IBM Software
WebSphere

Lab 2: Discover Liberty

Contents

Lab 2 Discover Liberty

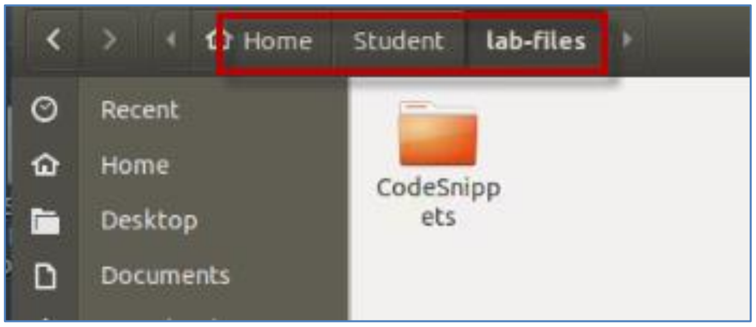
In this lab, you will explore the Liberty server configuration, installing applications onto Liberty, updating the server configurations, and updating an application.

As you explore Liberty, you will first use the WebSphere Developer Tools for Eclipse. Then, you will explore Liberty using the command line. Finally, you will be introduced to mode advanced configurations using jvm.options, bootstrap.properties, Liberty built-in variables, and logging and tracing.

TIP: Liberty is pre-installed on the VM environment provided. **{LAB_HOME}** refers to: **/home/ibmdemo/Student/WLP_21.0.0.3**

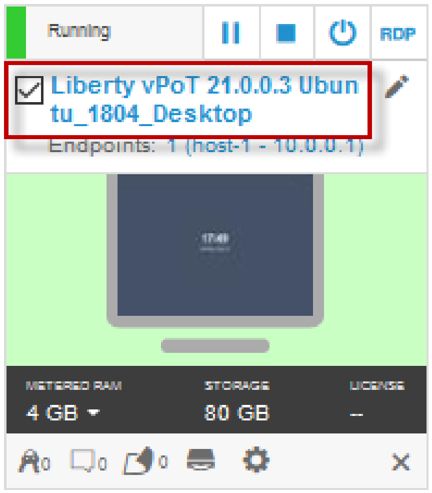
TIP: To reduce typing or copy & past of commands, you can find the related code snippets or commands in the VMWare image in the directory:

/home/ibmdemo/Student/lab-files/CodeSnippets/Bootcamp_Lab2_discover_CodeSnippets.txt



1.1 The lab environment

One (1) Linux VM has been provided for this lab.


Running
RDP
Liberty vPoT 21.0.0.3 Ubuntu_1804_Desktop
Endpoints: 1 (host-1 - 10.0.0.1)
METERED RAM
4 GB
STORAGE
80 GB
LICENSE
--

The **"Liberty vPOT … Desktop"** VM has the following software available:

- Application Project with Liberty
- Maven 3.6.0

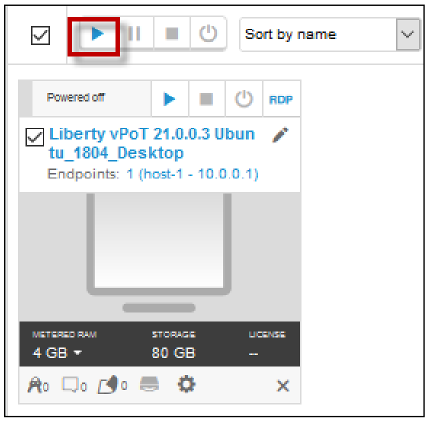
- The login credentials for the **Liberty vPOT … Desktop"** VM are:

  User ID: **ibmdemo**

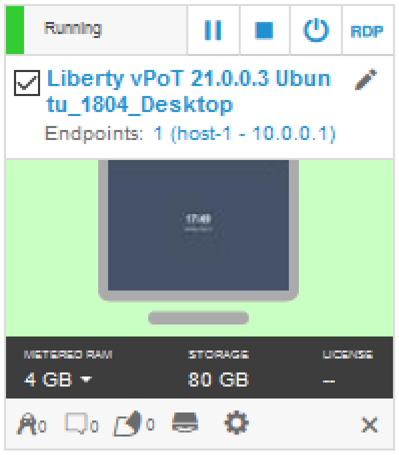
  Password: **passw0rd** (That is a numeric zero in passw0rd)

1.1.1 Login to the "Liberty vPOT … Desktop" VM and Get Started

__1. If the VM is **<u>not</u>** already started, start it by clicking the **Play** button.

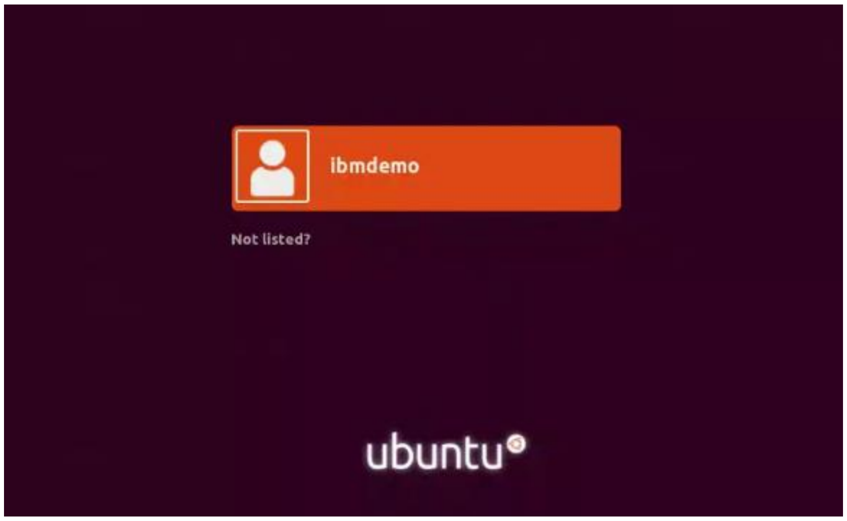

Sort by name
Powered off
RDP
Liberty vPoT 21.0.0.3 Ubuntu_1804_Desktop
Endpoints: 1 (host-1 - 10.0.0.1)
METERED RAM
4 GB
STORAGE
80 GB
LICENSE
--

__2. After the VM is started, click the **"Liberty vPOT … Desktop"** VM icon to access it.


Running
RDP
Liberty vPoT 21.0.0.3 Ubuntu_1804_Desktop
Endpoints: 1 (host-1 - 10.0.0.1)
METERED RAM
4 GB
STORAGE
80 GB
LICENSE
--

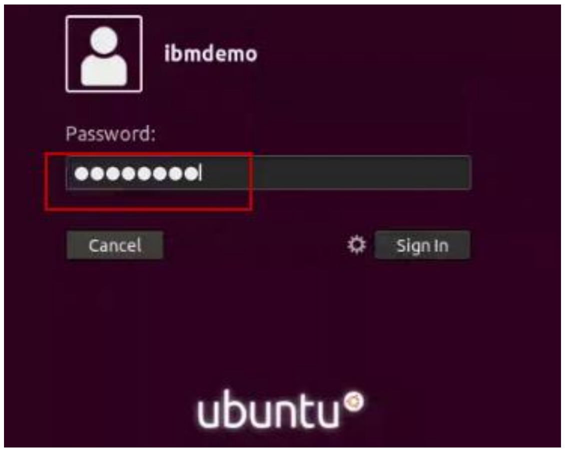
__3. Login with **ibmdemo** ID.

__a. Click on the "**ibmdemo**" icon on the Ubuntu screen.



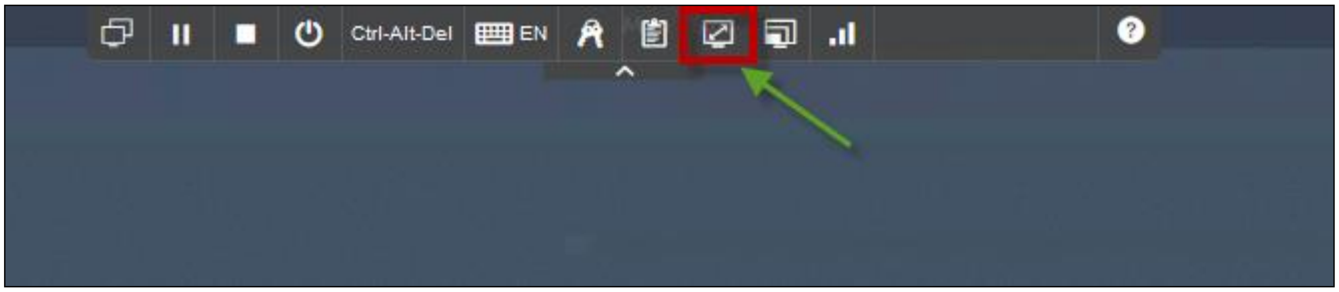
__b. When prompted for the password for “ibmdemo” user, enter “passw0rd” as the password:

Password: **`passw0rd`** (lowercase with a zero instead of the o)



__4. Resize the Skytap environment window for a larger viewing area while doing the lab.

From the Skytap menu bar, click on the “**Fit to Size**” icon. This will enlarge the viewing area to fit the size of your browser window.

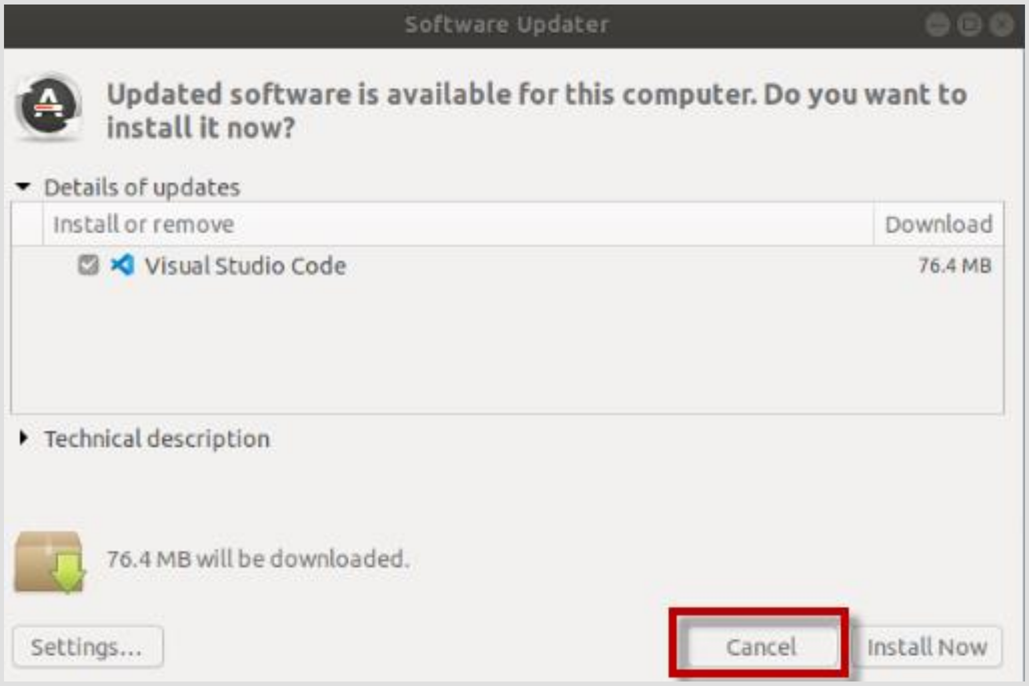


Important:

Click CANCEL…. If, at any time during the lab, you get a pop-up asking to install updated software onto the Ubuntu VM.

The one we experience is an update available for VS Code.

CLICK CANCEL!



Software Updater

Updated software is available for this computer. Do you want to install it now?

Details of updates

Install or remove	Download
Visual Studio Code	76.4 MB

Technical description

76.4 MB will be downloaded.

2 Explore Liberty via WDT

Note: proceed directly to section 3 if you only want to use the command line.

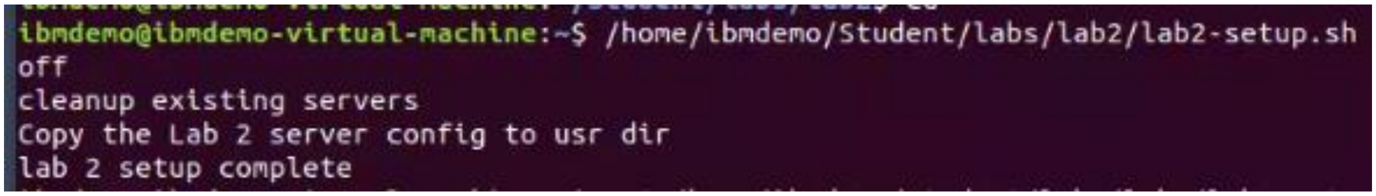
2.1 Setup Liberty for the lab

__1. Copy the Liberty server configuration, used in this lab, to the Liberty usr/servers directory.

__a. Open a new Terminal window

__b. Run the script to setup the Liberty server configuration used in this lab.

```
/home/ibmdemo/Student/labs/lab2/lab2-setup.sh
```



```
ibmdemo@ibmdemo-virtual-machine:~$ /home/ibmdemo/Student/labs/lab2/lab2-setup.sh
off
cleanup existing servers
Copy the Lab 2 server config to usr dir
lab 2 setup complete
```

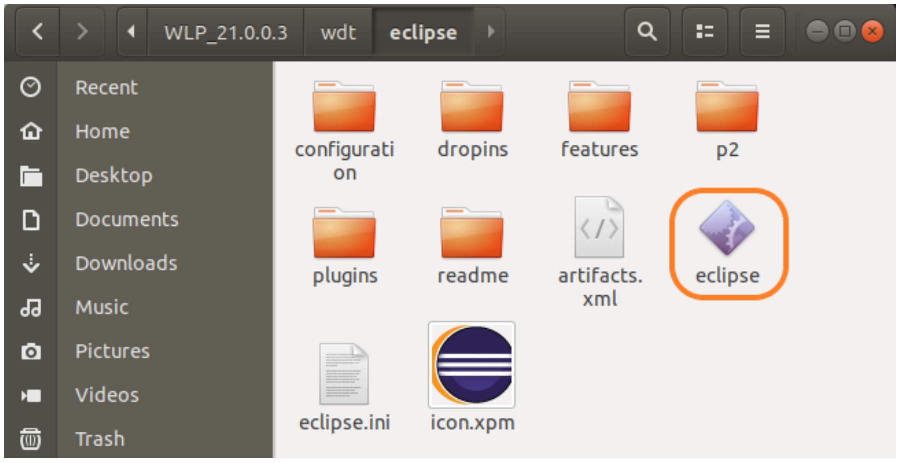
2.2 Ensure eclipse is started and you are in the correct workspace

__1. If Eclipse is not already started, start it now

__a. Use the File Explorer to navigate to the directory

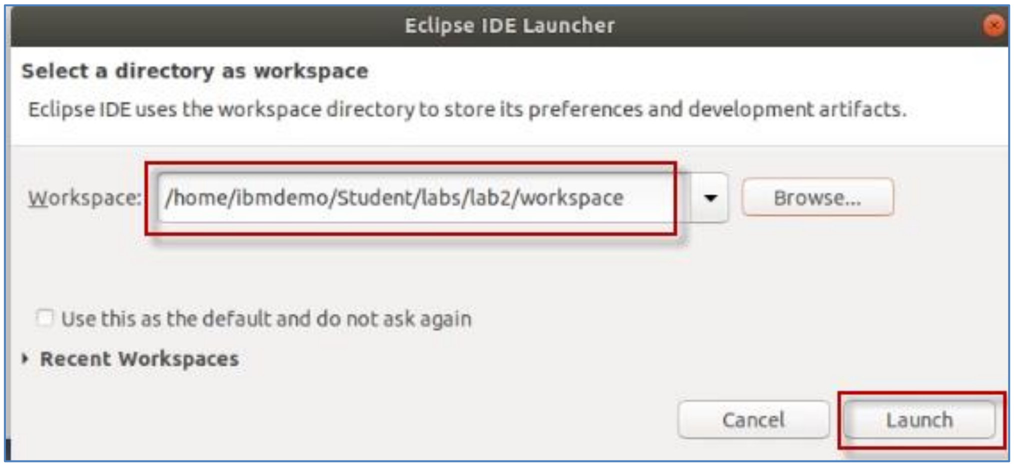
```
Home > Student > WLP_21.0.0.3 > wdt > eclipse
```

__b. Double-click on the **eclipse** executable to start Eclipse.



__c. When the Eclipse launcher prompts you to select a workspace, enter the following directory. Then click the **Launch** button.

```
/home/ibmdemo/Student/labs/lab2/workspace
```


Eclipse IDE Launcher
Select a directory as workspace
Eclipse IDE uses the workspace directory to store its preferences and development artifacts.
Workspace:
Browse...
Use this as the default and do not ask again
Recent Workspaces
Cancel
Launch

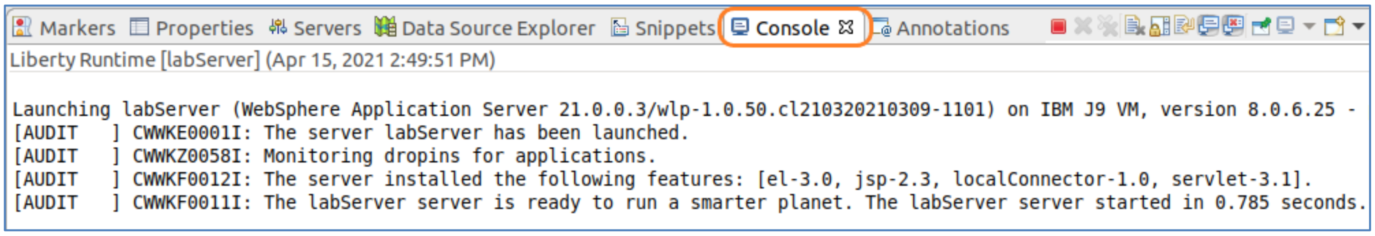
Workspace: /home/ibmdemo/Student/labs/lab2/workspace

2.3 Exploring the Liberty Server

__1. Start the server in eclipse.

__a. From the **Servers** view, select your **labServer** instance and click the **Start the server** button (). Alternatively, you can also right-click the server name and choose the **Start** option from the context menu.

__b. Switch to the **Console** view if necessary. Look at the messages to see how fast your server starts!

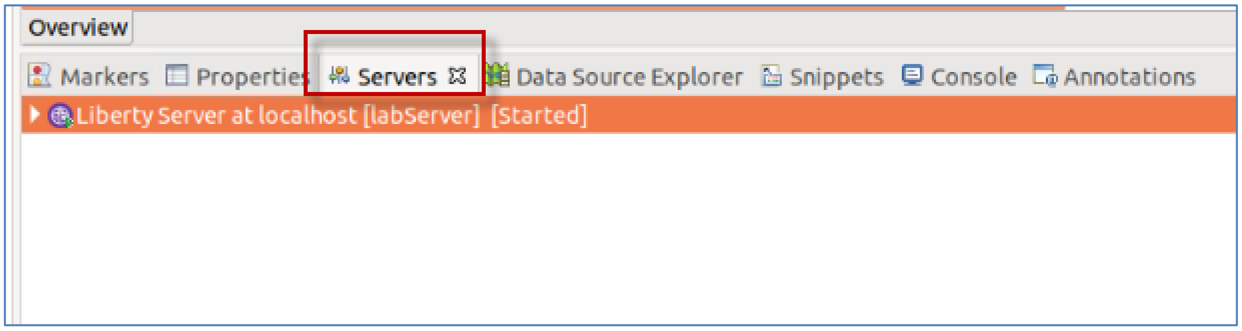


```
Liberty Runtime [labServer] (Apr 15, 2021 2:49:51 PM)

Launching labServer (WebSphere Application Server 21.0.0.3/wlp-1.0.50.cl210320210309-1101) on IBM J9 VM, version 8.0.6.25 -
[AUDIT   ] CWWKE0001I: The server labServer has been launched.
[AUDIT   ] CWWKZ0058I: Monitoring dropins for applications.
[AUDIT   ] CWWKF0012I: The server installed the following features: [el-3.0, jsp-2.3, localConnector-1.0, servlet-3.1].
[AUDIT   ] CWWKF0011I: The labServer server is ready to run a smarter planet. The labServer server started in 0.785 seconds.
```

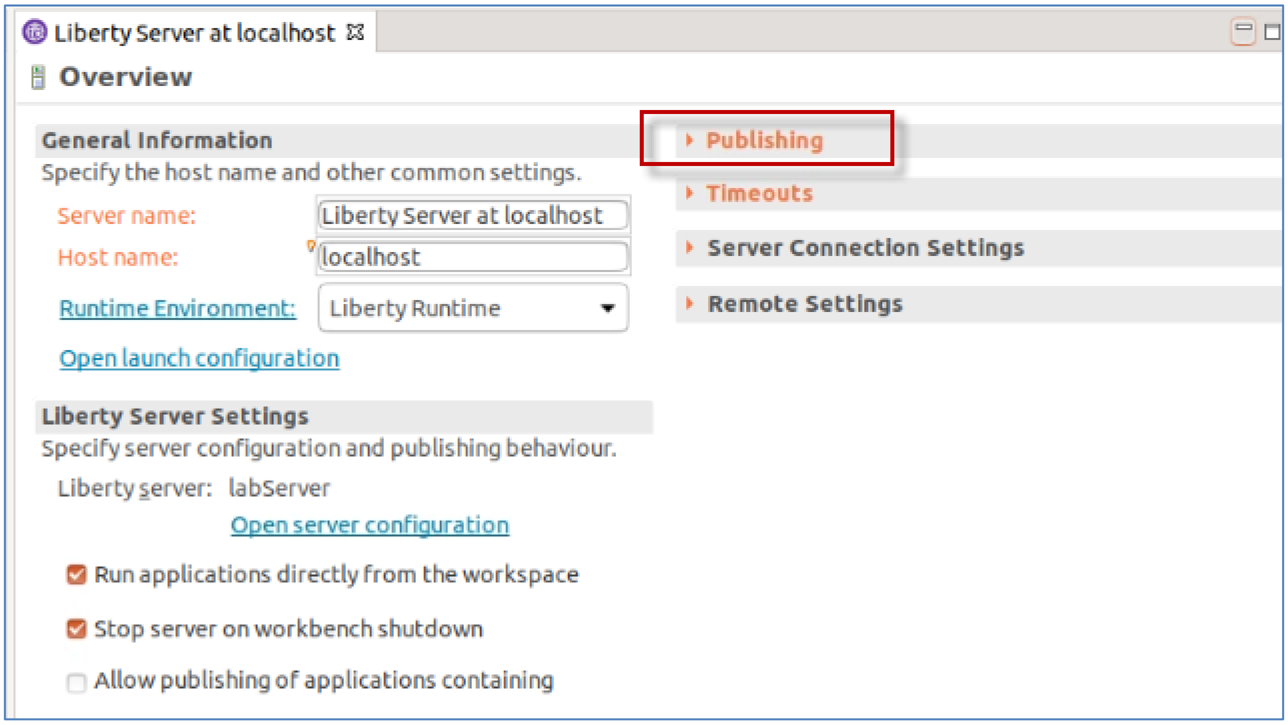
2.3.1 Explore using the WDT to make changes to the lab server configuration

__2. Open the Lab Server configuration using the WDT server configuration editor

__a. In the **Servers** view, double-click on your **labServer** server to open the server **Overview** .



__b. Expand the **Publishing** section and notice that the server is set to automatically detect and publish changes. Keep this default setting.



Liberty Server at localhost

Overview

General Information

Specify the host name and other common settings.

Server name: Liberty Server at localhost

Host name: localhost

Runtime Environment: Liberty Runtime

Open launch configuration

Liberty Server Settings

Specify server configuration and publishing behaviour.

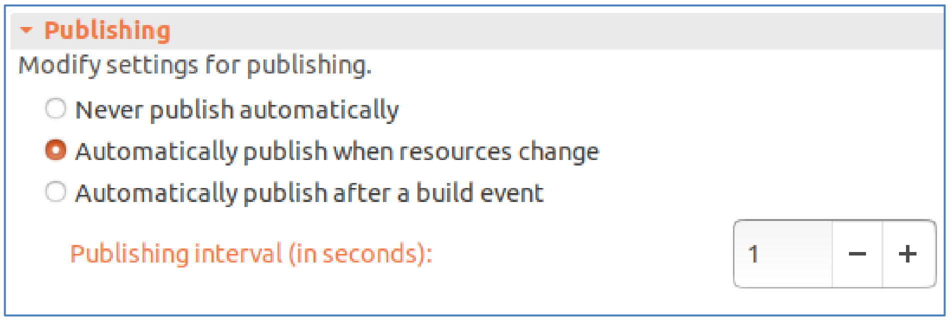
Liberty server: labServer

Open server configuration

[✓] Run applications directly from the workspace

[✓] Stop server on workbench shutdown

[ ] Allow publishing of applications containing



Publishing

Modify settings for publishing.

- ○ Never publish automatically
- ● Automatically publish when resources change
- ○ Automatically publish after a build event

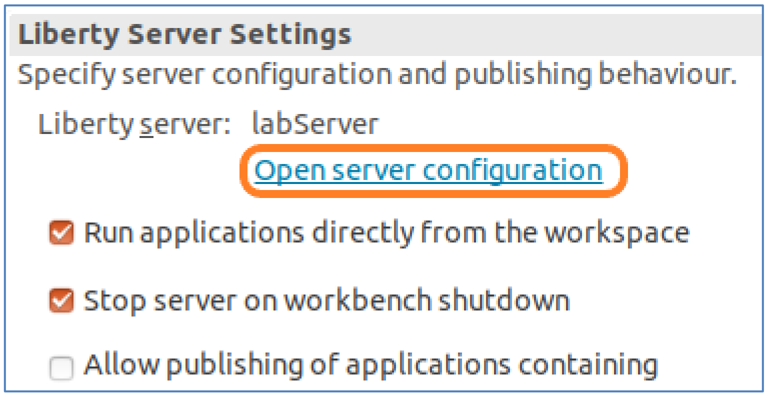
Publishing interval (in seconds): 1

__3. In this exercise, you will be deploying a simple servlet application, so try enabling the servlet feature on this server.

__a. On the **Overview** page, locate the **Liberty Server Settings** section

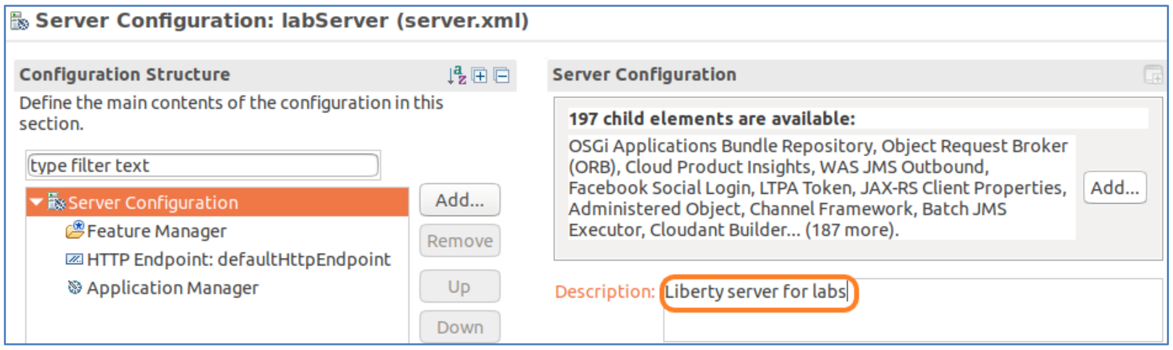
__b. Click the **Open server configuration** link to open the server.xml editor.

Note: To better see the content in Eclipse, you may need to resize the Eclipse views, or increase the size of the Eclipse desktop app


Liberty Server Settings
Specify server configuration and publishing behaviour.
Liberty server: labServer
Open server configuration
Run applications directly from the workspace
Stop server on workbench shutdown
Allow publishing of applications containing

__c. If the **Source** view of the **server,xml** file is displayed, switch to the **Design** view in the server configuration editor, by clicking on the “Design” tab in the editor pane

__d. Start by providing a meaningful description for your server, such as “Liberty server for labs”.


Server Configuration: labServer (server.xml)

Configuration Structure

Define the main contents of the configuration in this section.

Server Configuration

197 child elements are available:

OSGi Applications Bundle Repository, Object Request Broker (ORB), Cloud Product Insights, WAS JMS Outbound, Facebook Social Login, LTPA Token, JAX-RS Client Properties, Administered Object, Channel Framework, Batch JMS Executor, Cloudant Builder... (187 more).

Description: Liberty server for labs

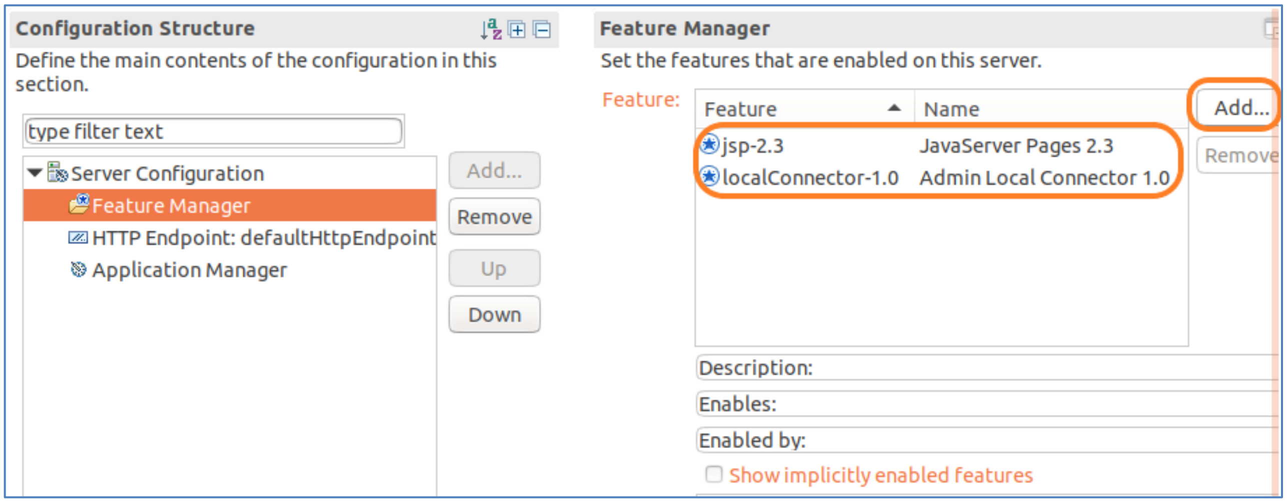
__e. To add a feature, such as servlet-4.0, go back in the **Configuration Structure** area, and exand it , if necessary, and note the **Feature Manager** configuration entry.

In this lab, the Feature Manager has already been added to the configuration

__f. Review the Feature Manager settings by clicking on the “**Feature Manager**” to view the list of features already configured.

Notice that the **jsp-2.3** and the **loalConnector-1.0** features have already been added to the server configuration.

__g. Click the **Add** button to add a new feature “servlet-4.0”

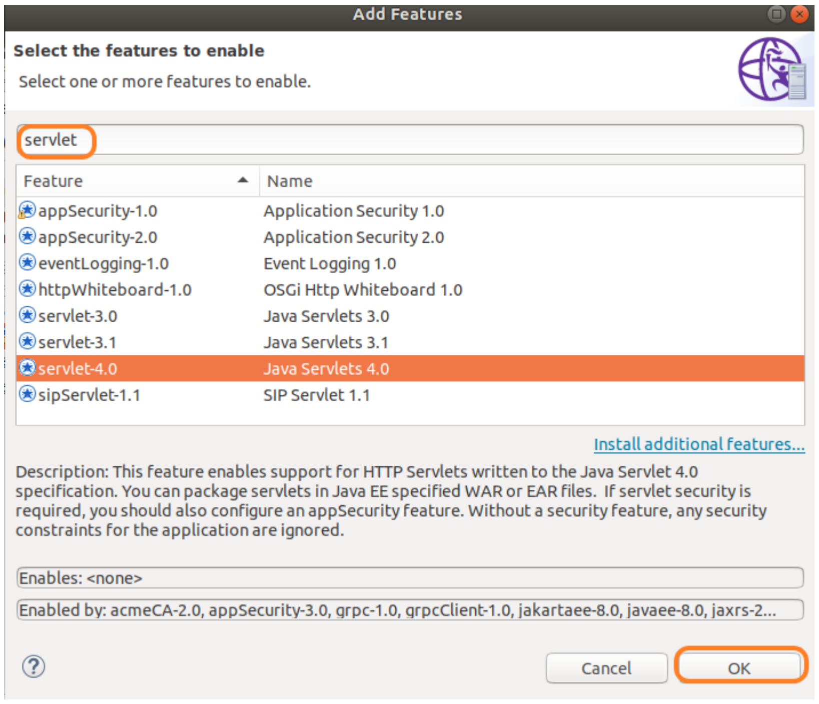

Configuration Structure
Define the main contents of the configuration in this section.
type filter text
Server Configuration
Feature Manager
HTTP Endpoint: defaultHttpEndpoint
Application Manager
Add... Remove Up Down

Feature Manager
Set the features that are enabled on this server.
Feature:

Feature	Name
jsp-2.3	JavaServer Pages 2.3
localConnector-1.0	Admin Local Connector 1.0

Add... Remove
Description:
Enables:
Enabled by:
Show implicitly enabled features

__h. In the pop-up, type **servlet** to filter to servlet related features. Then select **servlet-4.0.** Click **OK**


Add Features
Select the features to enable
Select one or more features to enable.
servlet
Install additional features...
Cancel
OK

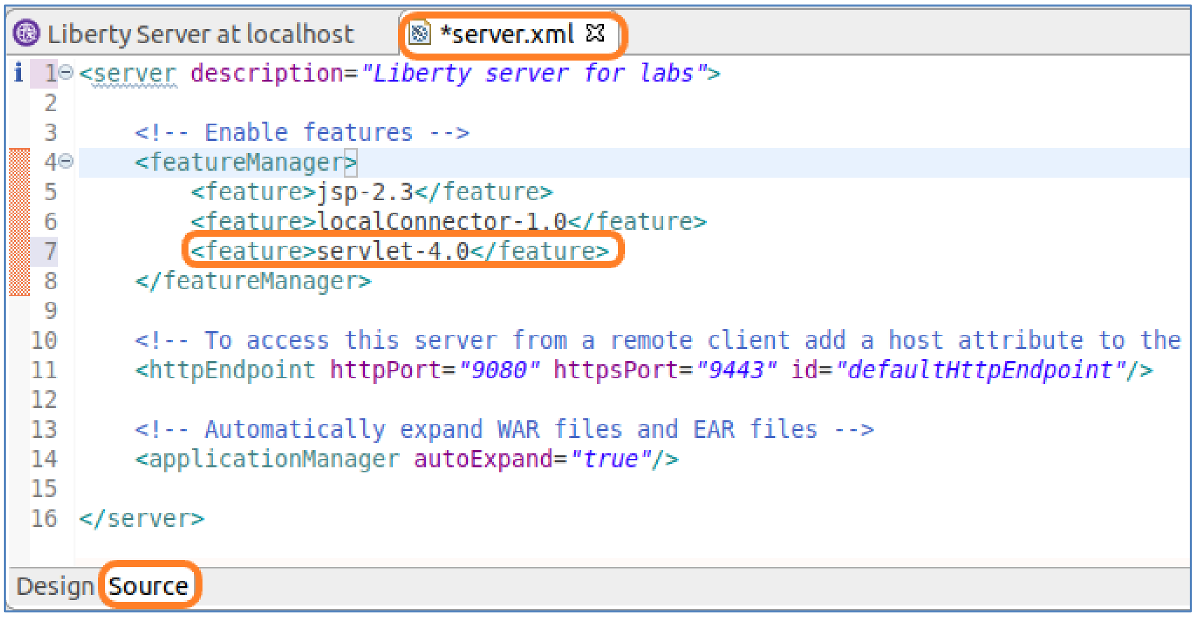
Feature	Name
appSecurity-1.0	Application Security 1.0
appSecurity-2.0	Application Security 2.0
eventLogging-1.0	Event Logging 1.0
httpWhiteboard-1.0	OSGi Http Whiteboard 1.0
servlet-3.0	Java Servlets 3.0
servlet-3.1	Java Servlets 3.1
servlet-4.0	Java Servlets 4.0
sipServlet-1.1	SIP Servlet 1.1

Description: This feature enables support for HTTP Servlets written to the Java Servlet 4.0 specification. You can package servlets in Java EE specified WAR or EAR files. If servlet security is required, you should also configure an appSecurity feature. Without a security feature, any security constraints for the application are ignored.

Enables: <none>

Enabled by: acmeCA-2.0, appSecurity-3.0, grpc-1.0, grpcClient-1.0, jakartaee-8.0, javaee-8.0, jaxrs-2...

__i. In the **server.xml** editor, switch to the **Source** tab at the bottom to see the XML source for this configuration file. You will see that a new **featureManager** element has been updated to contain the **servlet-4.0** feature.



```
<server description="Liberty server for labs">

    <!-- Enable features -->
    <featureManager>
        <feature>jsp-2.3</feature>
        <feature>localConnector-1.0</feature>
        <feature>servlet-4.0</feature>
    </featureManager>

    <!-- To access this server from a remote client add a host attribute to the
    <httpEndpoint httpPort="9080" httpsPort="9443" id="defaultHttpEndpoint"/>

    <!-- Automatically expand WAR files and EAR files -->
    <applicationManager autoExpand="true"/>

</server>
```

__j. Now you have a server that is configured to use the **servlet-4.0** feature.

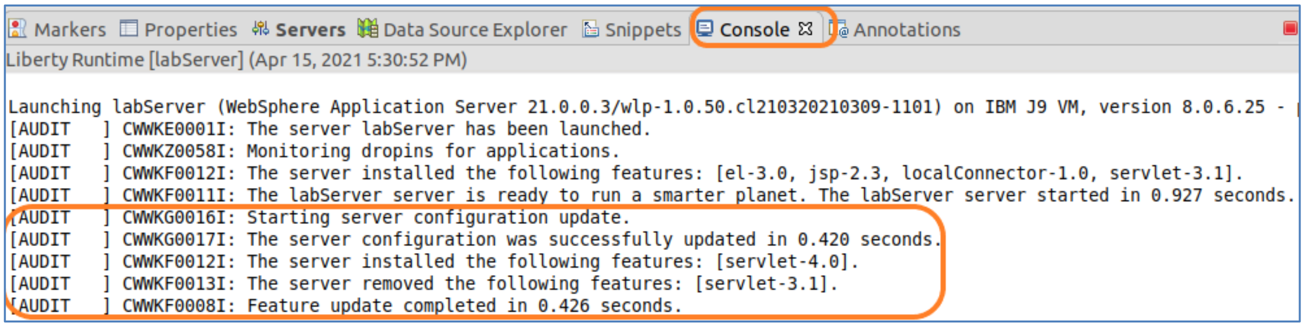
__k. Click the **Save** button (💾) to save your changes (or use CTRL+S)

TIP:

The Sample1 application is a Java Enterprise application that includes a simple **Java Servlet class**. For the Liberty Server to run the Sample1 application, the servlet feature must be configured in the server.xml file.

__4. Switch to the **Console** view, located at the bottom of the workbench and review the latest messages. These messages are showing that your Liberty server automatically detected the configuration update, processed the feature that you enabled, and is now listening for incoming requests.

You will notice that the server configuration was automatically updated, and the feature update was completed very quickly. In this example, it was less than one second.



```
Launching labServer (WebSphere Application Server 21.0.0.3/wlp-1.0.50.cl210320210309-1101) on IBM J9 VM, version 8.0.6.25 -
[AUDIT   ] CWWKE0001I: The server labServer has been launched.
[AUDIT   ] CWWKZ0058I: Monitoring dropins for applications.
[AUDIT   ] CWWKF0012I: The server installed the following features: [el-3.0, jsp-2.3, localConnector-1.0, servlet-3.1].
[AUDIT   ] CWWKF0011I: The labServer server is ready to run a smarter planet. The labServer server started in 0.927 seconds.
[AUDIT   ] CWWKG0016I: Starting server configuration update.
[AUDIT   ] CWWKG0017I: The server configuration was successfully updated in 0.420 seconds.
[AUDIT   ] CWWKF0012I: The server installed the following features: [servlet-4.0].
[AUDIT   ] CWWKF0013I: The server removed the following features: [servlet-3.1].
[AUDIT   ] CWWKF0008I: Feature update completed in 0.426 seconds.
```

__5. **Close** the server.xml in the editor pane.

__6. **Close** the **Liberty Server Overview** page in Eclipse

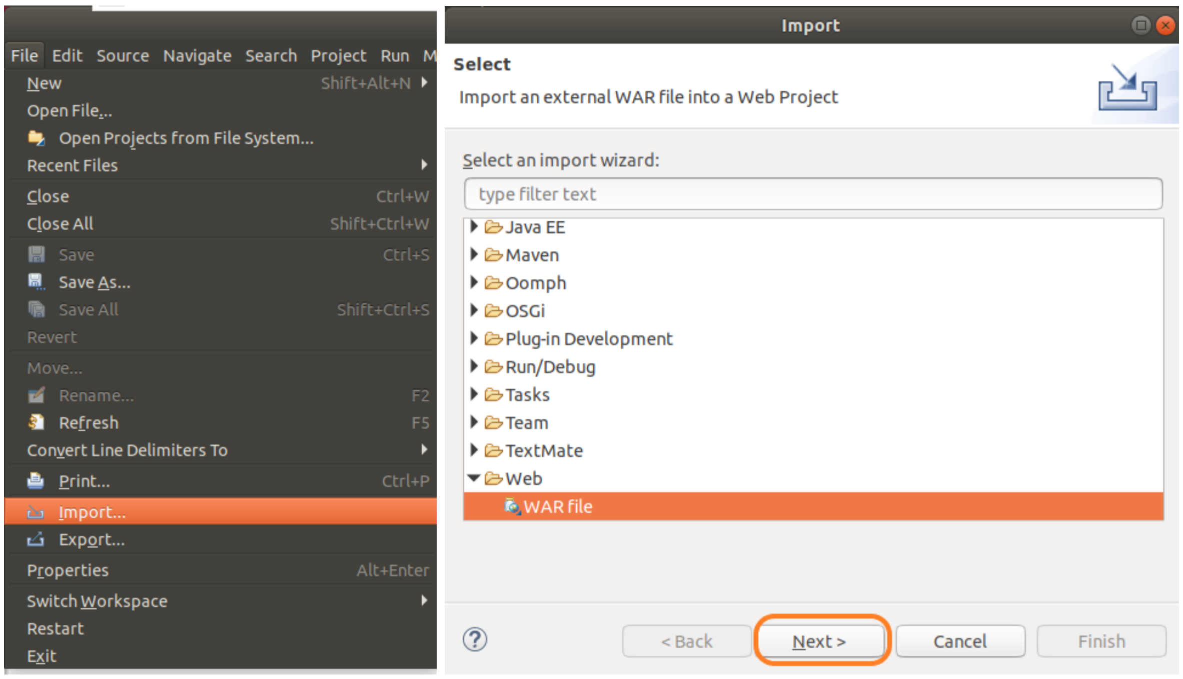
Now you are ready to start working with a sample application that uses the Servlet feature.

2.4 Deploying a sample application to Liberty

2.4.1 Import a sample application into Eclipse

__1. A simple servlet WAR file has been provided for this exercise; import it into your workbench.

__a. In Eclipse, go to **File > Import**. Expand the **Web** section, then select **WAR file**. Click **Next**.



__b. In the **WAR file** field, select **Browse**. Navigate to:

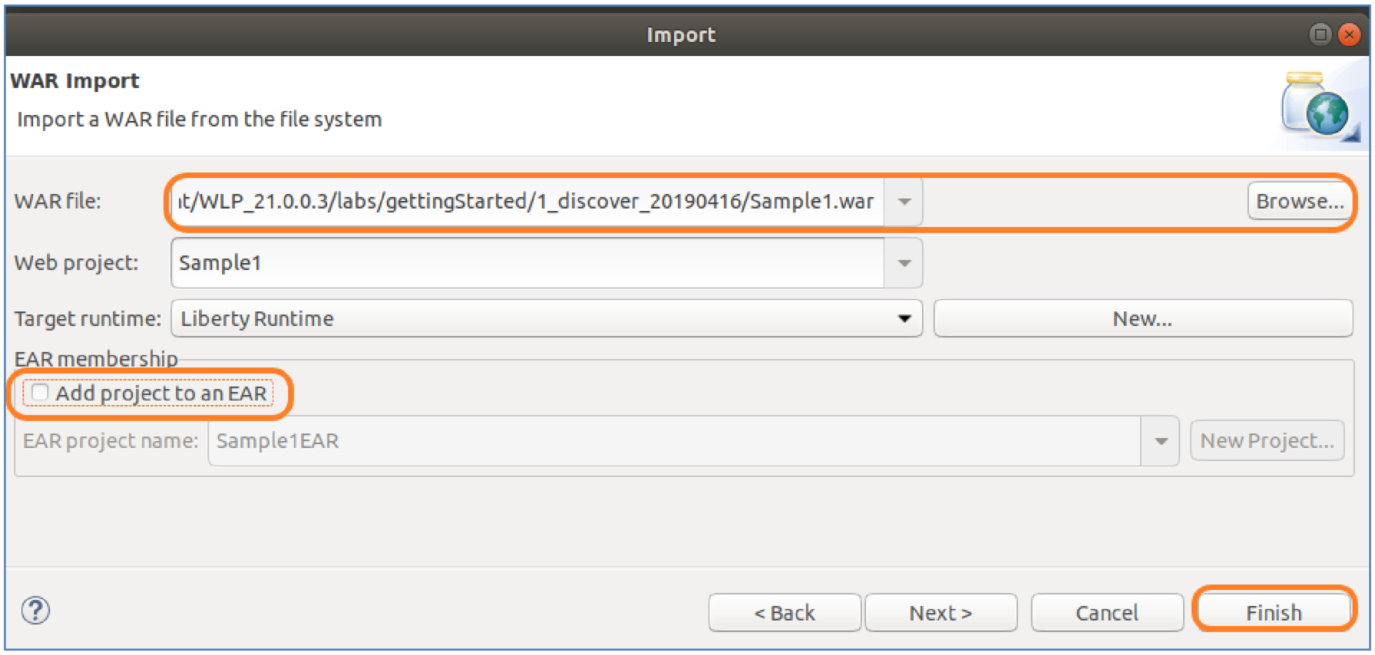
```
/home/ibmdemo/Student/WLP_21.0.0.3/labs/gettingStarted/1_discover_20190416/
Sample1.war
```

__c. Click **Open**.

__d. Ensure the **Target runtime** is set to **Liberty Runtime**

__e. **Unselect "Add project to an EAR"**

__f. Click **Finish**



Import

WAR Import

Import a WAR file from the file system

WAR file: ıt/WLP_21.0.0.3/labs/gettingStarted/1_discover_20190416/Sample1.war

Web project: Sample1

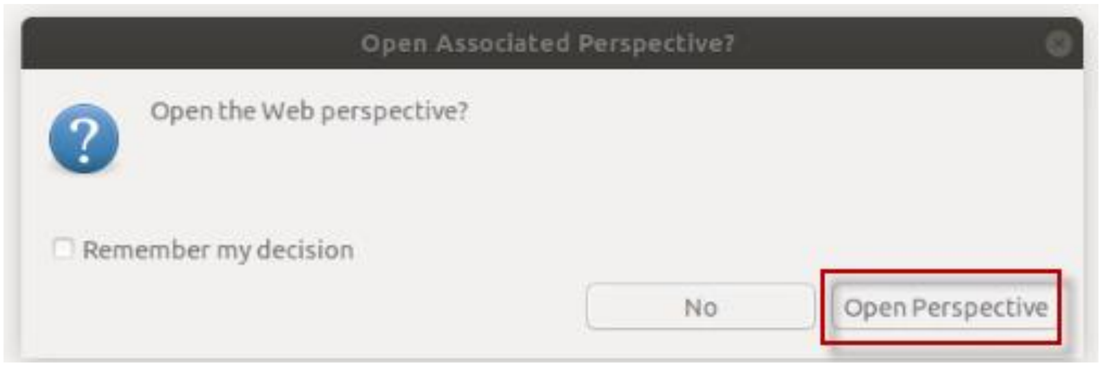
Target runtime: Liberty Runtime

EAR membership

☐ Add project to an EAR

EAR project name: Sample1EAR

__g. If prompted to open the web perspective, click **Open Perspective**.

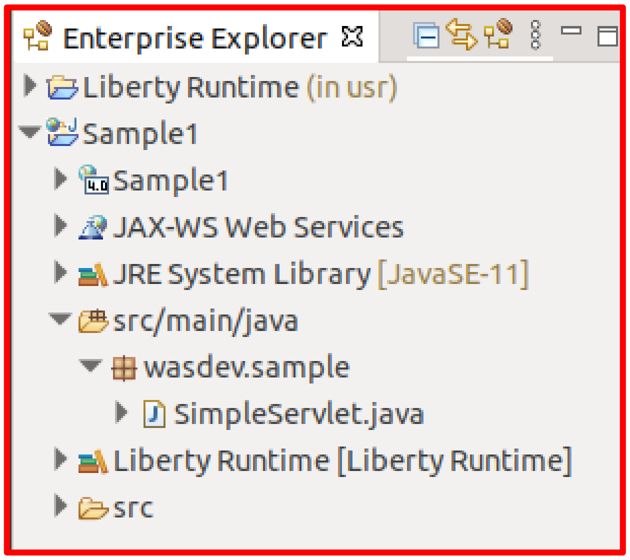


Open Associated Perspective?

Open the Web perspective?

☐ Remember my decision

__2. Now you have a **Sample1** web project in your workspace. Expand it in the **Enterprise Explorer** view to see the different components of the project.



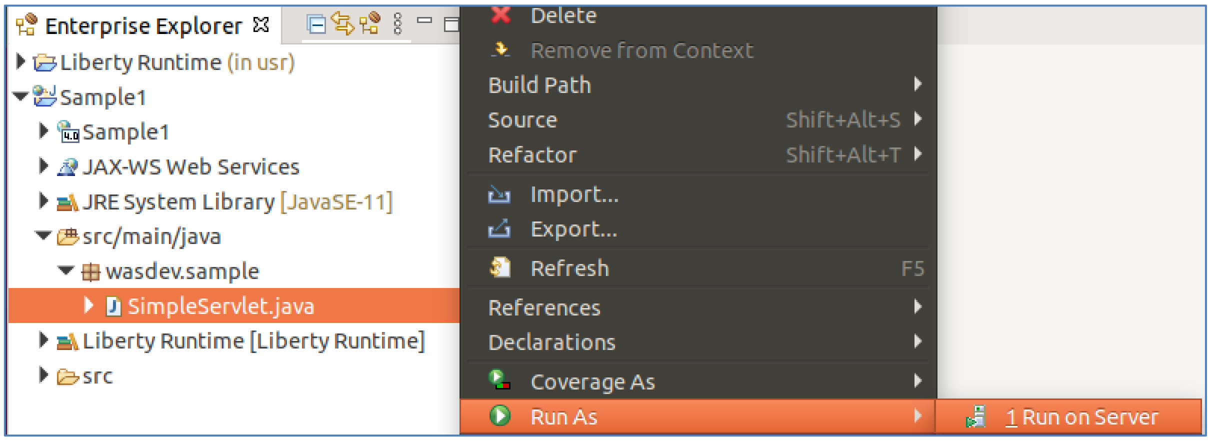
__3. **Start** the sample application.

__a. In the **Enterprise Explorer** pane, navigate to the **SimpleServlet.java** as shown below.

Sample1 -> src/main/java -> wasdev.sample -> SimpleServlet.java

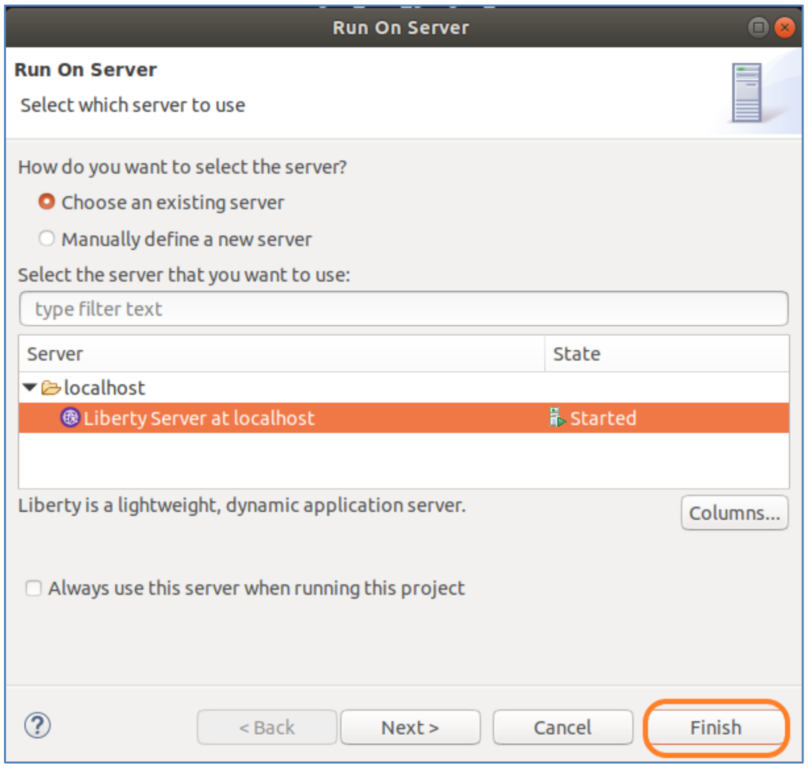
__b. Right-click on **SimpleServlet.java**.

__c. From the context menu, select **Run As > Run on Server**.

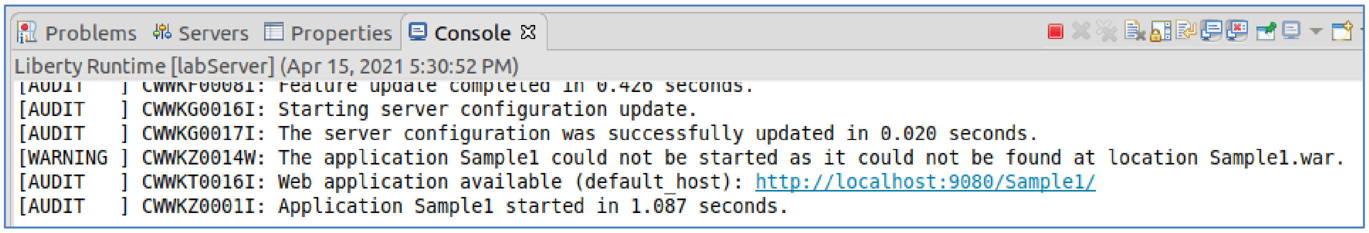


__d. In the **Run On Server** dialog:

- Verify that **Choose an existing server** is chosen.
- Under **localhost**, select the **Liberty Server** that you defined earlier. The server should be listed in **Started** state.
- Click **Finish**.


Run On Server
Run On Server
Select which server to use
How do you want to select the server?
Choose an existing server
Manually define a new server
Select the server that you want to use:
type filter text
Server
State
localhost
Liberty Server at localhost
Started
Liberty is a lightweight, dynamic application server.
Columns...
Always use this server when running this project
< Back
Next >
Cancel
Finish

__e. After a moment, your application will be installed and started. See the **Console** pane for the corresponding messages.



Liberty Runtime [labServer] (Apr 15, 2021 5:30:52 PM)

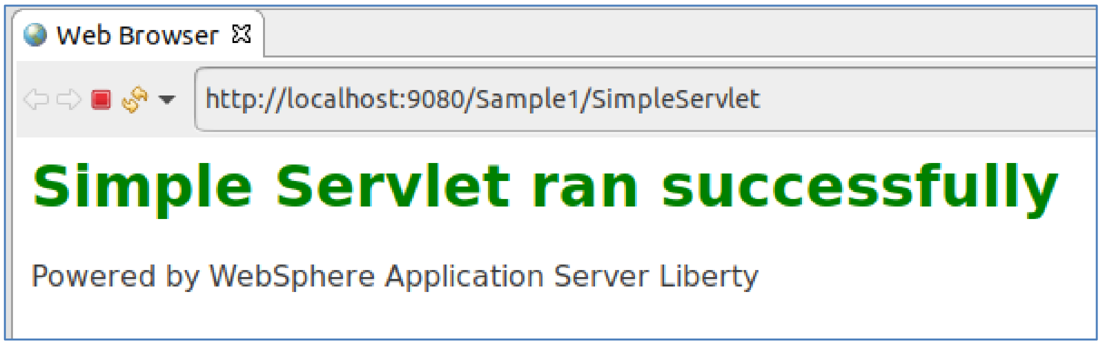
```
[AUDIT   ] CWWKF0008I: Feature update completed in 0.426 seconds.
[AUDIT   ] CWWKG0016I: Starting server configuration update.
[AUDIT   ] CWWKG0017I: The server configuration was successfully updated in 0.020 seconds.
[WARNING ] CWWKZ0014W: The application Sample1 could not be started as it could not be found at location Sample1.war.
[AUDIT   ] CWWKT0016I: Web application available (default_host): http://localhost:9080/Sample1/
[AUDIT   ] CWWKZ0001I: Application Sample1 started in 1.087 seconds.
```

__4. In the main panel of the workbench, a browser opened, pointing to

http://localhost:9080/Sample1/SimpleServlet.

__a. If you receive a 404 the first time, try to refresh the browser once the application is completely deployed and started.

__b. At this point, you should see the rendered HTML content generated by the simple servlet.



Simple Servlet ran successfully

Powered by WebSphere Application Server Liberty

In this section of the lab, you explored how to use the WebSphere Developer Tools (WDT) to start a Liberty Server, modify the server configuration, import a simple Java EE WAR module, and deploy the application to the server from with the development environment.

In the next section, you will explore using WDT to make changes to the application code, and have it hot deployed to the Liberty server, in your development environment, to illustrate a simple, yet robust development experience for Java developers using WDT and Liberty.

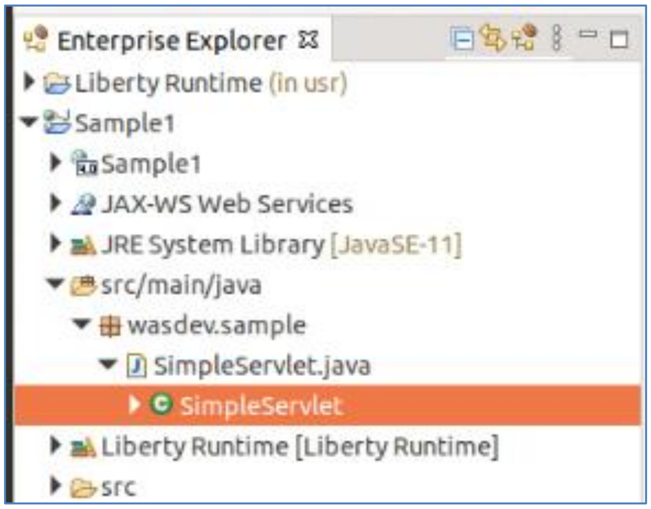
2.4.2 Modify the application to see how changes are automatically picked up in the running server.

__1. Open the servlet java source code.

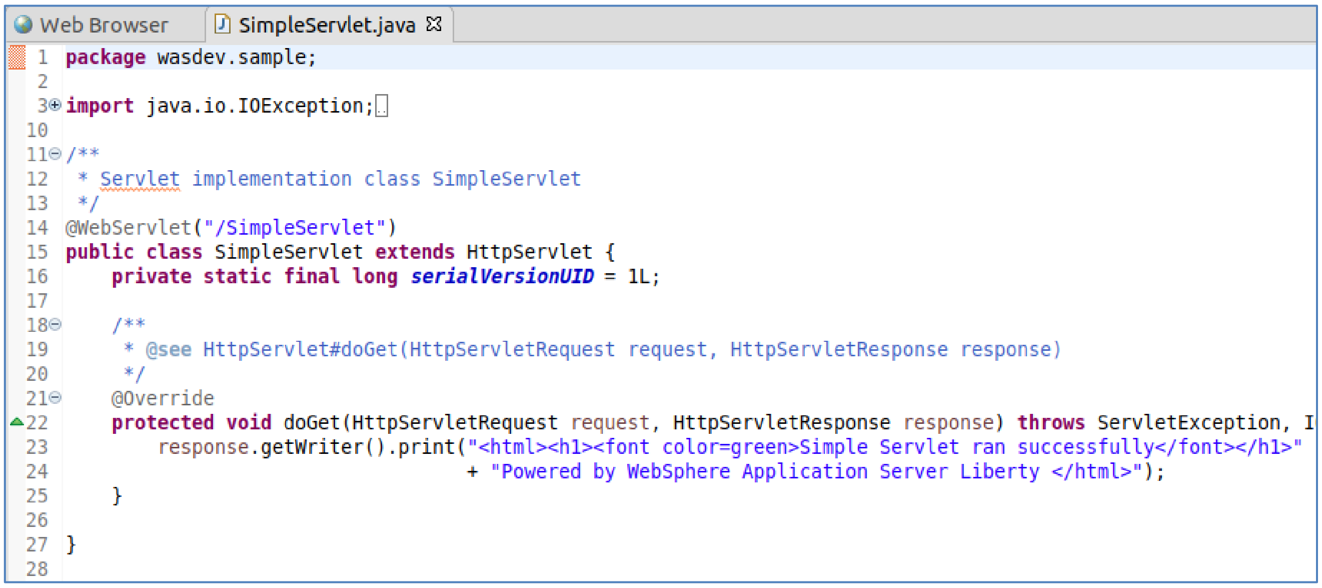
__a. In the **Enterprise Explorer** panel, expand the **Sample1** project, then go to:

Sample1 > src/main/java > wasdev.sample

__b. Double-click the **SimpleServlet**.java source file to open the Java editor for the servlet.



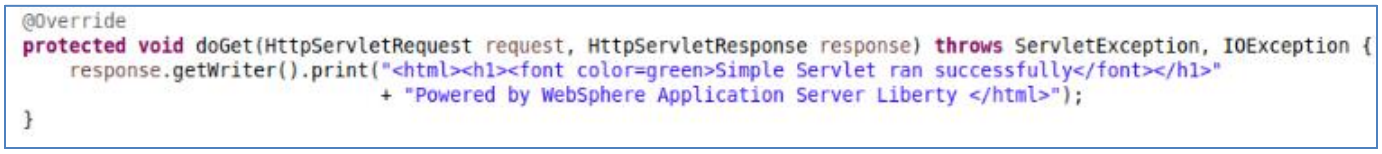
__c. This is the **SimpleServlet.java** source code



```
package wasdev.sample;

import java.io.IOException;
/**
 * Servlet implementation class SimpleServlet
 */
@WebServlet("/SimpleServlet")
public class SimpleServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, I
        response.getWriter().print("<html><h1><font color=green>Simple Servlet ran successfully</font></h1>"
                                  + "Powered by WebSphere Application Server Liberty </html>");
    }

}
```

__2. Examine the doGet() method in the SimpleServlet.java code

__a. This is a very simple servlet with a **doGet()** method that sends out an HTML snippet string as a response. Your **doGet()** method will look similar to this (some of the HTML tags might be a little different – that is ok).



```
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    response.getWriter().print("<html><h1><font color=green>Simple Servlet ran successfully</font></h1>"
                                + "Powered by WebSphere Application Server Liberty </html>");
}
```

__3. Modify the application and publish the change.

__a. In the **doGet()** method, Locate the **<h1>** heading element of the HTML string, and notice that it contains a **font tag** to set the color to "**green**".

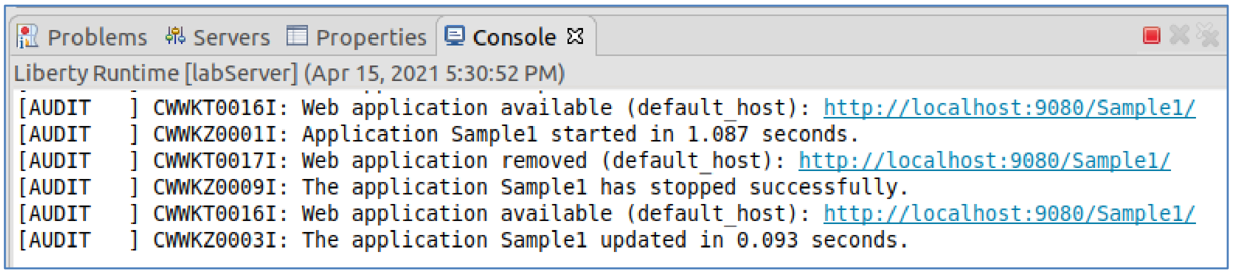
__b. Modify this string by changing the text green to **purple**, so your font tag will look read **<font color=purple>**.

```
 response.getWriter().print(
                "<html><h1><font color=purple>Simple Servlet ran 
successfully</font></html>"
                + "<html>Powered by WebSphere Application Server Liberty</html>");
```

__c. Save your changes to the Java source file by either clicking the **Save** button () or using **CTRL+S**.

__4. Recall that your server configuration is setup to automatically detect and **publish** application changes immediately. By saving the changes to your Java source file, you automatically triggered an application update on the server.

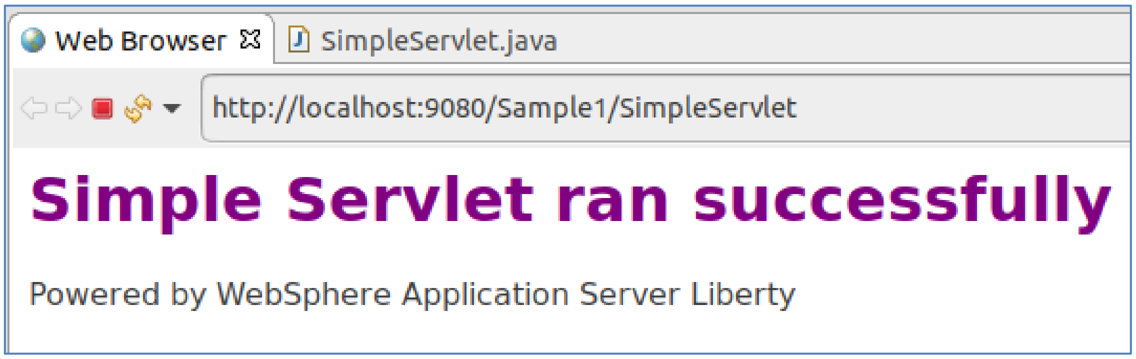
__a. To see this, go to the **Console** view at the bottom of the workbench. The application update started almost immediately after you saved the change to the application, and the update completed in seconds.



```
Liberty Runtime [labServer] (Apr 15, 2021 5:30:52 PM)
[AUDIT   ] CWWKT0016I: Web application available (default_host): http://localhost:9080/Sample1/
[AUDIT   ] CWWKZ0001I: Application Sample1 started in 1.087 seconds.
[AUDIT   ] CWWKT0017I: Web application removed (default_host): http://localhost:9080/Sample1/
[AUDIT   ] CWWKZ0009I: The application Sample1 has stopped successfully.
[AUDIT   ] CWWKT0016I: Web application available (default_host): http://localhost:9080/Sample1/
[AUDIT   ] CWWKZ0003I: The application Sample1 updated in 0.093 seconds.
```

__5. Access the updated application.

__a. Refresh the **browser** in your workbench to see the application change. The title should now be rendered in purple text.



Simple Servlet ran successfully

Powered by WebSphere Application Server Liberty

__6. Optionally continue to play around with application modifications and see how quickly those changes are available in the deployed application. Maybe put in some additional text to display on the page, or add extra HTML tags to see formatting changes (you could add a title tag to set the text displayed in the browser title bar, for example, **<head><title>Liberty Server</title></head>**).

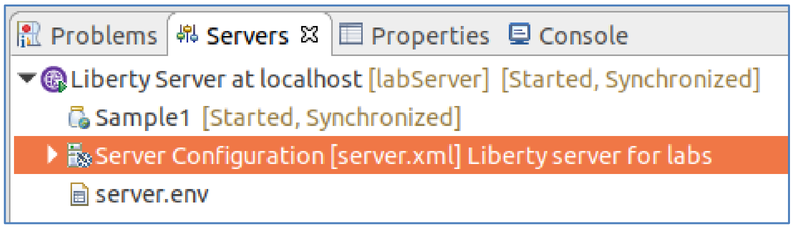
The key is that this **edit / publish / debug cycle is very simple and fast!**

2.4.3 Modify the server HTTP(s) ports to see that Liberty Serer configuration changes are also automatially picked up in the running server

With Liberty, server configuration changes are monitored, and dynamically updated in the running instance of the server. In this section, you will make some updates to the Liberty server.xml file and observe the dynamic server updates capability in Liberty.

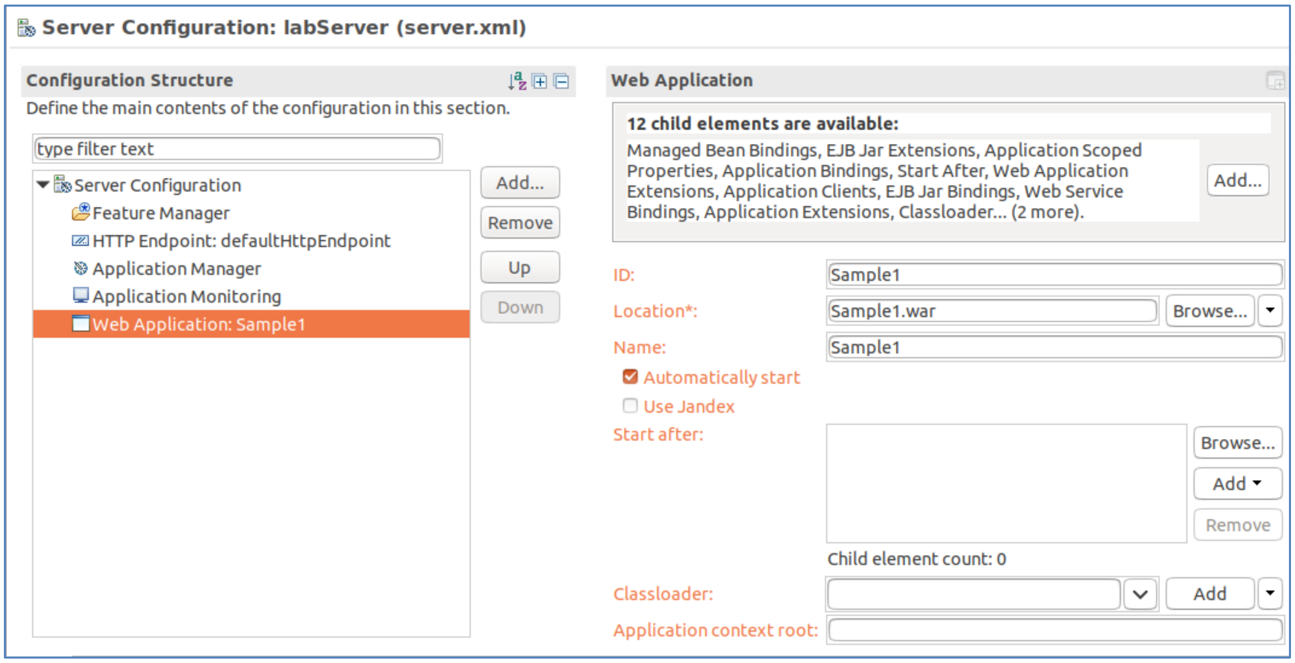
__1. Open the server configuration editor.

__a. In the **Servers** view, double-click on the labServer **Server Configuration** server to open the configuration server.xml editor.


Problems
Servers
Properties
Console
Liberty Server at localhost [labServer] [Started, Synchronized]
Sample1 [Started, Synchronized]
Server Configuration [server.xml] Liberty server for labs
server.env

__b. Ensure you are in the Design mode by selecting the **Design** tab on the Server Configuration editor.

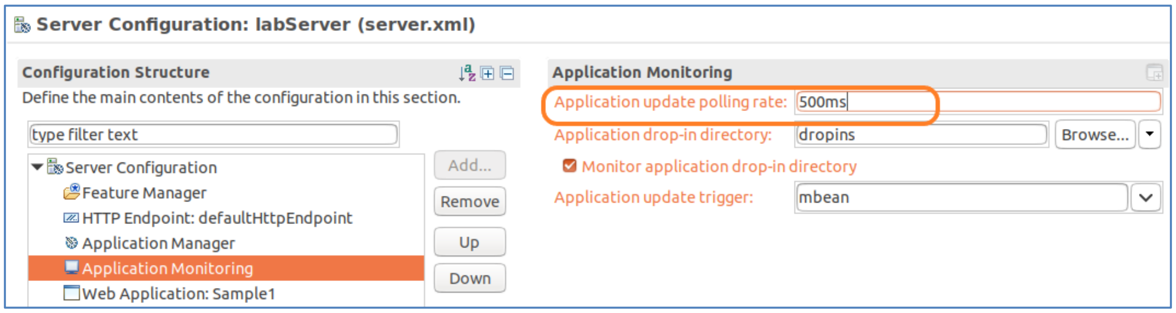
__2. Select the **Web Application: Sample1** item in the **Server Configuration** and look at its configuration details. From here, you can set basic application parameters, including the context root for the application, and to automatically start the application.



12 child elements are available:
Managed Bean Bindings, EJB Jar Extensions, Application Scoped Properties, Application Bindings, Start After, Web Application Extensions, Application Clients, EJB Jar Bindings, Web Service Bindings, Application Extensions, Classloader... (2 more).

ID: Sample1
Location*: Sample1.war
Name: Sample1
☑ Automatically start
☐ Use Jandex
Start after:
Child element count: 0
Classloader:
Application context root:

__3. Select the **Application Monitoring** item in the **Server Configuration** and look at its configuration details. You can see that the monitor polls for changes every **500ms** using an **mbean** trigger.



Server Configuration: labServer (server.xml)

Configuration Structure
Define the main contents of the configuration in this section.

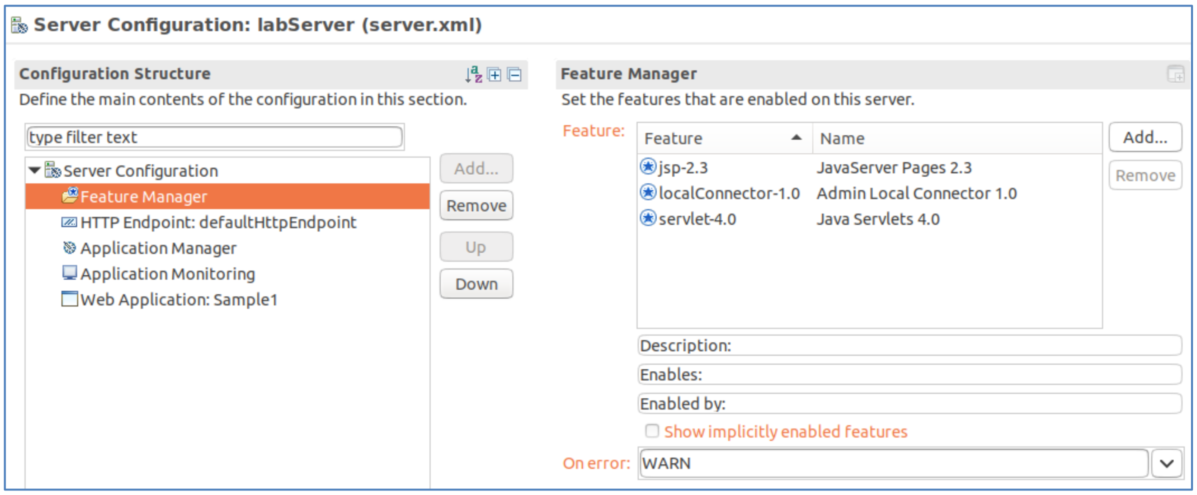
type filter text

- Server Configuration
  - Feature Manager
  - HTTP Endpoint: defaultHttpEndpoint
  - Application Manager
  - Application Monitoring
  - Web Application: Sample1

Application Monitoring

Application update polling rate: 500ms
Application drop-in directory: dropins
☑ Monitor application drop-in directory
Application update trigger: mbean

__4. Select the **Feature Manager** item to see the features that are configured on your server. You added the **servlet-4.0** feature because you knew that you were going to be running a servlet application. But the development tools automatically added the **localConnector-1.0.** feature to your server to support notifications and application updates. In fact, you would not have needed to add the servlet feature to your server at the beginning at all. The tools would have automatically enabled that feature, based on the content of the application.



Server Configuration: labServer (server.xml)

Configuration Structure
Define the main contents of the configuration in this section.

type filter text

- Server Configuration
  - Feature Manager
  - HTTP Endpoint: defaultHttpEndpoint
  - Application Manager
  - Application Monitoring
  - Web Application: Sample1

Feature Manager
Set the features that are enabled on this server.

Feature:

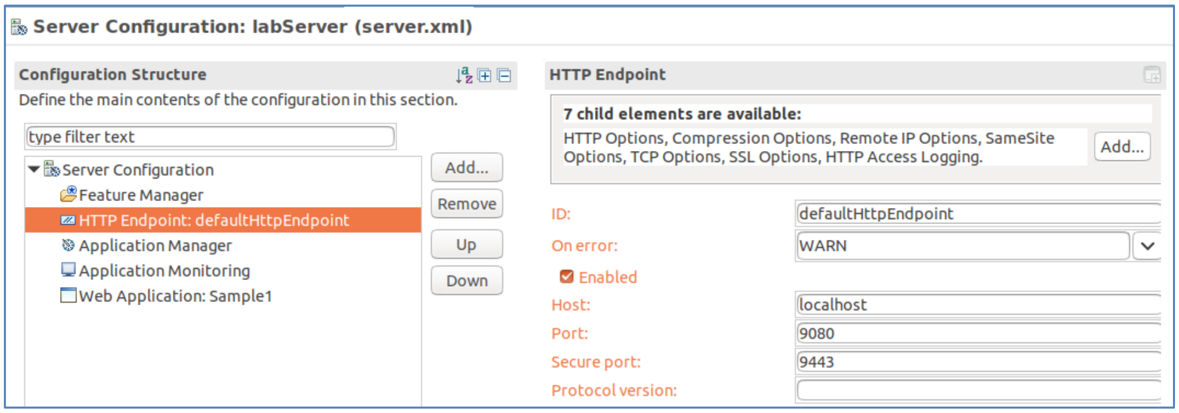
Feature	Name
jsp-2.3	JavaServer Pages 2.3
localConnector-1.0	Admin Local Connector 1.0
servlet-4.0	Java Servlets 4.0

Description:
Enables:
Enabled by:
☐ Show implicitly enabled features
On error: WARN

__5. Change the HTTP port.

Using the default HTTP port (9080) is an easy way to quickly bring up an application, but it is common to want to use a different port. This is an easy thing to change.

__a. In the **Configuration Structure** area, select **Server Configuration**, then select **HTTP Endpoint**


Server Configuration: labServer (server.xml)

Configuration Structure

Define the main contents of the configuration in this section.

- Server Configuration
  - Feature Manager
  - HTTP Endpoint: defaultHttpEndpoint
  - Application Manager
  - Application Monitoring
  - Web Application: Sample1

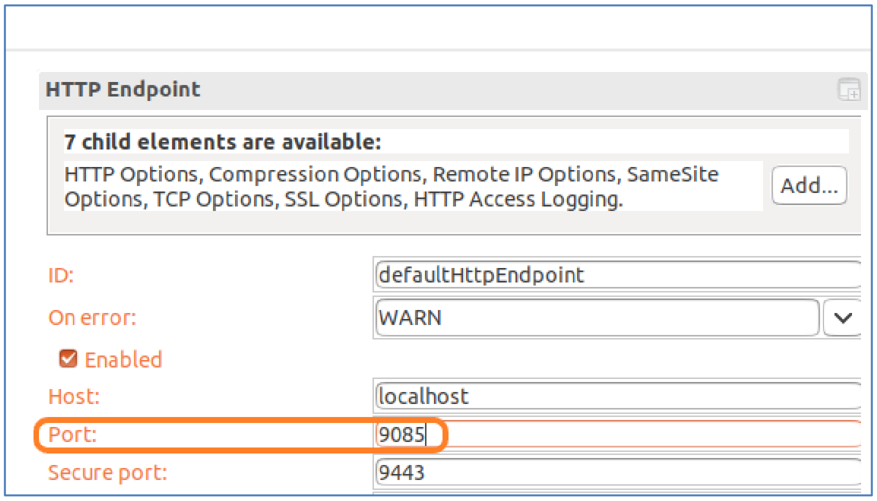
HTTP Endpoint

7 child elements are available:

HTTP Options, Compression Options, Remote IP Options, SameSite Options, TCP Options, SSL Options, HTTP Access Logging.

ID:	defaultHttpEndpoint
On error:	WARN
Enabled	✓
Host:	localhost
Port:	9080
Secure port:	9443
Protocol version:	

__b. In the **HTTP Endpoint Details** area, Change the HTTP Port to **9085**.


HTTP Endpoint

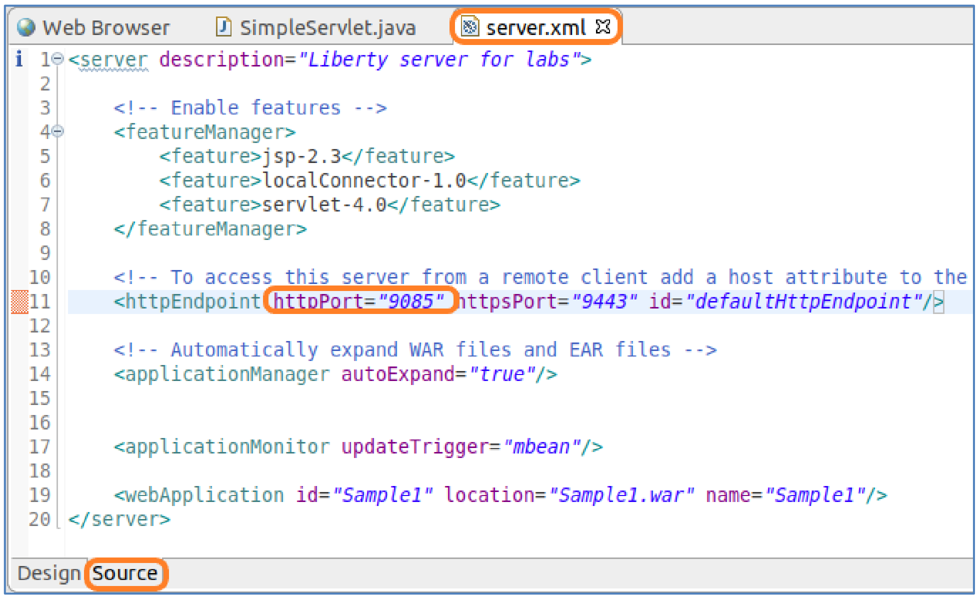
7 child elements are available:

HTTP Options, Compression Options, Remote IP Options, SameSite Options, TCP Options, SSL Options, HTTP Access Logging.

ID:	defaultHttpEndpoint
On error:	WARN
Enabled	✓
Host:	localhost
Port:	9085
Secure port:	9443

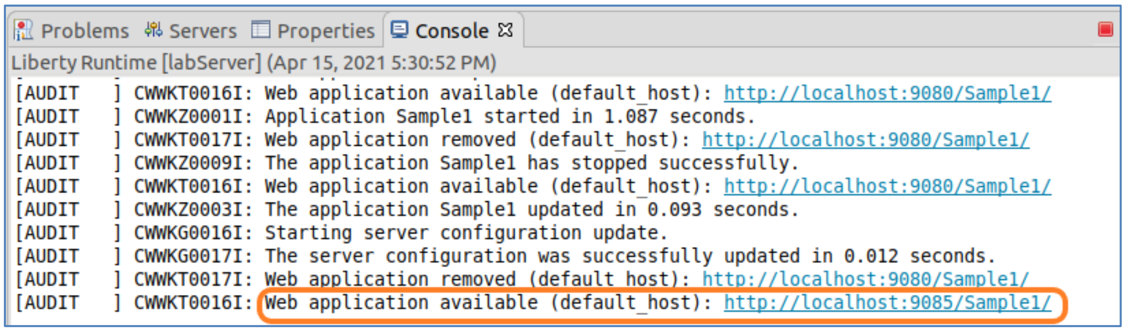
__c. **Save** your changes to the server configuration (CTRL+S).

__6. You can review your full server configuration in the **server.xml** source file. Back in the server configuration editor, switch to the **Source** tab at the bottom to view the full XML source for your server configuration.



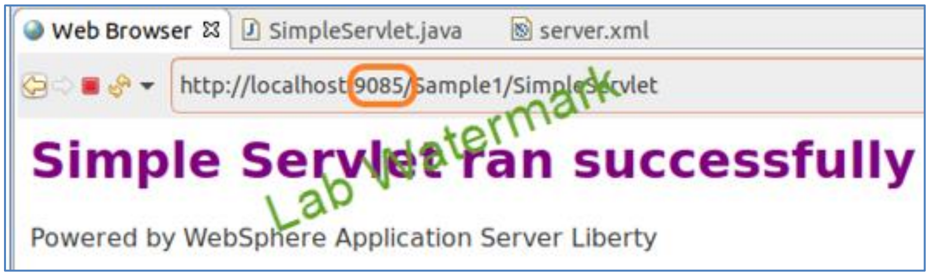
```
<server description="Liberty server for labs">

    <!-- Enable features -->
    <featureManager>
        <feature>jsp-2.3</feature>
        <feature>localConnector-1.0</feature>
        <feature>servlet-4.0</feature>
    </featureManager>

    <!-- To access this server from a remote client add a host attribute to the
    <httpEndpoint httpPort="9085" httpsPort="9443" id="defaultHttpEndpoint"/>

    <!-- Automatically expand WAR files and EAR files -->
    <applicationManager autoExpand="true"/>


    <applicationMonitor updateTrigger="mbean"/>

    <webApplication id="Sample1" location="Sample1.war" name="Sample1"/>
</server>
```

__7. After you saved your configuration changes, the configuration of your running server was automatically updated. The **Console** pane will show that the Sample1 servlet is now available on port 9085.



```
Liberty Runtime [labServer] (Apr 15, 2021 5:30:52 PM)
[AUDIT   ] CWWKT0016I: Web application available (default_host): http://localhost:9080/Sample1/
[AUDIT   ] CWWKZ0001I: Application Sample1 started in 1.087 seconds.
[AUDIT   ] CWWKT0017I: Web application removed (default_host): http://localhost:9080/Sample1/
[AUDIT   ] CWWKZ0009I: The application Sample1 has stopped successfully.
[AUDIT   ] CWWKT0016I: Web application available (default_host): http://localhost:9080/Sample1/
[AUDIT   ] CWWKZ0003I: The application Sample1 updated in 0.093 seconds.
[AUDIT   ] CWWKG0016I: Starting server configuration update.
[AUDIT   ] CWWKG0017I: The server configuration was successfully updated in 0.012 seconds.
[AUDIT   ] CWWKT0017I: Web application removed (default_host): http://localhost:9080/Sample1/
[AUDIT   ] CWWKT0016I: Web application available (default_host): http://localhost:9085/Sample1/
```

__8. Now, you can access your sample application using the new port. In the browser in your workbench, change the port from **9080** to **9085** and refresh the application.



http://localhost:9085/Sample1/SimpleServlet

Simple Servlet ran successfully

Powered by WebSphere Application Server Liberty

2.5 Add INFO logging output to console

WebSphere Traditional and Liberty provide the ability to set the logging level to any of the supported log levels defined in the documentation: https://www.ibm.com/docs/en/was-liberty/base?topic=liberty-logging-trace

AUDIT logging enables logging of "Significant event affecting server state or resources"

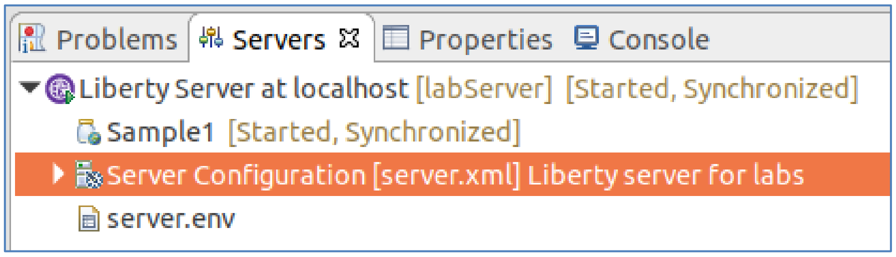
INFO logging enables of "General information outlining overall task progress"

By default, the Liberty Server has the console log level set to AUDIT. In this section, you will change the level of log messages written to the console from AUDIT to INFO, which will result in additional logging messages.

You will perform this activity in the server.xml file using the UI. It is also possible to set default logging options in the bootstrap.properties file. If the logging options are set in the bootstrap.properties file, the logging options will take effect very early in server startup, so it may be useful for debugging server initialization problems.

__1. Open the server configuration editor.

__a. In the **Servers** view, double-click on the labServer **Server Configuration** server to open the configuration server.xml editor.

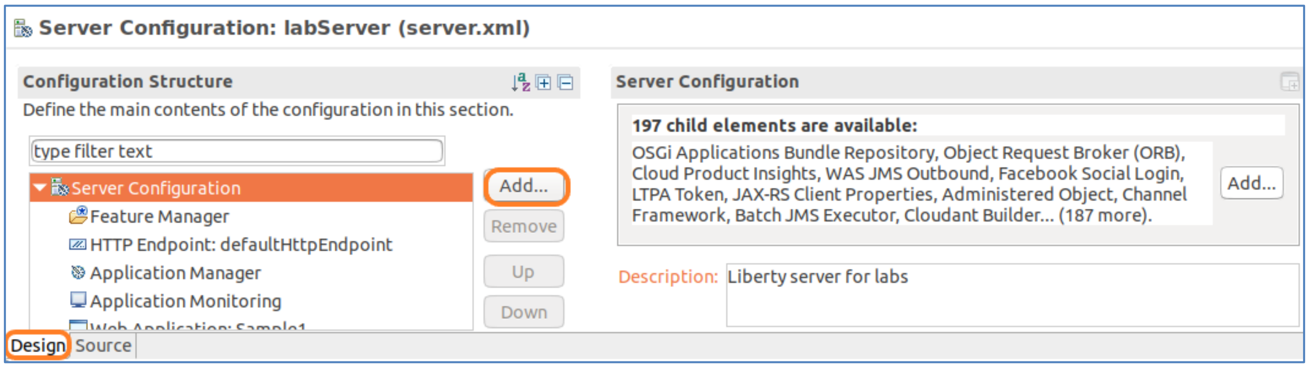


Liberty Server at localhost [labServer] [Started, Synchronized]
Sample1 [Started, Synchronized]
Server Configuration [server.xml] Liberty server for labs
server.env

__b. Ensure you are in the Design mode by selecting the **Design** tab on the Server Configuration editor.

__2. Add the **Logging** configuration option to the server

__a. Under the **Configuration Structure** section, Click on **Server Configuration.** And, then click the **Add button.**


Server Configuration: labServer (server.xml)

Configuration Structure

Define the main contents of the configuration in this section.

type filter text

- Server Configuration
  - Feature Manager
  - HTTP Endpoint: defaultHttpEndpoint
  - Application Manager
  - Application Monitoring

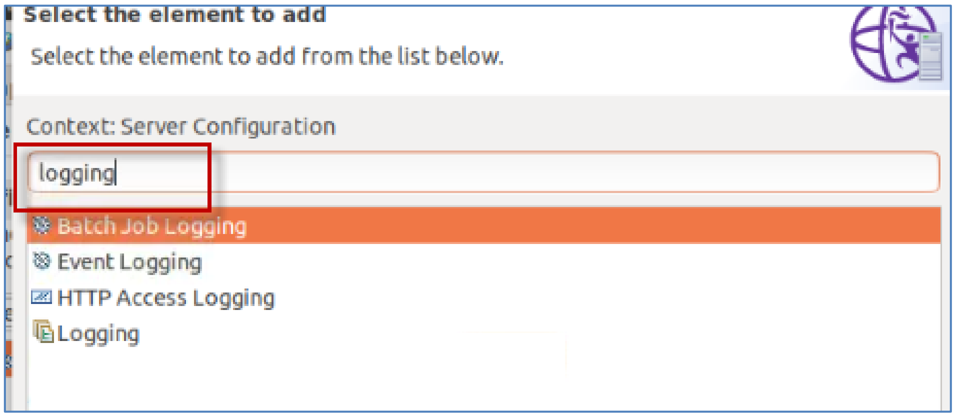
Server Configuration

197 child elements are available:

OSGi Applications Bundle Repository, Object Request Broker (ORB), Cloud Product Insights, WAS JMS Outbound, Facebook Social Login, LTPA Token, JAX-RS Client Properties, Administered Object, Channel Framework, Batch JMS Executor, Cloudant Builder... (187 more).

Description: Liberty server for labs

__b. Type **logging** in the “Context: Server Configuration” field to narrow the list of configuration options displayed


Select the element to add

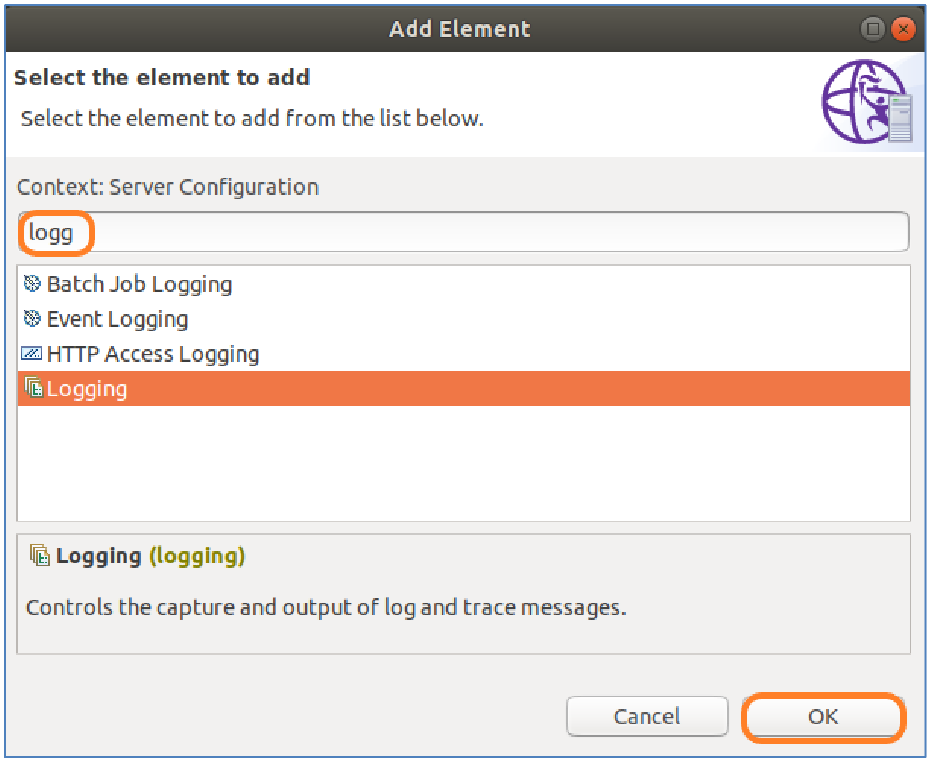
Select the element to add from the list below.

Context: Server Configuration

logging

- Batch Job Logging
- Event Logging
- HTTP Access Logging
- Logging

__c. On the **Add Element** dialog, select **Logging**, And, then click the **OK** button**.**


Add Element

Select the element to add

Select the element to add from the list below.

Context: Server Configuration

logg

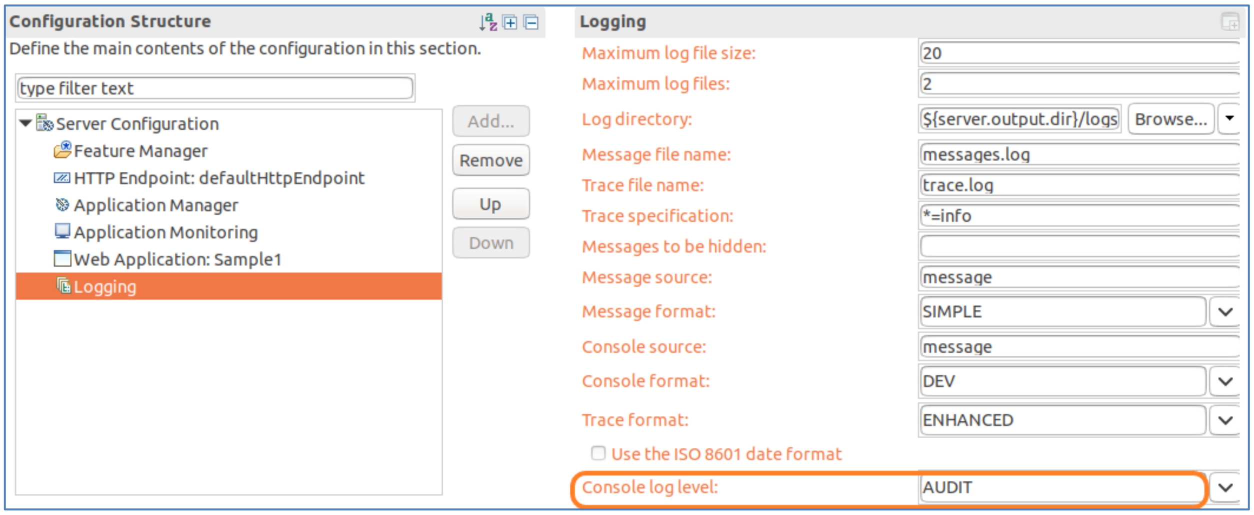
- Batch Job Logging
- Event Logging
- HTTP Access Logging
- Logging

Logging (logging)

Controls the capture and output of log and trace messages.

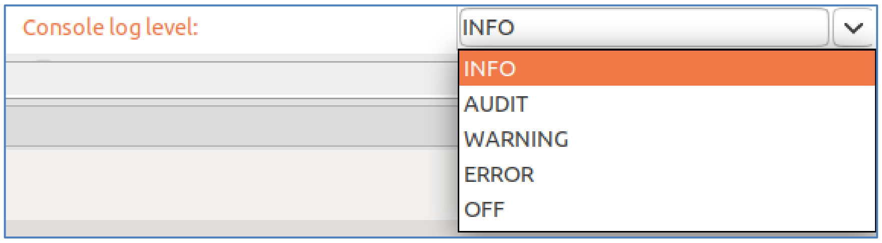
__d. The logging page displays the properties for the logging configuration, such as the name of the log files, the maximum size of log files, and the maximum number of log files to retain.

Additional configuration information is displayed regarding tracing. Notice that the **Console Log Level** is set to **AUDIT** by default.



Maximum log file size: 20
Maximum log files: 2
Log directory: ${server.output.dir}/logs
Message file name: messages.log
Trace file name: trace.log
Trace specification: *=info
Messages to be hidden:
Message source: message
Message format: SIMPLE
Console source: message
Console format: DEV
Trace format: ENHANCED
☐ Use the ISO 8601 date format
Console log level: AUDIT

__3.	Change the Console log level to **INFO** using the pull-down menu.

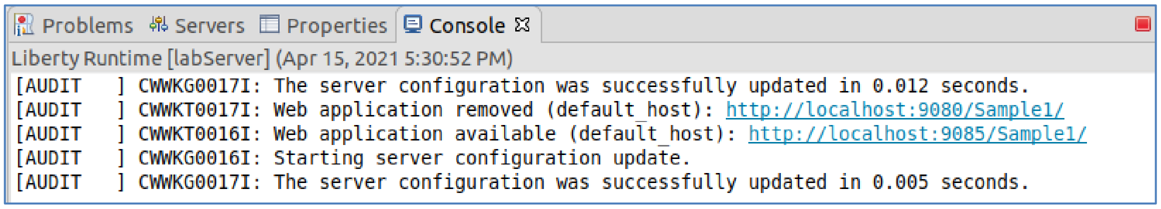


__a.	Switch to the **Source** view for the server.xml file to see the configuration changes added to server.xml.

```
<logging consoleLogLevel="INFO"></logging>
```

__b.	**Save** the configuration file.

The changes you made are dynamic and take effect immediately.



```
Liberty Runtime [labServer] (Apr 15, 2021 5:30:52 PM)
[AUDIT   ] CWWKG0017I: The server configuration was successfully updated in 0.012 seconds.
[AUDIT   ] CWWKT0017I: Web application removed (default_host): http://localhost:9080/Sample1/
[AUDIT   ] CWWKT0016I: Web application available (default_host): http://localhost:9085/Sample1/
[AUDIT   ] CWWKG0016I: Starting server configuration update.
[AUDIT   ] CWWKG0017I: The server configuration was successfully updated in 0.005 seconds.
```

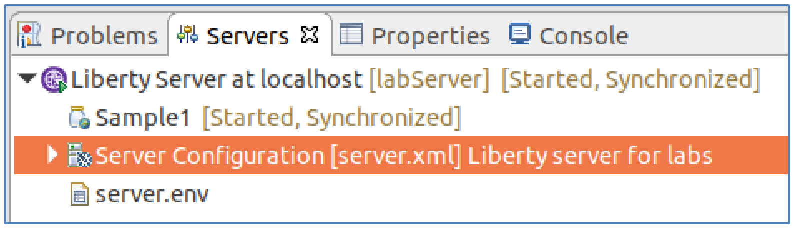
2.6 Update trace specification

By default, the Liberty Server trace specification is set to ***=info=enabled**. This is the same for Traditional WebSphere Application Server (WAS).

Updating the trace specification for debugging is easily performed using the server configuration editor. You can specify the trace specification in the UI, or copy / paste the trace specification directly into the server.xml file. In this section, you will specify a trace specification using the configuration editor. And, then, you will look at the result in the server.xml source file

__1. Open the server configuration editor, if it is not already opened.

__a. In the **Servers** view, double-click on the labServer **Server Configuration** server to open the configuration server.xml editor.



Liberty Server at localhost [labServer] [Started, Synchronized]
Sample1 [Started, Synchronized]
Server Configuration [server.xml] Liberty server for labs
server.env

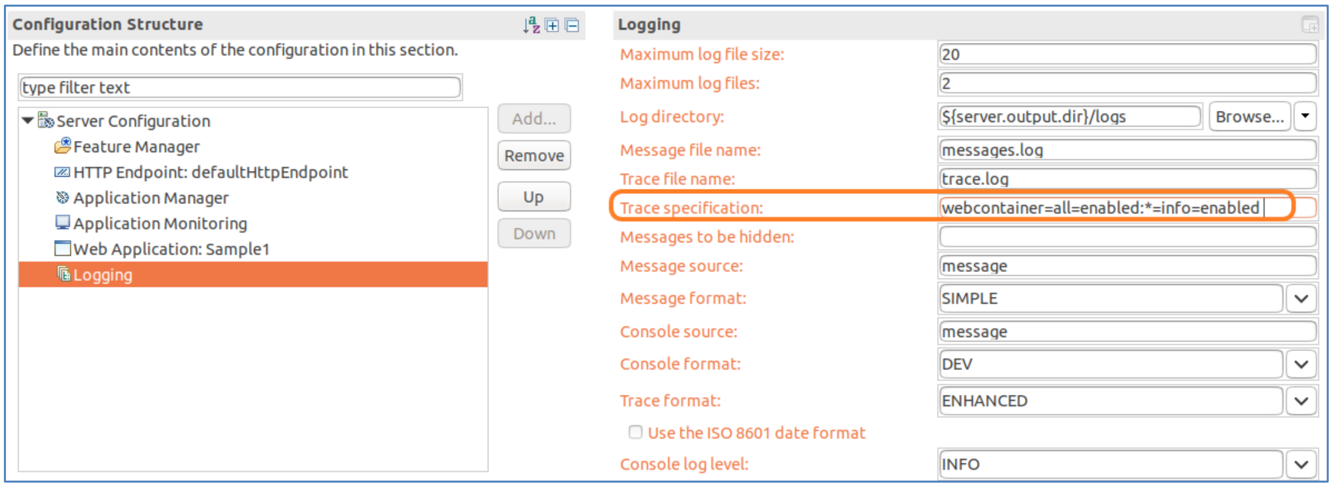
__b. Ensure you are in the Design mode by selecting the **Design** tab on the Server Configuration editor.

__2. Update the **Trace Specification** under the **logging** configuration.

__a. Click on **Logging** under the Server Configuration section. This displays the logging and trace details.

__b. Update the **Trace Specification** field with the following trace string:

```
webcontainer=all=enabled:*=info=enabled
```



Server Configuration
Feature Manager
HTTP Endpoint: defaultHttpEndpoint
Application Manager
Application Monitoring
Web Application: Sample1
Logging

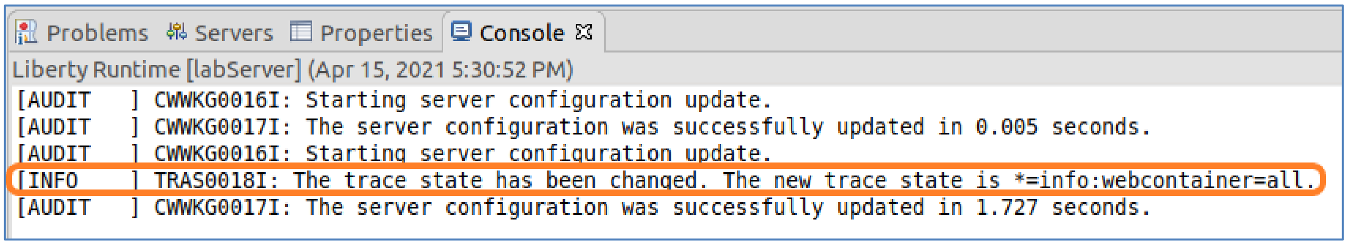
Maximum log file size:	20
Maximum log files:	2
Log directory:	${server.output.dir}/logs
Message file name:	messages.log
Trace file name:	trace.log
Trace specification:	webcontainer=all=enabled:*=info=enabled
Messages to be hidden:	
Message source:	message
Message format:	SIMPLE
Console source:	message
Console format:	DEV
Trace format:	ENHANCED
☐ Use the ISO 8601 date format	
Console log level:	INFO

__c. Switch to the **Source** tab on the configuration editor and view the logging configuration. :

```
<logging consoleLogLevel="INFO" traceSpecification="webcontainer=all=enabled:*=info=enabled
        "></logging>
```

__d. **Save** the configuration changes.

__e. Check the console view



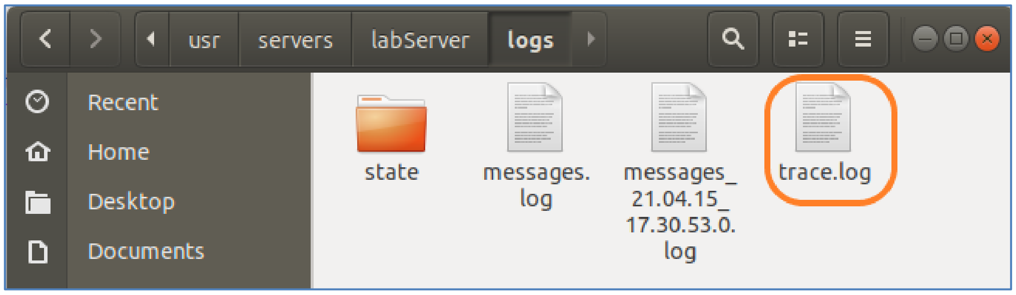
```
Liberty Runtime [labServer] (Apr 15, 2021 5:30:52 PM)
[AUDIT   ] CWWKG0016I: Starting server configuration update.
[AUDIT   ] CWWKG0017I: The server configuration was successfully updated in 0.005 seconds.
[AUDIT   ] CWWKG0016I: Starting server configuration update.
[INFO    ] TRAS0018I: The trace state has been changed. The new trace state is *=info:webcontainer=all.
[AUDIT   ] CWWKG0017I: The server configuration was successfully updated in 1.727 seconds.
```

__3. Verify that the **trace.log** file contains trace data.

__a. Navigate to the server logs directory.

```
Home > Student >  WLP_21.0.0.3 > wlp > usr > servers > labServer > logs
```

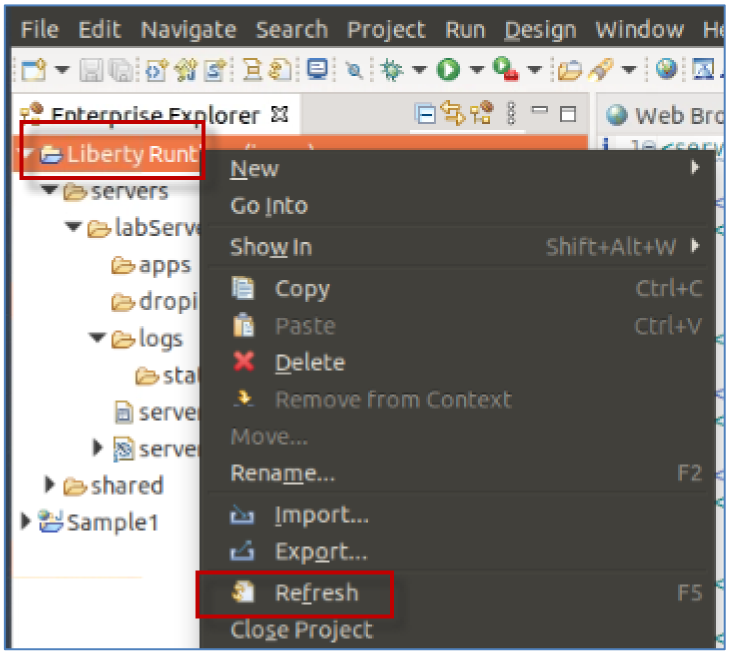
The trace.log file has been created and contains content.



__b. Double click on the **trace.log** file to view it in the text editor.

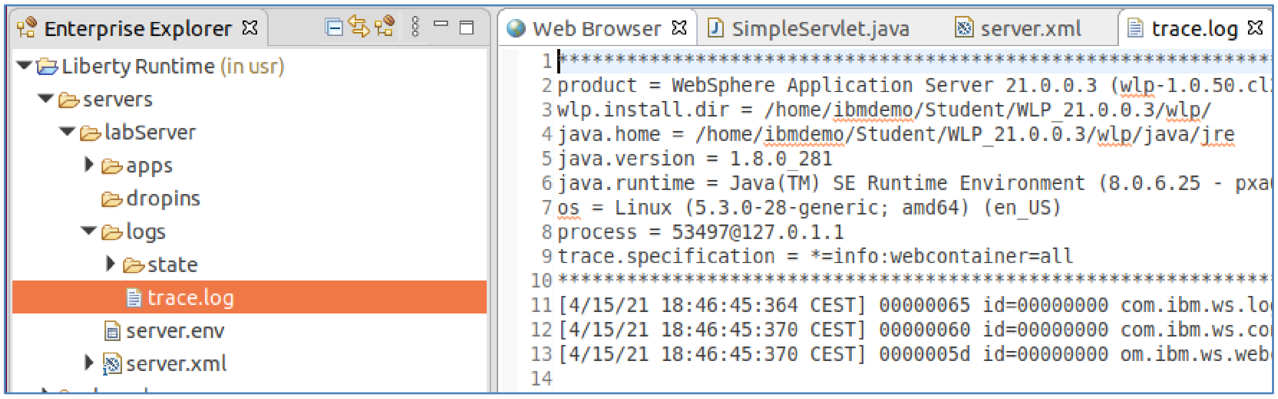
__4. You can also view the trace file in Eclipse. In the **Enterprise Explorer** view, expand the **Liberty Runtime** project, and its subdirectories, and you will find the **trace.log** file on the logs directory.

__a. Refresh the Liberty Runtime project so that the trace.log file will be visible in the view, since it was created outside of the eclipse IDE. **Right mouse click** on the **Liberty Runtime** project, and select **Refresh** from the context menu



__b. Navigate to Liberty Runtime > servers > labServer > logs

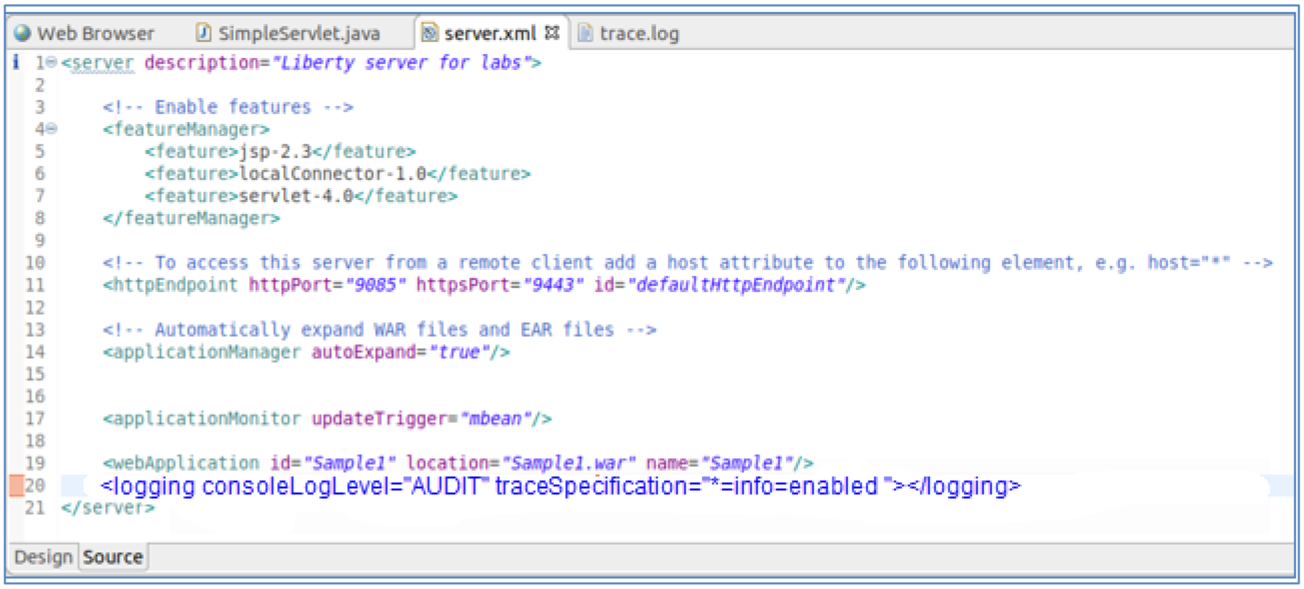
__c. The trace.log file is viewable from inside of the IDE



```
********************************************************************
product = WebSphere Application Server 21.0.0.3 (wlp-1.0.50.cl
wlp.install.dir = /home/ibmdemo/Student/WLP_21.0.0.3/wlp/
java.home = /home/ibmdemo/Student/WLP_21.0.0.3/wlp/java/jre
java.version = 1.8.0_281
java.runtime = Java(TM) SE Runtime Environment (8.0.6.25 - pxa
os = Linux (5.3.0-28-generic; amd64) (en_US)
process = 53497@127.0.1.1
trace.specification = *=info:webcontainer=all
********************************************************************
[4/15/21 18:46:45:364 CEST] 00000065 id=00000000 com.ibm.ws.lo
[4/15/21 18:46:45:370 CEST] 00000060 id=00000000 com.ibm.ws.co
[4/15/21 18:46:45:370 CEST] 0000005d id=00000000 om.ibm.ws.web
```

__5. **Very importantly, reset the trace specification back to the default value.**

__a. Switch to the **Source** tab on the configuration editor and update the logging configuration to:
<logging consoleLogLevel="AUDIT" traceSpecification="*=info=enabled "></logging>



```
<server description="Liberty server for labs">

    <!-- Enable features -->
    <featureManager>
        <feature>jsp-2.3</feature>
        <feature>localConnector-1.0</feature>
        <feature>servlet-4.0</feature>
    </featureManager>

    <!-- To access this server from a remote client add a host attribute to the following element, e.g. host="*" -->
    <httpEndpoint httpPort="9085" httpsPort="9443" id="defaultHttpEndpoint"/>

    <!-- Automatically expand WAR files and EAR files -->
    <applicationManager autoExpand="true"/>


    <applicationMonitor updateTrigger="mbean"/>

    <webApplication id="Sample1" location="Sample1.war" name="Sample1"/>
    <logging consoleLogLevel="AUDIT" traceSpecification="*=info=enabled "></logging>
</server>
```

__b. **Save** the configuration.

2.7 Customizing Liberty JVM Options

The generic JVM arguments are used to configure and adjust how the JVM executes.

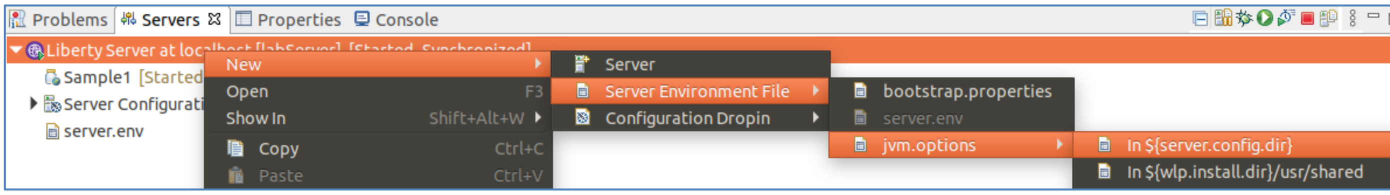
In this section of the lab, you will explore the JVM Options file. A common JVM option configuration is to set the minimum and maximum size of the JVM heap, based on the application runtime requirements.

The WebSphere Application Server Liberty is pre-configured with minimal settings defined. The following steps will direct you how to define custom generic JVM arguments such as heap settings for a Liberty server.

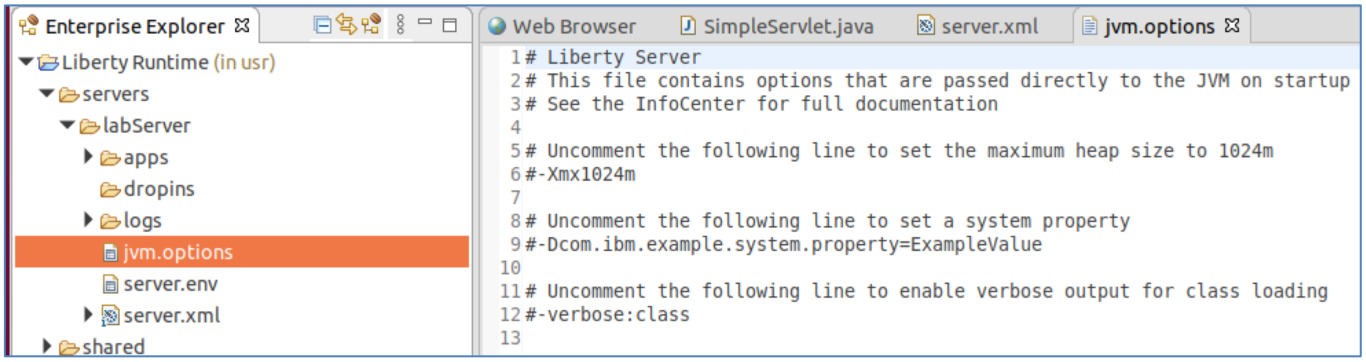
__1. First, create the jvm.options file, using the WebSPhere Developer tools

__a. In the Eclipse **Servers** view, right-click on the Liberty Server.

__b. Select from the context menu
New → Server Environment File → jvm.options → in ${server.config.dir}



This will create a **jvm.options** file in the server's configuration directory with the most commonly-used options available in comments:



```
# Liberty Server
# This file contains options that are passed directly to the JVM on startup
# See the InfoCenter for full documentation

# Uncomment the following line to set the maximum heap size to 1024m
#-Xmx1024m

# Uncomment the following line to set a system property
#-Dcom.ibm.example.system.property=ExampleValue

# Uncomment the following line to enable verbose output for class loading
#-verbose:class

```

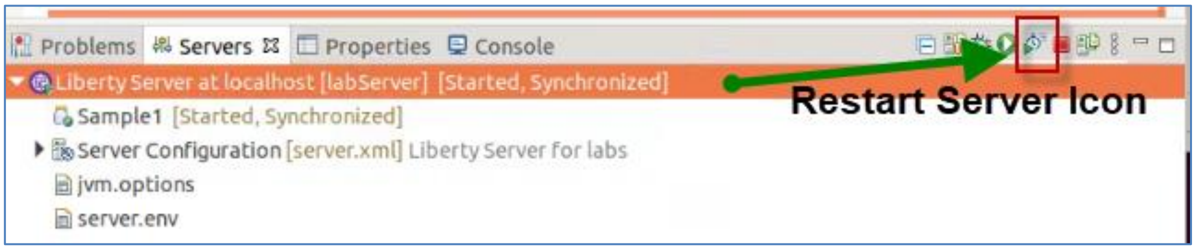
__2. If necessary, double click to open the file in the eclipse text editor

__3. Enter the following two lines in the **jvm.options** file to set the minimum and maximum heap size for the labServer server. The following options will set the min / max JVM heap size to 25 MB and 500 MB respectively.

```
-Xms25m

-Xmx500m
```

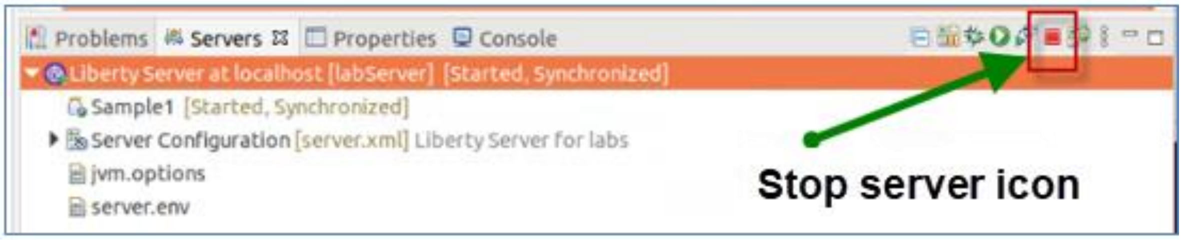
__4. **Save** the file. Ctrl + S

__5. **Restart** the server to enable changes



Liberty Server at localhost [labServer] [Started, Synchronized]
Sample1 [Started, Synchronized]
Server Configuration [server.xml] Liberty Server for labs
jvm.options
server.env

__6. **STOP** the server



Liberty Server at localhost [labServer] [Started, Synchronized]
Sample1 [Started, Synchronized]
Server Configuration [server.xml] Liberty Server for labs
jvm.options
server.env

__7. **Exit** Eclipse

This concludes the customization portion of the lab. In the next sections, you will be introduced to the Liberty configuration using the command line.

INFORMAITON ONLY! Do NOT run these steps below

- The default maximum heap size values of the JVM heap size is:

  –Xmx1024m

- VerboseGC can be enabled by specifying -verbose:gc in the jvm.options file.
- Verbose GC output will be logged to the following location by default:

  <wlp.install.dir>/usr/servers/<serverName>/logs/console.log

- Depending on your preferences, you might configure a single JVM or all Liberty JVMs with your options file. To apply these settings to all Liberty Servers, save jvm.options at:

  ${wlp.install.dir}/etc/jvm.options

  TIP: For this lab, the *${wlp.install.dir}* is "***{LAB_HOME}\wlp***"

  The changes will take effect for all JVMs that do not have a locally defined jvm.options file.

3 Explore Liberty via command line

In the previous section of the lab, you used the WebSphere Developer Tools in the Eclipse IDE to deploy an application and work with the Liberty configuration.

Creating servers, starting, and stopping servers, deploying applications, and overriding the server configuration can also be accomplished from the command line. In this section of the lab, you will experience how to work with Liberty from command line.

3.1 Create, start, and configure a new Liberty server

__1. Navigate to the Liberty directory

__a. Open a Terminal window and change to the Liberty installation directory

```
cd /home/ibmdemo/Student/WLP_21.0.0.3/wlp
```

__2. Create a new Liberty server named “myServer”. The “bin” directory contains the Liberty commands.

__a. Use the **server** command with the “create” option to create a new server named “myServer”. The server is created in a matter of seconds.

```
bin/server create myServer
```

__3. Start the Liberty server, using the “start” option on the **server** command. The server is started in a matter of seconds.

```
bin/server start myServer
```

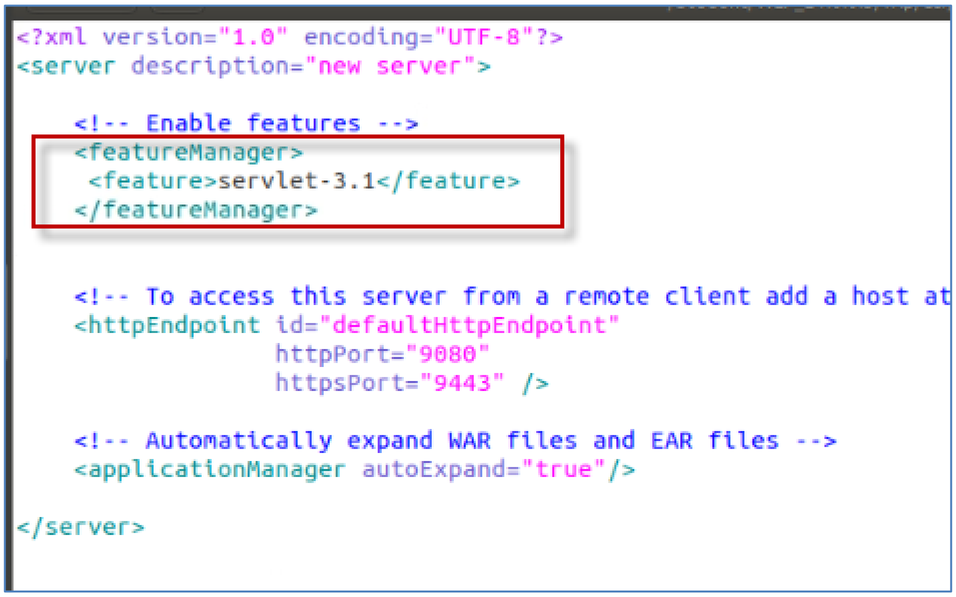
__4. Modify the server configuration to add the **servlet-3.1** feature that will be by the sample application you will deploy in a subsequent step.

__a. Open an editor to edit the **server.xml** file for the server names “myServer”.

```
gedit /home/ibmdemo/Student/WLP_21.0.0.3/wlp/usr/servers/myServer/server.xml
```

__b. Under the feature manager, replace existing featured with servlet-3.1 feature:

```
<featureManager>
     <feature>servlet-3.1</feature>
</featureManager>
```



```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
     <feature>servlet-3.1</feature>
    </featureManager>


    <!-- To access this server from a remote client add a host at
    <httpEndpoint id="defaultHttpEndpoint"
                  httpPort="9080"
                  httpsPort="9443" />

    <!-- Automatically expand WAR files and EAR files -->
    <applicationManager autoExpand="true"/>

</server>
```

__c. **Save** the changes

__d. **Close** the editor

__5. Look at the tail of {LAB_HOME}/wlp/usr/servers/myServer/logs/messages.log.

You should see messages about feature updates. The messages.log file is the main log file for Liberty, and by default, is located in the **logs** directory under the server configuration

```
tail /home/ibmdemo/Student/WLP_21.0.0.3/wlp/usr/servers/myServer/logs/messages.log
```

For example:

```
[4/16/19 14:29:29:276 CDT] 00000037 com.ibm.ws.kernel.feature.internal.FeatureManager
CWWKF0007I: Feature update started.

[4/16/19 14:29:29:400 CDT] 00000035 com.ibm.ws.config.xml.internal.ConfigRefresher
CWWKG0017I: The server configuration was successfully updated in 0.240 seconds.
```

Now you are ready to start working with a sample application that uses the Servlet feature.

3.2 Deploying a sample application to Liberty

In the first part of this lab, you used the WebSphere Developer Tools in the Eclipse IDE to deploy an application and work with the Liberty configuration.

In this section of the lab, you will deploy an application to Liberty using two different techniques.

First, you will simply copy the application WAR module into the Liberty "**dropins**" directory. The dropins directory is monitored by Liberty. As deployable units (WAR, EAR, JAR) are added to the directory, Liberty automatically deploys and starts the application on the Liberty server.

As the deployable units are removed from the dropins folder, the applications are stopped and removed from the running Liberty server.

Now, give it a try.

3.2.1 Deploy an application to dropins directory

Information:

The **dropins** directory can be used for applications that do not require extra configuration, such as security role mapping

__1. First, use the "**tail -f"** "command to view the Liberty server's **messages.log** file to see the messages that are generated once the application war file is copied to the "dropins" folder for your server.

__a. Open a new Terminal window

__b. Use the "**tail -**f" command to view the messages.log file.

```
tail -f
/home/ibmdemo/Student/WLP_21.0.0.3/wlp/usr/servers/myServer/logs/messages.log
```

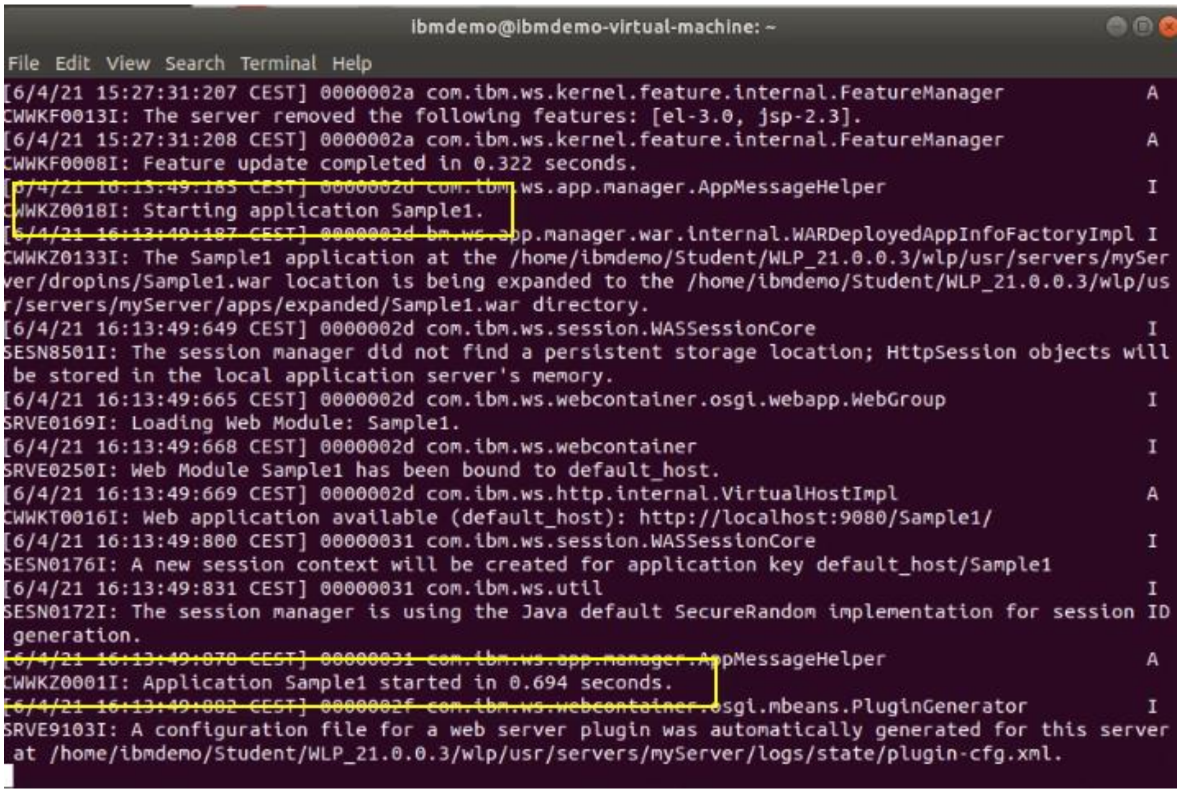
__2. The easiest way to deploy an application to Liberty is to copy it to the server's **dropins** directory.

The **Sample1.war** is provided for you in the lab. It is the same application that was deployed using the Eclipse IDE in the first part of the lab.

__a. Return to the Terminal Window that is at the directory: ~/Student/WLP_21.0.0.3/wlp __b. Copy the provided **Sample1.war** application to the dropins folder of your “myServer” server.

```
cp
/home/ibmdemo/Student/WLP_21.0.0.3/labs/gettingStarted/1_discover_*/Sample1.w
ar /home/ibmdemo/Student/WLP_21.0.0.3/wlp/usr/servers/myServer/dropins
```

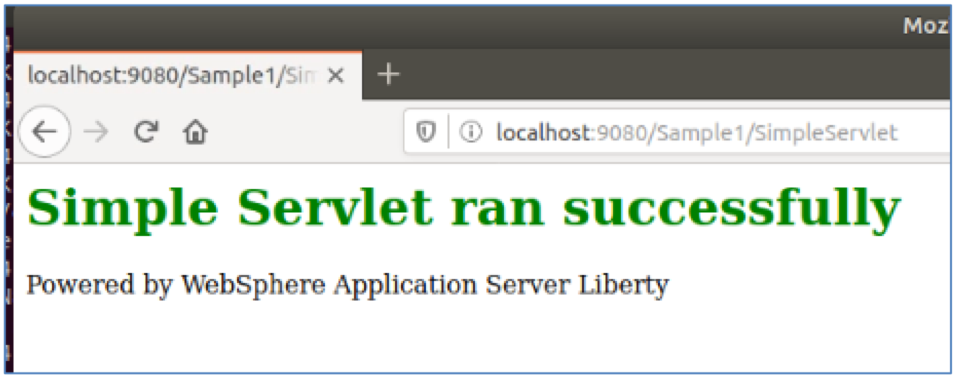
__3. Check the server’s **messages.log** to ensure that application deployment has taken place. You will see messages showing the Sample1 application being started.


ibmdemo@ibmdemo-virtual-machine: ~
File Edit View Search Terminal Help

```
[6/4/21 15:27:31:207 CEST] 0000002a com.ibm.ws.kernel.feature.internal.FeatureManager           A
CWWKF0013I: The server removed the following features: [el-3.0, jsp-2.3].
[6/4/21 15:27:31:208 CEST] 0000002a com.ibm.ws.kernel.feature.internal.FeatureManager           A
CWWKF0008I: Feature update completed in 0.322 seconds.
[6/4/21 16:13:49:185 CEST] 0000002d com.ibm.ws.app.manager.AppMessageHelper                     I
CWWKZ0018I: Starting application Sample1.
[6/4/21 16:13:49:187 CEST] 0000002d com.ibm.ws.app.manager.war.internal.WARDeployedAppInfoFactoryImpl I
CWWKZ0133I: The Sample1 application at the /home/ibmdemo/Student/WLP_21.0.0.3/wlp/usr/servers/mySer
ver/dropins/Sample1.war location is being expanded to the /home/ibmdemo/Student/WLP_21.0.0.3/wlp/us
r/servers/myServer/apps/expanded/Sample1.war directory.
[6/4/21 16:13:49:649 CEST] 0000002d com.ibm.ws.session.WASSessionCore                           I
SESN8501I: The session manager did not find a persistent storage location; HttpSession objects will
 be stored in the local application server's memory.
[6/4/21 16:13:49:665 CEST] 0000002d com.ibm.ws.webcontainer.osgi.webapp.WebGroup                I
SRVE0169I: Loading Web Module: Sample1.
[6/4/21 16:13:49:668 CEST] 0000002d com.ibm.ws.webcontainer                                     I
SRVE0250I: Web Module Sample1 has been bound to default_host.
[6/4/21 16:13:49:669 CEST] 0000002d com.ibm.ws.http.internal.VirtualHostImpl                    A
CWWKT0016I: Web application available (default_host): http://localhost:9080/Sample1/
[6/4/21 16:13:49:800 CEST] 00000031 com.ibm.ws.session.WASSessionCore                           I
SESN0176I: A new session context will be created for application key default_host/Sample1
[6/4/21 16:13:49:831 CEST] 00000031 com.ibm.ws.util                                             I
SESN0172I: The session manager is using the Java default SecureRandom implementation for session ID
 generation.
[6/4/21 16:13:49:878 CEST] 00000031 com.ibm.ws.app.manager.AppMessageHelper                     A
CWWKZ0001I: Application Sample1 started in 0.694 seconds.
[6/4/21 16:13:49:882 CEST] 0000002f com.ibm.ws.webcontainer.osgi.mbeans.PluginGenerator         I
SRVE9103I: A configuration file for a web server plugin was automatically generated for this server
 at /home/ibmdemo/Student/WLP_21.0.0.3/wlp/usr/servers/myServer/logs/state/plugin-cfg.xml.
```

__4. Check the application is running by opening a browser at:

http://localhost:9080/Sample1/SimpleServlet

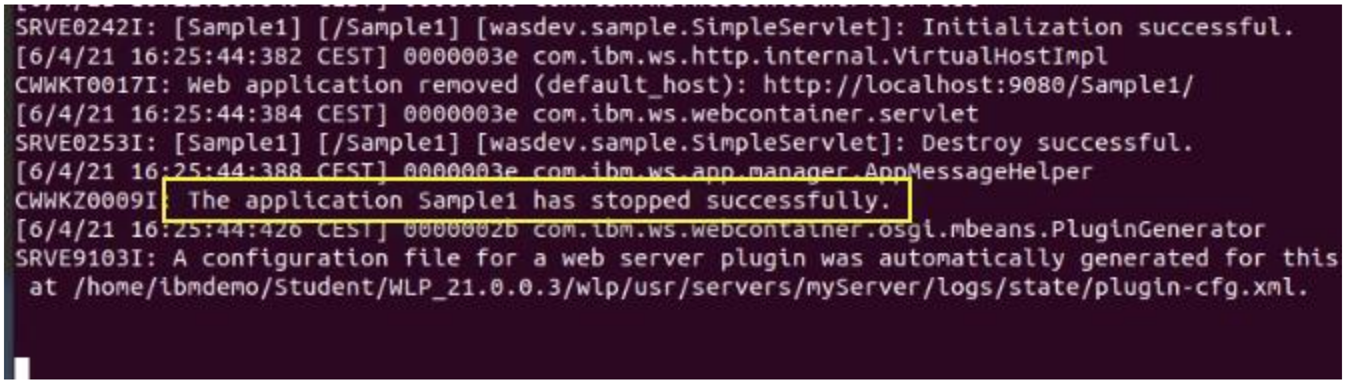

Moz
localhost:9080/Sample1/Sir
localhost:9080/Sample1/SimpleServlet

Simple Servlet ran successfully

Powered by WebSphere Application Server Liberty

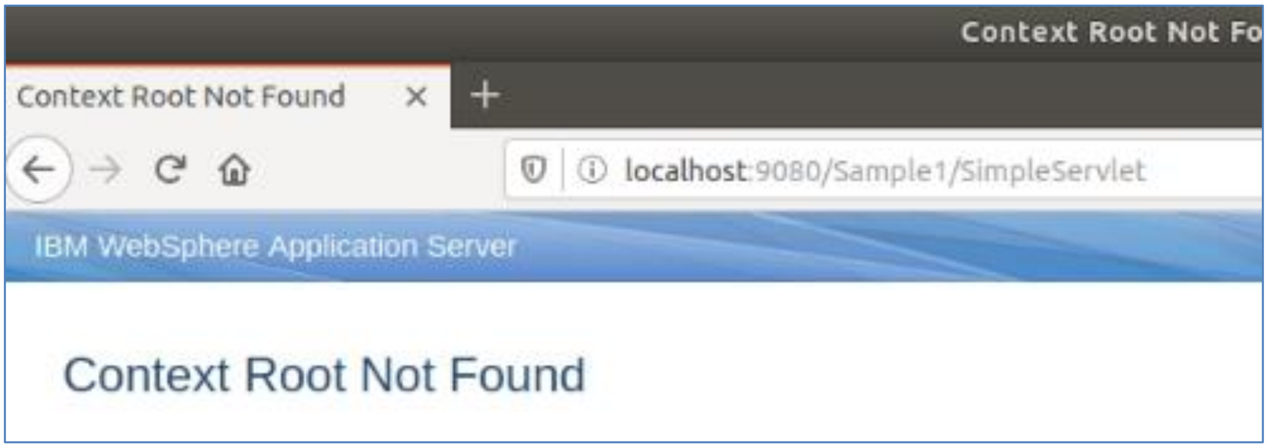
__5. Delete the Sample1.war file from the **dropins** directory, then check that it's no longer accessible from the browser.

```
rm /home/ibmdemo/Student/WLP_21.0.0.3/wlp/usr/servers/myServer/dropins/Sample1.war
```



```
SRVE0242I: [Sample1] [/Sample1] [wasdev.sample.SimpleServlet]: Initialization successful.
[6/4/21 16:25:44:382 CEST] 0000003e com.ibm.ws.http.internal.VirtualHostImpl
CWWKT0017I: Web application removed (default_host): http://localhost:9080/Sample1/
[6/4/21 16:25:44:384 CEST] 0000003e com.ibm.ws.webcontainer.servlet
SRVE0253I: [Sample1] [/Sample1] [wasdev.sample.SimpleServlet]: Destroy successful.
[6/4/21 16:25:44:388 CEST] 0000003e com.ibm.ws.app.manager.AppMessageHelper
CWWKZ0009I: The application Sample1 has stopped successfully.
[6/4/21 16:25:44:426 CEST] 0000002b com.ibm.ws.webcontainer.osgi.mbeans.PluginGenerator
SRVE9103I: A configuration file for a web server plugin was automatically generated for this
 at /home/ibmdemo/Student/WLP_21.0.0.3/wlp/usr/servers/myServer/logs/state/plugin-cfg.xml.
```

http://localhost:9080/Sample1/SimpleServlet


Context Root Not Fo
Context Root Not Found
localhost:9080/Sample1/SimpleServlet
IBM WebSphere Application Server

Context Root Not Found

3.2.2 Deploy the application by adding it to the server.xml file

While the **dropins** directory can be used for applications that do not require extra configuration, deploying the application by adding it to the server.xml file provides the freedom to configure the Liberty server based on the application configuration requirements.

In this section, you will deploy the Sample1 application by adding it to the server.xml fie.

In this case, you must put the application in one of the following locations:

- ${server.config.dir}/apps (that is, *server_directory*/user/servers/*server_name*/apps)
- ${shared.app.dir} (that is, *liberty_install_location*/usr/shared/apps)

__1. Copy the Sample1.war file to the ${server.config.dir}/apps directory.

```
cp
/home/ibmdemo/Student/WLP_21.0.0.3/labs/gettingStarted/1_discover_*/Sample1.w
ar /home/ibmdemo/Student/WLP_21.0.0.3/wlp/usr/servers/myServer/apps
```

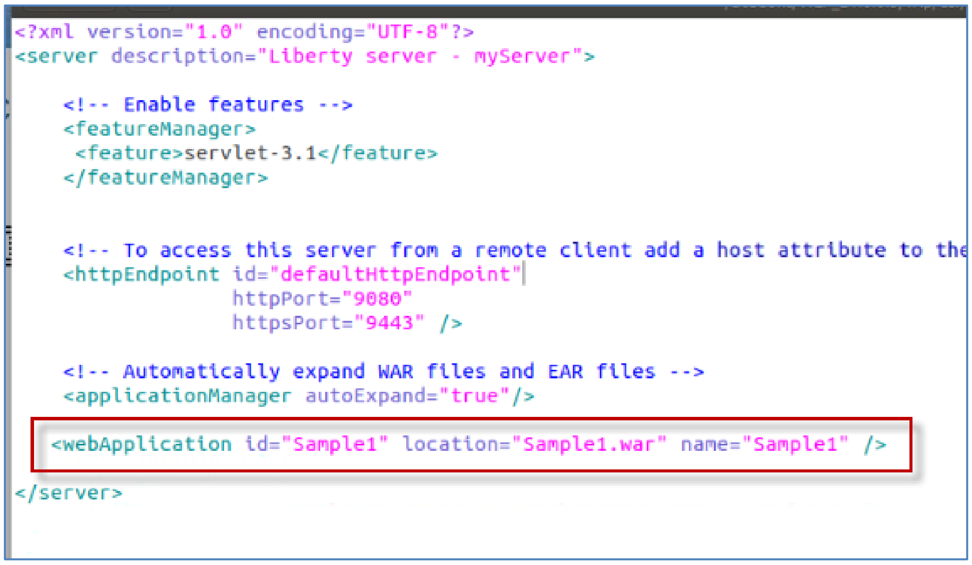
__2. Add the Sample1.war application configuration to the server.xml file.

__a. Use gedit to edit the **server.xml** and add the Sample1.war to the configuration, which points to the “apps” directory, by default.

```
gedit /home/ibmdemo/Student/WLP_21.0.0.3/wlp/usr/servers/myServer/server.xml
```

__b. Add the following line to the server,xml file, as illustrated below:

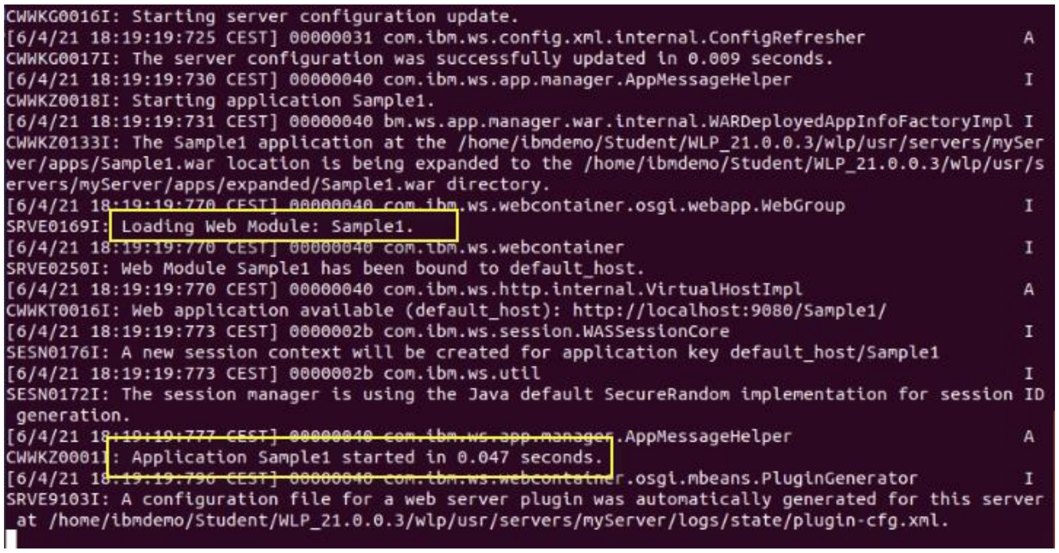
```
<webApplication id="Sample1" location="Sample1.war" name="Sample1" />
```



```
<?xml version="1.0" encoding="UTF-8"?>
<server description="Liberty server - myServer">

    <!-- Enable features -->
    <featureManager>
     <feature>servlet-3.1</feature>
    </featureManager>


    <!-- To access this server from a remote client add a host attribute to the
    <httpEndpoint id="defaultHttpEndpoint"
                  httpPort="9080"
                  httpsPort="9443" />

    <!-- Automatically expand WAR files and EAR files -->
    <applicationManager autoExpand="true"/>

      <webApplication id="Sample1" location="Sample1.war" name="Sample1" />

</server>
```

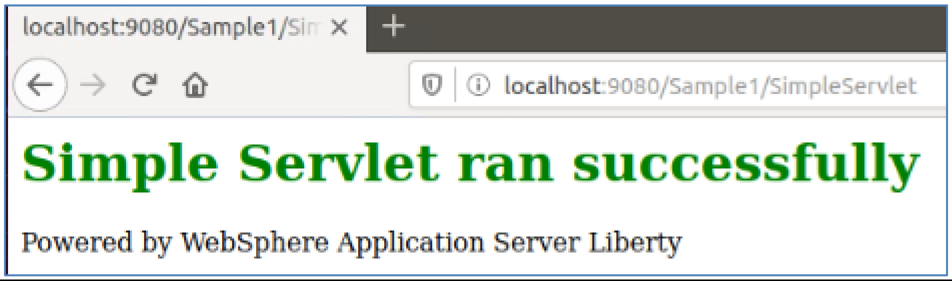
__c. **Save** and **CLOSE** the server.xml file

__3. Check the server's **messages.log** that the application is started.



```
CWWKG0016I: Starting server configuration update.
[6/4/21 18:19:19:725 CEST] 00000031 com.ibm.ws.config.xml.internal.ConfigRefresher          A
CWWKG0017I: The server configuration was successfully updated in 0.009 seconds.
[6/4/21 18:19:19:730 CEST] 00000040 com.ibm.ws.app.manager.AppMessageHelper                 I
CWWKZ0018I: Starting application Sample1.
[6/4/21 18:19:19:731 CEST] 00000040 bm.ws.app.manager.war.internal.WARDeployedAppInfoFactoryImpl I
CWWKZ0133I: The Sample1 application at the /home/ibmdemo/Student/WLP_21.0.0.3/wlp/usr/servers/mySer
ver/apps/Sample1.war location is being expanded to the /home/ibmdemo/Student/WLP_21.0.0.3/wlp/usr/s
ervers/myServer/apps/expanded/Sample1.war directory.
[6/4/21 18:19:19:770 CEST] 00000040 com.ibm.ws.webcontainer.osgi.webapp.WebGroup            I
SRVE0169I: Loading Web Module: Sample1.
[6/4/21 18:19:19:770 CEST] 00000040 com.ibm.ws.webcontainer                                 I
SRVE0250I: Web Module Sample1 has been bound to default_host.
[6/4/21 18:19:19:770 CEST] 00000040 com.ibm.ws.http.internal.VirtualHostImpl                A
CWWKT0016I: Web application available (default_host): http://localhost:9080/Sample1/
[6/4/21 18:19:19:773 CEST] 0000002b com.ibm.ws.session.WASSessionCore                       I
SESN0176I: A new session context will be created for application key default_host/Sample1
[6/4/21 18:19:19:773 CEST] 0000002b com.ibm.ws.util                                         I
SESN0172I: The session manager is using the Java default SecureRandom implementation for session ID
 generation.
[6/4/21 18:19:19:777 CEST] 00000040 com.ibm.ws.app.manager.AppMessageHelper                 A
CWWKZ0001I: Application Sample1 started in 0.047 seconds.
[6/4/21 18:19:19:796 CEST] 00000040 com.ibm.ws.webcontainer.osgi.mbeans.PluginGenerator     I
SRVE9103I: A configuration file for a web server plugin was automatically generated for this server
 at /home/ibmdemo/Student/WLP_21.0.0.3/wlp/usr/servers/myServer/logs/state/plugin-cfg.xml.
```

__4. Check the application is running by opening a browser at:
http://localhost:9080/Sample1/SimpleServlet


localhost:9080/Sample1/Sin
localhost:9080/Sample1/SimpleServlet

Simple Servlet ran successfully

Powered by WebSphere Application Server Liberty

3.3 Add INFO logging output to console

By default, the Liberty Server has the console log level set to AUDIT. In this section, you will change the level of log messages written to the console from AUDIT to INFO.

You will perform this activity by modifying **server.xml** file using an editor. It is also possible to set default logging options in the bootstrap.properties file. If the logging options are set in the bootstrap.properties file, the logging options will take effect very early in server startup, so it may be useful for debugging server initialization problems.

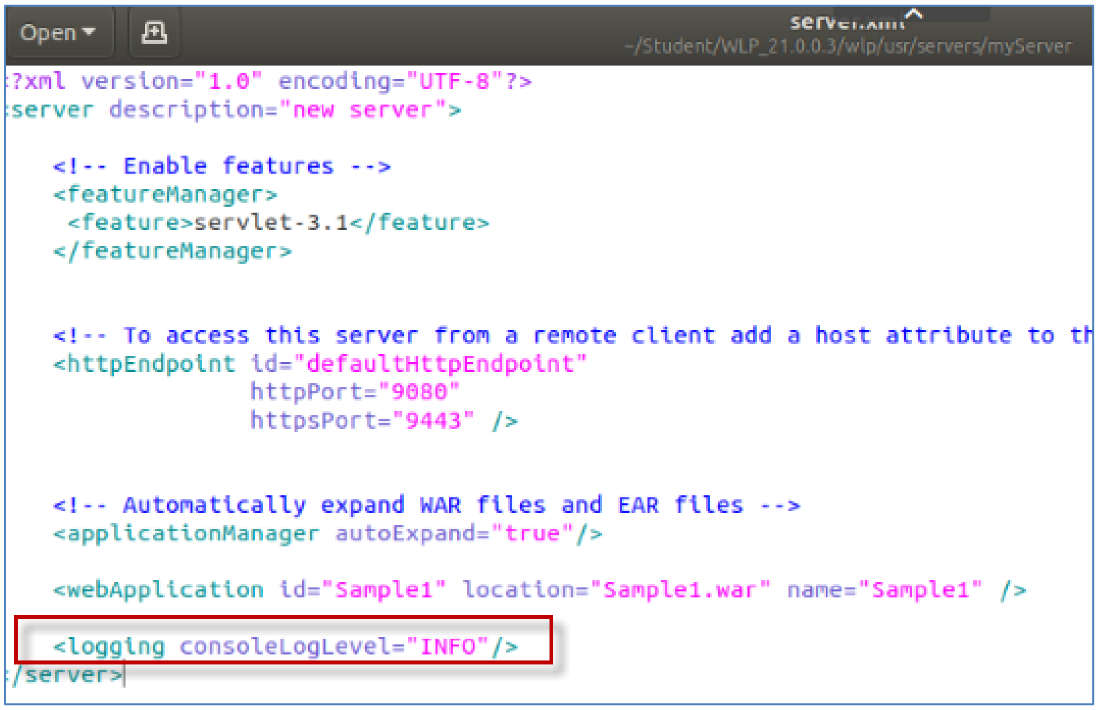
__1. Edit **server.xml** and add the logging configuration:

__a. Open the server.xml with the editor

```
gedit /home/ibmdemo/Student/WLP_21.0.0.3/wlp/usr/servers/myServer/server.xml
```

__b. Add the following line to the server.xml file to update the logging level from AUDIT to INFO.

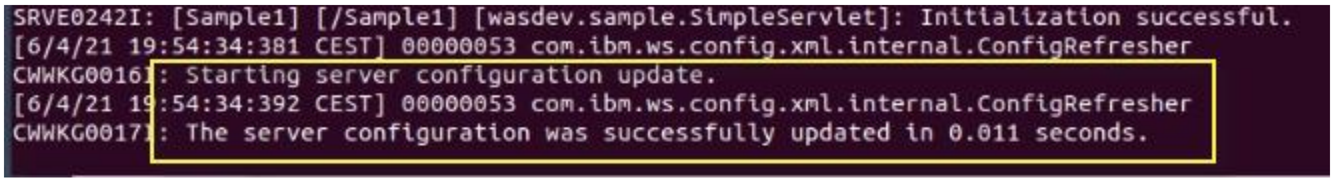
```
<logging consoleLogLevel="INFO"/>
```



```
?xml version="1.0" encoding="UTF-8"?>
server description="new server">

    <!-- Enable features -->
    <featureManager>
     <feature>servlet-3.1</feature>
    </featureManager>


    <!-- To access this server from a remote client add a host attribute to th
    <httpEndpoint id="defaultHttpEndpoint"
                  httpPort="9080"
                  httpsPort="9443" />


    <!-- Automatically expand WAR files and EAR files -->
    <applicationManager autoExpand="true"/>

    <webApplication id="Sample1" location="Sample1.war" name="Sample1" />

    <logging consoleLogLevel="INFO"/>
/server>
```

__c. **Save** the changes, and check messages.log that the configuration change was updated.

__d. From the terminal window where “**tail -f**” command is running, verify the server configuration was updated.



```
SRVE0242I: [Sample1] [/Sample1] [wasdev.sample.SimpleServlet]: Initialization successful.
[6/4/21 19:54:34:381 CEST] 00000053 com.ibm.ws.config.xml.internal.ConfigRefresher
CWWKG0016I: Starting server configuration update.
[6/4/21 19:54:34:392 CEST] 00000053 com.ibm.ws.config.xml.internal.ConfigRefresher
CWWKG0017I: The server configuration was successfully updated in 0.011 seconds.
```

3.4 Update trace specification

By default, the Liberty Server trace specification is set to *=info=enabled. This is the same for Traditional WebSphere Application Server.

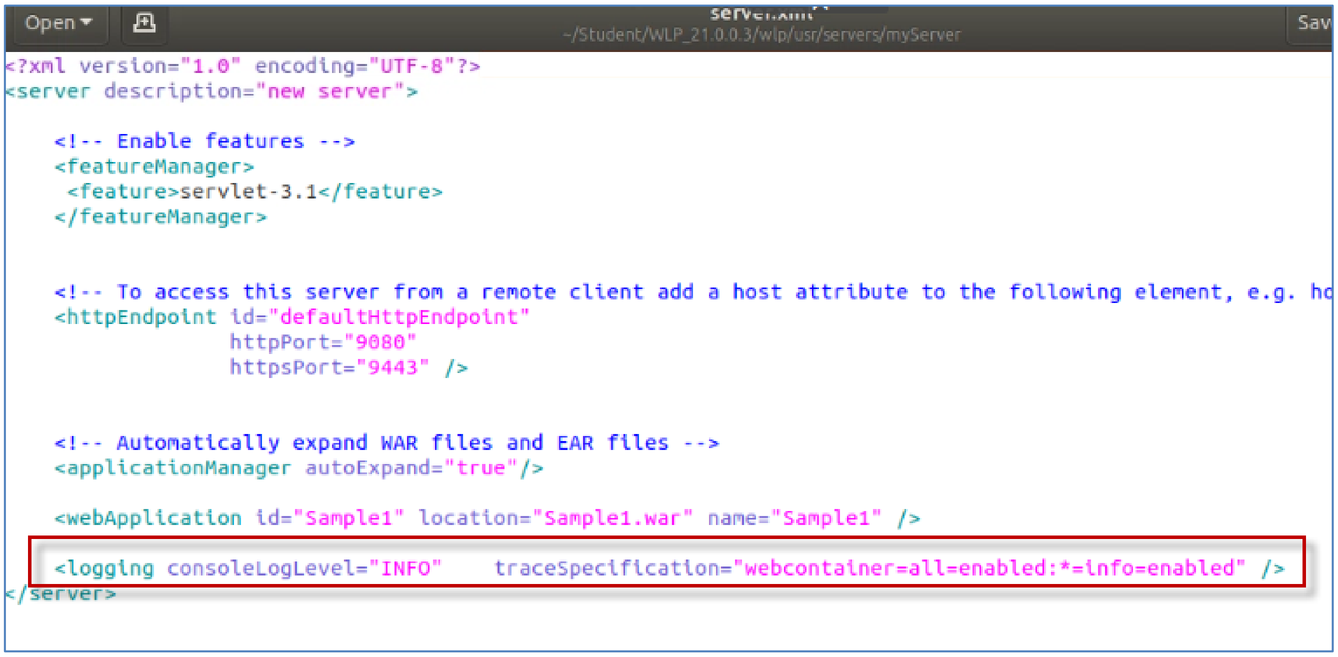
Updating the trace specification for debugging is easily performed by updating server.xml.

__1. Update the logging stanza to include a trace specification for tracing the webcontainer.

__a. Use gedit to edit **server.xml** and **update** the logging stanza to:

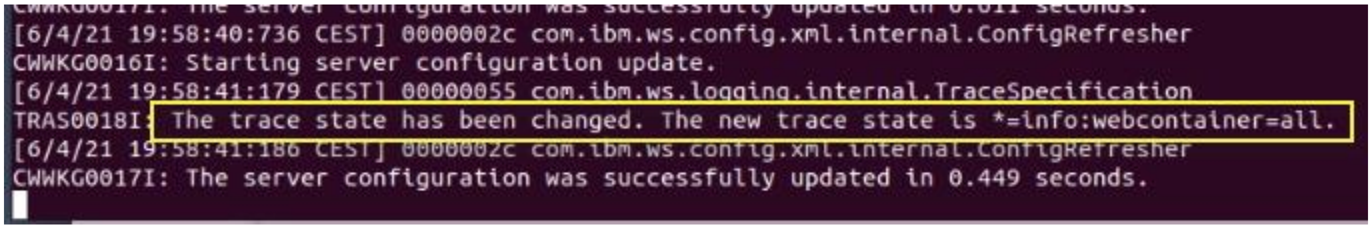
```
<logging consoleLogLevel="INFO"
traceSpecification="webcontainer=all=enabled:*=info=enabled" />
```

__b. **Save** the server.xml file:



```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
     <feature>servlet-3.1</feature>
    </featureManager>


    <!-- To access this server from a remote client add a host attribute to the following element, e.g. ho
    <httpEndpoint id="defaultHttpEndpoint"
                  httpPort="9080"
                  httpsPort="9443" />


    <!-- Automatically expand WAR files and EAR files -->
    <applicationManager autoExpand="true"/>

    <webApplication id="Sample1" location="Sample1.war" name="Sample1" />

    <logging consoleLogLevel="INFO"    traceSpecification="webcontainer=all=enabled:*=info=enabled" />
</server>
```

__c. Verify the messages.lof file logged the updtaed trace specfication



```
[6/4/21 19:58:40:736 CEST] 0000002c com.ibm.ws.config.xml.internal.ConfigRefresher
CWWKG0016I: Starting server configuration update.
[6/4/21 19:58:41:179 CEST] 00000055 com.ibm.ws.logging.internal.TraceSpecification
TRAS0018I: The trace state has been changed. The new trace state is *=info:webcontainer=all.
[6/4/21 19:58:41:186 CEST] 0000002c com.ibm.ws.config.xml.internal.ConfigRefresher
CWWKG0017I: The server configuration was successfully updated in 0.449 seconds.
```

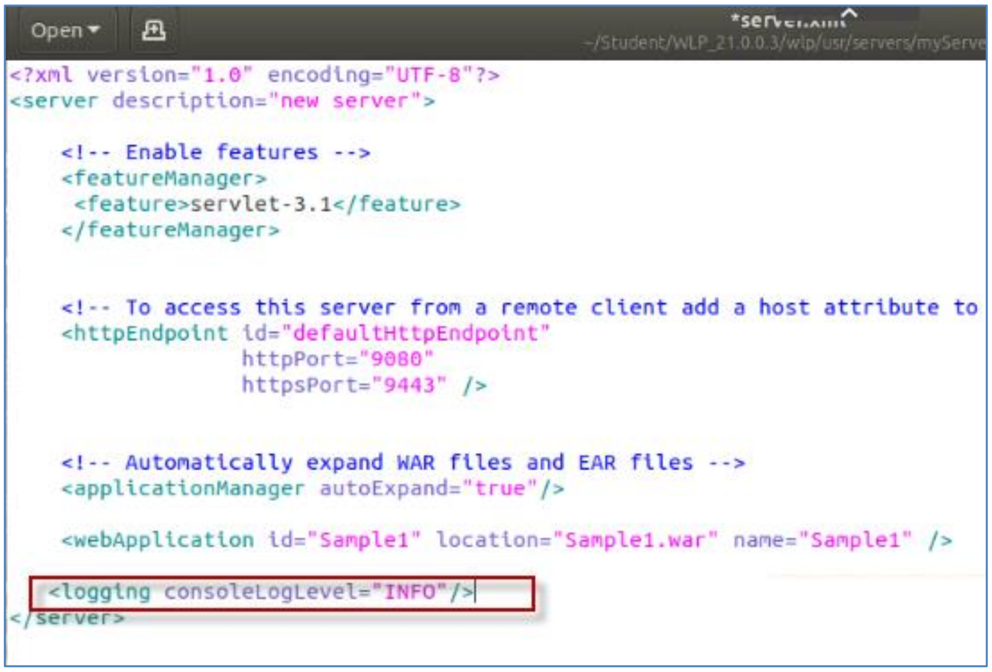
__2. Use **CTRL-C** to **Stop** the "**tail -f**" command that is running in the terminal window.

__3. Verify that the trace file contains trace data. The trace file is located at:

```
cat /home/ibmdemo/Student/WLP_21.0.0.3/wlp/usr/servers/myServer/logs/trace.log
```

__**4. Very important: avoid excessive tracing by reverting the trace specification back to :**

```
<logging consoleLogLevel="INFO"/>
```



```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
     <feature>servlet-3.1</feature>
    </featureManager>


    <!-- To access this server from a remote client add a host attribute to
    <httpEndpoint id="defaultHttpEndpoint"
                  httpPort="9080"
                  httpsPort="9443" />


    <!-- Automatically expand WAR files and EAR files -->
    <applicationManager autoExpand="true"/>

    <webApplication id="Sample1" location="Sample1.war" name="Sample1" />

   <logging consoleLogLevel="INFO"/>
</server>
```

__5. **Save** the changes to the server.xml file.

__6. **Close** the editor

3.5 Customizing Liberty JVM Options

The generic JVM arguments are used to configure and adjust how the JVM executes.

The WebSphere Application Server Liberty is pre-configured with minimal settings defined. The following steps will direct you how to define custom generic JVM arguments such as heap settings for a Liberty server.

__1. Create a **jvm.options** file that affects all servers

__a. Change directory to {LAB_HOME}/wlp/usr/servers/myServer

```
cd /home/ibmdemo/Student/WLP_21.0.0.3/wlp/usr/servers/myServer
```

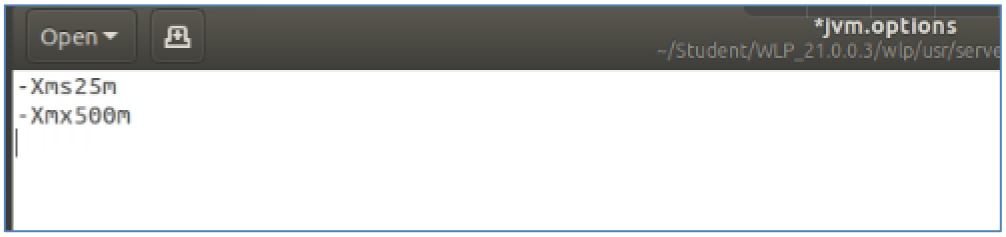
__b. Create a new file named **jvm.options** in the “myServer” directory

```
gedit  jvm.options
```

__c. Add the following lines to set the minimum and maximum heap size. The following options will set the min / max JVM heap size to 25 MB and 500 MB respectively.

-Xms25m

-Xmx500m



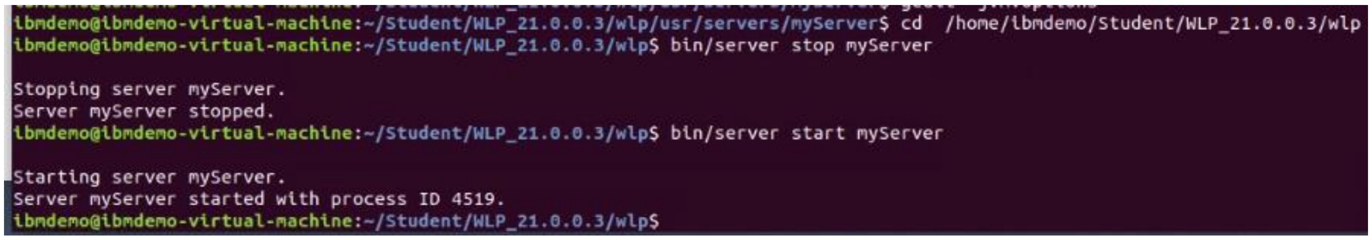
```
-Xms25m
-Xmx500m
```

__2. **Save** and **close** the jvm.options file

__3. **Restart** the server for the changes to take affect

```
cd  /home/ibmdemo/Student/WLP_21.0.0.3/wlp

bin/server stop myServer
```

```
bin/server start myServer
```



```
ibmdemo@ibmdemo-virtual-machine:~/Student/WLP_21.0.0.3/wlp/usr/servers/myServer$ cd  /home/ibmdemo/Student/WLP_21.0.0.3/wlp
ibmdemo@ibmdemo-virtual-machine:~/Student/WLP_21.0.0.3/wlp$ bin/server stop myServer

Stopping server myServer.
Server myServer stopped.
ibmdemo@ibmdemo-virtual-machine:~/Student/WLP_21.0.0.3/wlp$ bin/server start myServer

Starting server myServer.
Server myServer started with process ID 4519.
ibmdemo@ibmdemo-virtual-machine:~/Student/WLP_21.0.0.3/wlp$
```

__4. **Stop** the server

```
cd  /home/ibmdemo/Student/WLP_21.0.0.3/wlp

bin/server stop myServer
```

This concludes the customization portion of the lab using command line. In the next sections, you will be introduced to the Liberty configuration files for customizing the server initialization and environment settings.

4 Introducing Liberty Environment Variable Configuration

You can customize the Liberty environment using certain specific variables to support the placement of product binaries and shared resources.

The Liberty environment variables are specified using **server.env** file.

You can use **server.env** file at the installation and server levels to specify environment variables such as JAVA_HOME, WLP_USER_DIR and WLP_OUTPUT_DIR.

Important:

NOTE: You will **NOT** modify the default environment configuration in this lab.

Review the information in this section to become familiar with the environment variables that are available for customizing the Liberty environment.

The following Liberty specific variables can be used to customize the Liberty environment:

- **${wlp.install.dir}**

  This configuration variable has an inferred location. The installation directory is always set to the parent of the directory containing the launch script or the parent of the /lib directory containing the target jar files.

  TIP: For this lab, the *${wlp.install.dir}* is ""**{LAB_HOME}/wlp**"

- **WLP_USER_DIR**

  This environment variable can be used to specify an alternate location for the ${wlp.install.dir}/usr. This variable can only be an absolute path. If this is specified, the runtime environment looks for shared resources and server definition in the specified directory.

  The ${server.config.dir} is equivalent to ${wlp.user.dir}/servers/serverName and can be set to a different location when running a server (to use configuration files from a location outside wlp.user.dir).

  TIP: For this lab, the ${server.config.dir} is depending on the used server either "**{LAB_HOME}/wlp*/usr/servers/labServer***" or "**{LAB_HOME}/wlp*/usr/servers/myServer***".

- **WLP_OUTPUT_DIR**

  This environment variable can be used to specify an alternate location for server generated output such as logs, the workarea directory and generated files. This variable can only be an absolute path.

  If this environment variable is specified, ${server.output.dir} is set to the equivalent of WLP_OUTPUT_DIR/serverName. If not specified, the ${server.output.dir} is the same as ${server.config.dir} .

  TIP: For this lab, the ${server.output.dir} is **{LAB_HOME}/wlp*/usr/servers/labServer***” or “**{LAB_HOME}/wlp*/usr/servers/myServer***”, which is the same as ${server.config.dir}.

5 Introducing Liberty bootstrap.properties

In this section of the lab, you will gain an understanding of how and when bootstrap properties are required during environment initialization.

Important:

NOTE: You will **NOT** modify the default environment configuration in this lab.

This information is provided in the lab for your reference. .

Bootstrap properties are used to initialize the runtime environment for a particular server. Generally, they are attributes that affect the configuration and initialization of the runtime.

Bootstrap properties are set in a text file named **bootstrap.properties**. This file should be located in the server directory alongside the configuration root file server.xml.

By default, the server directory is **usr/servers/*server_name***.

The **bootstrap.properties** file contains two types of properties:

- A small, predefined set of initialization properties.
- Any custom properties you choose to define which you can then use as variables in other configuration files (that is, server.xml and included files).

You can create the bootstrap.properties through any file creation mechanism, or by using the same method shown above for creation of the jvm.options file in eclipse. You can edit the bootstrap.properties file using a text editor, or using the editor that is part of the Liberty developer tools.

Changes to the bootstrap.properties file are applied when the server is restarted.

TIP:

As an example, the logging service can be controlled through the server configuration (server.xml) file. Occasionally you need to set logging properties so they can take effect before the server configuration files are processed.

In this case you set them in the bootstrap.properties file instead of the server configuration.

You do not usually need to do this to get logging from your own code, which is loaded after server configuration processing, but you might need to do this to analyze problems in early server start or configuration processing.

=== END OF LAB ===