HCL Digital Experience

HDX-DEV-300
DX JSP Portlet Development Lab

Edition: July 2024

Herbert Hilhorst – herbert.hilhorst@hcl-software.com

Thorsten Reinheimer - thorsten.reinheimer@hcl-software.com

Table of Contents

Author(s)

This document was created by the following Subject Matter Experts:

Herbert Hilhorst
Company:
HCLSoftware

Bio

Herbert Hilhorst is an HCL Digital Experience (DX) Technical Advisor at HCLSoftware.

Contact: herbert.hilhorst@hcl-software.com

Thorsten Reinheimer
Company:
HCLSoftware

Bio

Thorsten Reinheimer is an HCL Digital Experience (DX) Technical Lead at HCLSoftware.

Contact: thorsten.reinheimer@hcl-software.com

Introduction

This hands-on lab gets you started on Java Portlet Development on the HCL Digital Experience (DX) platform.

In this DX developer lab, you play the role of Gene, a developer for the fictitious Woodburn Studio company.

Gene Hayes, Developer, based in Chicago (USA)

You will need to develop, deploy, test, update and debug a new Java Portlet on HCL DX.

Prerequisites

1. Completion of the HDX-INTRO course as this gives you access to your own DX instance on HCL SoFy
2. Completion of HDX-DEV-100 as this helps you setting up the DXClient and deploy DX standalone as a Docker instance
3. Completion of the HDX-DEV-300 DX - Setup a Java Development Environment lab to set up your local Java development environment. In this lab, the instructions will be based on Visual Studio Code.
4. Access to download the Lab Resources:
   In the same place where you have found this lab, you will find corresponding resources which you may download and unzip in your Desktop. This helps you to run the lab more easily, and you may later replace them by your own ones.

You will be using the following user IDs and passwords:

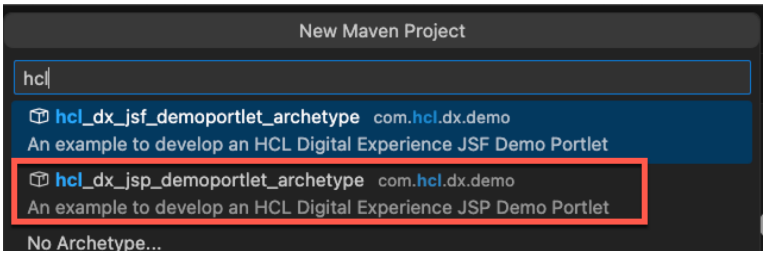
Purpose	User	Password
SoFy Login	Your official email id	Your password
SoFy Solution Console Login	sol-admin	<from SoFy solution>
DX Login	testuser	testuser
DX Login	wpsadmin	wpsadmin

Lab Overview

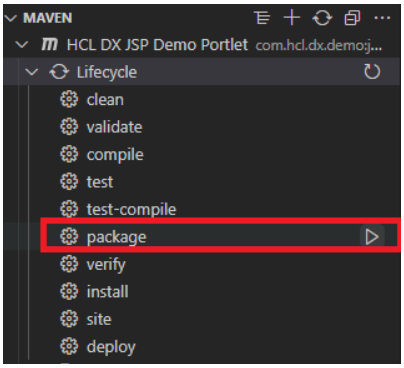
In this lab, you will explore how to build JSP portlets with Microsoft Visual Studio Code. Before you start with this lab, please make sure that you have completed **Part 1: DX Java Development with Visual Code Studio** of the **HDX-DEV-300 DX Setup a Java Development Environment Lab**, as this lab documents the steps setting up Java, Maven with the DX JSP archetype and Microsoft Visual Studio.
Notice that if you need to develop JSF portlets, you will be pointed out what resources to use and the flow is very similar.

There are several parts in this lab, shortly introduced now.

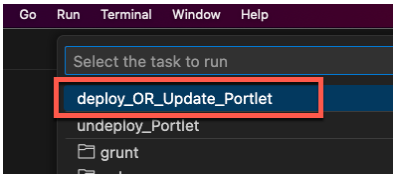
Part 1: Create a first JSP Portlet
You will use Microsoft Visual Studio Code and Maven to create your first JSP Portlet, a JSP portlet called HCL DX JSP Demo Portlet.



Part 2: Build and Deploy to DX
Then you will build and deploy your new JSP Portlet you just created. Using Maven, you will generate the JSP Portlet WAR file.

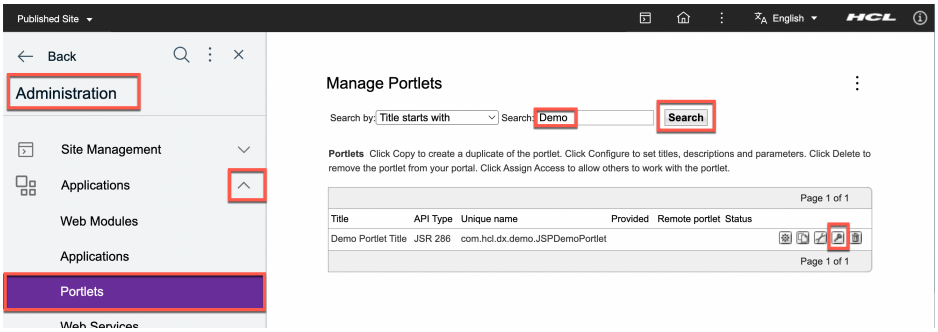


Which you deploy to your DX server using the DXClient with some helpers.



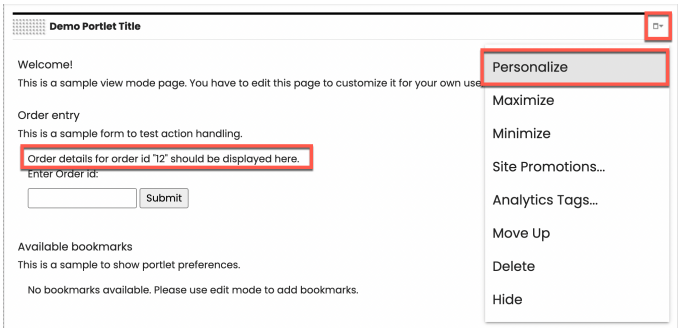
Part 3: Secure on DX

You will secure the Portlet.



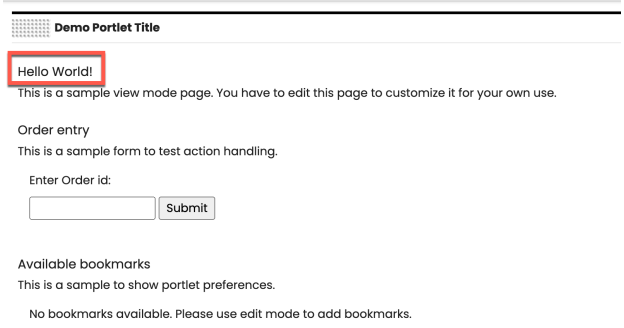
Part 4: Test on DX

You will add the Portlet to a page and test it.



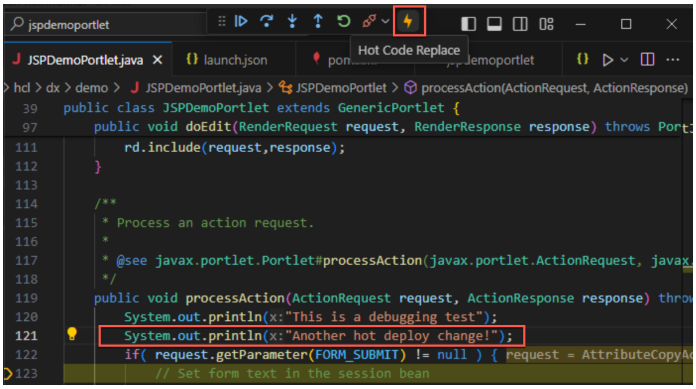
Part 5: Update a JSP Portlet

You will learn how to make updates to the JSP Portlet, deploy and test it again.



Optional Part 6: Debug the Portlet

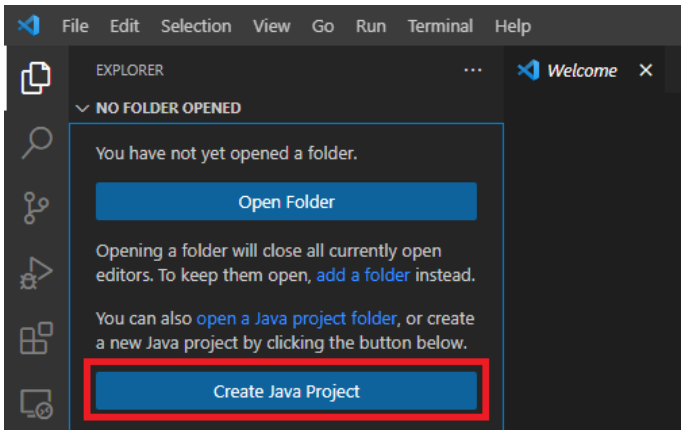
And finally, you will learn how to debug your JSP Portlet.



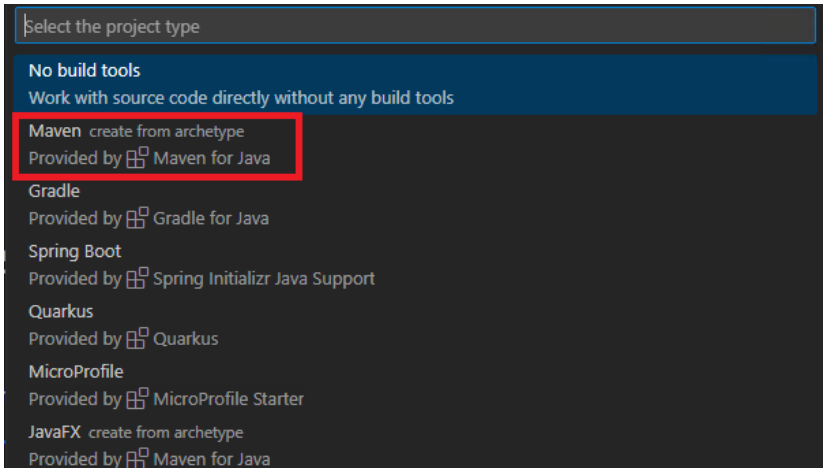
Part 1: Create a first JSP Portlet

You will use Microsoft Visual Studio Code and Maven to create your first JSP portlet, called HCL DX JSP Demo Portlet.

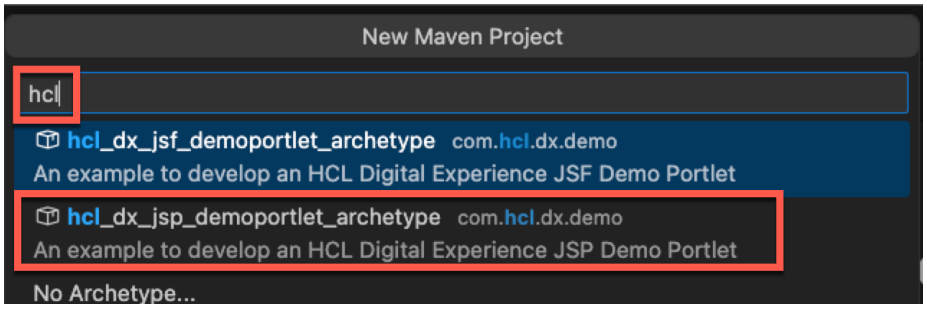
1. First create a new Java project. Start Microsoft Visual Studio Code in a new empty workspace and then click **Create Java Project**.



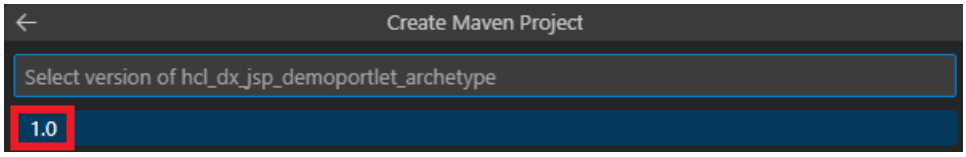
2. In the **Select the project type** section, click **Maven – create from archetype**.



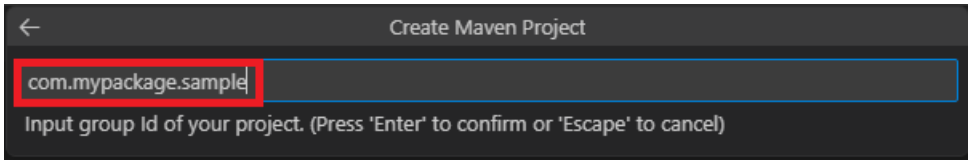
3. Then search for **hcl** and click **hcl_dx_jsp_demoportlet_archetype** (or hcl_jsf_demoportlet_archetype if you want to develop with JSF).



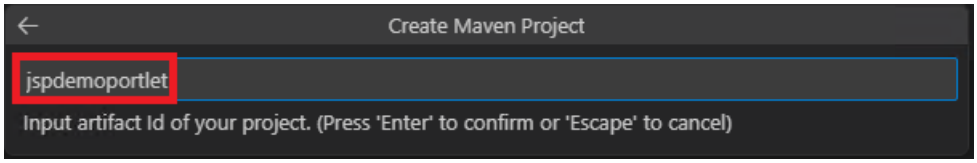
4. Then select **1.0** as version.



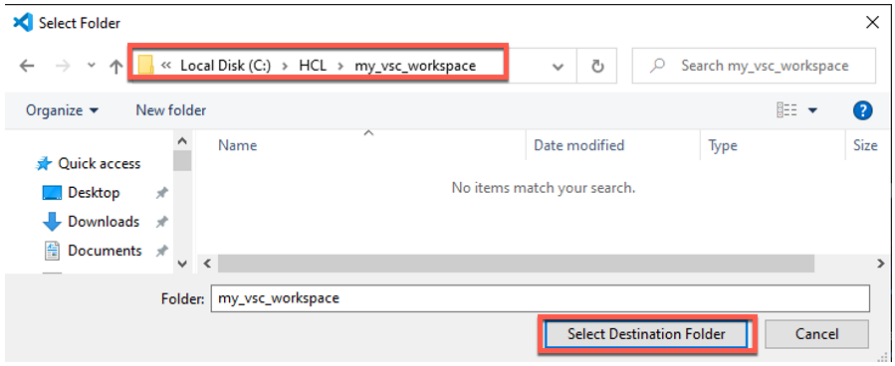
5. Then specify a group id of your project. This group ID should reflect your own package name in which the HCL DX JSP Demo Portlet will be stored. For example, use **com.mypackage.sample**. Enter the name and hit enter.



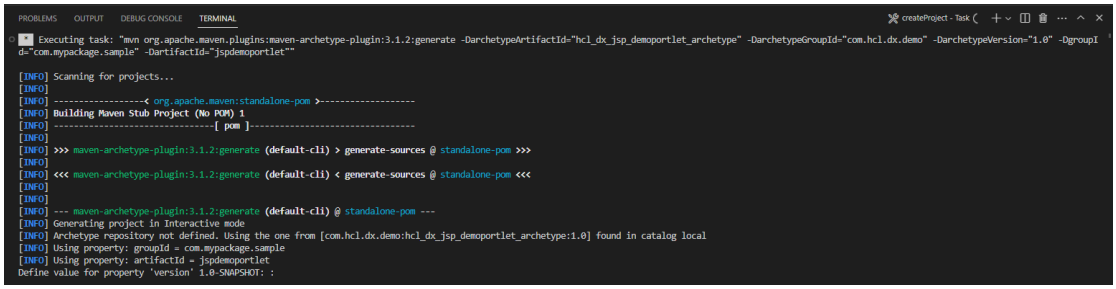
6. Then provide the artifact id, which is like the class name in java. For example, use **jspdemoportlet**. Enter the name and hit enter.



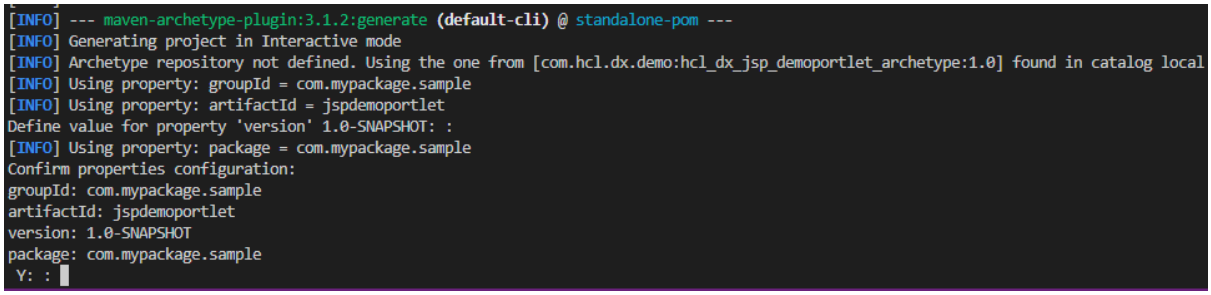
7. As soon as the artifact id is specified, a window pops up to select the destination folder of your new project. The folder needs to exist already, so you may want to create a new empty directory, e.g. using your File Explorer. For example: **C:\HCL\my_vsc_workspace**. Select your folder and then click **Select Destination Folder**.



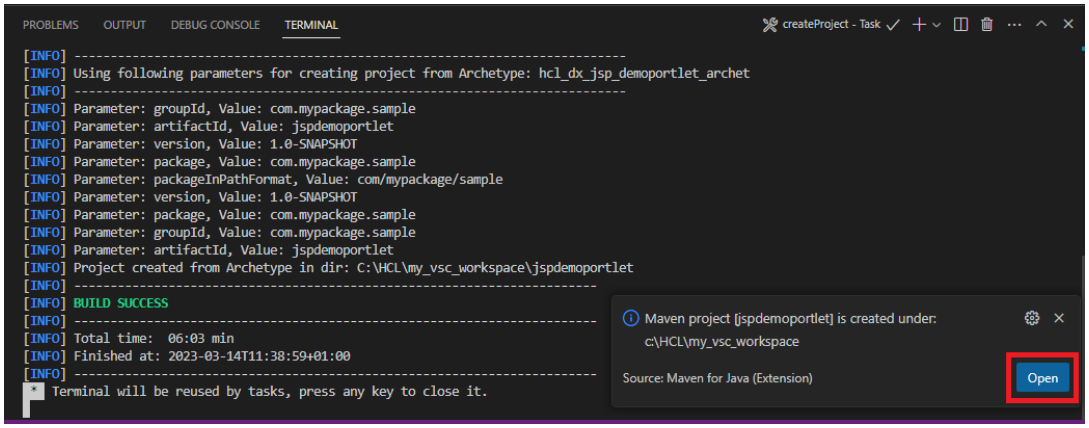
8. The Maven process is then automatically started and generates the resources. The progress can be monitored in the TERMINAL tab.



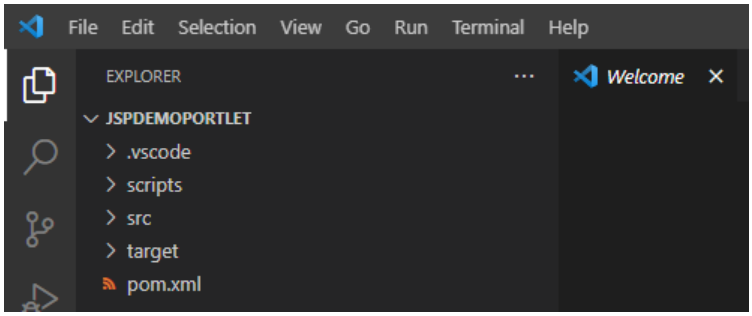
9. The project generation process stops as soon as the **Define value for property 'version' 1.0-SNAPSHOT: :** line is printed. Accept the default value **1.0-SNAPSHOT** by just hinting the ENTER key. Then, you need to confirm the auto-generated package name, you selected before. Confirm it with another ENTER key.



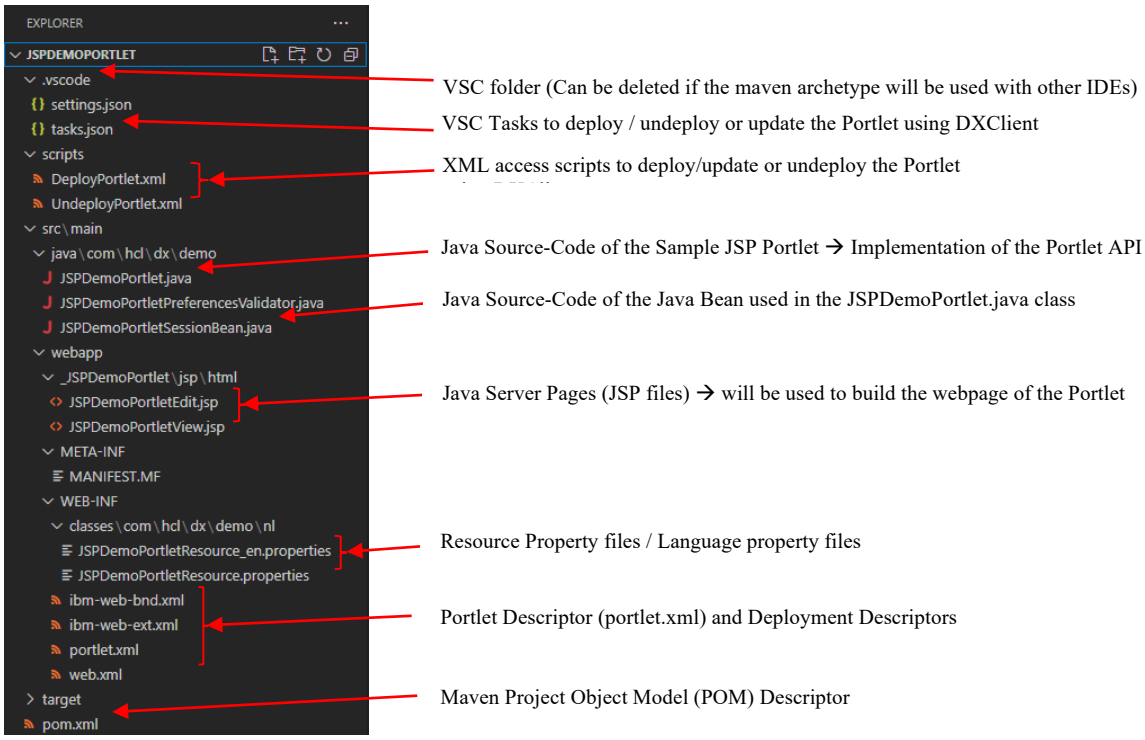
10. Finally, the project generation task continues and completes with the green message **BUILD SUCCESS**. In addition, a pop up asks to open the new project. Click **Open** to open it.



11. When the new project folder is loaded, the following resources are generated.



12. Expand the folders and examine the generated project files:

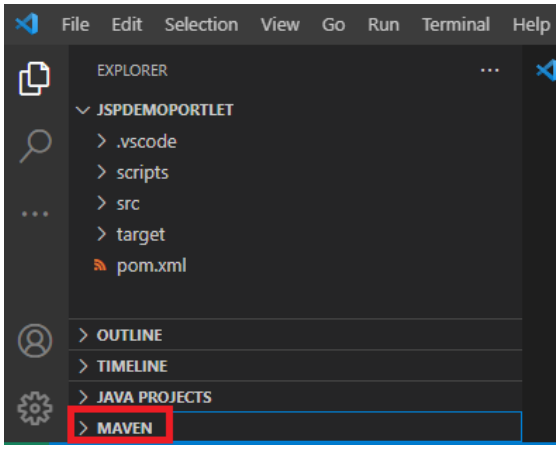


Congratulations! You have successfully created your first JSP Portlet in Microsoft Visual Studio Code.

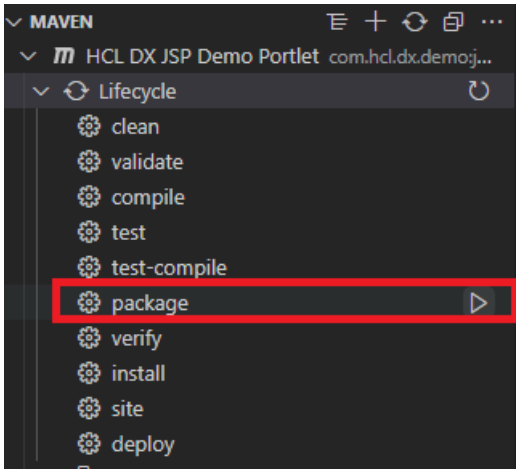
Part 2: Build and Deploy to DX

In this part you will build and deploy your new JSP Portlet you just created. Using Maven, you will generate the JSP Portlet WAR file which you deploy to your DX server using the DXClient.

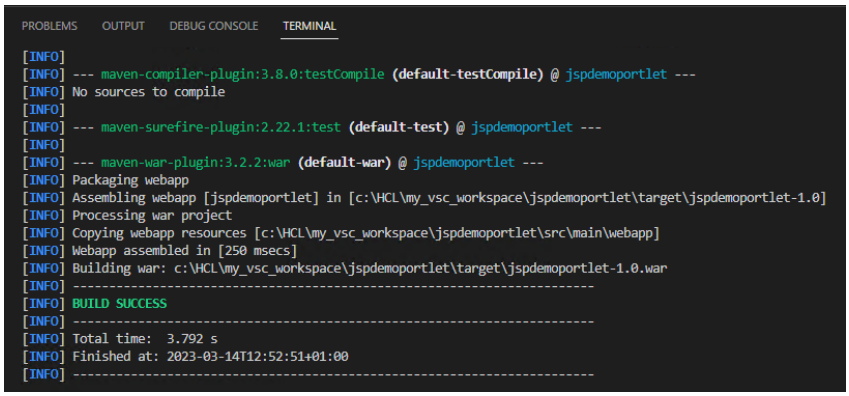
1. Use the Maven package function to build your portlet first. Expand the Maven section in the Visual Studio Code Project



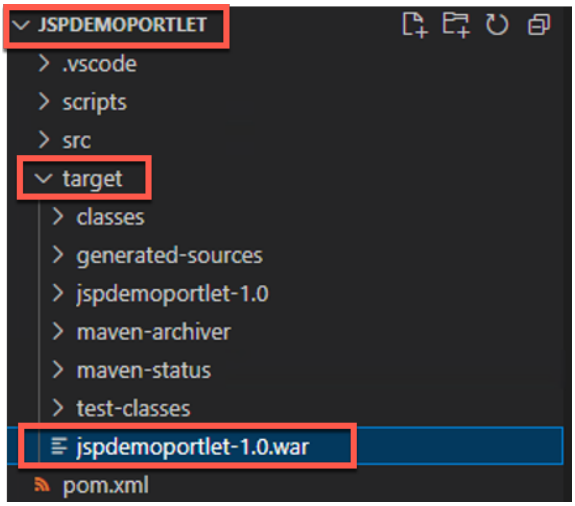
2. Then expand **HCL DX JSP Demo Portlet** - **Lifecycle** and click **package**.



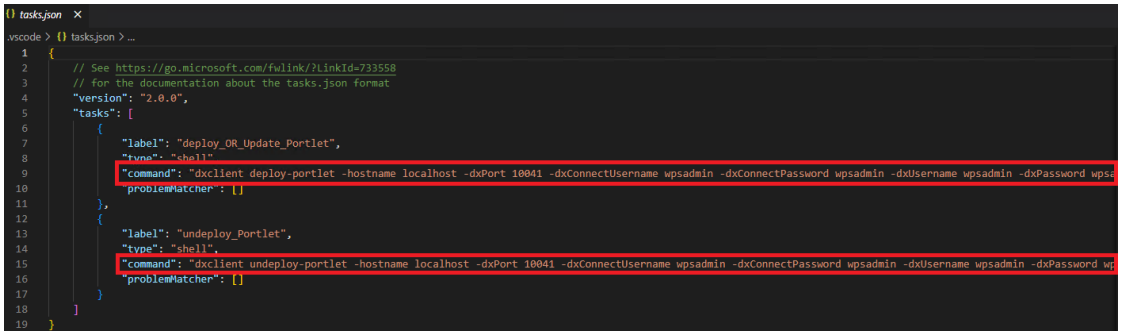
3. The TERMINAL tab opens and shows the build process. When successfully completed, it should show **BUILD SUCCESS**.



4. It has generated the JSP Demo Portlet jspdemoportlet-1.0.war file in the target folder.

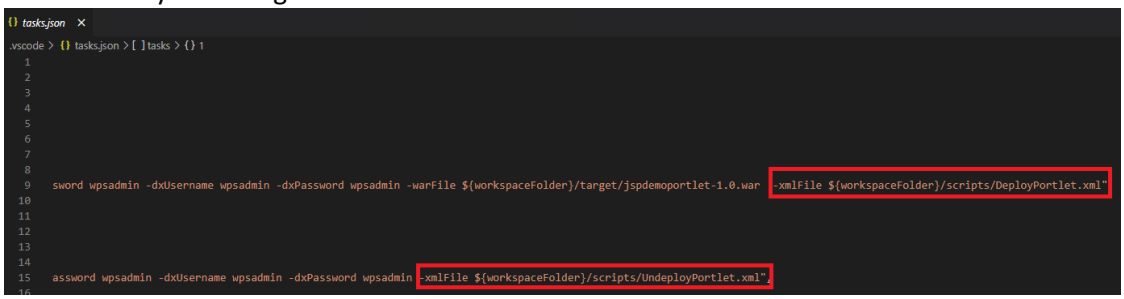


5. Now you will deploy this WAR file of your JSP demo portlet on a HCL DX server running on HCL SoFy. The generated sample code contains "Visual Studio Code – Tasks" that can be used for deploying the WAR file on any HCL DX server. For that, please expand the **.vscode** folder and check the source code of the **tasks.json** file. The tasks are defined in a JSON format. Each task has a label, type, command and problemMatcher object. Notice that the command line uses a DXClient command that deploys/updates and undeploys your portlet, described in details here https://opensource.hcltechsw.com/digital-experience/latest/extend_dx/development_tools/dxclient/dxclient_artifact_types/portlets/. It is configured to run on a localhost by default.

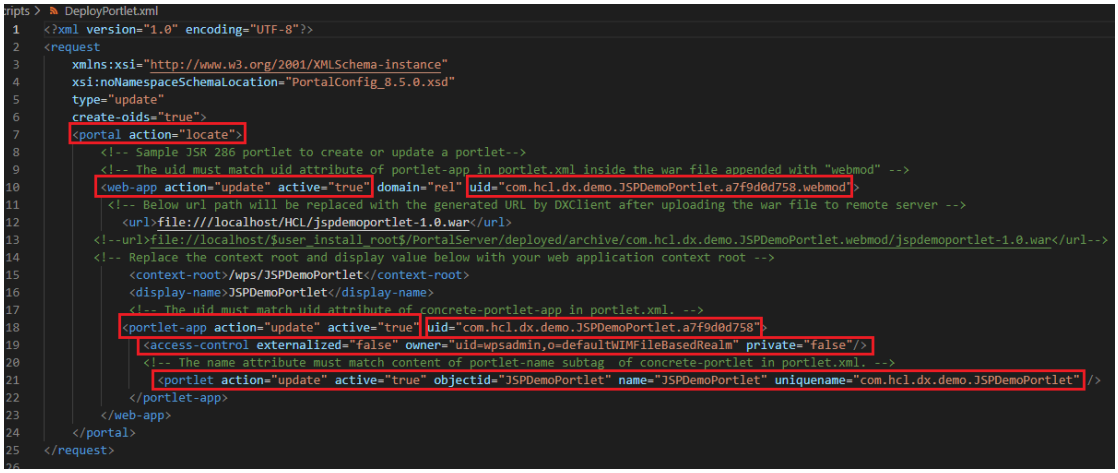


6. You need to update its parameters of the command lines for DXClient to point to your DX server. With a DX server on HCL SoFy, it may look like
hostname: dx.sbx0000.play.hclsofy.com
dxPort: 443
dxProtocol: https

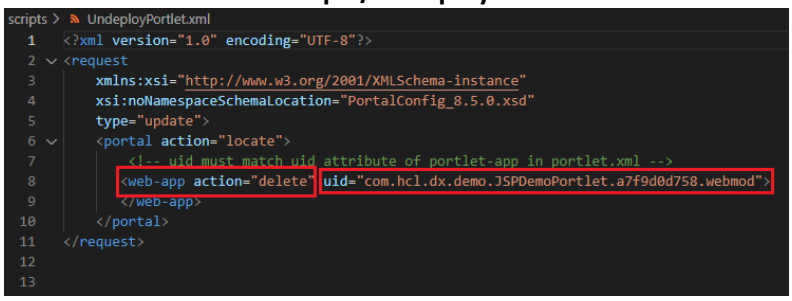
7. Notice the parameter **-xmlFile** that is set to the provided XMLAccess files DeployPortlet.xml and UndeployPortlet.xml in the scripts folder. These files describe what is needed to run that action. **Save** your changes.



8. Then open the **scripts/DeploymentPortlet.xml** file. It includes details about the portlet-Id, permissions, and the location in which it should be installed or from which location it should be uninstalled. In red are the important information marked. Specially check the uid values of the web-app and portlet-app tag, as these IDs will be used later to update and undeploy the application.

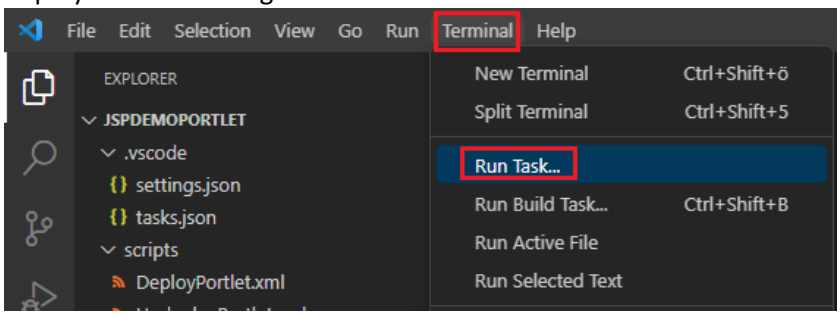


9. Also have a look at the **scripts/UndeployPortlet.xml**.



10. Important! The uid=com.hcl.dx.demo.JSPDemoPortlet.a7f9d0d758 needs to match the id attribute in the portlet.xml file! If that is missing or the ID do not match, the deployment/update or undeployment may fail! You may want to double check that.

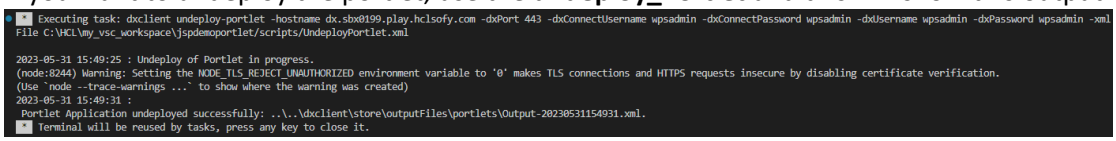
11. Using these auto-generated tasks, deploying the HCL DX Demo Portlet is quite easy. Run the deployment task using **Terminal** - **Run Task…**



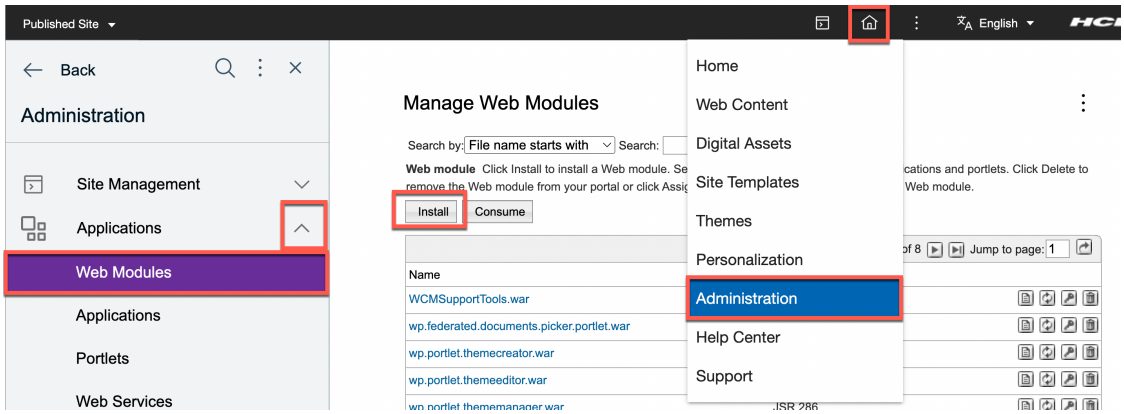
12. Then click **deploy_OR_Update_Portlet** to deploy (and later update) our portlet. Notice the **undeploy_Portlet** task you may use to undeploy it.

13. This starts the deployment in the TERMINAL window with this output information.

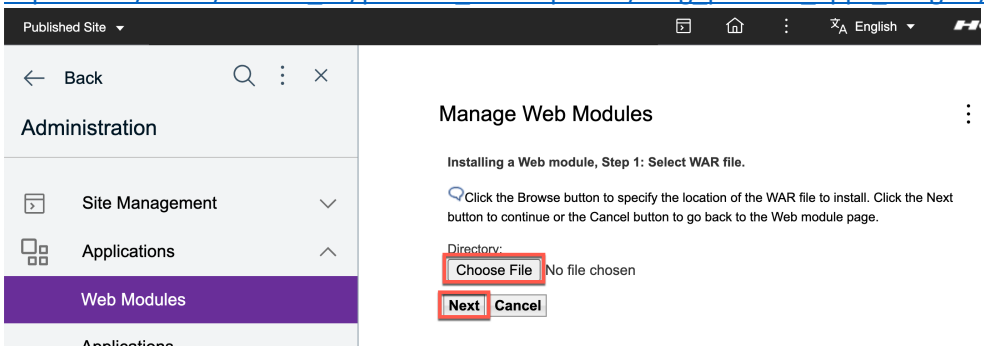
14. If you want to undeploy the portlet, use the **undeploy_Portlet** and this will show this output.



15. Optionally, you may also use the Practitioner Studio Administration to manually deploy your portlet. Open the **Practitioner Studio** – **Administration** – **Applications** – **Web Modules** and click **Install**.



16. Then select the WAR file of your portlet and use **Next** to deploy it. More details in https://opensource.hcltechsw.com/digital-experience/latest/extend_dx/portlets_development/mng_portlets_apps_widgets/adctinsp/.

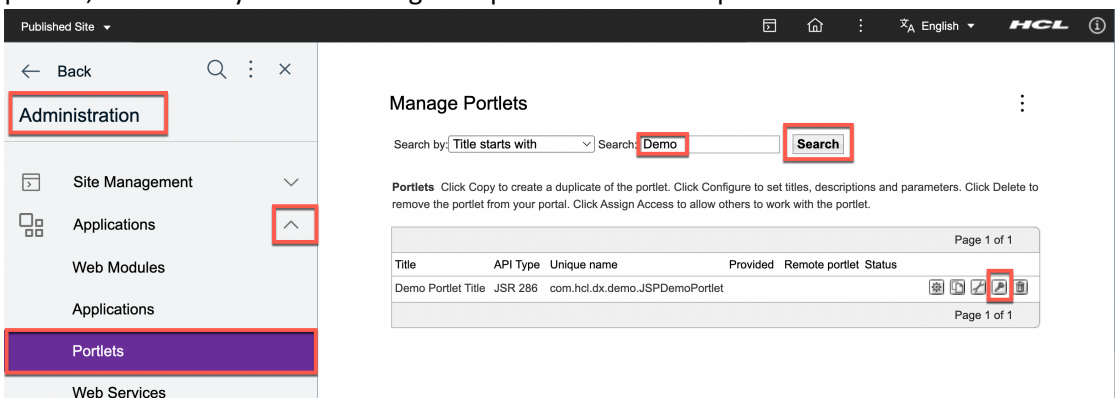


Congratulations! You have learnt how to build a JSP Demo Portlet WAR file (Deployment artifact) and deploy it easily in Visual Studio Code.

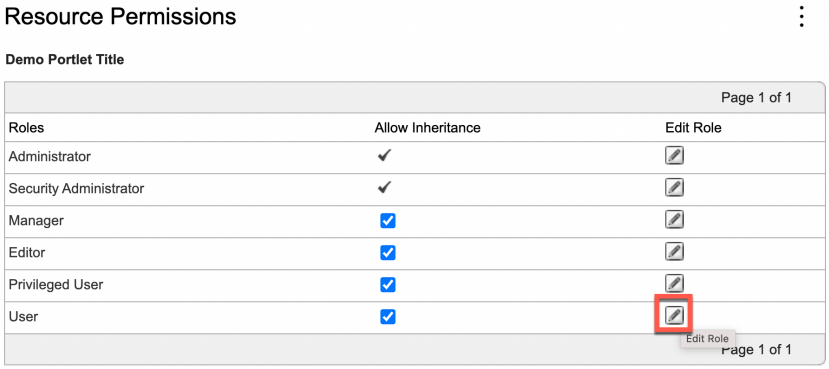
Part 3: Secure on DX

In this part, you will secure your deployed portlet.

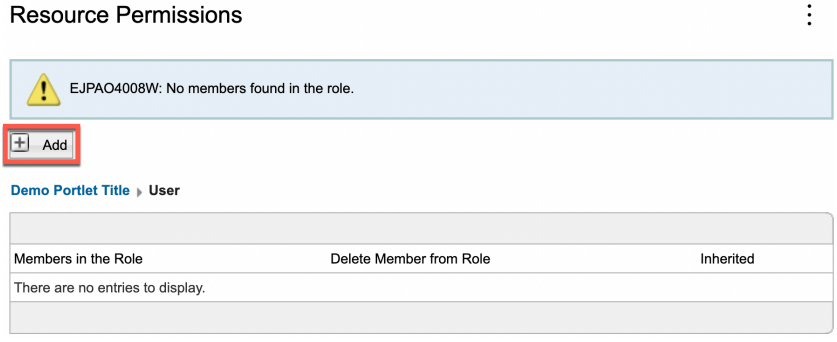
1. Open the **Practitioner Studio** – **Administration** – **Applications** – **Portlets** page and search your new portlet, using the default **Title starts with**. Enter **Demo** and click **Search**. On your portlet, click the key icon to manage the permissions of the portlet.



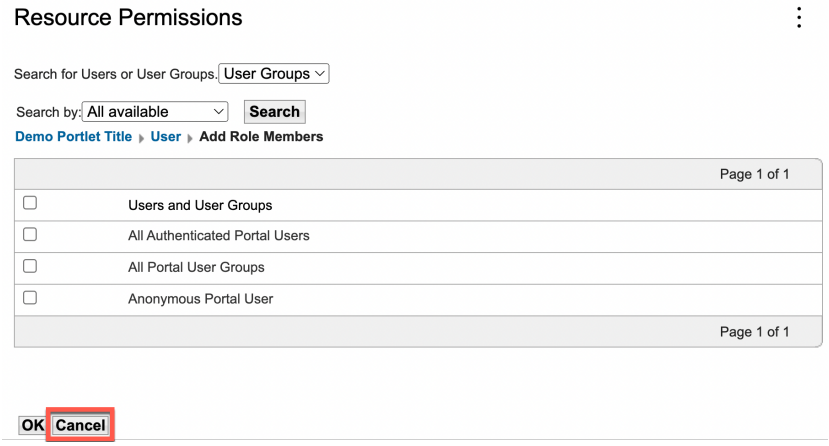
2. You may use different roles for your portlet. They all have Allow Inheritance checked. Check out what user access has been inherited. Click the **Edit Role** icon for **User**.



3. Here you can select users and groups to be added to this role. Click **Add**.



4. Notice that it inherited authenticated portal users and anonymous users. If you wish to change this, you need to uncheck the inheritance above and then add your own specific users and/or groups. Keep it like this for now and click **Cancel**.

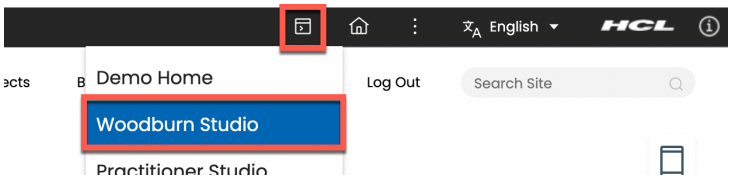


Congratulations! You have learnt how to secure your portlet.

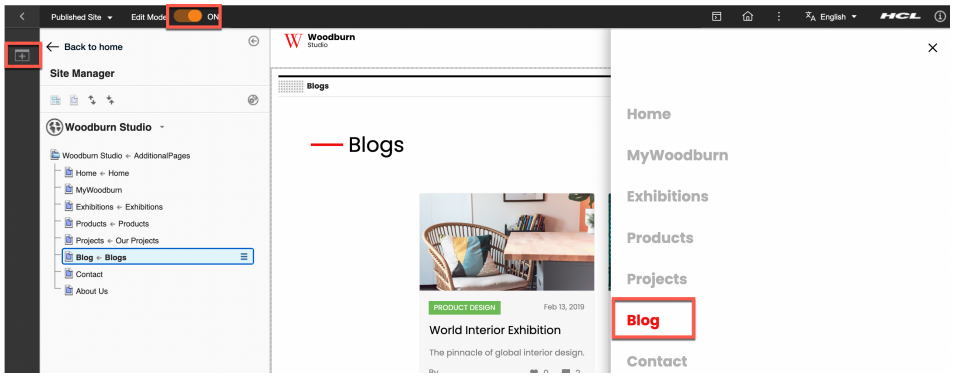
Part 4: Test on DX

In this part, you will add the Portlet to a page and test it.

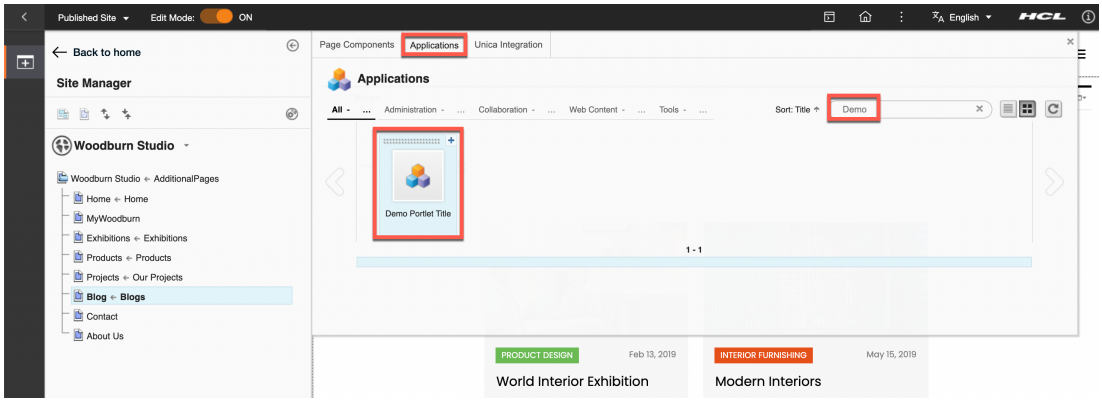
1. First go to a page where you want to add this portlet, this could be the Woodburn Studio – Blogs page. First go to the Woodburn Studio site.



2. Then go to the Blog page, enable the **Edit Mode** and click the Add page components and applications icon  .

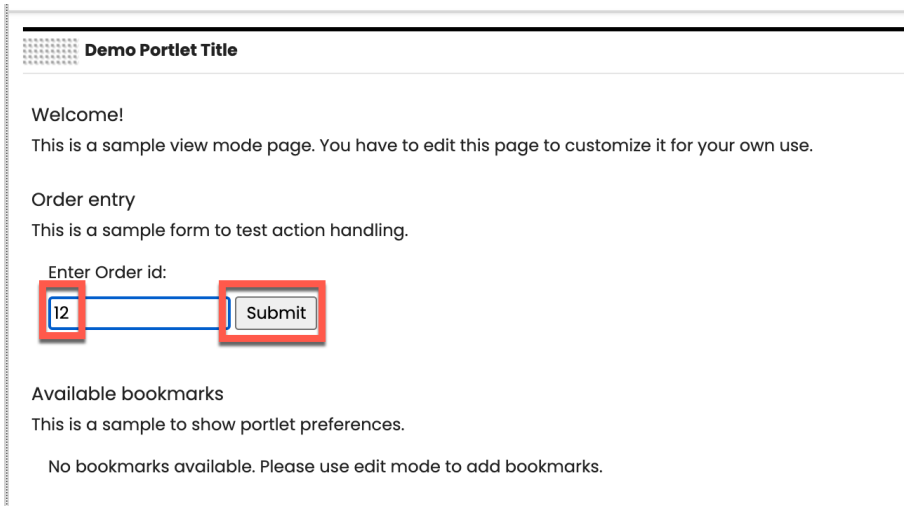


3. The Java portlets are under Applications. Click **Applications**, search for **Demo** and click your Demo Portlet to see its details.

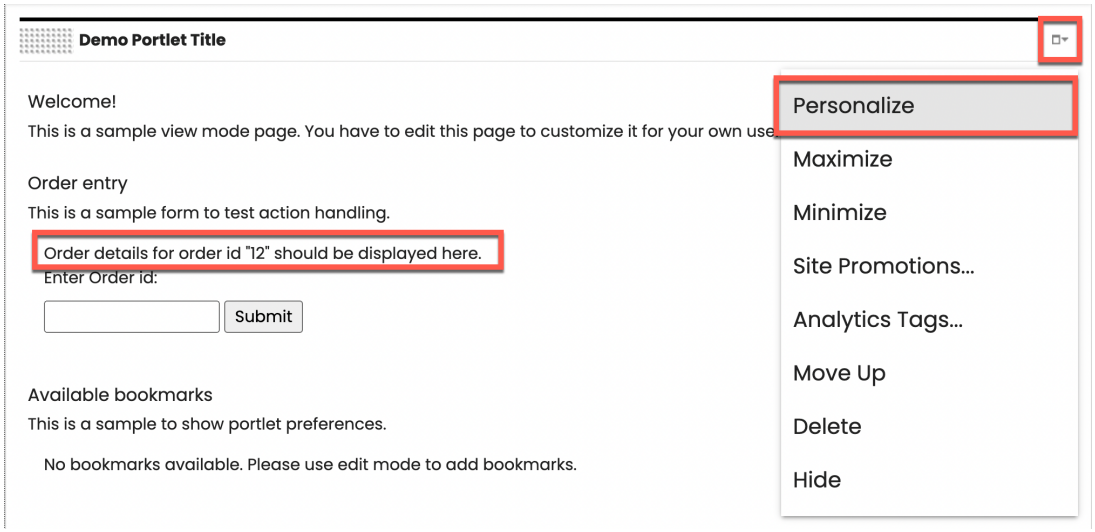


4. Here you see the details. Then add it to your page. Click **Add to Page…** and close the secondary toolbar.

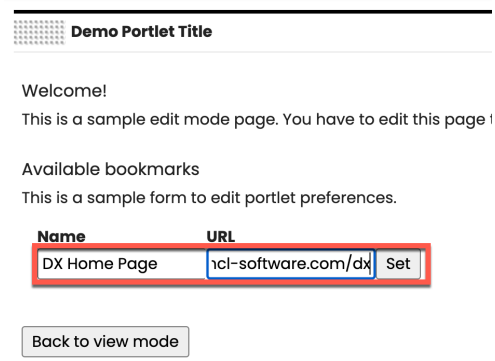
5. Now try it out. Enter and Order id, like 12 and click **Submit**.



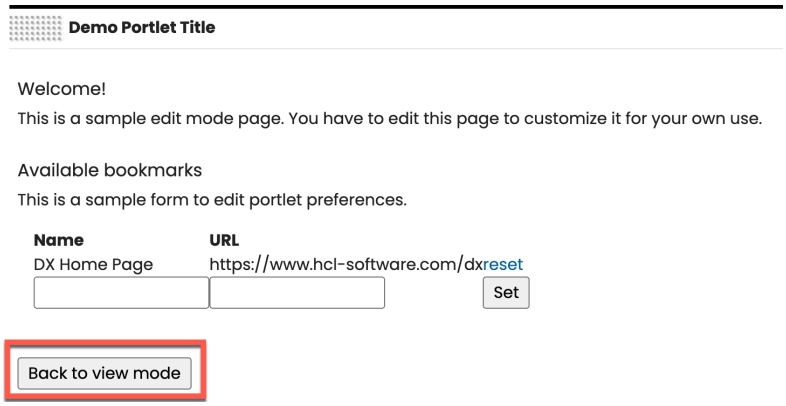
6. You see it has tried to find order details for this id. Then check out the bookmarks. They are managed under the portlet preferences. Open the portlet menu and click **Personalize**.



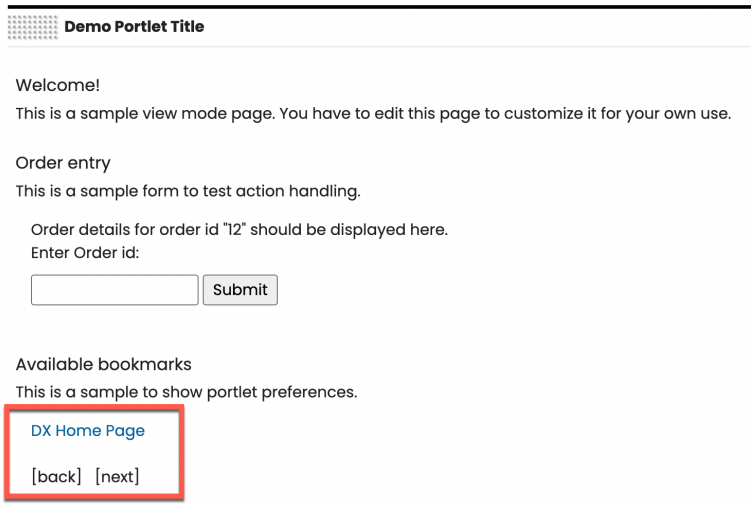
7. Then add a bookmark, e.g. for DX Home Page with URL https://www.hcl-software.com/dx, and click **Set**.



8. And notice, it has been added. Then click **Back to view mode**.



9. And you see your bookmark.

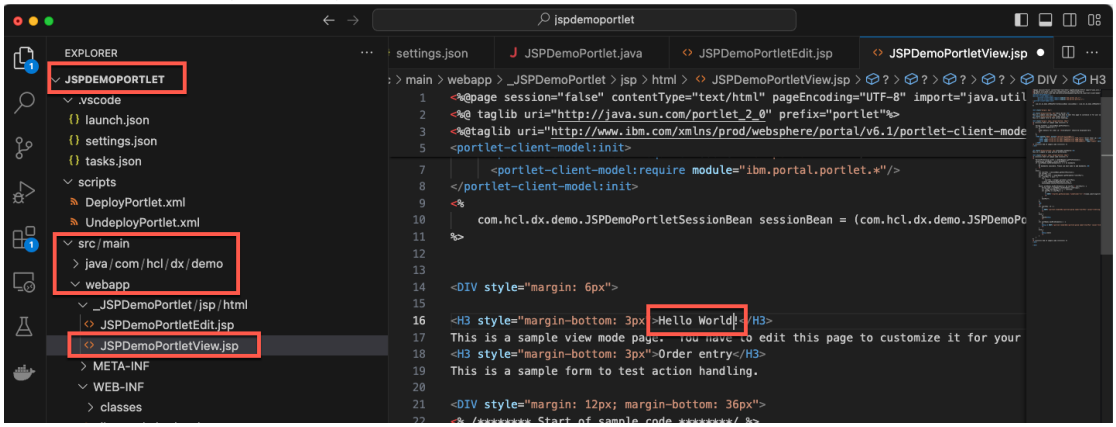


Congratulations! You have now learnt how to add the portlet to a page and test it.

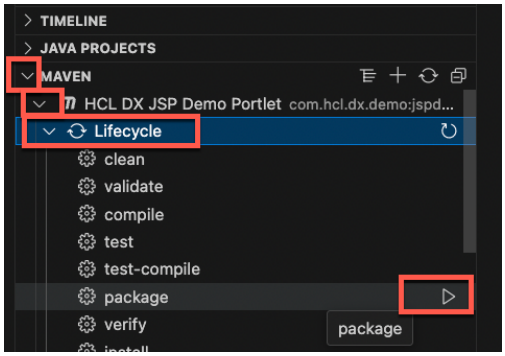
Part 5: Update a JSP Portlet

In this part, you will learn how to make updates to the JSP Portlet, deploy and test it again.

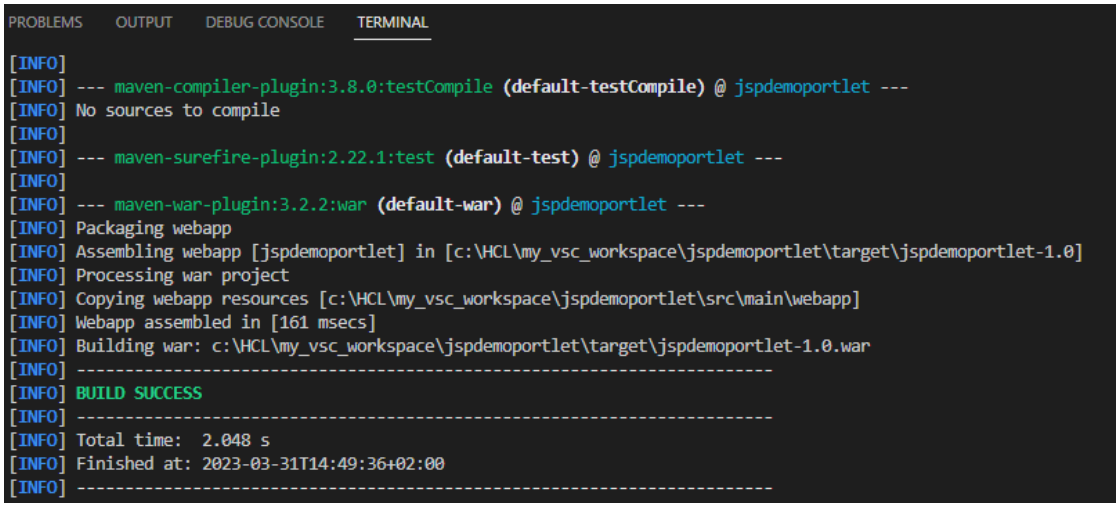
1. First make a change in your JSP Portlet in Visual Studio Code application. For example, change the Welcome! message to **Hello World!** of your portlet. This is managed in the **src\webapp_JSPDemoPortlet\jsp\html\ JSPDemoPortletView.jsp**. Open it and change it in line 16 and save it (CTRL + S).



2. Then repackage it again using Maven, like you did for your first deployment. On the left side expand **MAVEN** - **HCL DX JSP Demo Portlet** - **Lifecycle** and click **package**.



3. The TERMINAL shows the packaging process and should end with BUILD SUCCESS.



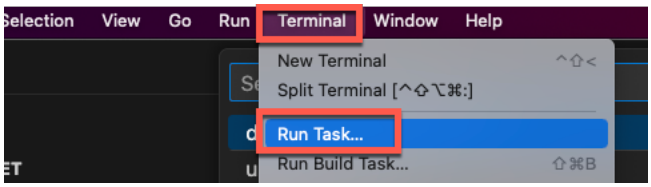
4. Then deploy it again. When a portlet is deployed on DX, a unique ID is specified in the portlet.xml file to ensure that the portlet is unique in the whole DX environment.

```
<?xml version="1.0" encoding="UTF-8"?>
<portlet-app xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd" version="2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd"
id="com.hcl.dx.demo.JSPDemoPortlet.a7f9d0d758">
```

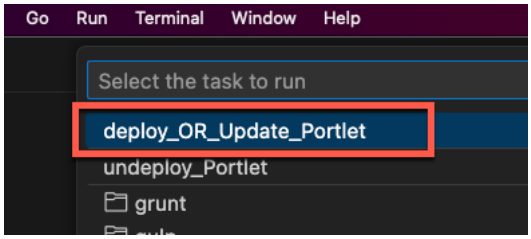
5. If the portlet is updated and the ID will not be changed, another deployment of the same portlet lead updates just its code and you do not have to add the portlet to a page again. In the DeployPortlet.xml file you have seen before, there are already the **action=”update”** commands included to allows you to deploy new and updates of that portlet.

```
scripts > DeployPortlet.xml
<?xml version="1.0" encoding="UTF-8"?>
<request
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="PortalConfig_8.5.0.xsd"
    type="update"
    create-oids="true">
    <portal action="locate">
        <!-- Sample JSR 286 portlet to create or update a portlet-->
        <!-- The uid must match uid attribute of portlet-app in portlet.xml inside the war file appended with "webmod" -->
        <web-app action="update" active="true" domain="rel" uid="com.hcl.dx.demo.JSPDemoPortlet.a7f9d0d758.webmod">
          <!-- Below url path will be replaced with the generated URL by DXClient after uploading the war file to remote server -->
            <url>file:///localhost/HCL/jspdemoportlet-1.0.war</url>
         <!--url>file://localhost/$user_install_root$/PortalServer/deployed/archive/com.hcl.dx.demo.JSPDemoPortlet.webmod/jspdemoportlet-1.0.war</url-->
         <!-- Replace the context root and display value below with your web application context root -->
            <context-root>/wps/JSPDemoPortlet</context-root>
            <display-name>JSPDemoPortlet</display-name>
            <!-- The uid must match uid attribute of concrete-portlet-app in portlet.xml. -->
          <portlet-app action="update" active="true" uid="com.hcl.dx.demo.JSPDemoPortlet.a7f9d0d758">
            <access-control externalized="false" owner="uid=admin,o=defaultWIMFileBasedRealm" private="false"/>
            <!-- The name attribute must match content of portlet-name subtag  of concrete-portlet in portlet.xml. -->
              <portlet action="update" active="true" objectid="JSPDemoPortlet" name="JSPDemoPortlet" uniquename="com.hcl.dx.demo.JSPDemoPortlet" />
          </portlet-app>
        </web-app>
    </portal>
</request>
```

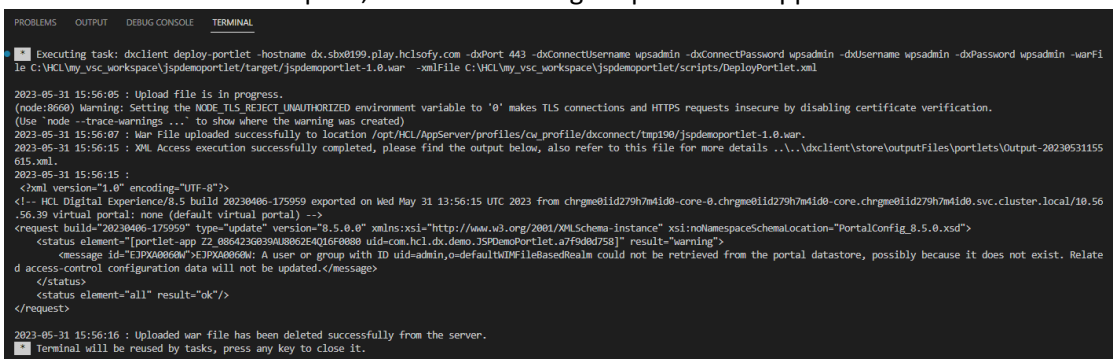
6. Run the deploy or update portlet task again. In the Visual Studio Code menu, click **Terminal** - **Run Task…**



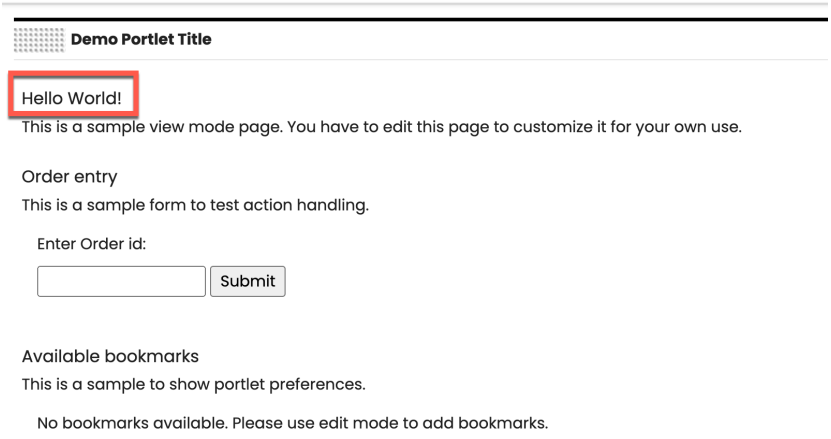
7. And select again **deploy_OR_Update_Portlet** to update the portlet using dxclient.



8. The TERMINAL window opens, and the following output should appear:



9. Reload your test page and notice it now shows Hello World!

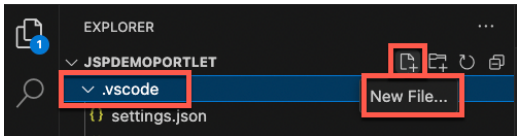


Congratulations! You have successfully updated, deployed, and tested your JSP portlet.

Optional Part 6: Debug a JSP Portlet

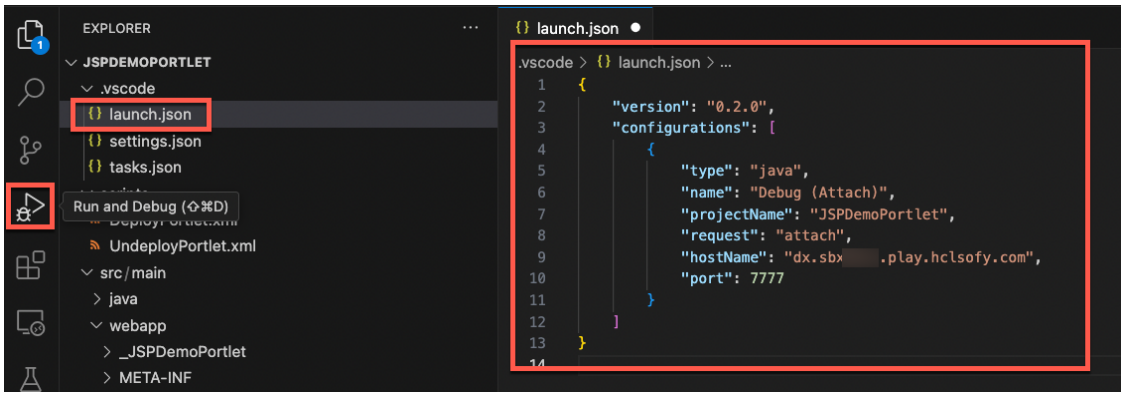
And finally, you will learn how to debug your JSP Portlet. This does not work with a DX server on HCL SoFy, as the port 7777 is not open.

1. You first need to set up debugging. You need to create a launch.json file in the .vscode folder of your project. In VSC, select your **.vscode** folder and click **New File…**

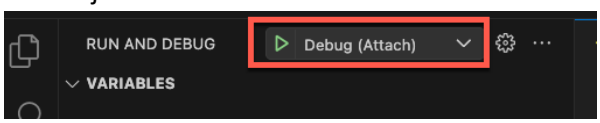


2. Add the following configuration information. Ensure the hostname and port match that of your DX server. Save the changes and click the **Run and Debug** icon.

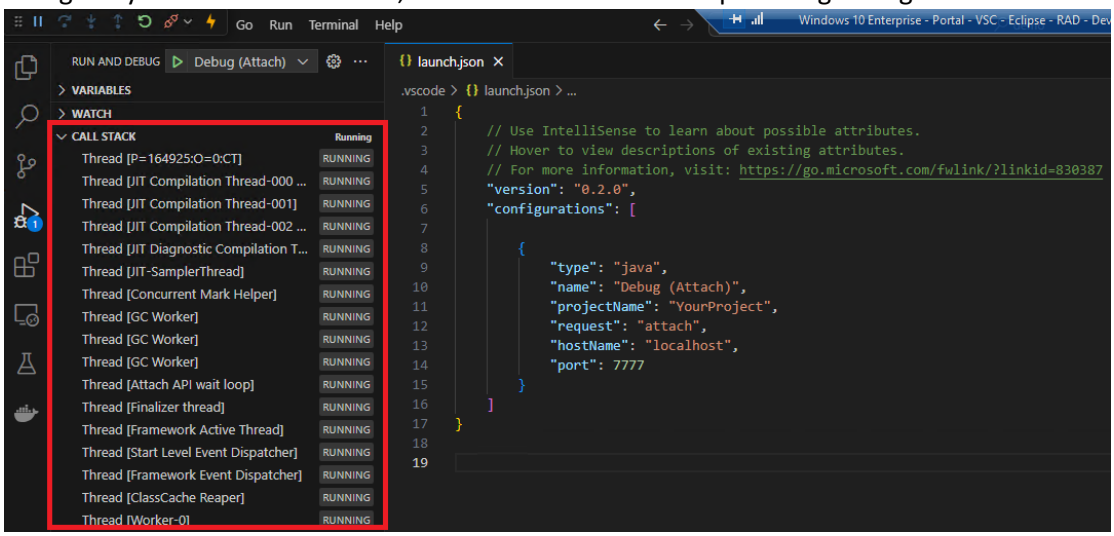
```
{
  "version": "0.2.0",
  "configurations": [

    {
      "type": "java",
      "name": "Debug",
      "projectName": "<Your project name, e.g. JSPDemoPortlet>",
      "request": "attach",
      "hostName": "<Your DX server hostname, e.g. dx.sbx0000.play.hclsofy.com>",
      "port": 7777
    }
  ]
}
```



3. Then start the remote debugging. Click **Debug (Attach)**, or other name you gave it in the launch.json file.

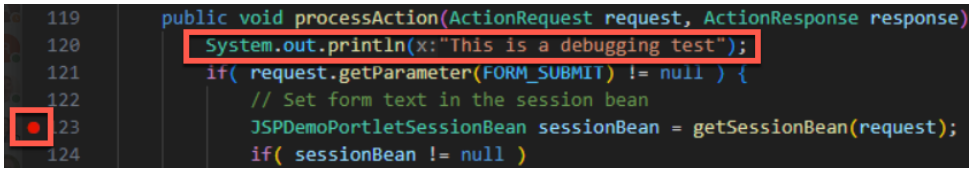


4. This gives you access to variables, a watch list and call stack providing debug information.

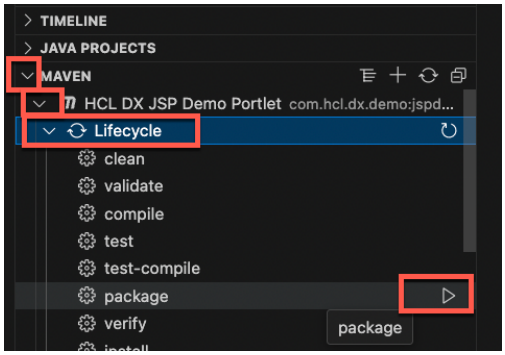


5. Now add a breakpoint and print something out to the SystemOut log file in your portlet. Open src\main\java\com\hcl\dx\demo\JSPDemoPortlet.java and add in line 120 this Java code that writes debug information to the SystemOut.log and left click on the left of line 123 to enable the breakpoint, as shown.

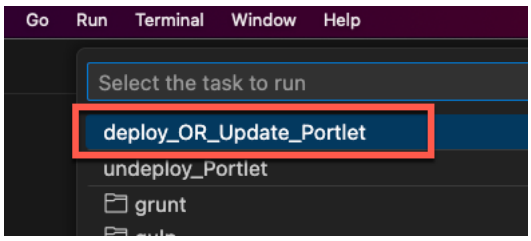
   SystemOut.println("This is a debugging test");



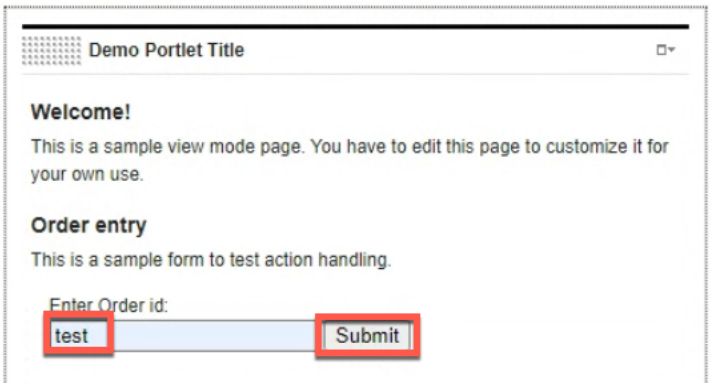
6. Then update the portlet again, using the same way as you did for deploying it. First package again.



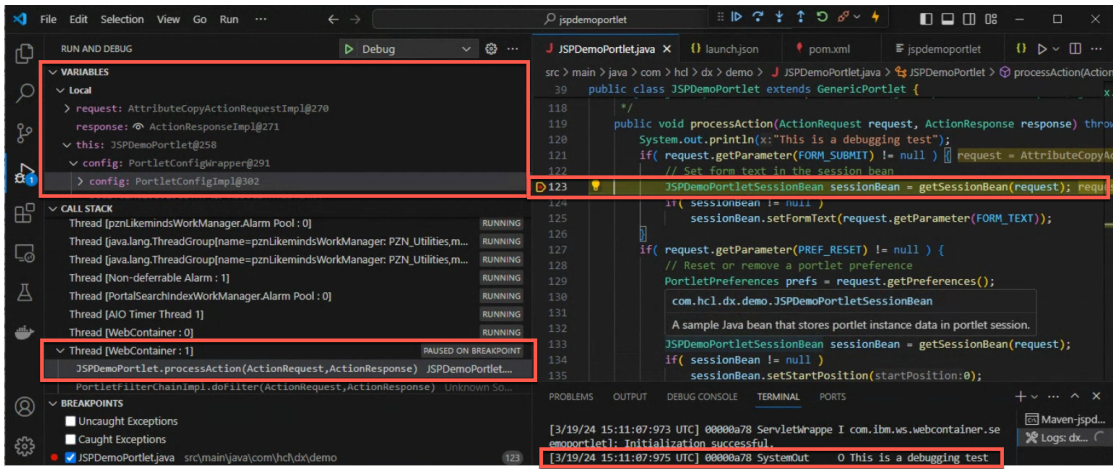
7. And then deploy.



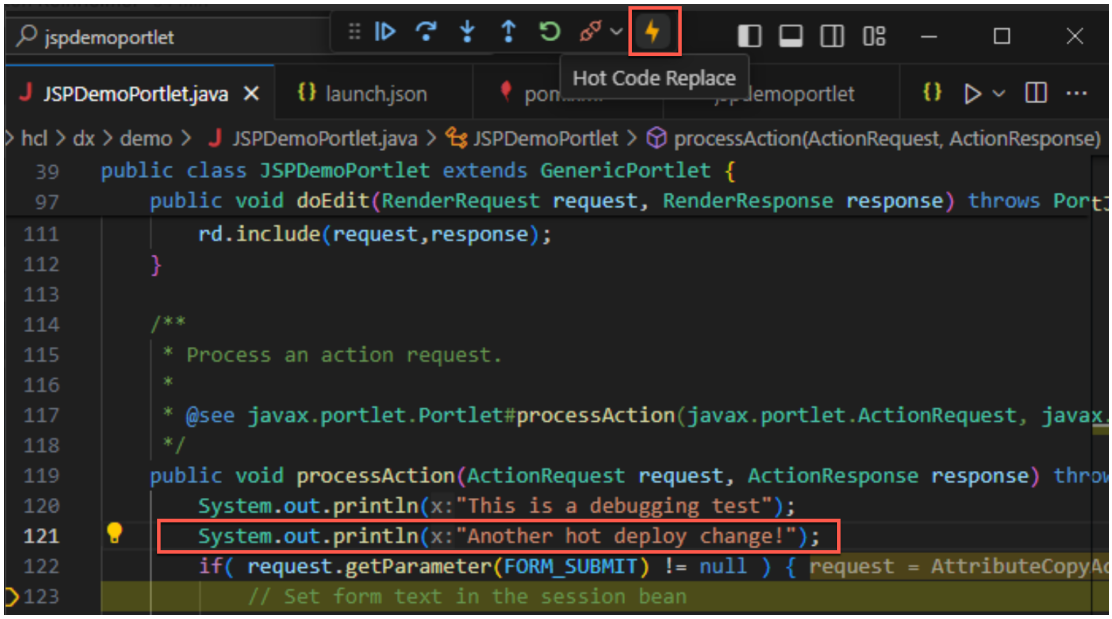
8. Then go to your test page again, enter a test order ID and click **Submit**.



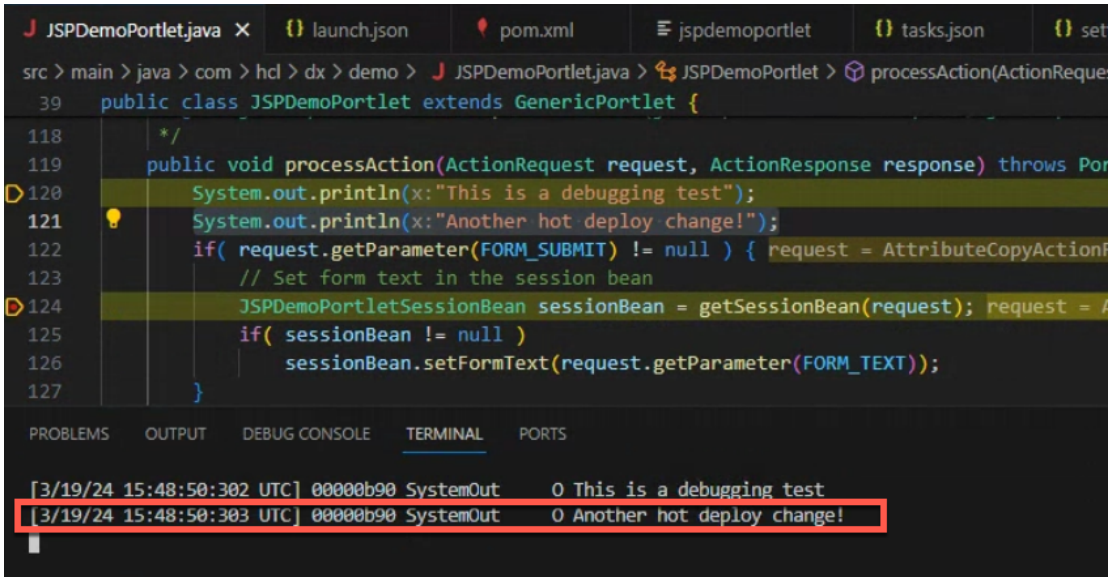
9. Now that the debug mode is enabled and a breakpoint is set, you will notice that your VSC is blinking shortly, and the execution of that code exactly stops at the line of code for which the different kind of variables and thread can be reviewed. The line 123 is now highlighted. And you see your debug print out in the SystemOut.log in your TERMINAL.



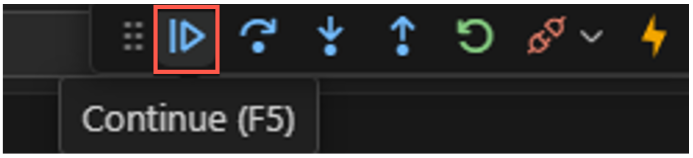
10. When you are in debug mode, the following menu shows allowing you to walk through your code. You have actions to continue, step over, step into, step out, restart, disconnect/stop and hot code replace. Now make change to the code and test this immediately on your portlet. For example, add another print line, as shown.



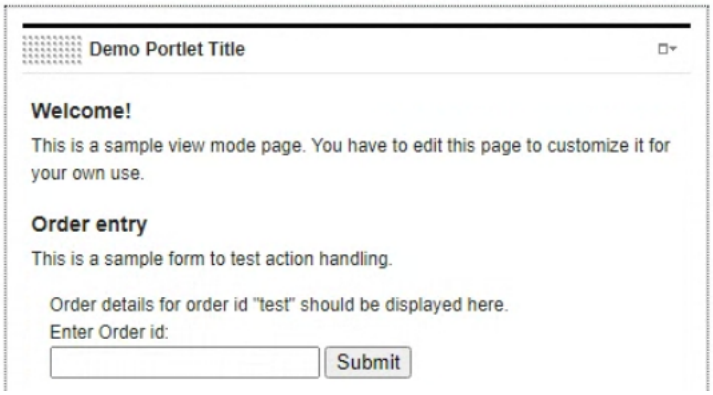
11. You now see that the TERMINAL shows the output of your hot deployed code.



12. Now push to the continue button (first icon or just click to F5), the code will continue to execute to the next breakpoint, or end, and the rest of your portlet should be visible.



13. And your page now gets updated.



Congratulations! You have learnt how to debug your JSP portlet.

Conclusion

Using this lab tutorial, you have learned how to easily develop, deploy, test, update and debug JSP portlets for DX using Visual Studio Code.

Enjoy developing your own Java portlets now.

Resources

Refer to the following resources to learn more:

HCL Digital Experience Home - https://hclsw.co/dx

HCL Digital Experience on HCL SoFy - https://hclsofy.com/

HCL Software - https://hclsw.co/software

HCL Product Support - https://hclsw.co/product-support

HCL DX Product Documentation - https://hclsw.co/dx-product-documentation

HCL DX Support Q&A Forum - https://hclsw.co/dx-support-forum

HCL DX Video Playlist on YouTube - https://hclsw.co/dx-video-playlist

HCL DX Product Ideas - https://hclsw.co/dx-ideas

HCL DX Product Demos - https://hclsw.co/dx-product-demo

HCL DX Did You Know? Videos - https://hclsw.co/dx-dyk-videos

HCL DX GitHub - https://hclsw.co/dx-github

Legal statements

This edition applies to version 9.5, release 216 of HCL Digital Experience and to all subsequent releases and modifications until otherwise indicated in new editions.

When you send information to HCL Technologies Ltd., you grant HCL Technologies Ltd. a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Disclaimers

This report is subject to the HCL Terms of Use (https://www.hcl.com/terms-of-use) and the following disclaimers:

The information contained in this report is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied, including but not limited to the implied warranties of merchantability, non-infringement, and fitness for a particular purpose. In addition, this information is based on HCL's current product plans and strategy, which are subject to change by HCL without notice. HCL shall not be responsible for any direct, indirect, incidental, consequential, special or other damages arising out of the use of, or otherwise related to, this report or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from HCL or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of HCL software.

References in this report to HCL products, programs, or services do not imply that they will be available in all countries in which HCL operates. Product release dates and/or capabilities referenced in this presentation may change at any time at HCL's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. The underlying database used to support these reports is refreshed on a weekly basis. Discrepancies found between reports generated using this web tool and other HCL documentation sources may or may not be attributed to different publish and refresh cycles for this tool and other sources. Nothing contained in this report is intended to, nor shall have the effect of, stating.

or implying that any activities undertaken by you will result in any specific sales, revenue growth, savings or other results. You assume sole responsibility for any results you obtain or decisions you make as a result of this report. Notwithstanding the HCL Terms of Use (https://www.hcl.com/terms-of-use), users of this site are permitted to copy and save the reports generated from this tool for such users own internal business purpose. No other use shall be permitted.

