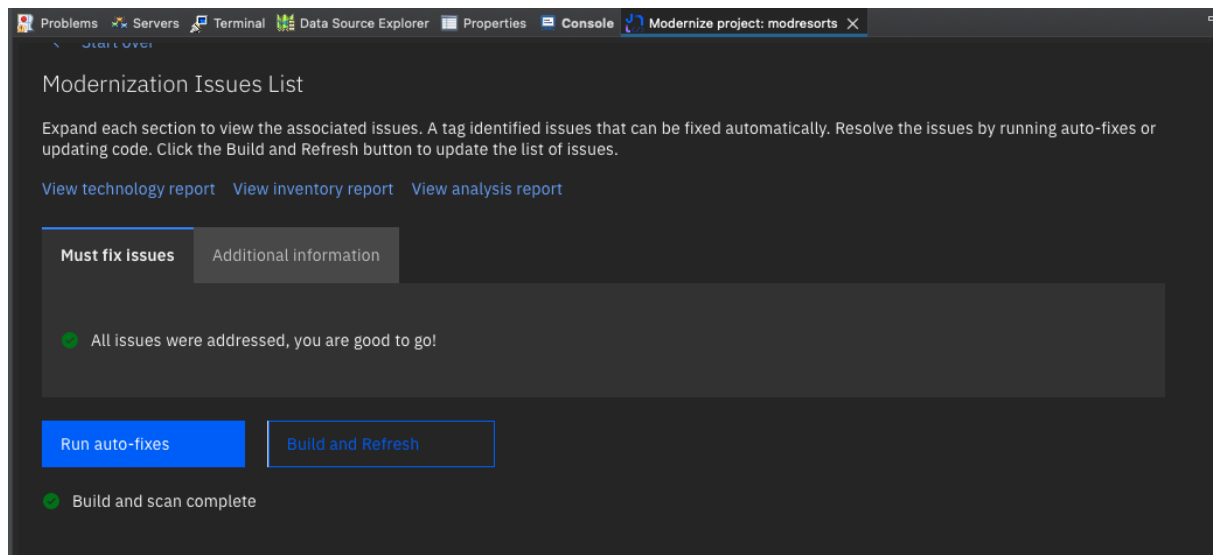
- Select the entire class (beginning at “public class” and ending at the final enclosing “}” and click the **Help me** button, in the Modernize project panel.

14. The IDE will automatically switch to the chat window, and you will see that watsonx Code Assistant is working on the solution. After a couple of moments, you should get a code snippet returned: (*code snippet shown below*)

- The code snippet should contain a method (probably called **escapeHTML**) and a separate snippet that show you how to replace the current call to the **ResponseUtils** utility with the new method.
- Add the new method to the class, and replace the call to **ResponseUtils** with a call to the new method as described.
- **NOTE:** The solution should also return an alternative solution, which involves using the **Apache Commons** Text library to solve the problem. You may review that as an alternative, but no need to take any action on that in this scenario.



15. After saving your work, go back to the **Modernize project** panel, and click **Build and refresh**. After a couple of moments, you should observe all the issues are fixed.



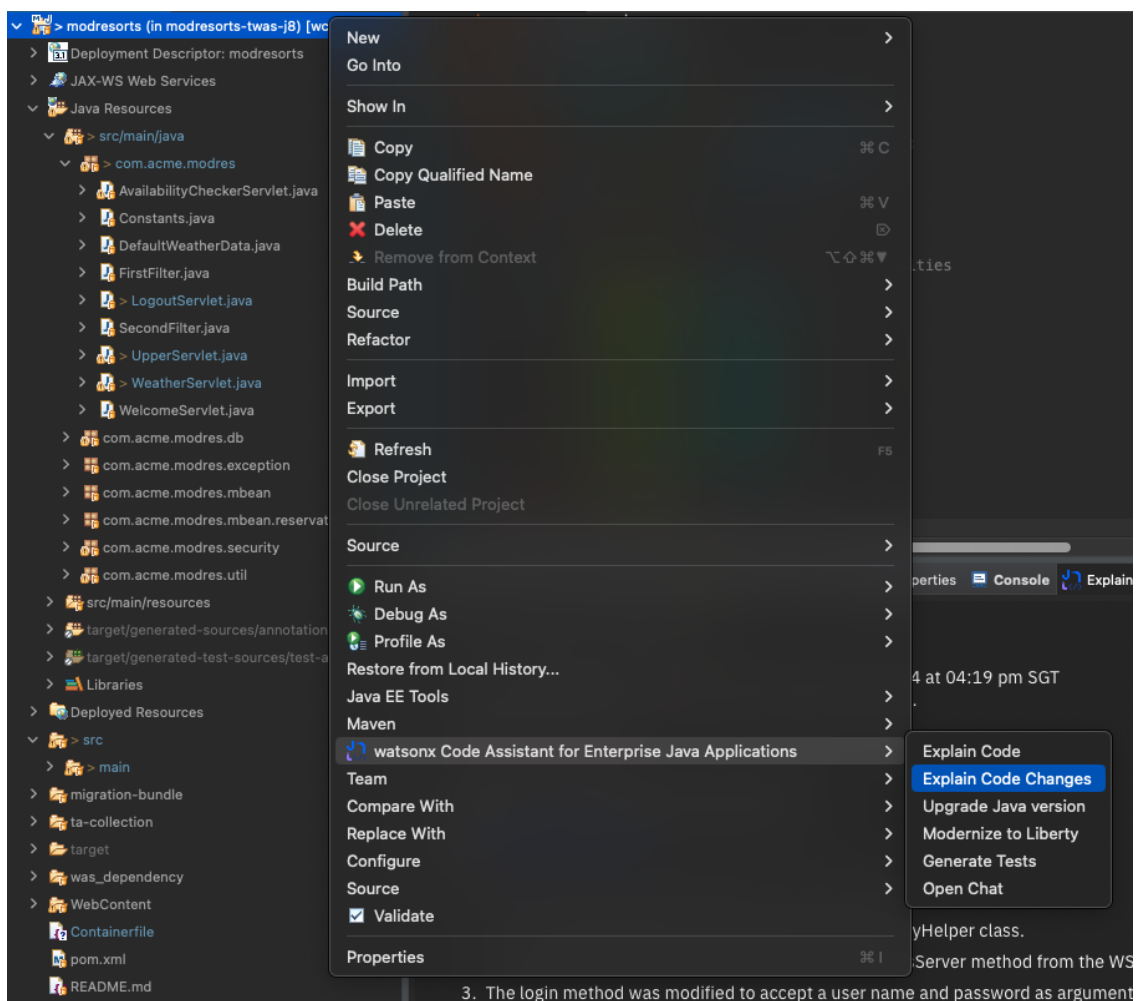
16. You can now return the <http://localhost:9080/resorts> to see that the application is running (Note: if you previously stopped the server in Liberty Tools, you need to restart it at this point). Let's look at the logout function now and verify that it works.

- Click the logout button.
- You should see a login page.
- For ease of demo setup, security for the application is actually turned off. You can enter any credentials and click login.

ModResorts is now a Liberty application!

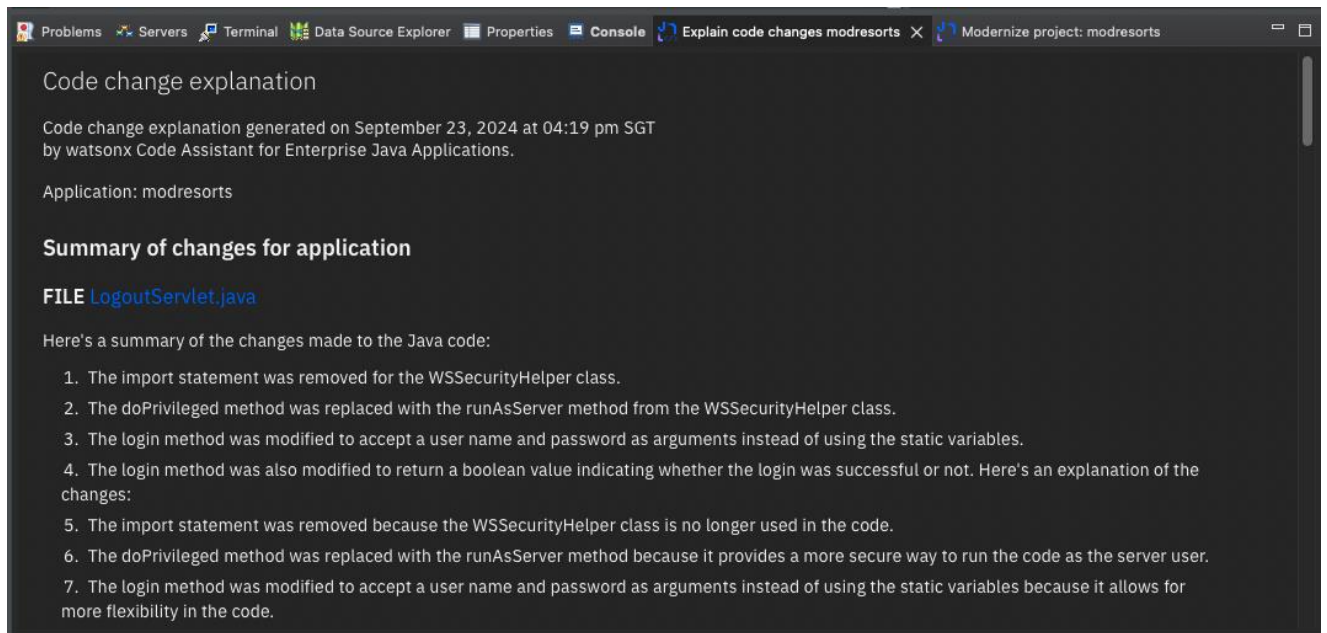
USE CASE 3: Explain Code Changes

1. It would be good to see what all changes have been made to the code, as they can later be included in the documentation or in the release notes
2. Right click on the project, click on **watsonx code assistant for enterprise java applications** and select **Explain Code Changes**:



3. The login method was modified to accept a user name and password as argument

3. In the console below, the code changes will show to explain all the changes that have been done to application as part of the modernization process.



The screenshot shows an IDE window with the 'Console' tab selected. The console displays a 'Code change explanation' generated on September 23, 2024, at 04:19 pm SGT by watsonx Code Assistant for Enterprise Java Applications. The application being modernized is 'modresorts'. The explanation is titled 'Summary of changes for application' and specifically for the file 'LogoutServlet.java'. It provides a summary of changes made to the Java code, listing seven items:

1. The import statement was removed for the WSSecurityHelper class.
2. The doPrivileged method was replaced with the runAsServer method from the WSSecurityHelper class.
3. The login method was modified to accept a user name and password as arguments instead of using the static variables.
4. The login method was also modified to return a boolean value indicating whether the login was successful or not. Here's an explanation of the changes:
5. The import statement was removed because the WSSecurityHelper class is no longer used in the code.
6. The doPrivileged method was replaced with the runAsServer method because it provides a more secure way to run the code as the server user.
7. The login method was modified to accept a user name and password as arguments instead of using the static variables because it allows for more flexibility in the code.