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

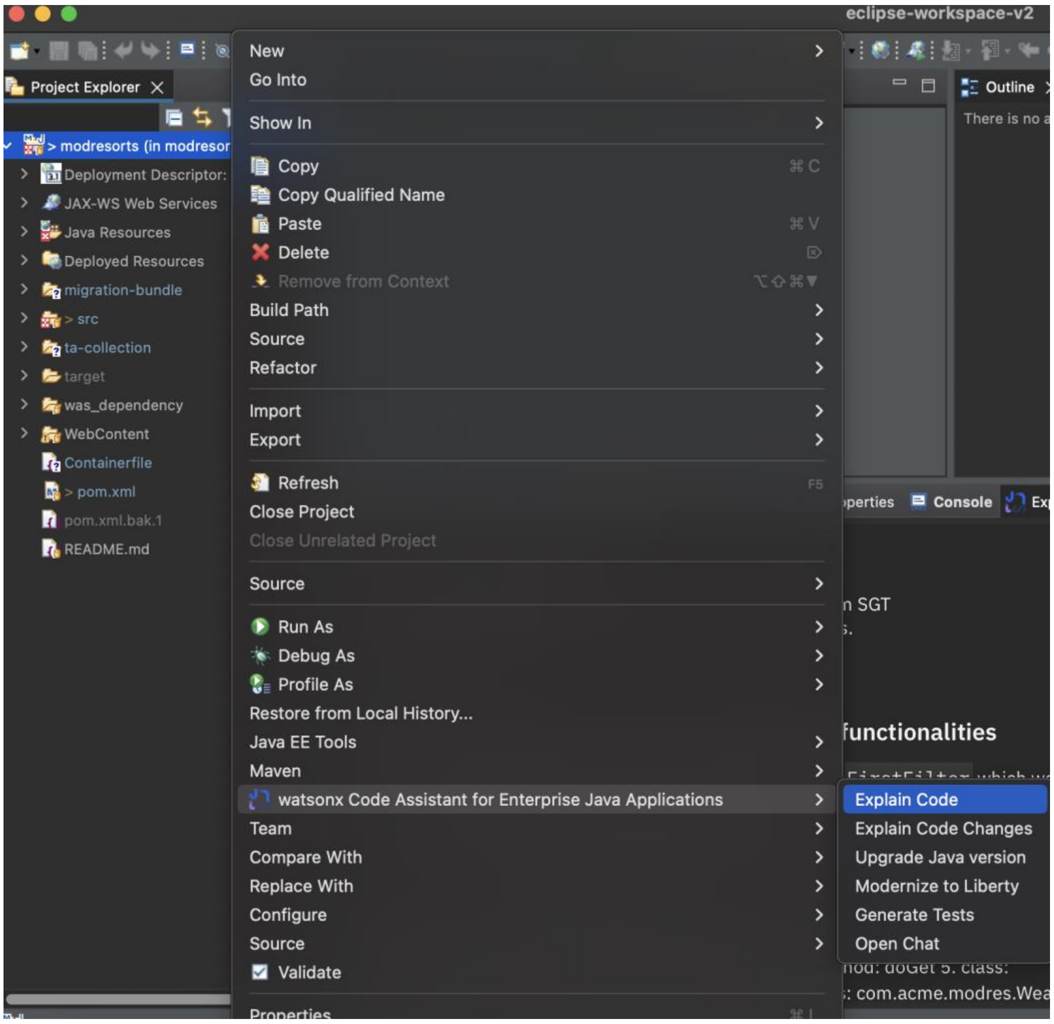
Build Project

1. Open a terminal, and go to your project folder, and navigate to was_dependency folder.
2. Once inside the folder, run the following command to build project:

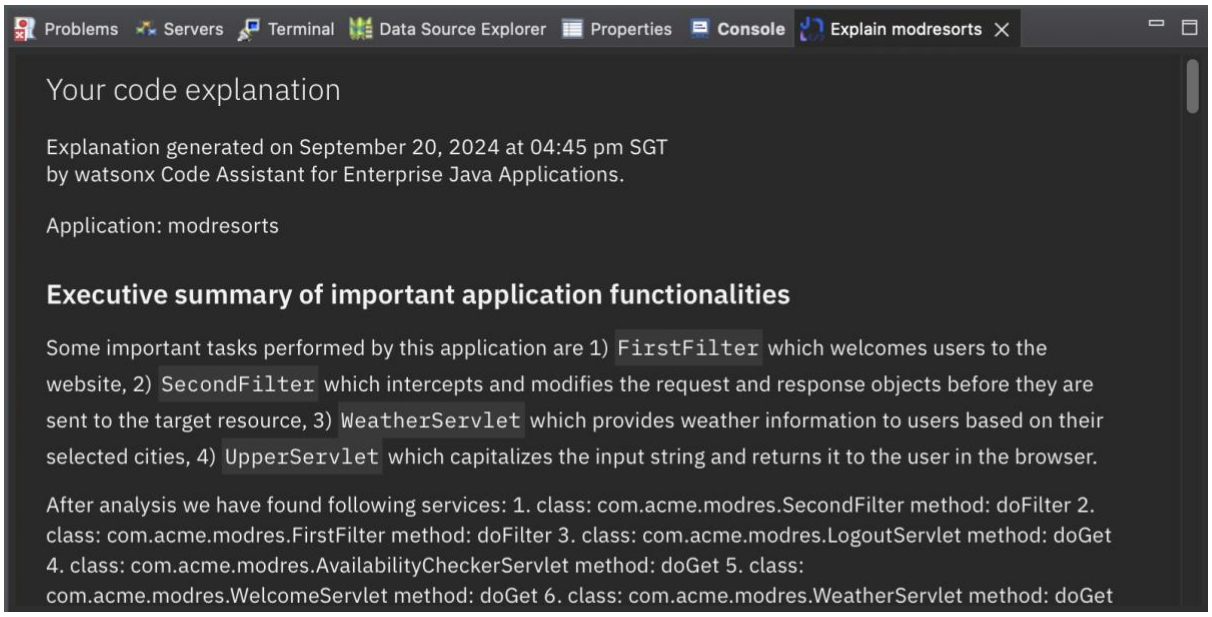
```
mvn install:install-file -Dfile=was_public.jar -DpomFile=was_public-9.0.0.pom
```

Explain Application

1. Git clone the project from the repository.
2. 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, click open and finish.
3. Once your IDE is setup, we will get an explanation of the entire project to understand what the app is doing.



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



Your code explanation

Explanation generated on September 20, 2024 at 04:45 pm SGT
by watsonx Code Assistant for Enterprise Java Applications.

Application: modresorts

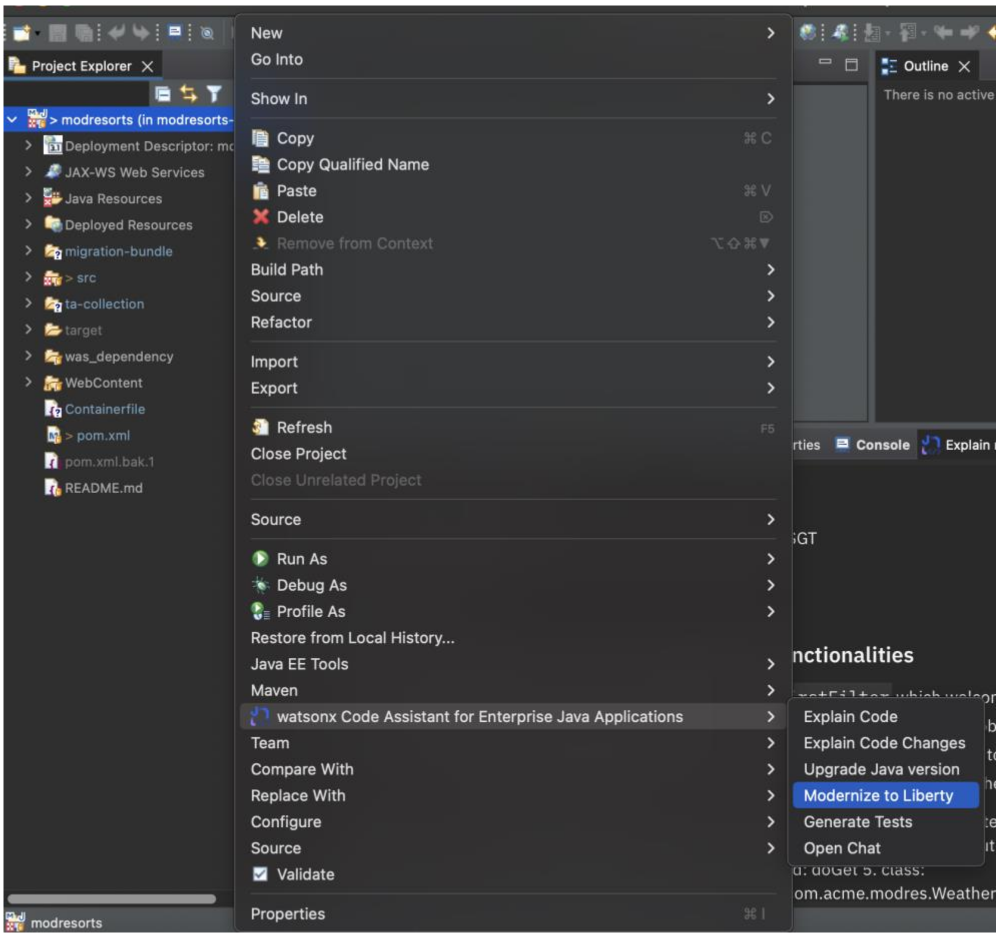
Executive summary of important application functionalities

Some important tasks performed by this application are 1) `FirstFilter` which welcomes users to the website, 2) `SecondFilter` which intercepts and modifies the request and response objects before they are sent to the target resource, 3) `WeatherServlet` which provides weather information to users based on their selected cities, 4) `UpperServlet` which capitalizes the input string and returns it to the user in the browser.

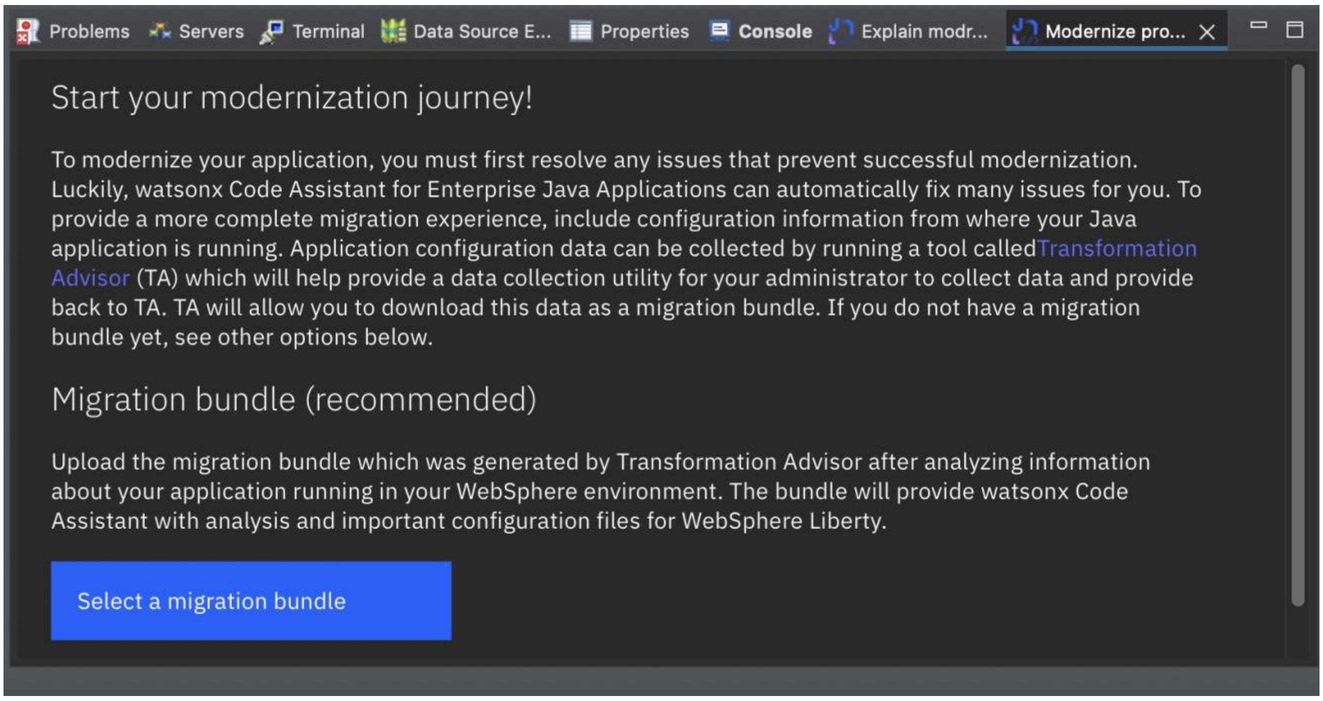
After analysis we have found following services: 1. class: com.acme.modres.SecondFilter method: doFilter 2. class: com.acme.modres.FirstFilter method: doFilter 3. class: com.acme.modres.LogoutServlet method: doGet 4. class: com.acme.modres.AvailabilityCheckerServlet method: doGet 5. class: com.acme.modres.WelcomeServlet method: doGet 6. class: com.acme.modres.WeatherServlet method: doGet

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:



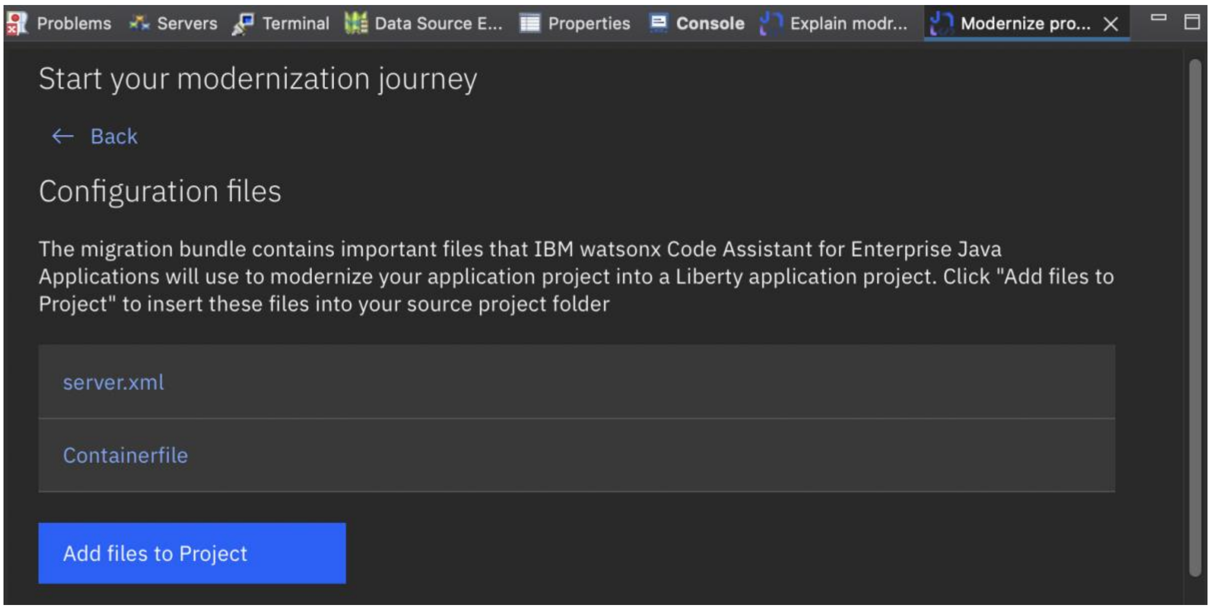
Start your modernization journey!

To modernize your application, you must first resolve any issues that prevent successful modernization. Luckily, watsonx Code Assistant for Enterprise Java Applications can automatically fix many issues for you. To provide a more complete migration experience, include configuration information from where your Java application is running. Application configuration data can be collected by running a tool calledTransformation Advisor (TA) which will help provide a data collection utility for your administrator to collect data and provide back to TA. TA will allow you to download this data as a migration bundle. If you do not have a migration bundle yet, see other options below.

Migration bundle (recommended)

Upload the migration bundle which was generated by Transformation Advisor after analyzing information about your application running in your WebSphere environment. The bundle will provide watsonx Code Assistant with analysis and important configuration files for WebSphere Liberty.

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



Start your modernization journey

← Back

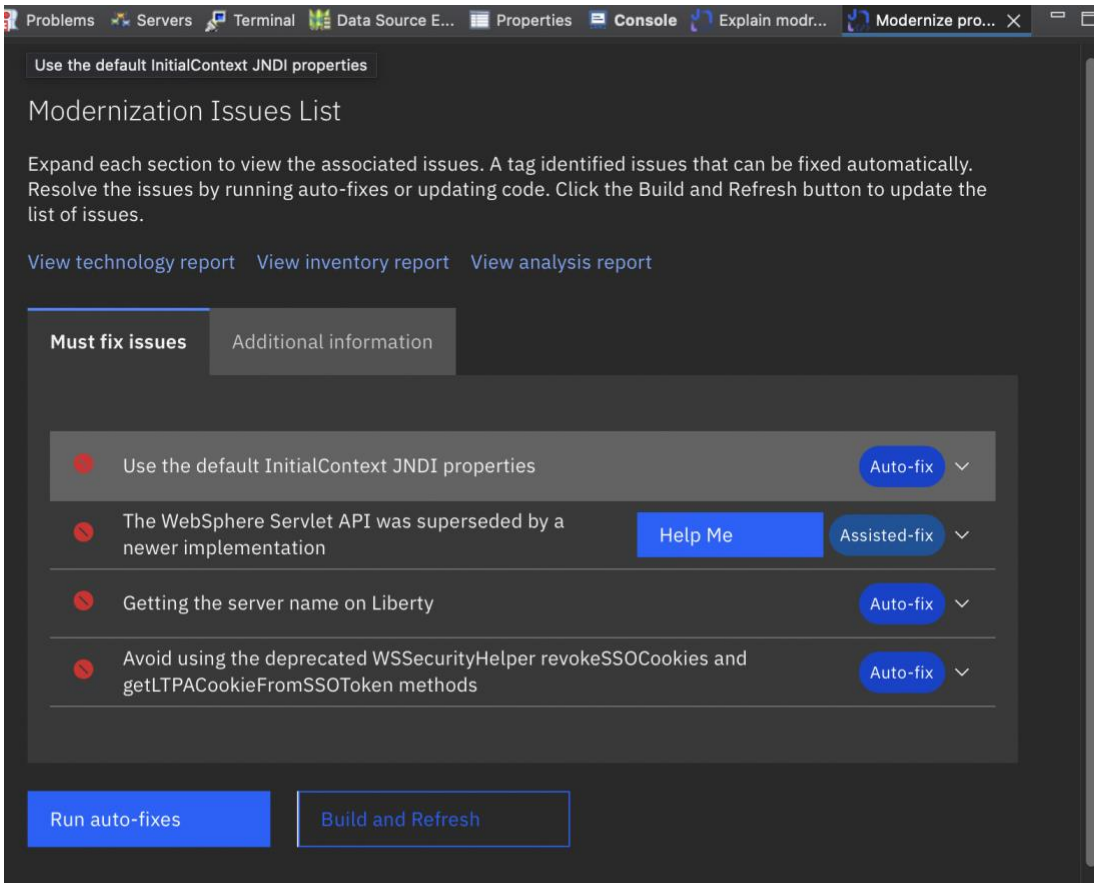
Configuration files

The migration bundle contains important files that IBM watsonx Code Assistant for Enterprise Java Applications will use to modernize your application project into a Liberty application project. Click "Add files to Project" to insert these files into your source project folder

server.xml

Containerfile

5. Click the button to add the files to the project. You should see the following panel:



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.

Use the default InitialContext JNDI properties — Auto-fix

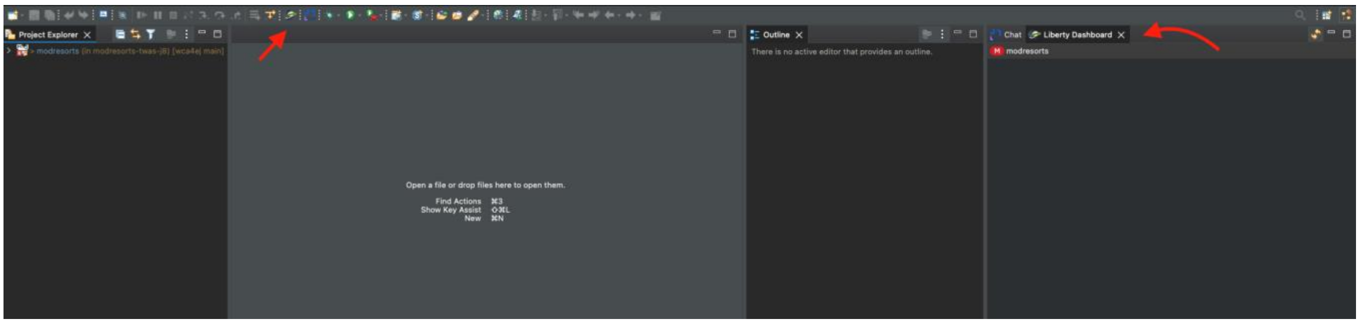
The WebSphere Servlet API was superseded by a newer implementation — Help Me — Assisted-fix

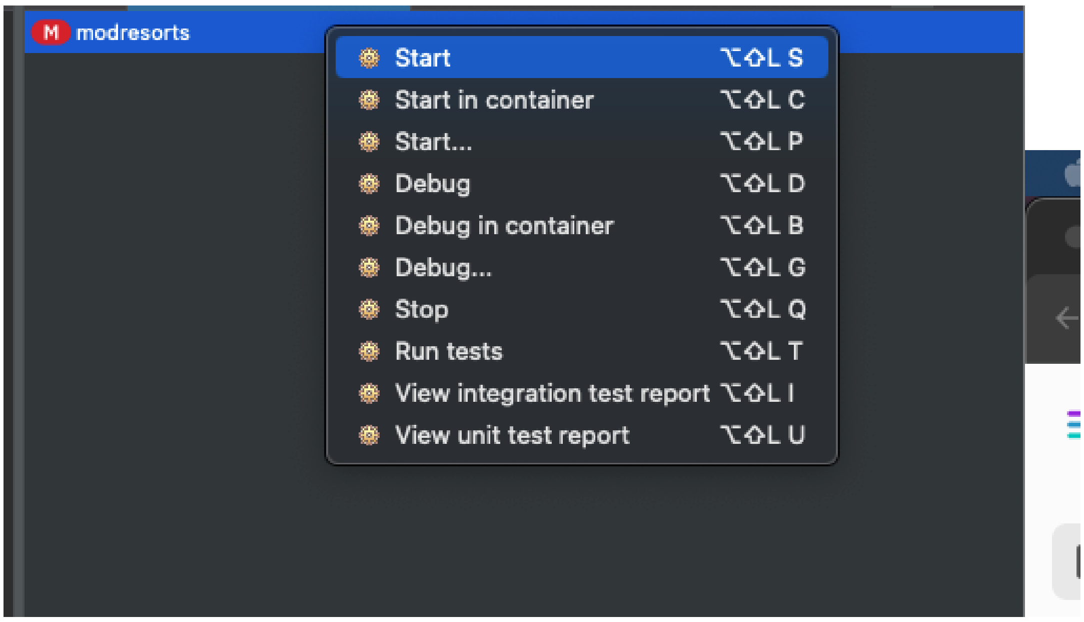
Getting the server name on Liberty — Auto-fix

Avoid using the deprecated WSSecurityHelper revokeSSOCookies and getLTPACookieFromSSOToken methods — Auto-fix

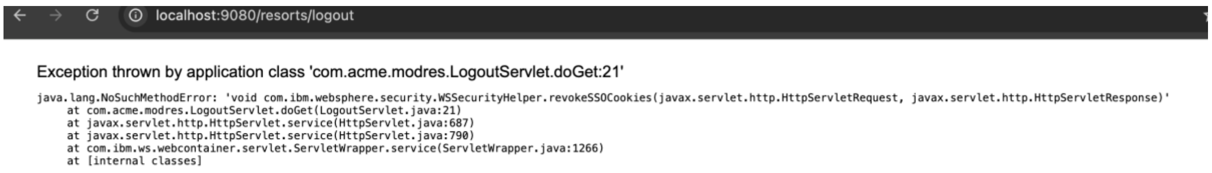
You will see 4 issues that must be fixed before the application can run successfully in Liberty. Three of the issues have auto fixes, and one issue has an assisted fix using the watsonx Code Assistant.

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:
    -





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:

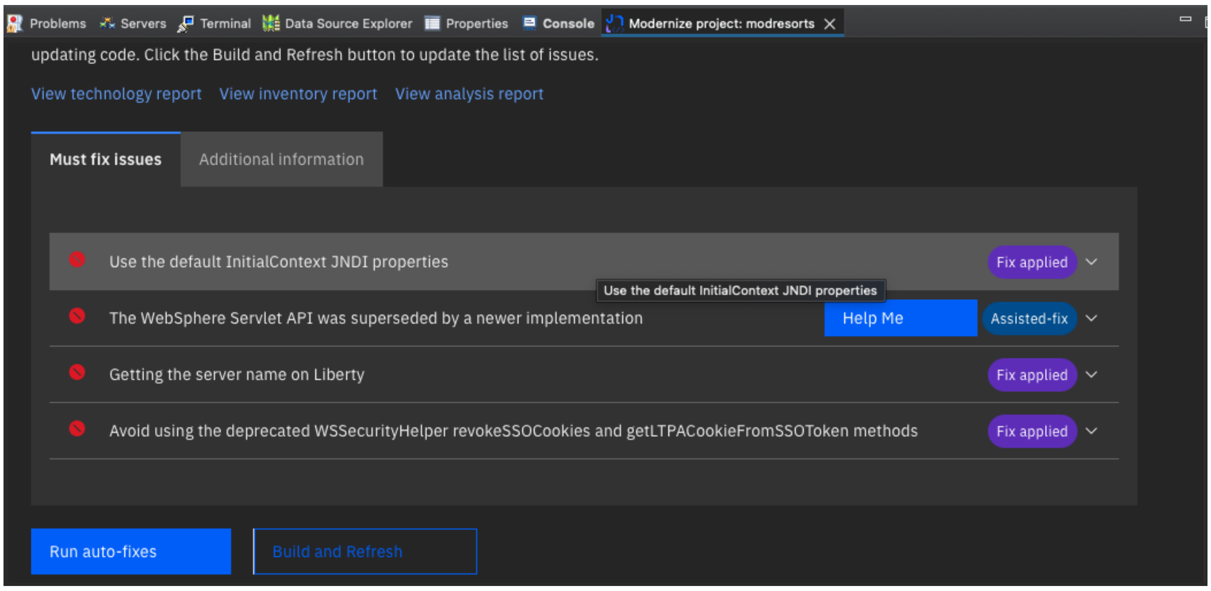


Exception thrown by application class 'com.acme.modres.LogoutServlet.doGet:21'

```
java.lang.NoSuchMethodError: 'void com.ibm.websphere.security.WSSecurityHelper.revokeSSOCookies(javax.servlet.http.HttpServletRequest, javax.servlet.http.HttpServletResponse)'
    at com.acme.modres.LogoutServlet.doGet(LogoutServlet.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:



updating code. Click the Build and Refresh button to update the list of issues.

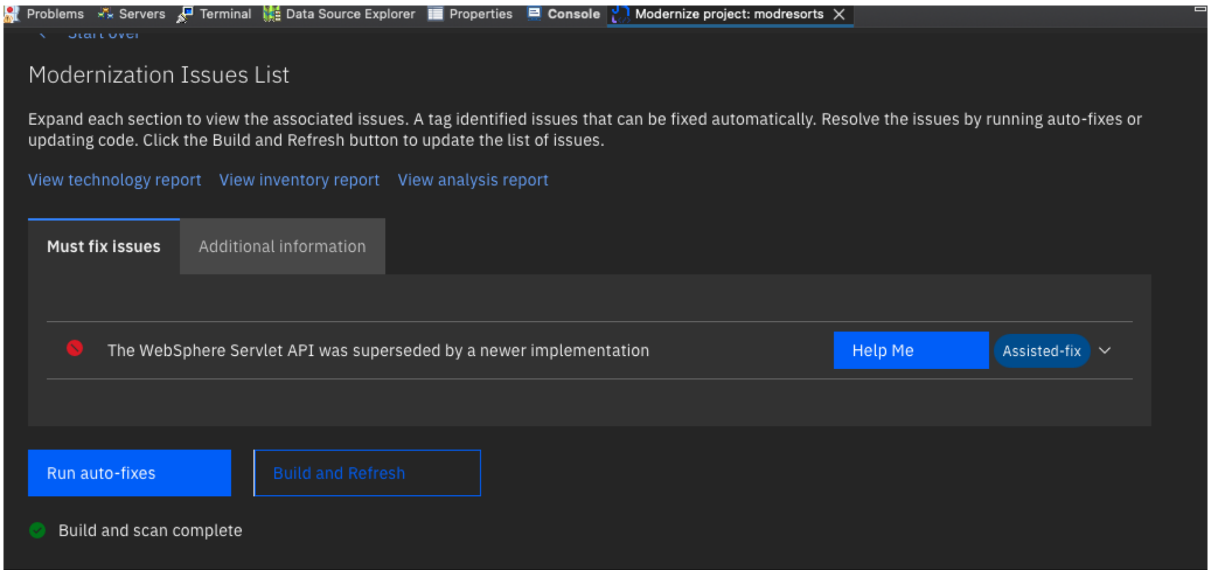
View technology report View inventory report View analysis report

Must fix issues | Additional information

- Use the default InitialContext JNDI properties — Fix applied
- The WebSphere Servlet API was superseded by a newer implementation — Help Me — Assisted-fix
- Getting the server name on Liberty — Fix applied
- Avoid using the deprecated WSSecurityHelper revokeSSOCookies and getLTPACookieFromSSOToken methods — Fix applied

Use the default InitialContext JNDI properties

Run auto-fixes | Build and Refresh



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

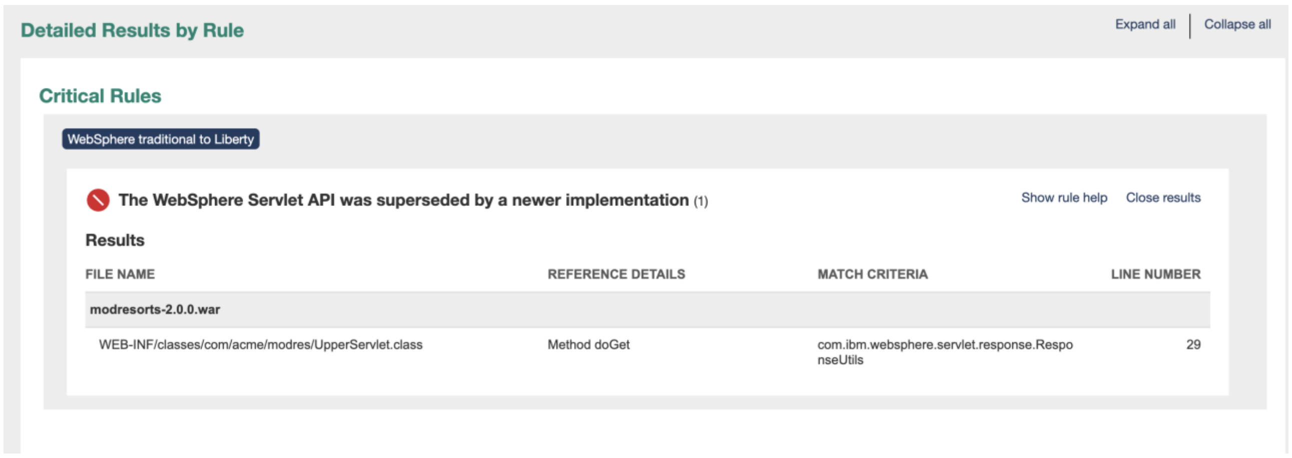
Must fix issues | Additional information

- The WebSphere Servlet API was superseded by a newer implementation — Help Me — Assisted-fix

Run auto-fixes | Build and Refresh

Build and scan complete

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

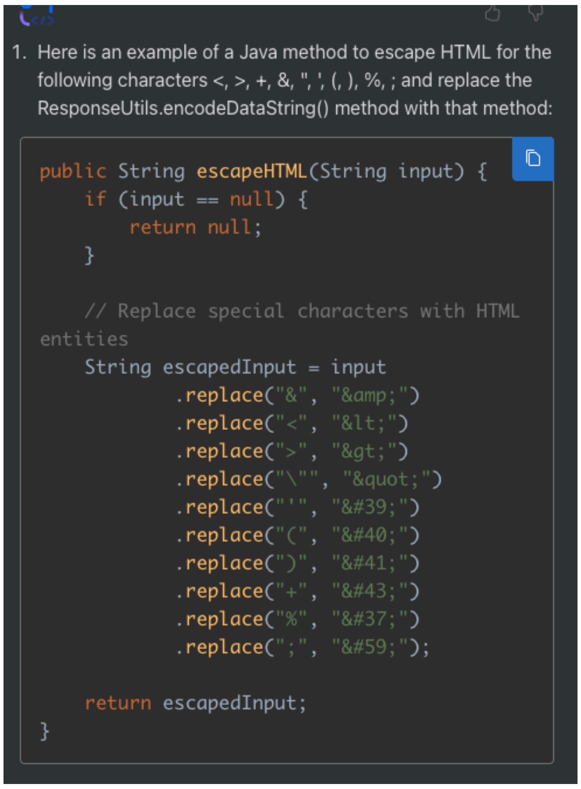


The WebSphere Servlet API was superseded by a newer implementation (1)

Results

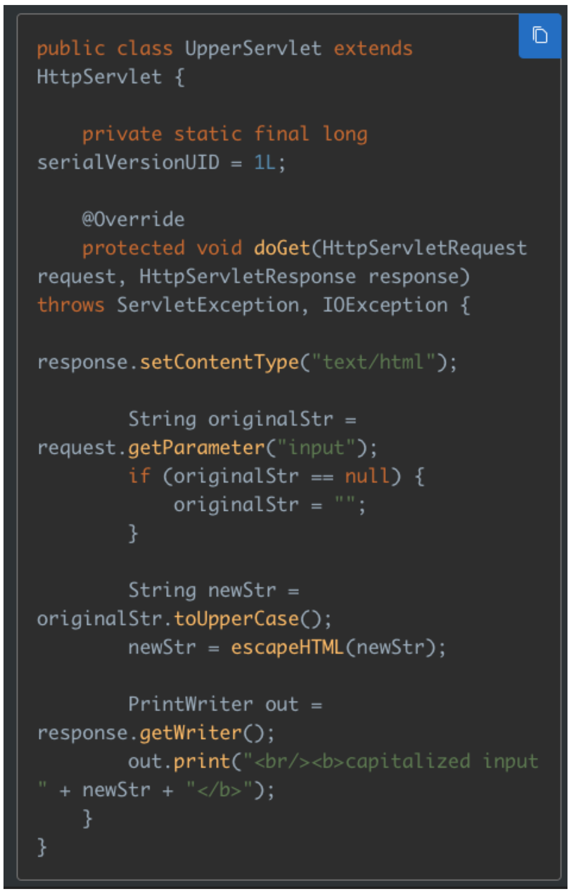
FILE NAME	REFERENCE DETAILS	MATCH CRITERIA	LINE NUMBER
modresorts-2.0.0.war			
WEB-INF/classes/com/acme/modres/UpperServlet.class	Method doGet	com.ibm.websphere.servlet.response.ResponseUtils	29

12. 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.
13. 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:
    - 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.



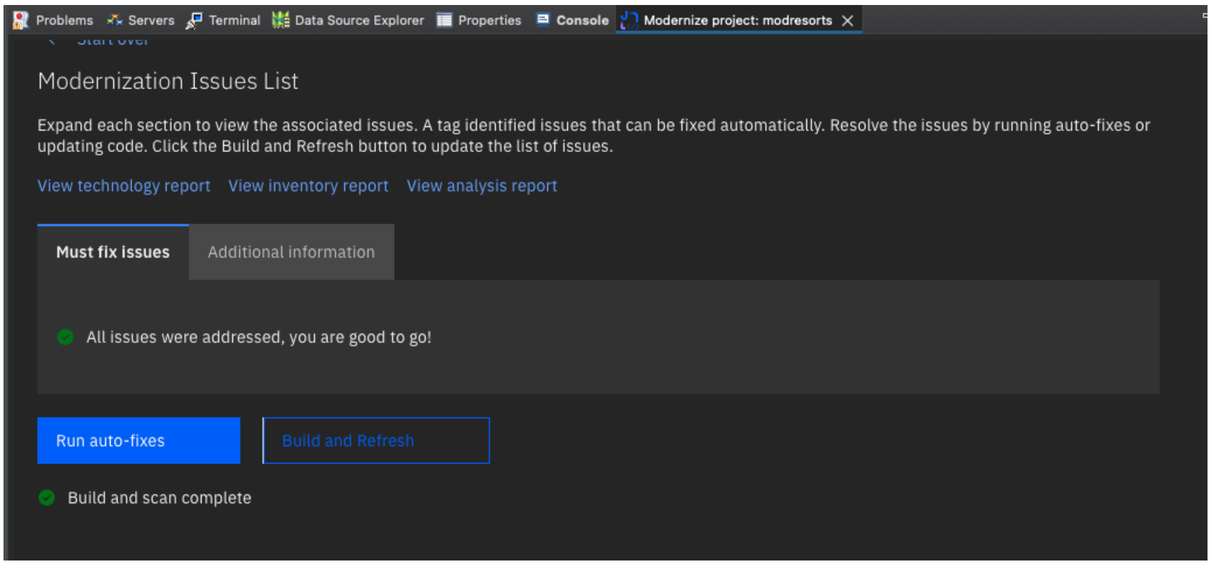
1. Here is an example of a Java method to escape HTML for the following characters <, >, +, &, ", ', (,), %, ; and replace the ResponseUtils.encodeDataString() method with that method:

```java
public String escapeHTML(String input) {
    if (input == null) {
        return null;
    }

    // Replace special characters with HTML
entities
    String escapedInput = input
            .replace("&", "&amp;")
            .replace("<", "&lt;")
            .replace(">", "&gt;")
            .replace("\"", "&quot;")
            .replace("'", "&#39;")
            .replace("(", "&#40;")
            .replace(")", "&#41;")
            .replace("+", "&#43;")
            .replace("%", "&#37;")
            .replace(";", "&#59;");

    return escapedInput;
}
```



```java
public class UpperServlet extends
HttpServlet {

    private static final long
serialVersionUID = 1L;

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

response.setContentType("text/html");

        String originalStr =
request.getParameter("input");
        if (originalStr == null) {
            originalStr = "";
        }

        String newStr =
originalStr.toUpperCase();
        newStr = escapeHTML(newStr);

        PrintWriter out =
response.getWriter();
        out.print("<br/><b>capitalized input
" + newStr + "</b>");
    }
}
```

14. 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.



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.

All issues were addressed, you are good to go!

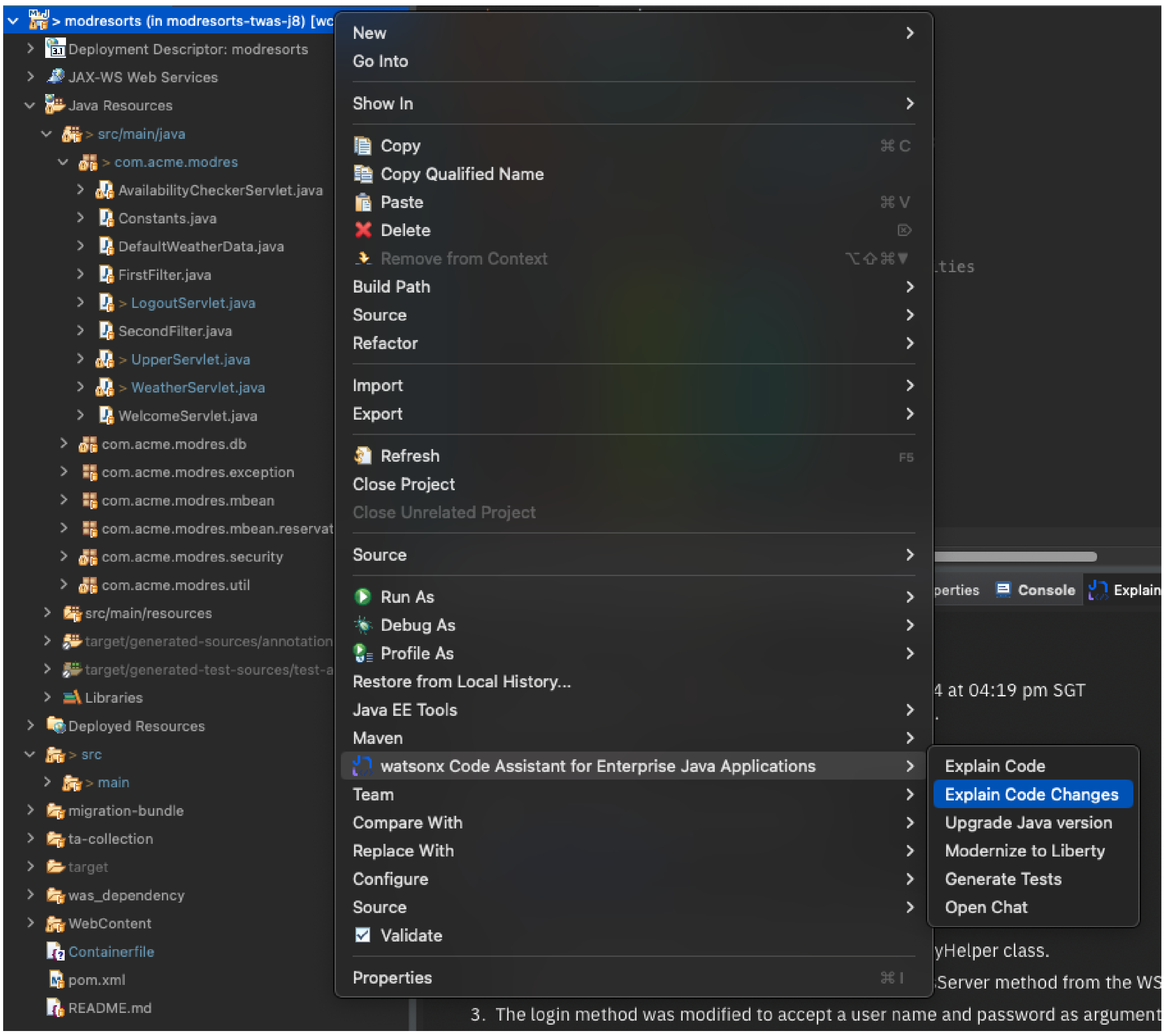
Build and scan complete

15. 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!

<u>Explain Code Changes:</u>

1. Before proceeding, 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:



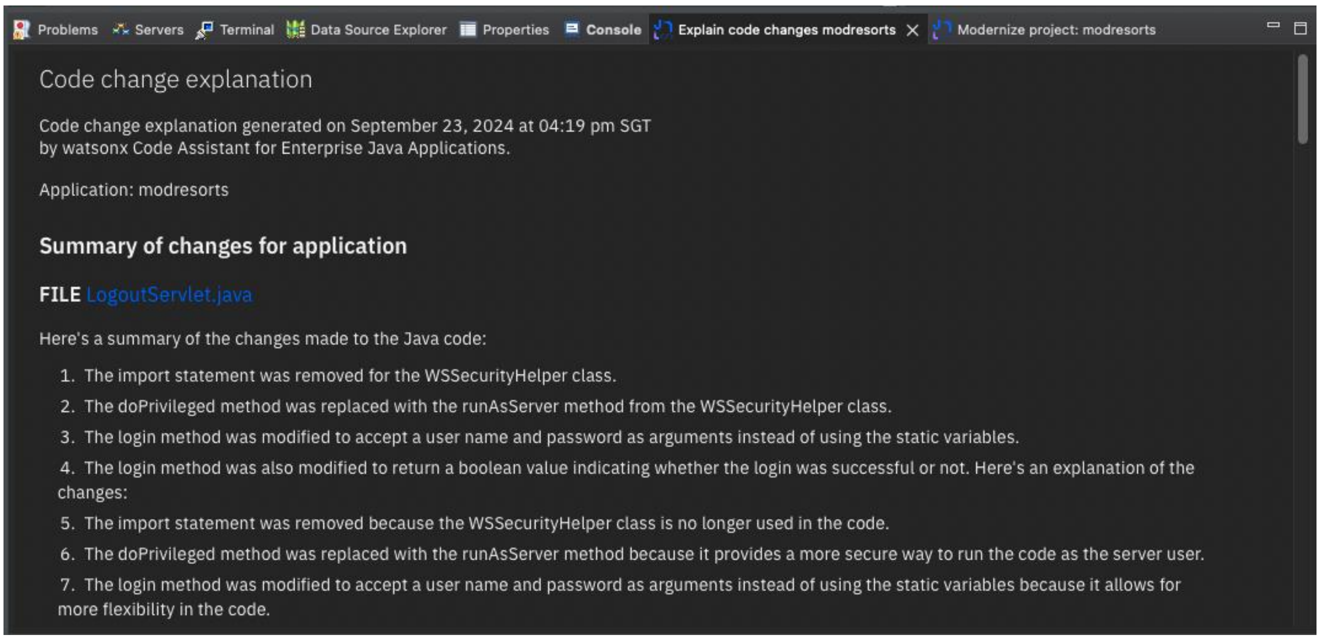
4 at 04:19 pm SGT

yHelper class.

Server method from the WS

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


Problems
Servers
Terminal
Data Source Explorer
Properties
Console
Explain code changes modresorts
Modernize project: modresorts

Code change explanation

Code change explanation generated on September 23, 2024 at 04:19 pm SGT
by watsonx Code Assistant for Enterprise Java Applications.

Application: modresorts

Summary of changes for application

FILE LogoutServlet.java

Here's a summary of the changes made to the Java code:

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.