

Discover Liberty



Contents

LAB 1

DISCOVER LIBERTY.....3

1.1 EXPLORING THE LIBERTY SERVER4

1.2 DEPLOYING A SAMPLE APPLICATION TO LIBERTY10

1.2.1 IMPORT A SAMPLE APPLICATION INTO ECLIPSE10

1.2.2 MODIFY THE APPLICATION.....15

1.2.3 MODIFY THE SERVER HTTP(S) PORTS.....18

1.3 ADD INFO LOGGING OUTPUT TO CONSOLE22

1.4 UPDATE TRACE SPECIFICATION25

1.5 CUSTOMIZING LIBERTY JVM OPTIONS28

1.6 INTRODUCING LIBERTY ENVIRONMENT VARIABLE CONFIGURATION.....30

1.7 INTRODUCING LIBERTY BOOTSTRAP.PROPERTIES32

1.8 RUNNING LIBERTY AS A WINDOWS SERVICE33

APPENDIX A. NOTICES.....35

APPENDIX B. TRADEMARKS AND COPYRIGHTS.....37

Lab 1 Discover Liberty

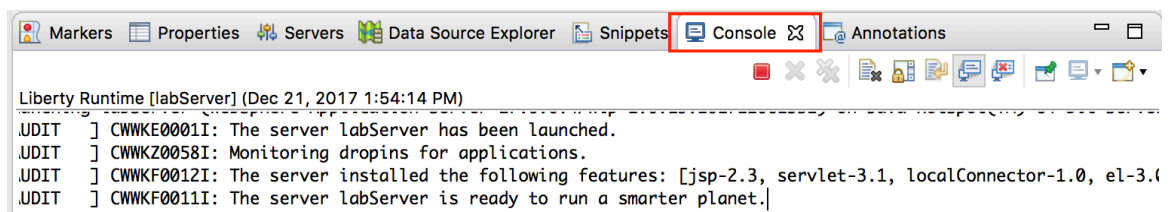
In this lab we will perform the initial set up required for all the labs and explore Liberty. The instructions assume a Windows environment, but Linux and Mac differences are presented. Where applicable, substitute with Linux or Mac equivalent, such as path names.

Please refer to the following table for file and resource location references on different operating systems.

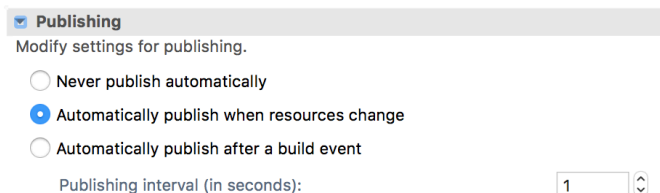
Location Ref.	OS	Absolute Path
{LAB_HOME}	Windows	C:\WLP_<version>
	Linux	~/WLP_<version> Or your choice
	Mac OSX	~/WLP_<version> Or your choice

1.1 Exploring the Liberty Server

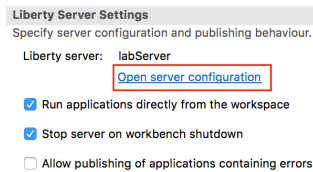
- ___1. Start the server in eclipse.
 - ___a. From the **Servers** view, select your **labServer** instance and click the **Start the server** button (🟢). Alternatively, you can also right-click the server name and choose the **Start** option from the context menu.
 - ___b. Switch to the **Console** view if necessary. Look at the messages to see how fast your server starts!



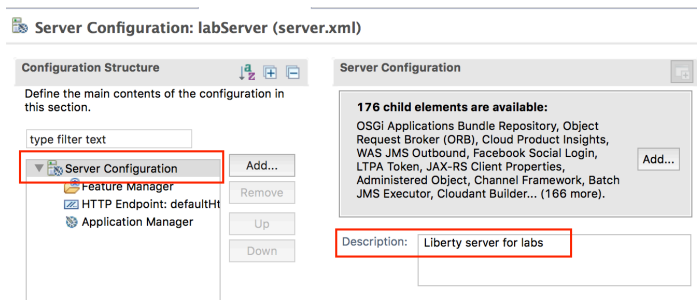
- ___2. Modify the lab server configuration.
 - ___a. In the **Servers** view, double-click on your **labServer** server to open the server Overview (or right-click and select **Open** from the context menu).
 - ___b. First, expand the **Publishing** section and notice that the server is set to automatically detect and publish changes. Keep this default setting.



- ___c. In this exercise, you will be deploying a simple servlet application, so try enabling the servlet feature on this server. On the **Overview** page, locate the **Liberty Server Settings** section, and click the **Open server configuration** link to open the server.xml editor.




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




- ___e. To add a feature, such as servlet-3.1, go back in the **Configuration Structure** area, and determine if the **Feature Manager** has already been added to the configuration.


- ___i. The Feature Manager will already exist in the configuration if the Liberty Server configuration already has features defined, such as jsp-2.3. Review the Feature Manager settings. In this lab, the Feature Manager has already been added to the configuration. Click on “Feature Manager” to view the list of features already configured.


 **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

Add...

Remove


Up

Down

Feature Manager

Set the features that are enabled on this server.

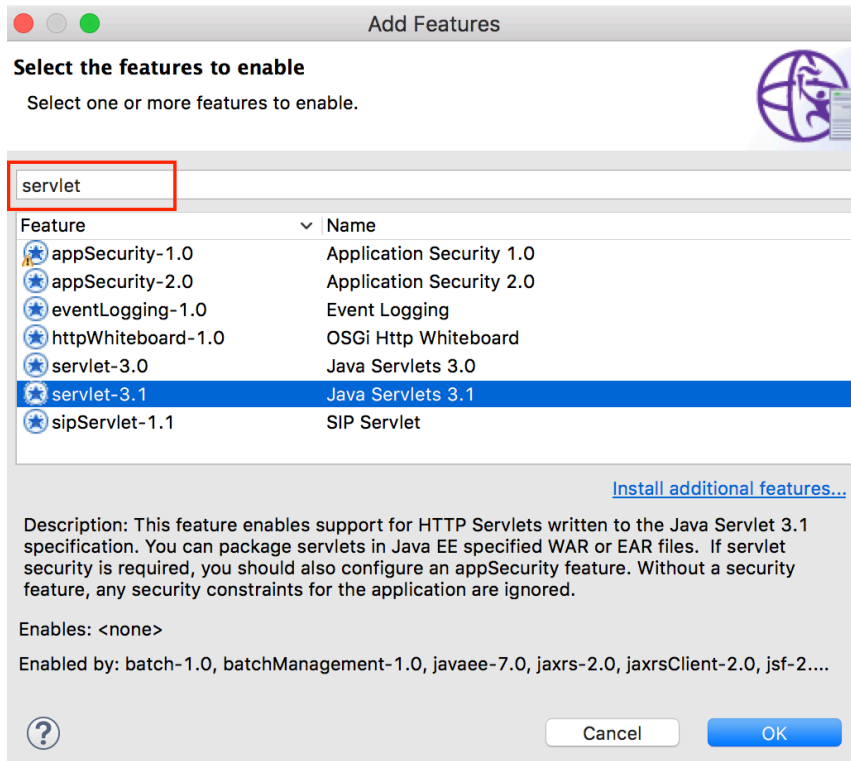
Feature:

Feature	Name
 jsp-2.3	JavaServer Pages 2.3

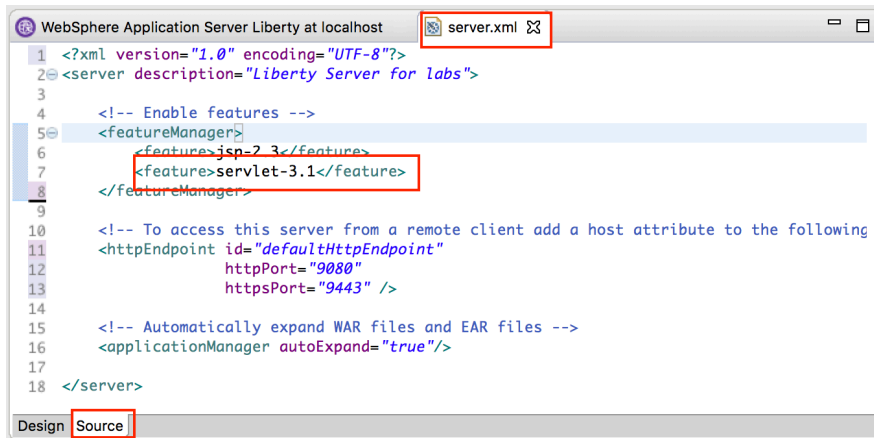
Add... Remove


Description:

- __f. Click the Add button
- __g. In the pop-up, type **servlet** to filter to servlet related features. Then select **servlet-3.1**. Click **OK**



- ___h. 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 added, and that it contains the **servlet-3.1** feature.



- ___i. Now you have a server that is configured to use the **servlet-3.1** feature. Click the **Save** button () to save your changes (or use CTRL+S)

Switch to the **Console** panel 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.

- __i. 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.

```
[AUDIT ] CWWKE0001I: The server labServer has been launched.  
[AUDIT ] CWWKZ0058I: Monitoring dropins for applications.  
[AUDIT ] CWWKF0012I: The server installed the following features: [jsp-2.3, servlet-3.1, localConnector-1.  
[AUDIT ] CWWKF0011I: The server labServer is ready to run a smarter planet.  
[AUDIT ] CWWKG0016I: Starting server configuration update.  
[AUDIT ] CWWKF0008I: Feature update completed in 0.021 seconds.  
[AUDIT ] CWWKG0017I: The server configuration was successfully updated in 0.036 seconds.
```

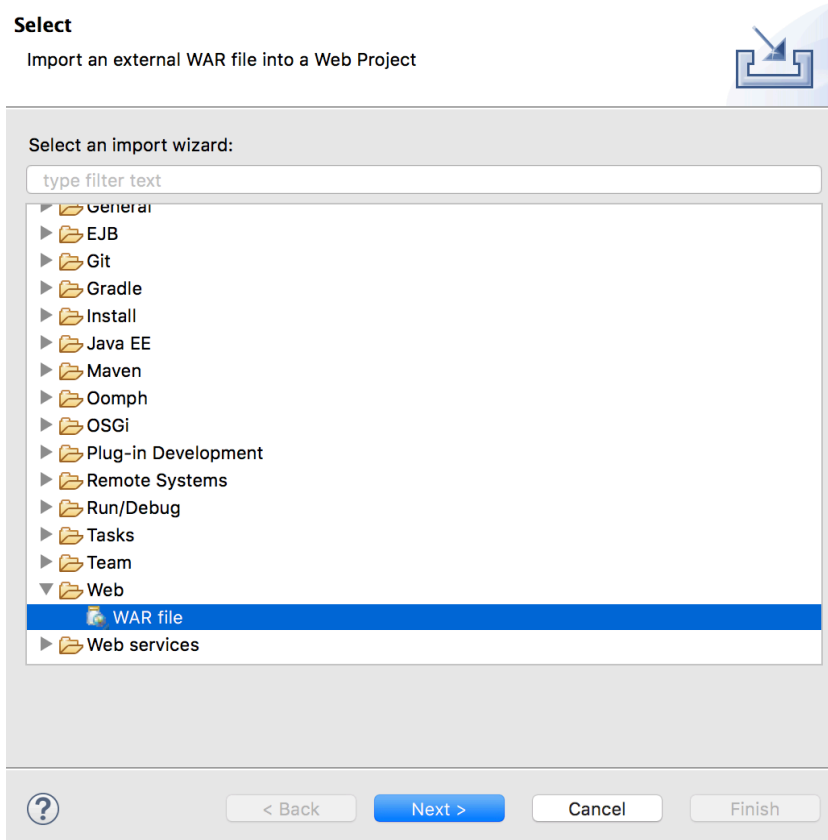
- __j. Now you are ready to start working with a sample application that uses the Servlet feature.

1.2 Deploying a sample application to Liberty

1.2.1 Import a sample application into Eclipse


___3. 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 **{LAB_HOME}\labs\gettingStarted\1_discover_<timestamp>\Sample1.war** and click **Open**.
- ___c. Ensure the **Target runtime** is set to **WebSphere Application Server Liberty**.
- ___d. **Unselect “Add project to an EAR”**
- ___e. Click **Finish**

WAR Import
Import a WAR file from the file system



WAR file:

Web project:

Target runtime:

EAR membership

☐ Add project to an EAR

EAR project name:

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

- __g. Now you have a **Sample1** web project in your workspace. You can expand it in the **Enterprise Explorer** view to see the different components of the project.



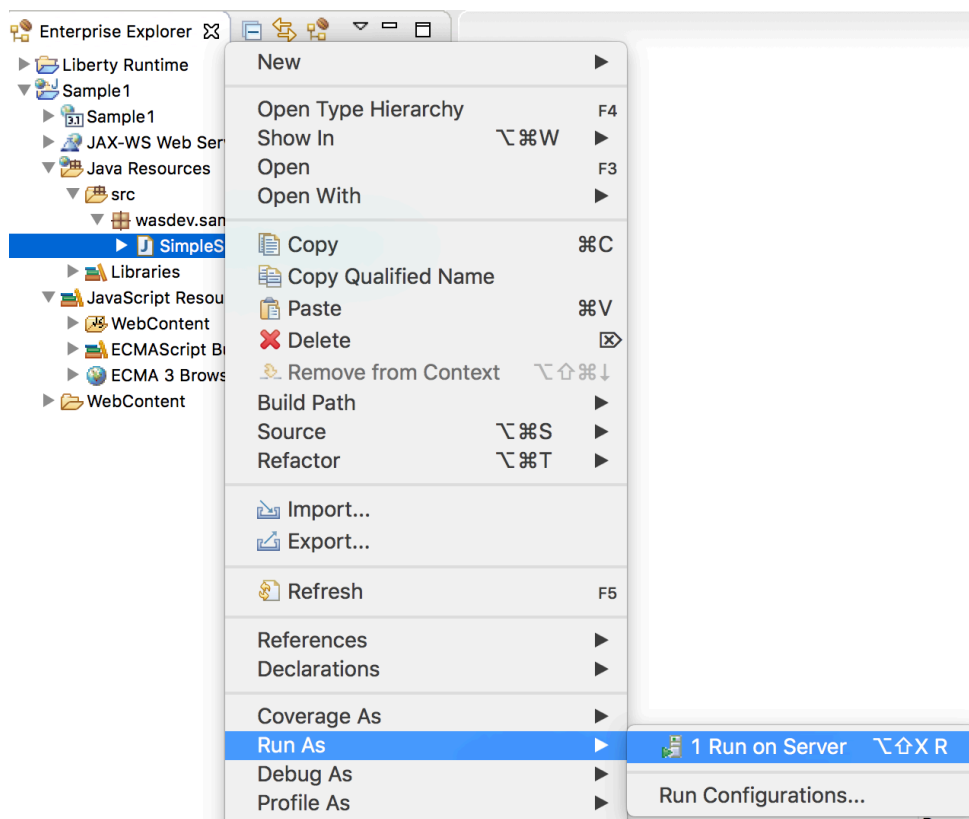
- __4. Start the sample application.

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

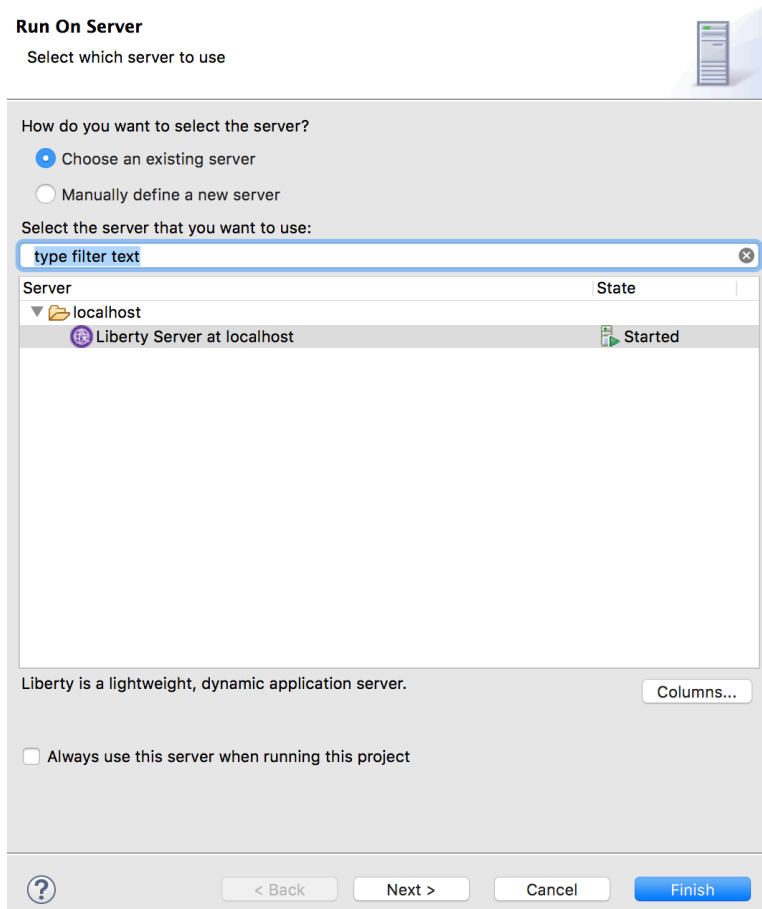
__i. **Sample1 -> Java resources -> src -> wasdev.sample -> SimpleServlet.java**

- __b. Right-click on **SimpleServlet.java**. **Note:** If right click on SimpleServlet.class file you will get a Run As > Run Configuration, select the java file.

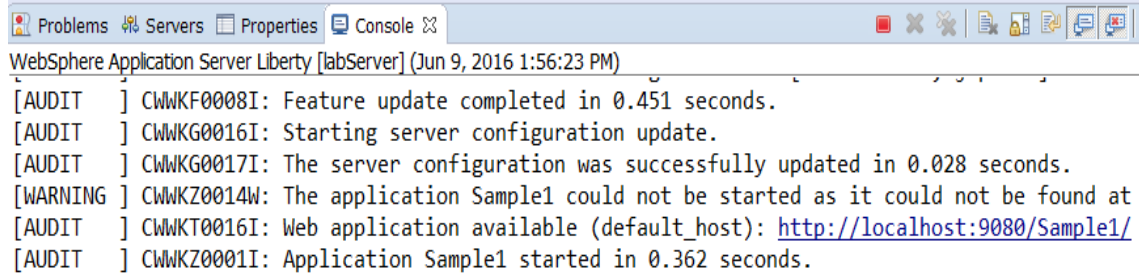
- __c. From the context menu, select **Run As > Run on Server**.



- __d. In the **Run On Server** dialog, verify that **Choose an existing server** is chosen.
- __i. Under **localhost**, select the **Liberty Server** that you defined earlier. The server should be listed in **Started** state.
- __ii. Click **Finish**.

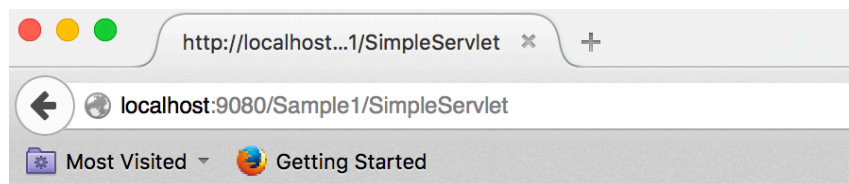


- ___e. After a moment, your application will be installed and started. See the **Console** pane for the corresponding messages.



```
WebSphere Application Server Liberty [labServer] (Jun 9, 2016 1:56:23 PM)
[AUDIT   ] CWWKF0008I: Feature update completed in 0.451 seconds.
[AUDIT   ] CWWKG0016I: Starting server configuration update.
[AUDIT   ] CWWKG0017I: The server configuration was successfully updated in 0.028 seconds.
[WARNING ] CWWKZ0014W: The application Sample1 could not be started as it could not be found at
[AUDIT   ] CWWKT0016I: Web application available (default_host): http://localhost:9080/Sample1/
[AUDIT   ] CWWKZ0001I: Application Sample1 started in 0.362 seconds.
```

- ___f. In the main panel of the workbench, a browser also opened, pointing to <http://localhost:9080/Sample1/SimpleServlet>.
- ___g. If you receive a 404 the first time, try to refresh the browser once the application is completely deployed and started.
- ___h. At this point, you should see the rendered HTML content generated by the simple servlet.



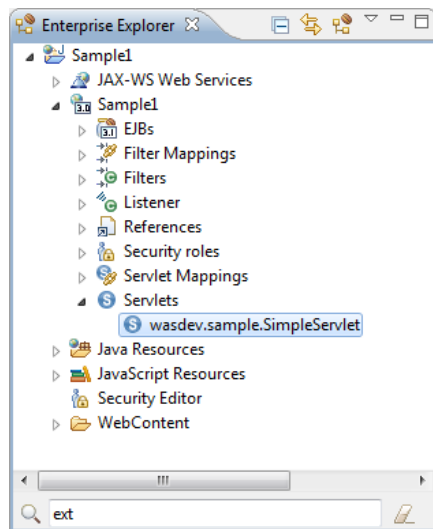
Simple Servlet ran successfully

Powered by WebSphere Application Server Liberty

1.2.2 Modify the application

___1. Open the servlet java source code.

- ___a. In the **Enterprise Explorer** panel, expand the **Sample1** project, then go to **Sample1 > Servlets**. Double-click the **wasdev.sample.SimpleServlet** entry to open the Java editor for the servlet.



- ___b. This is how the SimpleServlet.java source looks in the editor:

```
SimpleServlet.java  http://localhost:9080/Sample1/SimpleServlet
1  package wasdev.sample;
2
3  import java.io.IOException;
10
11 /**
12  * Servlet implementation class SimpleServlet
13  */
14 @WebServlet("/SimpleServlet")
15 public class SimpleServlet extends HttpServlet {
16     private static final long serialVersionUID = 1L;
17
18     /**
19      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
20      */
21     @Override
22     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
23         response.getWriter().print("<html><h1><font color=green>Simple Servlet ran successfully</font></h1></html>"
24                                   + "<html>Powered by WebSphere Application Server Liberty</html>");
25     }
26
27 }
28
```

- ___c. 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).

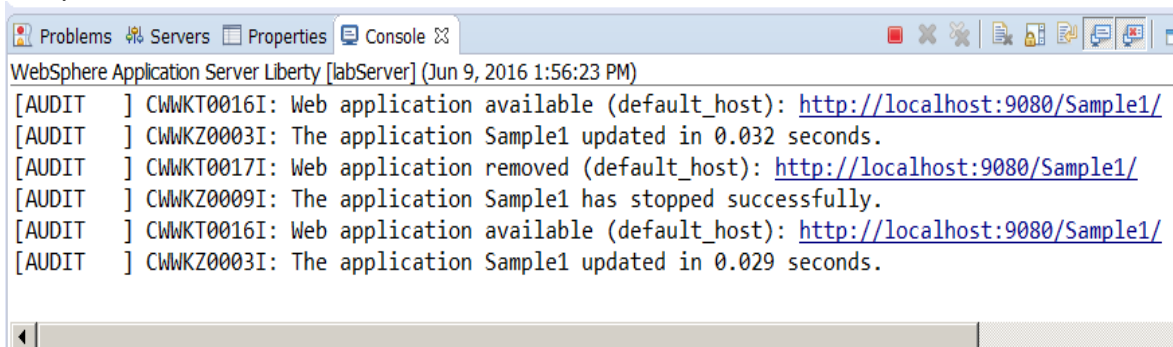
```
/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    response.getWriter().print(
        "<html><h1><font color=green>Simple Servlet ran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

- ___2. 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. Modify this string by changing the text green to **purple**, so your font tag will look read ****.

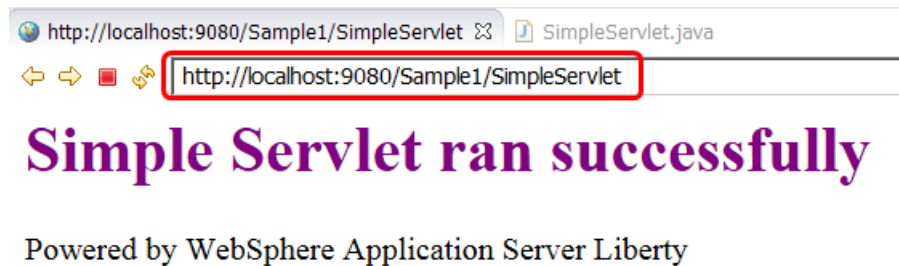
```
response.getWriter().print(
    "<html><h1><font color=purple>Simple Servlet ran
successfully</font></html>"
    + "<html>Powered by WebSphere Application Server Liberty</html>");
```

- ___b. Save your changes to the Java source file by either clicking the **Save** button (📁) or using **CTRL+S**.
- ___c. 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.
- ___d. 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.



__3. 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.

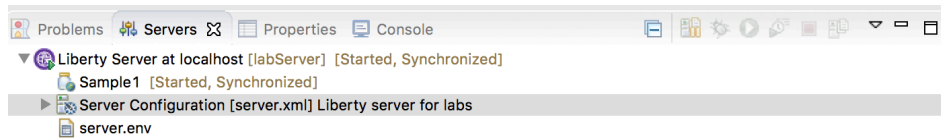


- __b. Optionally continue to play around with application modifications and see how quickly those changes are available in the deployed application. Maybe put in some additional text to display on the page, or add extra HTML tags to see formatting changes (you could add a title tag to set the text displayed in the browser title bar, for example, `<head><title>Liberty Server</title></head>`).
- __c. The key is that this **edit / publish / debug cycle is very simple and fast!**

1.2.3 Modify the server HTTP(s) ports

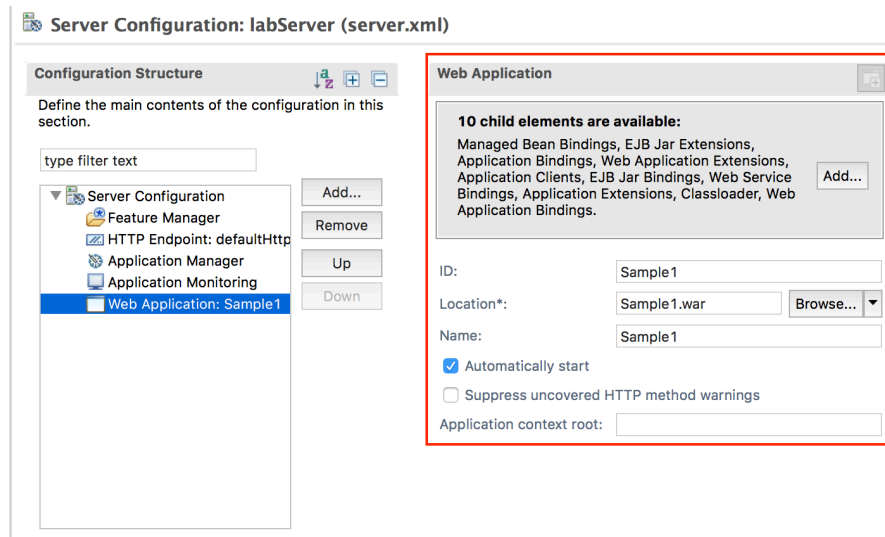
___1. Open the server configuration editor.

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

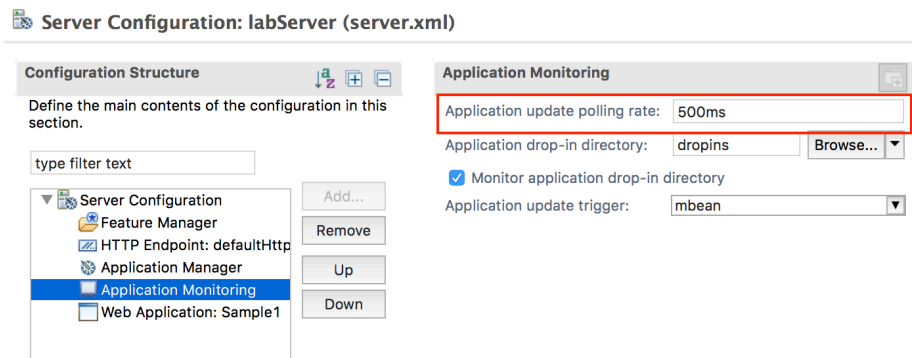


___b. Ensure you are in the Design mode by selecting the **Design** tab on the Server Configuration editor.

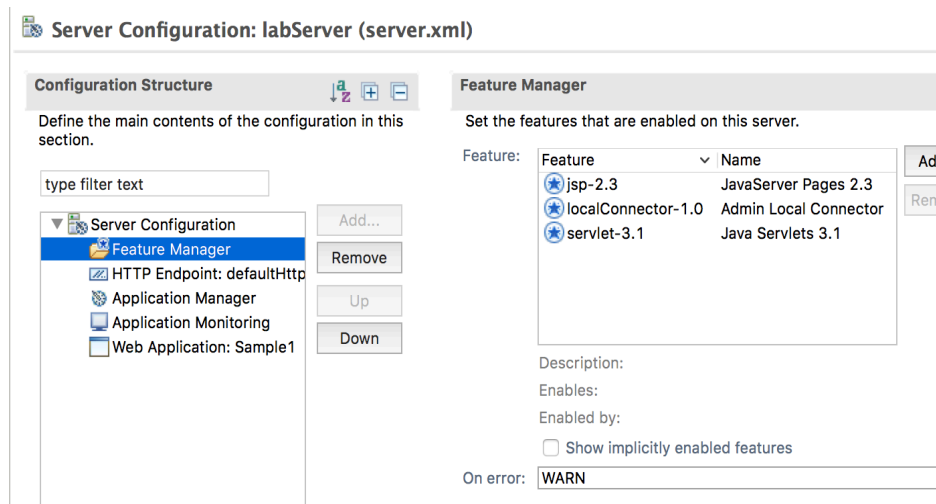
___c. Select the **Web Application: Sample1** item in the **Server Configuration** and look at its configuration details. From here, you can set basic application parameters, including the context root for the application.



- ___d. 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. You did not add any JMX features to your server to support mbean notification – so how is that working?



- ___e. Select the **Feature Manager** item to see the features that are configured on your server. You added the **servlet-3.1** 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.



__2. Change the HTTP port.

- __a. 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.
- __b. 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.

type filter text

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

Add... Remove Up Down

HTTP Endpoint

4 child elements are available:
HTTP Options, TCP Options, SSL Options, HTTP Access Logging. Add...

ID: defaultHttpEndpoint

On error: WARN

☒ Enabled

Host: localhost

Port: 9080

Secure port: 9443

HTTP options reference: Add

SSL options reference: Add

TCP options reference: Add

HTTP access logging reference: Add

- __c. In the **HTTP Endpoint Details** area, Change the HTTP Port to 9580.
- __i. Update the **Port** field to **9580**.

HTTP Endpoint

4 child elements are available:
HTTP Options, TCP Options, SSL Options, HTTP Access Logging. Add...

ID: defaultHttpEndpoint

On error: WARN

☒ Enabled

Host: localhost

Port: 9085

Secure port: 9443

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

- ___e. 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.

```

1 <?xml version='1.0' encoding='UTF-8'?>
2 <server description="Liberty server for labs">
3   <!-- Enable features -->
4   <featureManager>
5     <feature>jsp-2.2</feature>
6     <feature>servlet-3.1</feature>
7     <feature>localConnector-1.0</feature>
8   </featureManager>
9
10  <!-- To access this server from a remote client add a host attribute
11  <httpEndpoint httpPort="9580" httpsPort="9443" id="defaultHttpEndpoint"
12  </httpEndpoint>
13  <!-- Automatically expand WAR files and EAR files -->
14  <applicationManager autoExpand="true"/>
15
16  <applicationMonitor updateTrigger="mbean"/>
17
18  <webApplication id="Sample1" location="Sample1.war" name="Sample1"
19 </server>

```

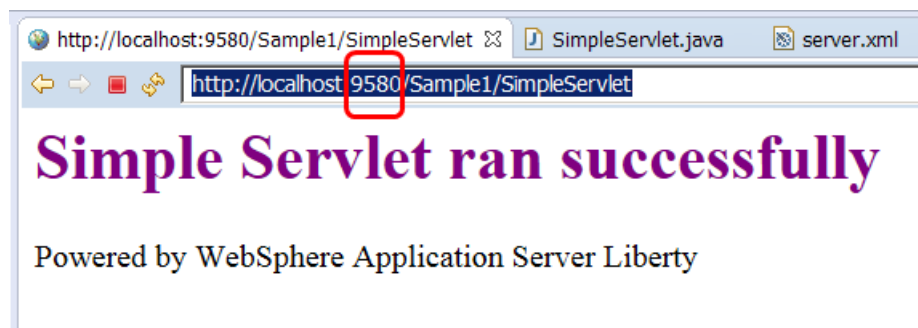
- ___f. After you saved your configuration changes, the configuration of your running server was automatically updated. The **Console** pane will show that the Sample1 servlet is now available on port 9580.

```

Liberty Runtime [labServer] (Dec 21, 2017 1:54:14 PM)
[AUDIT] CWWKG0016I: Starting server configuration update.
[AUDIT] CWWKG0017I: The server configuration was successfully updated in 0.009 seconds.
[AUDIT] CWWKT0017I: Web application removed (default host): http://localhost:9080/Sample1/
[AUDIT] CWWKT0016I: Web application available (default host): http://localhost:9085/Sample1/

```

- ___g. Now, you can access your sample application using the new port. In the browser in your workbench, change the port from **9080** to **9580** and refresh the application.



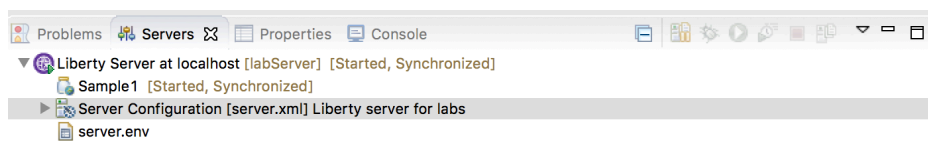
1.3 Add INFO logging output to console

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 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.

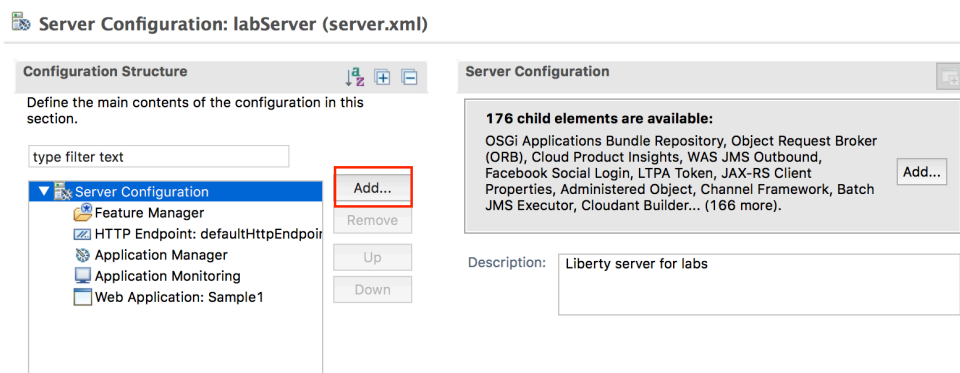
___a. In the **Servers** view, double-click on the labServer **Server Configuration** server to open the configuration server.xml editor.



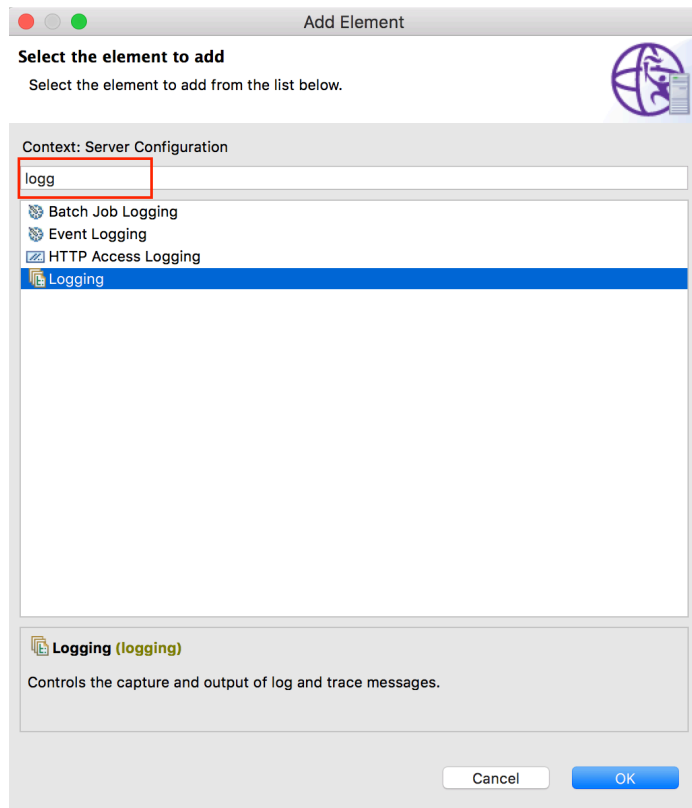
___b. Ensure you are in the Design mode by selecting the **Design** tab on the Server Configuration editor.

___2. Add the Logging configuration option to the server

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

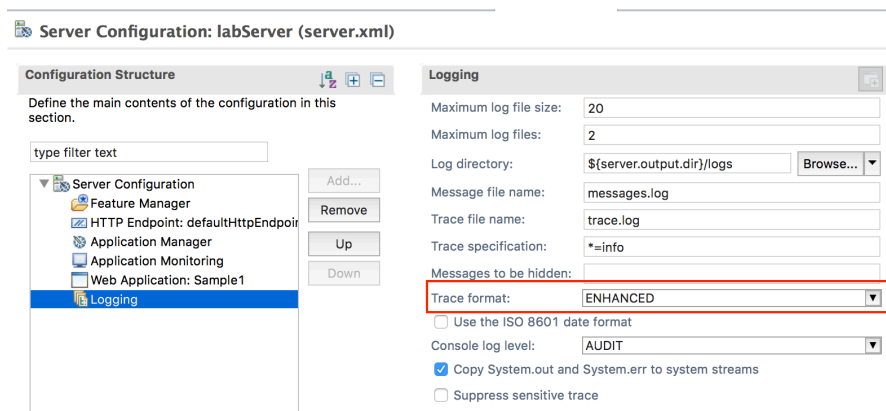


- ___d. On the **Add Element** dialog, select **Logging**, And, then click the **OK** button.

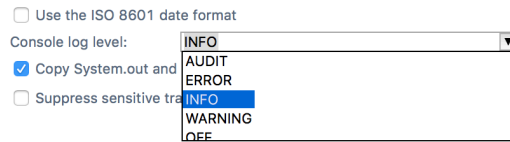


- ___c. The logging page displays the properties for the logging configuration, such as the name of the log files, the maximum size of log files, and the maximum number of log files to retain.

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



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

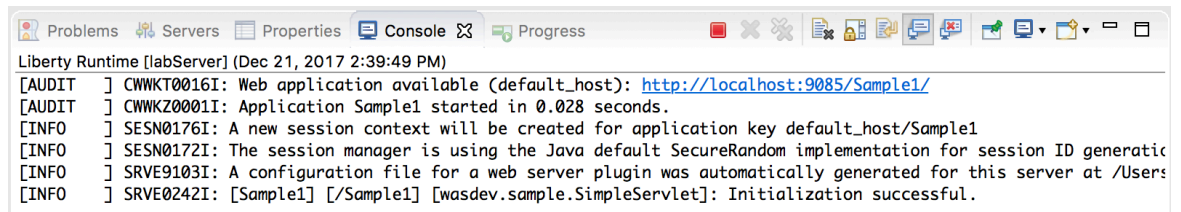


__a. Switch to the **Source** view for the server.xml file to see the configuration changes added to server.xml.

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

__b. **Save** the configuration file.

The changes you made are dynamic and take effect immediately.

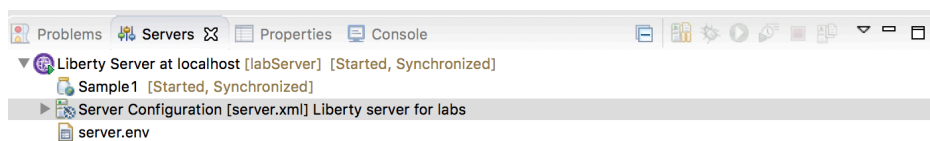


1.4 Update trace specification

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



- ___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:

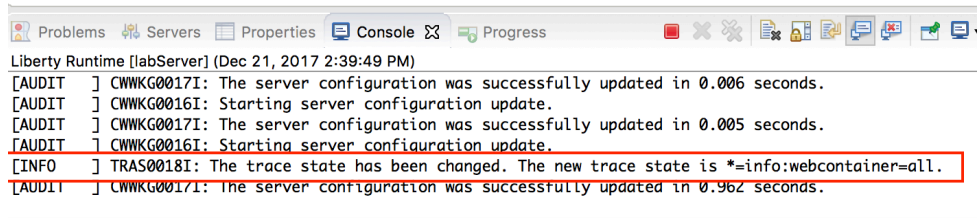
```
webcontainer=all=enabled:*=info=enabled
```

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

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

- __d. **Save** the configuration changes.

- __e. Check the console view



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

- __a. From a Windows Explorer, navigate to the server logs directory.

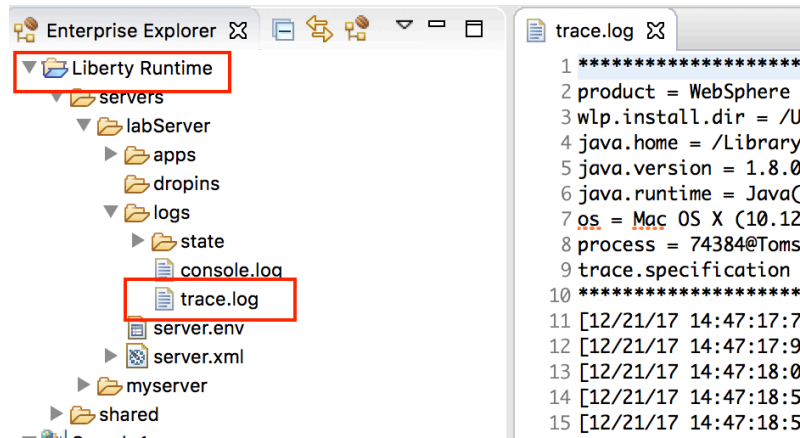
{LAB_HOME}\wlp\usr\servers\labServer\logs

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

Name	Size
messages.log	4 KB
messages_12.07.10_14.22.2...	6 KB
trace.log	71 KB

- __b. You can view the **trace.log** file using **Notepad**.

- __c. You can also view the trace file in Eclipse. In the Enterprise Explorer view, expand the WebSphere Application Server Liberty project, and its subdirectories, and you will find the trace.log file on the logs directory. You may need to right-click on a higher directory and select '**refresh**' to see any newly-created directories and files.



__4. **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:

```

--
16
17 <applicationMonitor updateTrigger="mbean"/>
18 <logging consoleLogLevel="AUDIT" traceSpecification="webcontainer=all=enabled:*=info=enabled"/>
19
20 <webApplication id="Sample1" location="Sample1.war" name="Sample1"/>
21 </server>

```

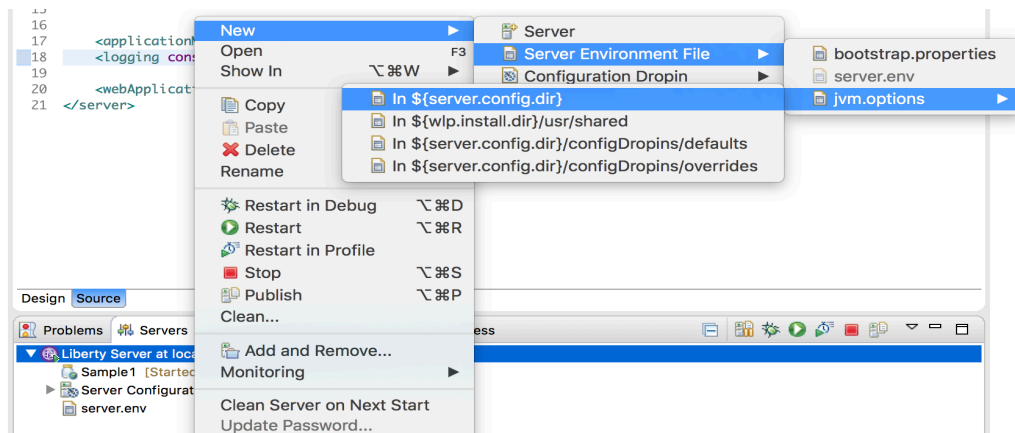
- __b. **Save** the configuration.

1.5 Customizing Liberty JVM Options

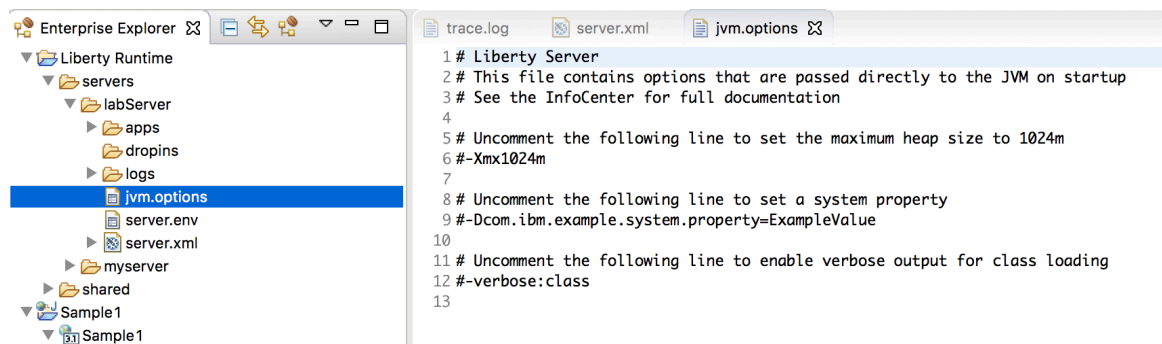
The generic JVM arguments are used to configure and adjust how the JVM executes.

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. In the eclipse Servers view, right-click on the localhost server. And, then select **New → Server Environment File → jvm.options** from the context menu.



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



- ___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

TIP:

- 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.

Restart the server to enable changes.

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

1.6 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.

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 “`{LAB_HOME}\wlp\usr\servers\labServer`”

- **`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 `${server.output.dir}` is `{LAB_HOME}\wlp\usr\servers\labServer`, which is the same as `${server.config.dir}`.

1.7 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.

NOTE: You will not modify any of the default environment initialization. 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.

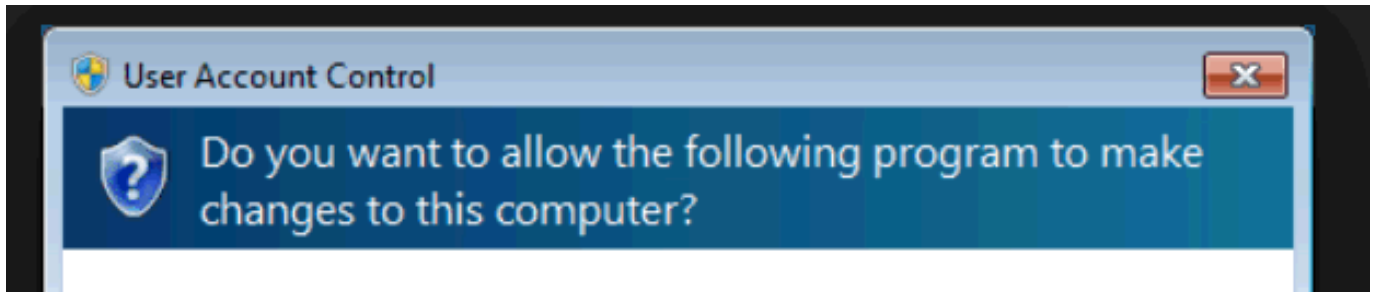
In this case you set them in the `bootstrap.properties` file instead of the server configuration.

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.

1.8 Running Liberty as a Windows Service

A Liberty server can be registered as a Microsoft Windows Service program when running on Windows. After registering as a Windows Service, Liberty can then be started and stopped manually as a Service, or the Windows "Services" program can be used to change the Liberty service to start automatically when the Windows Server starts.

The commands to Register/Start/Stop/Unregister Liberty as Service use the same **server.bat** file that is used when starting the Liberty server normally. You should get a Windows prompt to make changes to your computer. Accept these changes.



1. Register the labServer as a Windows service by running these commands:

```
cd {LAB_HOME}\wlp\bin
```

```
server registerWinService labServer
```

2. Start the labServer Windows service. This can be done with the following command:

```
server startWinService labServer
```

Alternatively, the server can be started through the Windows Services interface. Start the Windows "Services" program and find the entry that corresponds to the server name 'labServer'. Using the Windows Services program the service can be modified to start in Automatic mode, to start under a specified Account ID (and password), and to start with specified Liberty Server command line parameters.

3. Stop the labServer Windows service. This can be done with the following command:

```
server stopWinService labServer
```

Alternatively, the server can be stopped through the Windows Services interface.

4. Unregister the labServer Windows service. This can be done with the following command:

```
server unregisterWinService labServer
```

The Liberty Service can also be administered via the Windows Registry, typically using the Windows **regedit** program. The entries for the Service will be located in the Registry:

HKEY_LOCAL_MACHINE->SOFTWARE->Wow6432Node->Apache Software Foundation->Procrun
2.0-><labServer>

and

HKEY_LOCAL_MACHINE->SYSTEM->CurrentControlSet->services-><labServer>

This completes the lab exercises.

Appendix A. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have

been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental. All references to fictitious companies or individuals are used for illustration purposes only.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Appendix B. Trademarks and copyrights

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	AIX	CICS	ClearCase	ClearQuest	Cloudscape
Cube Views	DB2	developerWorks	DRDA	IMS	IMS/ESA
Informix	Lotus	Lotus Workflow	MQSeries	OmniFind	
Rational	Redbooks	Red Brick	RequisitePro	System i	
<i>System z</i>	<i>Tivoli</i>	<i>WebSphere</i>	<i>Workplace</i>	<i>System p</i>	

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of The Minister for the Cabinet Office, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

NOTES

NOTES



© Copyright IBM Corporation 2017.

The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. This information is based on current IBM product plans and strategy, which are subject to change by IBM without notice. Product release dates and/or capabilities referenced in these materials may change at any time at IBM'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.

IBM, the IBM logo and ibm.com are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.



Please Recycle
