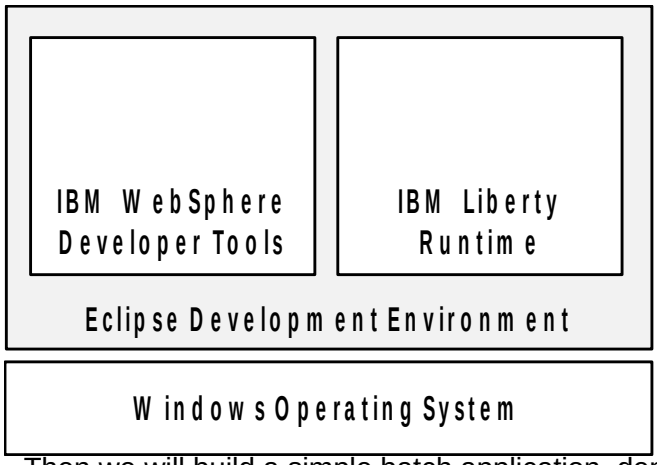
IBM WebSphere Java Batch Lab

What are we going to do?

First we are going to set up a development environment on your workstation.

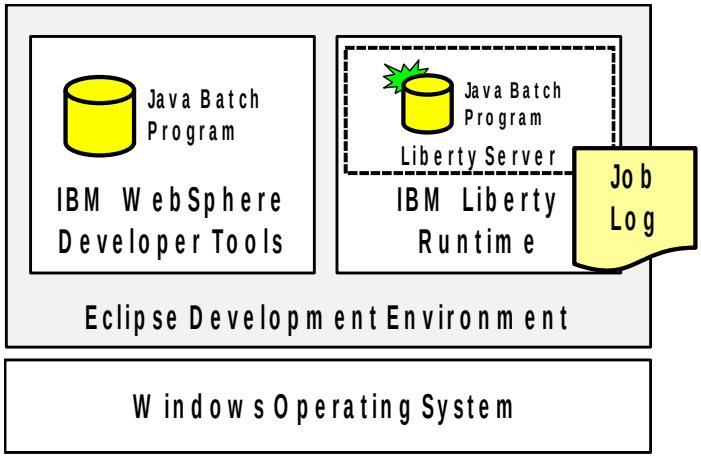


Download and install Eclipse

Install WebSphere Developer Tools

Install IBM Liberty into Eclipse

Then we will build a simple batch application, deploy it into a Liberty server we will define, run the job, and look at the results.



Create a Java Batch "batchlet" program
You will do this in WDT

Create a Liberty server
This will provide the runtime for the batch program

Deploy and run the Java batch job
You will see it be submitted, execute and complete

Review Job Log and see results
You will see "Hello World!" in the log

Setting Things Up

In this section we will provide you with information to help you set up Eclipse and the WebSphere Developer Tools on your workstation.

Downloading and installing Eclipse

- ☐ Go to this URL:
  https://developer.ibm.com/wasdev/getstarted/
- ☐ Use the following link at that site to download a ZIP file that includes Eclipse:

Installing from within Eclipse

Complete the following steps to install the WebSphere Liberty using Eclipse:

1. If you don't already have Eclipse, install Eclipse Oxygen for Java EE Developers.

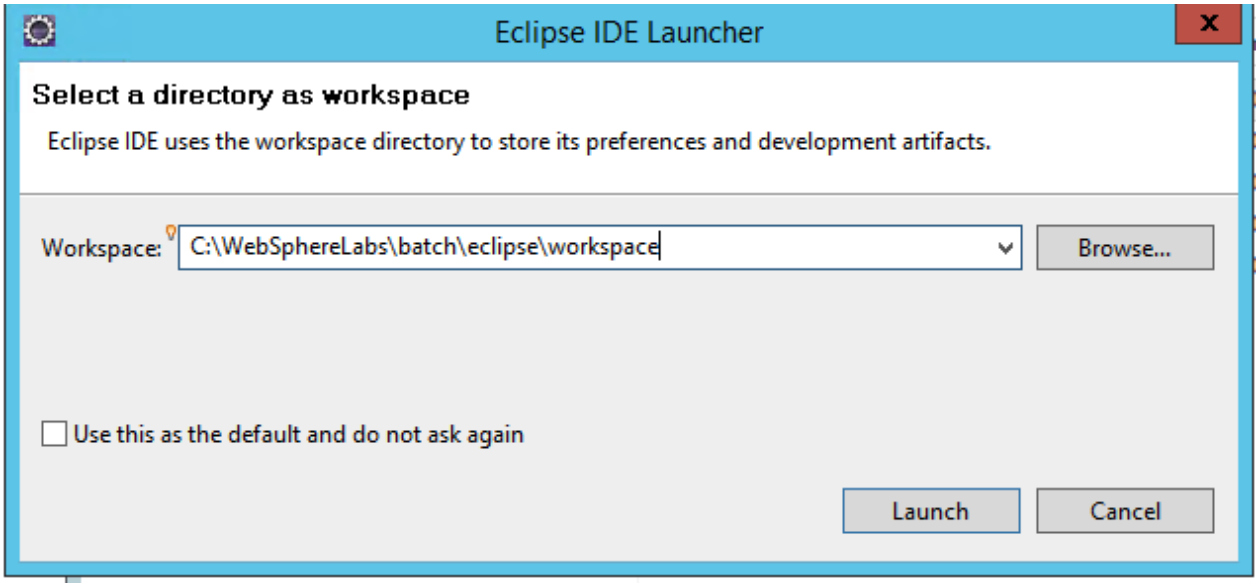
- ☐ When you get it downloaded, run the .exe to install it..
- ☐ The `eclipse.exe` file is what you double-click on to launch Eclipse.
- ☐ If you find it has trouble locating Java to use, edit the `eclipse.ini` file and add:

```
--launcher.defaultAction
openFile
-vm
<path_to_Java>/bin/javaw.exe
--launcher.appendVmargs
```

The objective is to get Eclipse to launch to the "Welcome" screen.

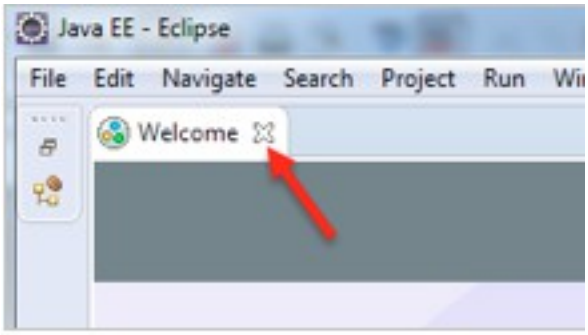
- ☐ If you get a security warning asking if you want to run the file, click "Run". You should then see a splash screen:

- It will then ask you for the location of the "Workspace":



Provide the value **`C:\WebSphereLabs\batch\eclipse\workspace`** and click "Launch".

Close the 'Welcome" panel if it shows at startup.



Installing WebSphere Developer Tools into your Eclipse

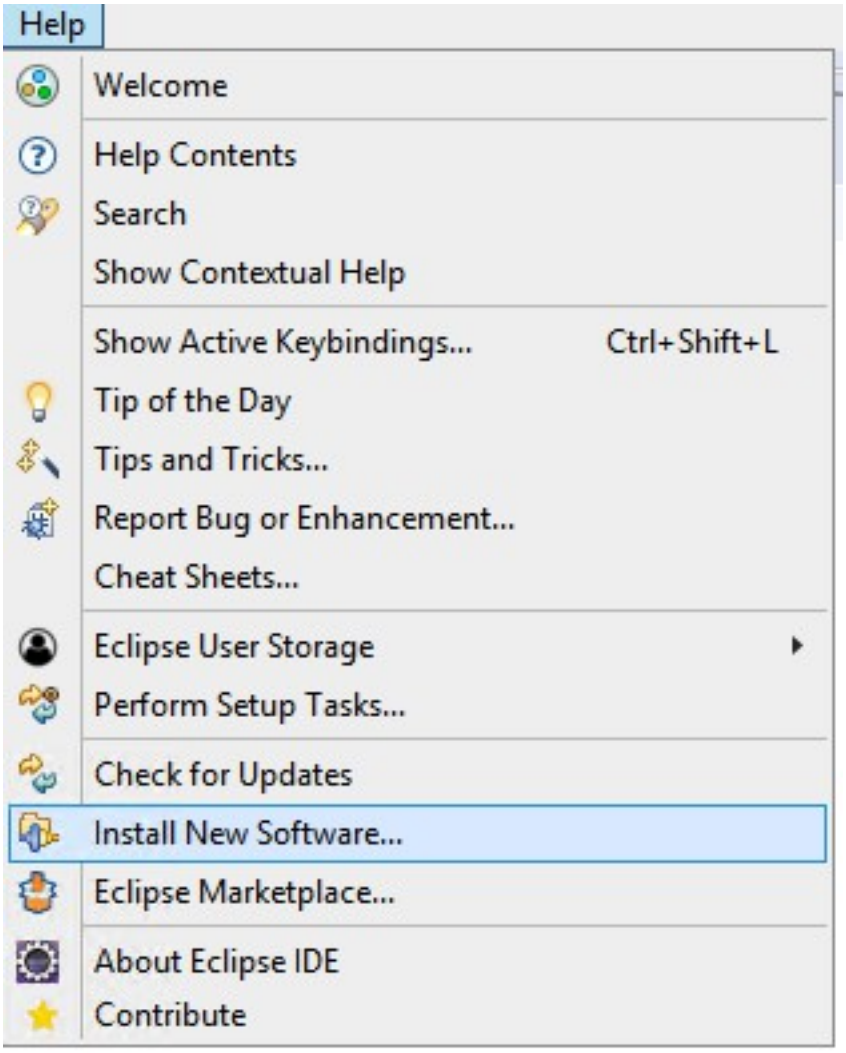
Once you have Eclipse launched, we need to install WDT.

Begin here: https://developer.ibm.com/wasdev/docs/websphere-developer-tools-releases/

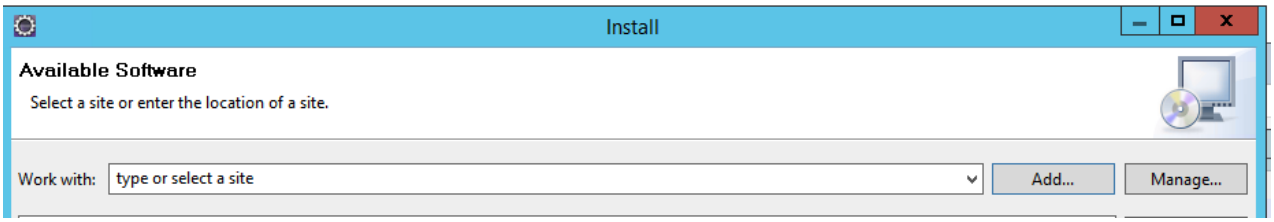
Under "Current Releases" select "Get it now!" for the Eclipse version you just installed.

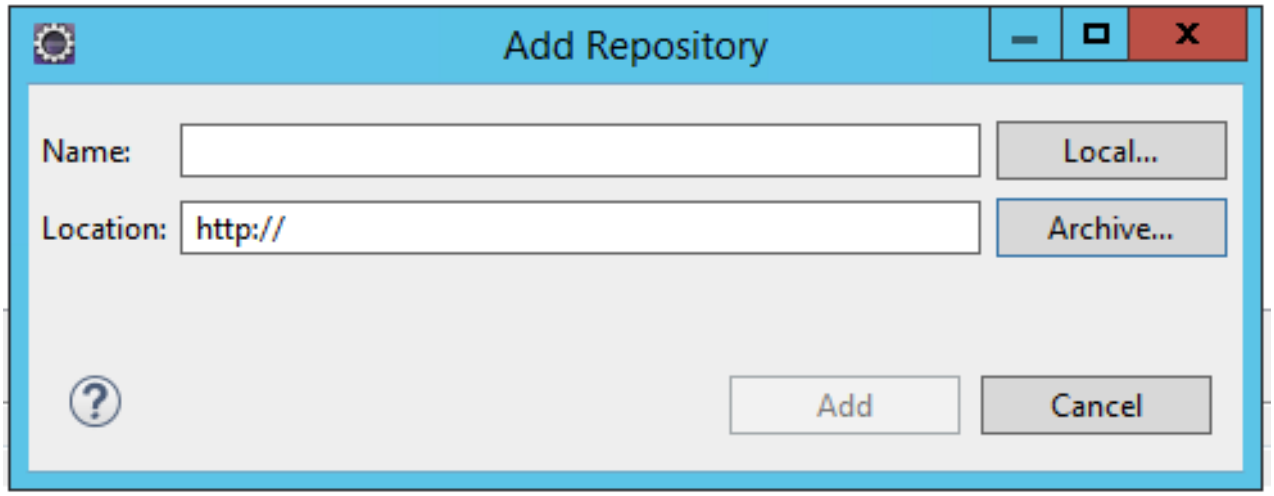
This will open another page with two download links. Make sure you get the one that says "WebSphere Developer Tools for WebSphere Application Server Liberty and WebSphere Application Server traditional". It is usually the second one.

Once it downloads, go back to Eclipse and select "Help->Install New Software"



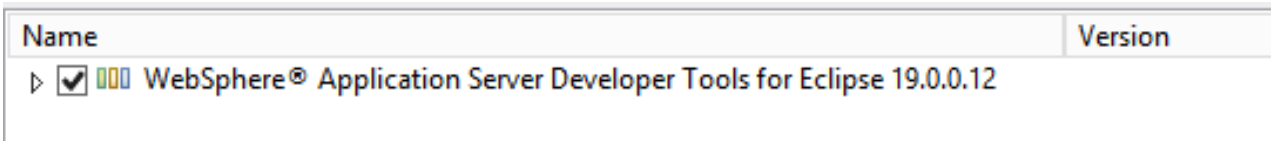
Click the **Add** button and then click the **Archive** button.





Browse to the install image you downloaded earlier and select it. Then click **Add** in the Add Repository Dialog.

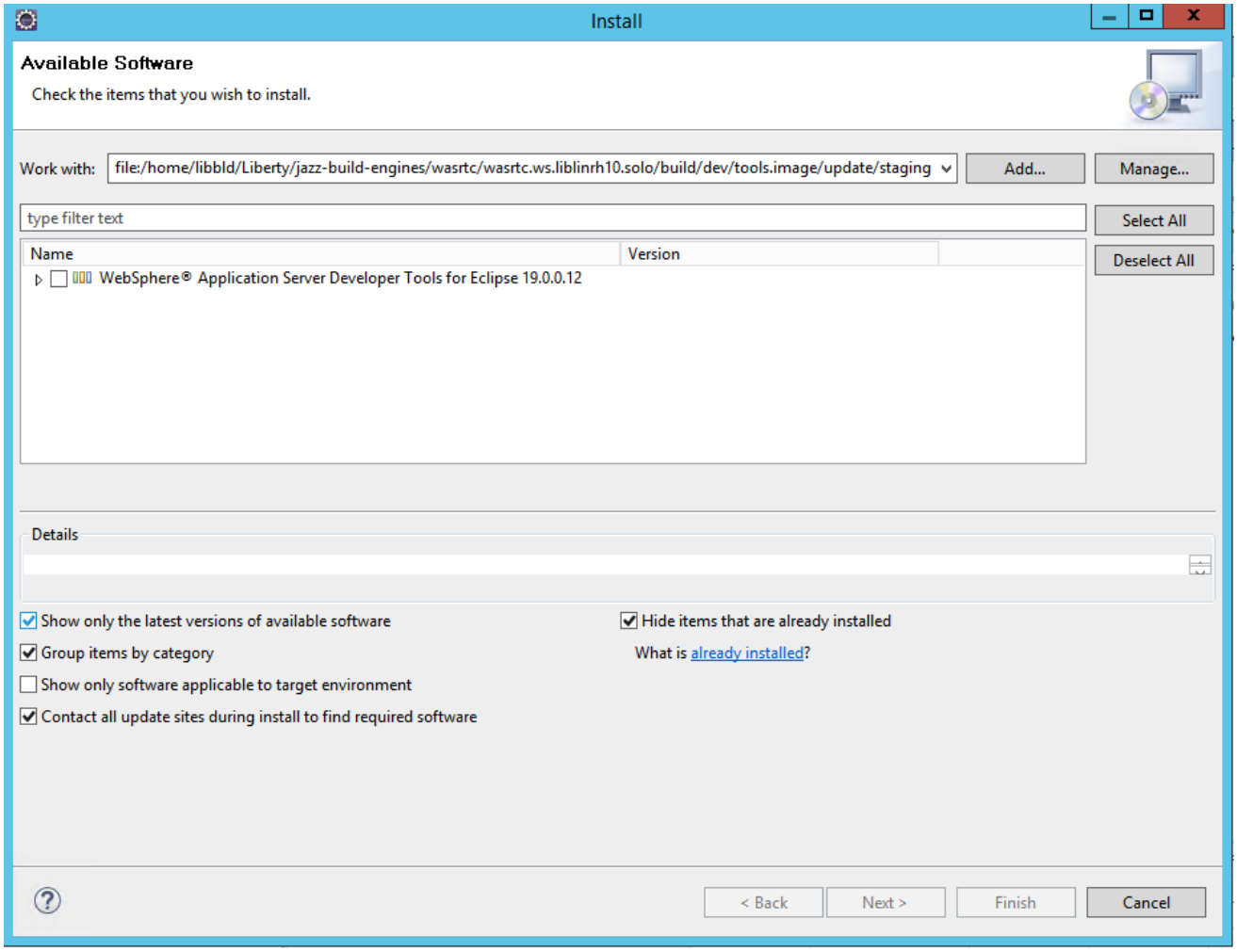
Next click the checkbox next to "WebSphere Application Server Developer Tools for Eclipse 19.0.0.12" (or whatever level is current).



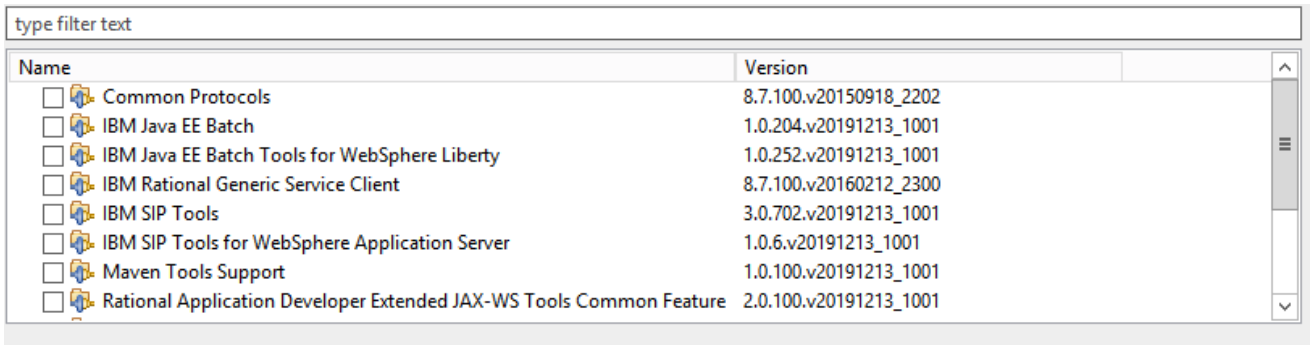
Click **Next** and finish the wizard, restarting Eclipse when prompted.

Adding the Batch WDT Feature

After Eclipse restarts, go through the steps above again (Help->Install Software, then add the archive). Get back to the point where you selected the Developer tools to install, but this don't don't check the box next to it. Instead uncheck the "Group items by Category" box farther down the screen.



Unchecking it will change the list of things to install to look something like this:

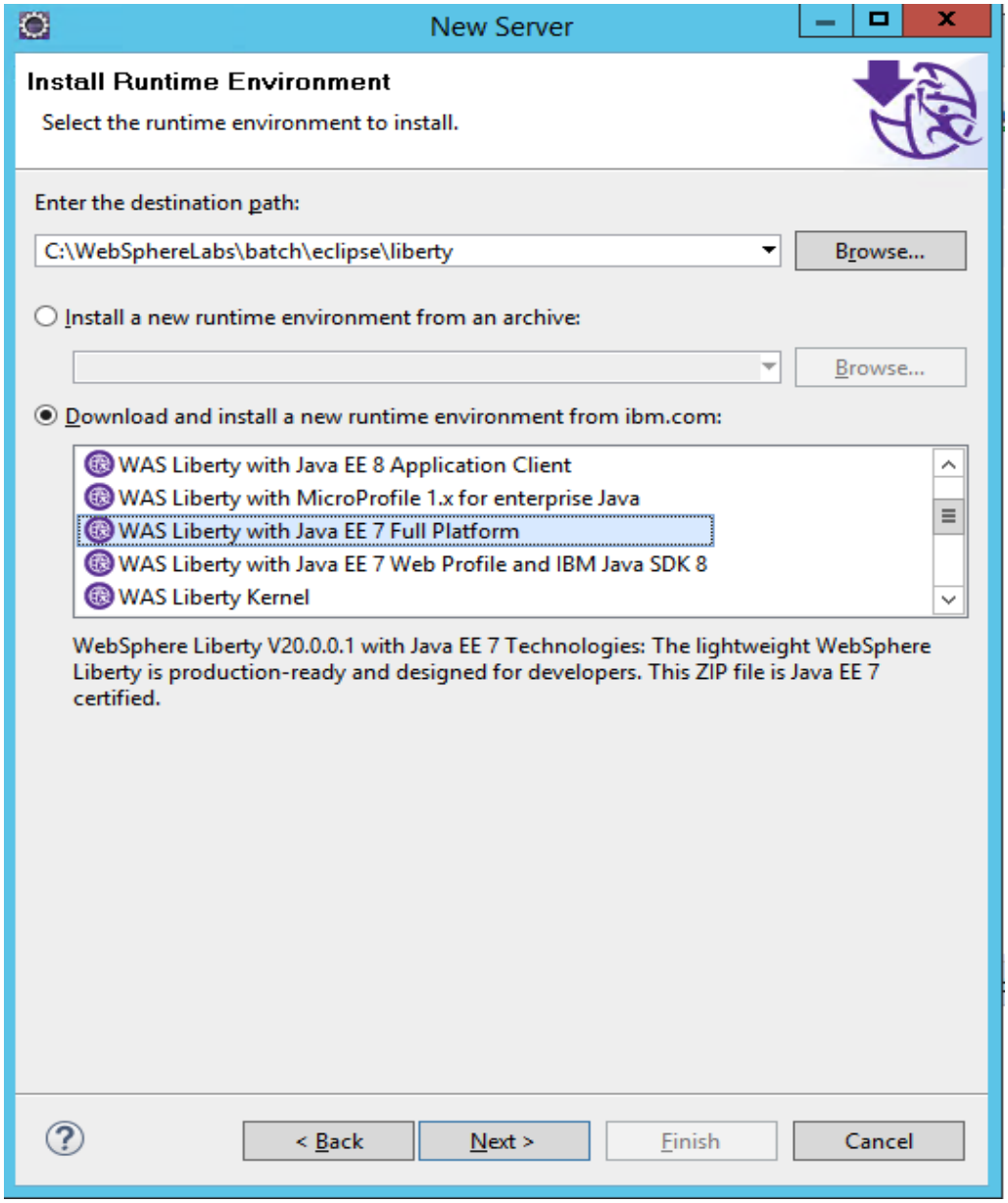


Select the "IBM Java EE Batch" and "IBM Java EE Batch Tools for WebSphere Liberty", click **Next**, and follow the wizard again, restarting Eclipse when prompted.

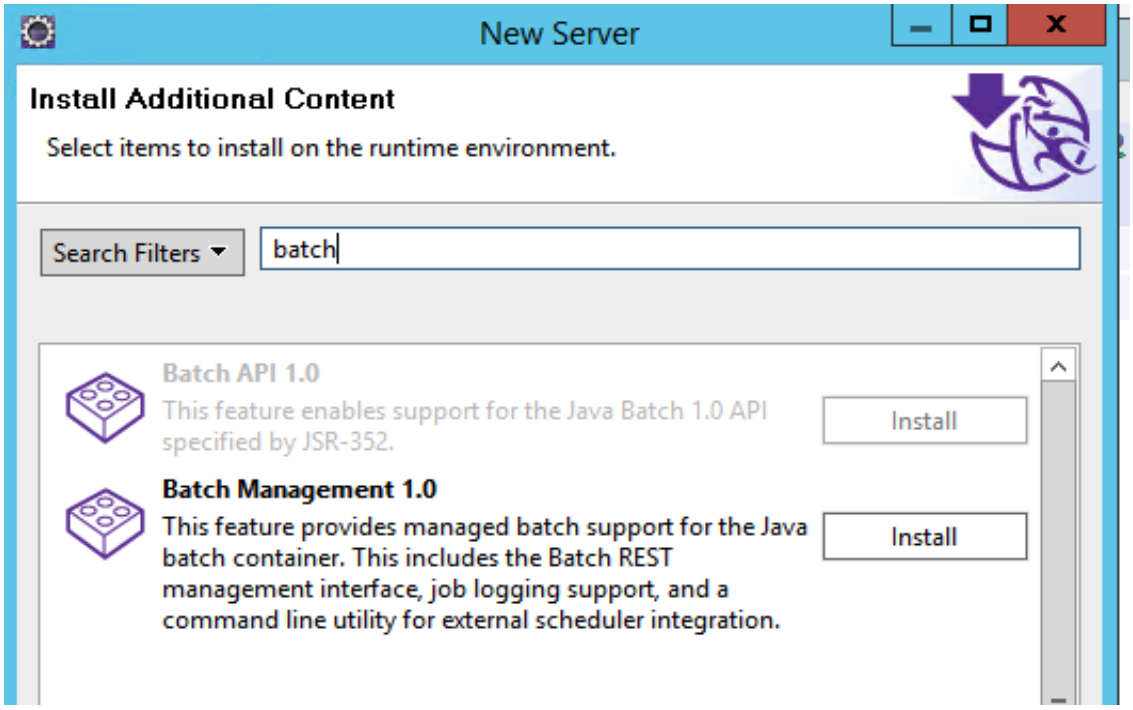
Create a Server

Your Java batch job needs to run in an environment that supports the JSR 352 batch standard. Liberty with Java EE 7 provides that. We'll define a new server and install Liberty along the way. Do the following:

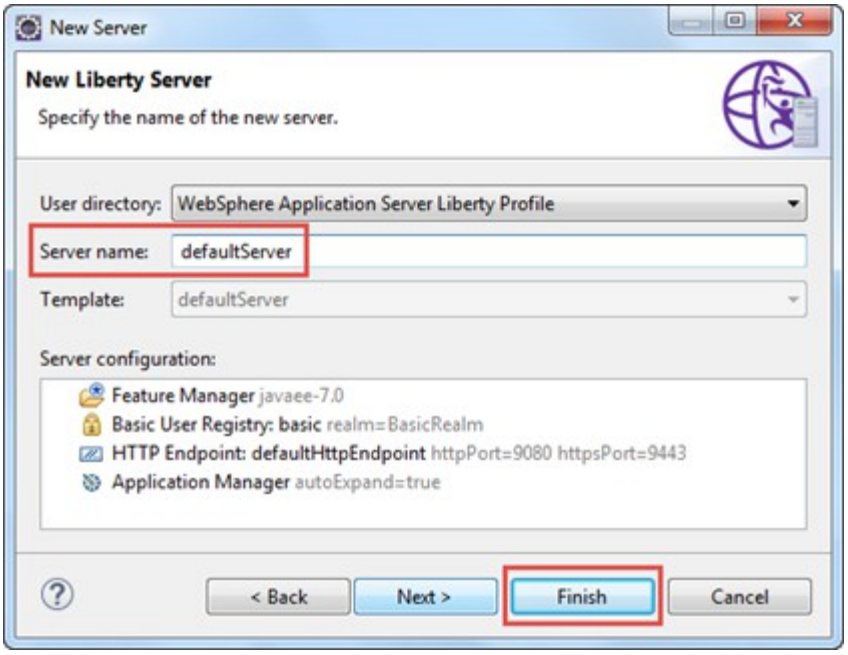
- ☐ In the lower portion of the screen, click on the "Servers" tab:
- ☐ Position your mouse in that server tab pane, right-click and select *New Server*.
- ☐ Select "Liberty Server" and then click "Next":
- ☐ Next, select the radio button "Install from an archive or a repository". Keep the "Use default JRE" radio button default and then click "Next":
- ☐ The "destination path" is where you want Liberty installed (it must be an empty folder), and the "Repository" is an IBM-hosted site from which the files will be downloaded. Be sure to select "Java EE 7 Full Platform" to get the JSR 352 Java Batch function.



- ☐ When you get to the "Install Additional Content" panel, search for "batch" and select the "Batch Management" feature to install.



- ☐ Accept the license agreement and then click "Finish." The install will begin.
- ☐ Then, note the server name ("defaultServer," which is what you'll use for this lab), then click "Finish":



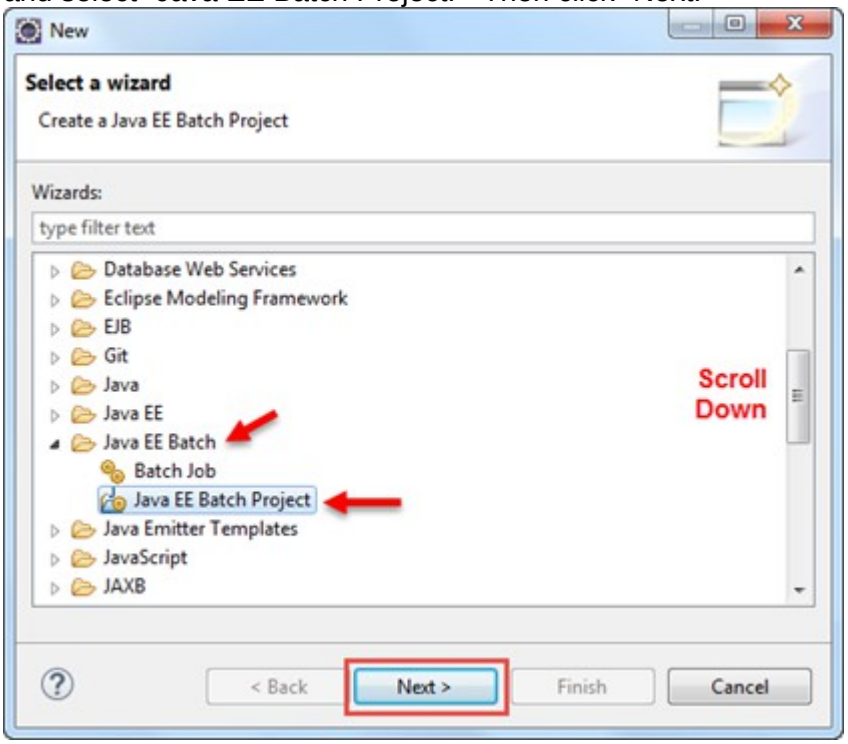
- ☐ You should see your new server in the "Server" tab:

Create a New Batch Job

In this section you will go through the steps to define a batch project and batch class file in preparation for running the batch job in your server. Do the following:

- ☐ On the left side of your screen you should see the "Enterprise Explorer" tab. Position your mouse in that pane, right click, and select *New* *Other*:

- ☐ In the wizard panel that appears, scroll down a little, locate "Java EE Batch," open that folder and select "Java EE Batch Project." Then click "Next."



- ☐ In the next window, give the project a name (such as "MyBatch") and click "Finish":

- ☐ On the left side of the screen in the "Enterprise Explorer" pane you should see something like this:

- ☐ With your project selected, Right-Click and select *New* *Batch Job*:

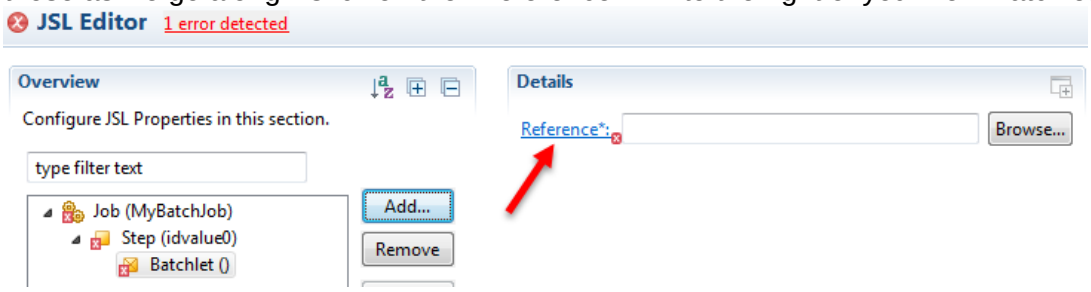
- ☐ Then give your job a name (such as "MyBatchJob") and click "Finish":

- ☐ At this point you should see the JSL Editor:

- ☐ Click on the "Add" button, select "Step" and then click "OK":

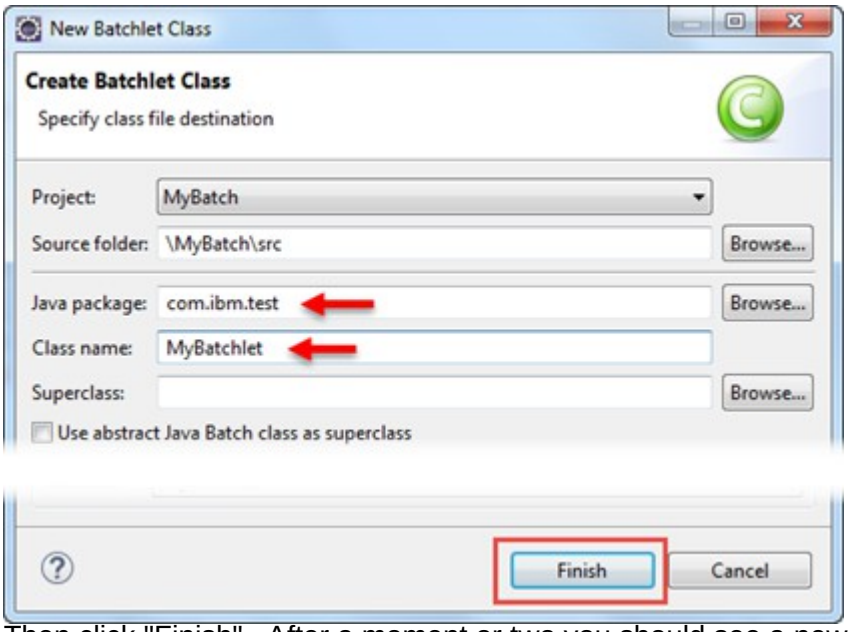
- ☐ With the new step highlighted (should do that automatically), click on "Add" again, and this time select "Batchlet" and "OK":

- ☐ At this point you will notice various "red X" symbols indicating errors. That's okay; you will fix those as we go along. Click on the "Reference" link to the right of your new Batchlet step:



- ☐ In the "Create Batchlet Class" wizard, provide the following:

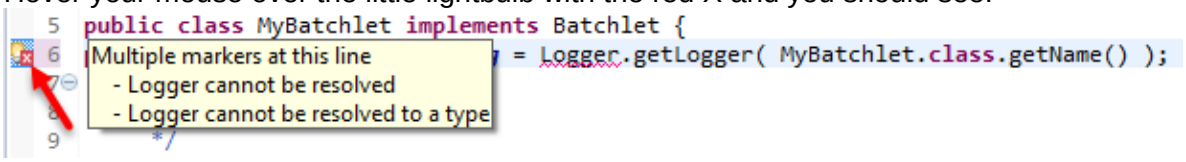
Java package:	**`com.ibm.test`**
Class name:	**`MyBatchlet`**



Then click "Finish". After a moment or two you should see a new tab open that represents the source for `MyBatchlet.java`:

- ☐ You're going to add a few things to this source, and to keep typing to a minimum we have provided a "Copy and Paste" file with the code to be added. That is supplied here:
  `WP102544_Lab Copy-and-Paste.txt`
  Open that text file in the default text editor.
- ☐ Add the Logger private static line (the first line in the text file) as shown here:

  (The red X error indicator will be resolved next.)
- ☐ Hover your mouse over the little lightbulb with the red X and you should see:



- ☐ Click on the red-X light bulb and then double-click on **`java.util.logging`** to select that to import:

  The red-X should turn into a yellow warning:

```
7  public class MyBatchlet implements Batchlet {
8  private static final Logger log = Logger.getLogger( MyBatchlet.class.getName() );
9      /**
10      * Default constructor.
```

That's expected and okay for now. It will be resolved in the next few steps.

- ☐ Towards the bottom of the source file add the next line from the Copy-and-Paste file:

  Again, a red-X will appear. You will resolve that next.
- ☐ Click on the red-X symbol and double-click on "Import 'Level' (java.util.logging)" to import it:

  The red-X goes away:

  And you'll notice the yellow warning on your earlier "private static final" is also gone:

- ☐ Notice the little asterisk on the tab for this source editor:

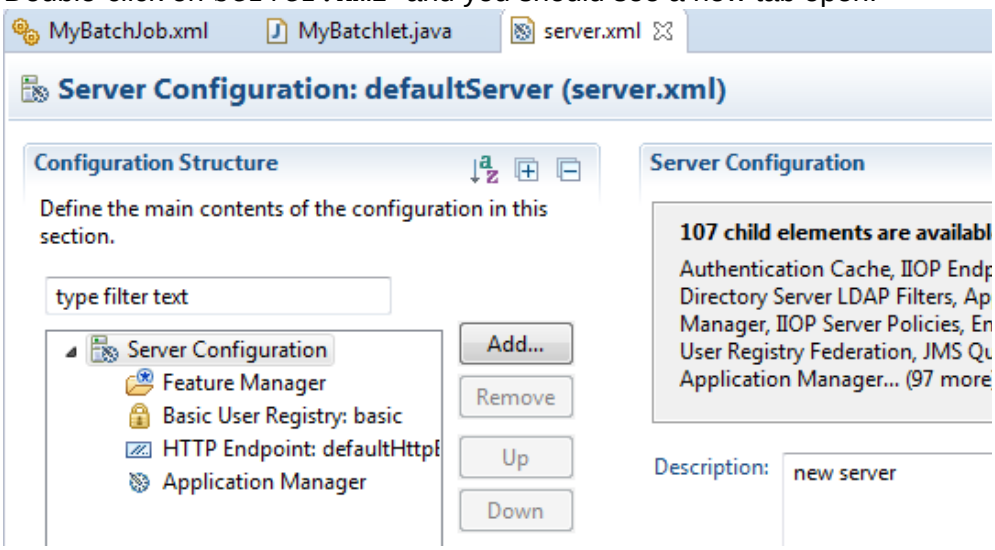
  That means changes have been made and are not yet saved.
- ☐ Save the changes by selecting **Ctrl-s** ... the asterisk will go away, indicating changes have been saved.
- ☐ Click on the "MyBatchJob.xml" tab and you'll see an asterisk there as well:

  Select **Ctrl-s** to save the changes there as well.

Earlier we created our Liberty server, but now we need to update the configuration to support running our batch job. We'll first add the batchManagement feature to allow remote management of batch jobs over a REST API and then add some simple security configuration (since the REST interface is secure).

- ☐ Go over to the left side of the screen and open up the Liberty server so you can see the `server.xml` file:

- ☐ Double-click on `server.xml` and you should see a new tab open:



- ☐ Select "Feature Manager" and then click the "Add" button:

- ☐ A window will open with a long list of features to enable. Select **`batchManagement-1.0`** and click "OK":

  You should see the following:

- ☐ Next, select "Basic User Registry" and click the "Add" button:

- ☐ Then select "User" and click "OK":

- ☐ Create an ID of **`Test`** and then click the "Set" button:

  Note: The "user name" could be anything (Bob, Mary, whatever). We'll use Test for this lab.
- ☐ For the password, provide **`testpwd`** (case sensitive) and click "OK"

  Note: this simply XOR encodes the password.
- ☐ Now, click on the "Source" tab at the bottom of the `server.xml` screen:

- ☐ In the source editor, locate the `<keyStore>` element and supply a dummy password[1] of **`password`**:

```
14          Then uncomment the keyStore element. -->
15      <!--
16      <keyStore password="password"/>
17      -->
18
```

- ☐ Then, remove the comments around `<keyStore>`:

```
14          Then uncomment the keyStore element. -->
15      <!--
16      <keyStore password="password"/>
17      -->
18
```

  You should now have:

- ☐ Go to the Copy-and-Paste file you used earlier and retrieve the five lines for the "authorization roles," and paste into the source XML above `<basicRegistry>` as shown here:

  Note: that provides the ID you created earlier ("Test") the authority to access the batch administrator role. this is required to access the REST interface of the batch runtime to submit and view jobs.
- ☐ Select **Ctrl-s** to save the changes in this editor. The three tabs should all show "no asterisk," meaning all editor window changes have been saved:

  You are now ready to deploy and run your batch job.

Deploy Batch Job

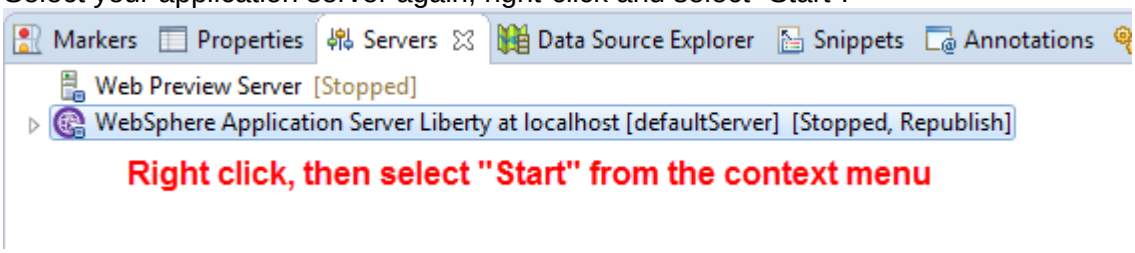
Your batch job is complete. The next step is to deploy it to your Liberty server. Do the following:

- ☐ At the bottom of the screen, select your Liberty server, right-click and select "Add and Remove":

- ☐ You should see your application (MyBatchWAR) available for adding. Select it, then click the "Add" button:

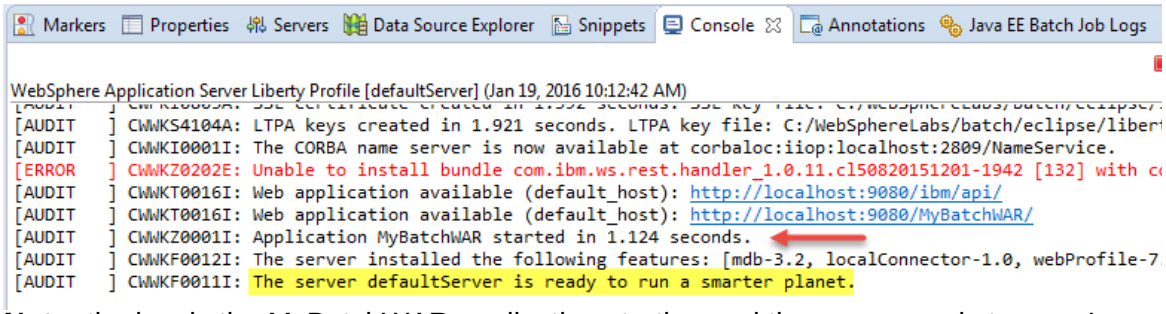
  You should see the application moved to the right side of the wizard. Then click "Finish":

- ☐ Select your application server again, right-click and select "Start":



- ☐ A "console" tab should open and you should see something like this:

1 We are using the built-in keystore of Liberty. The password does not really matter, but one is required. So we populate with a dummy value as shown.

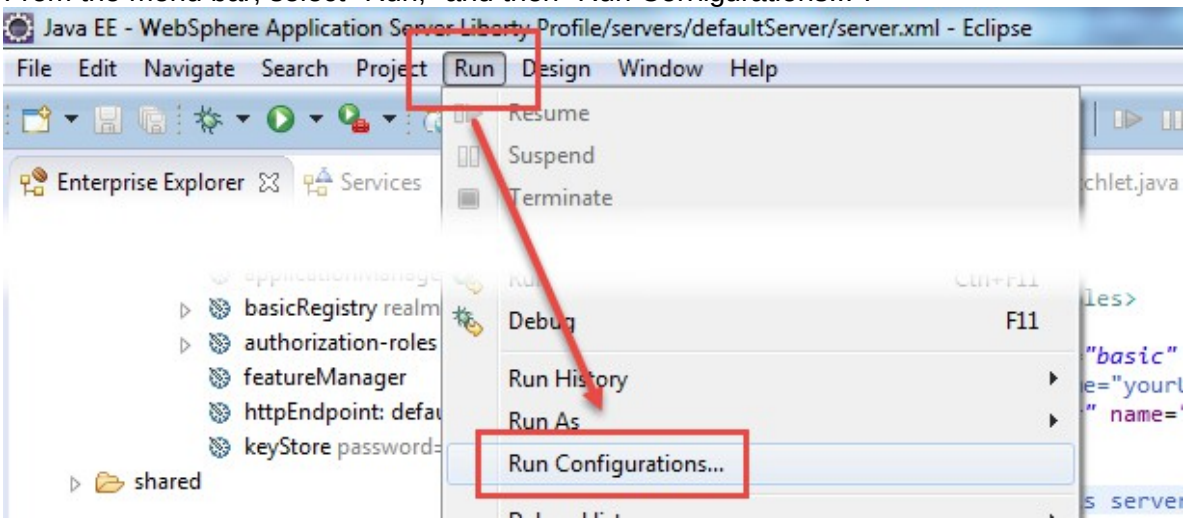


Note: the key is the MyBatchWAR application starting and the server ready to run. Ignore the red error message.

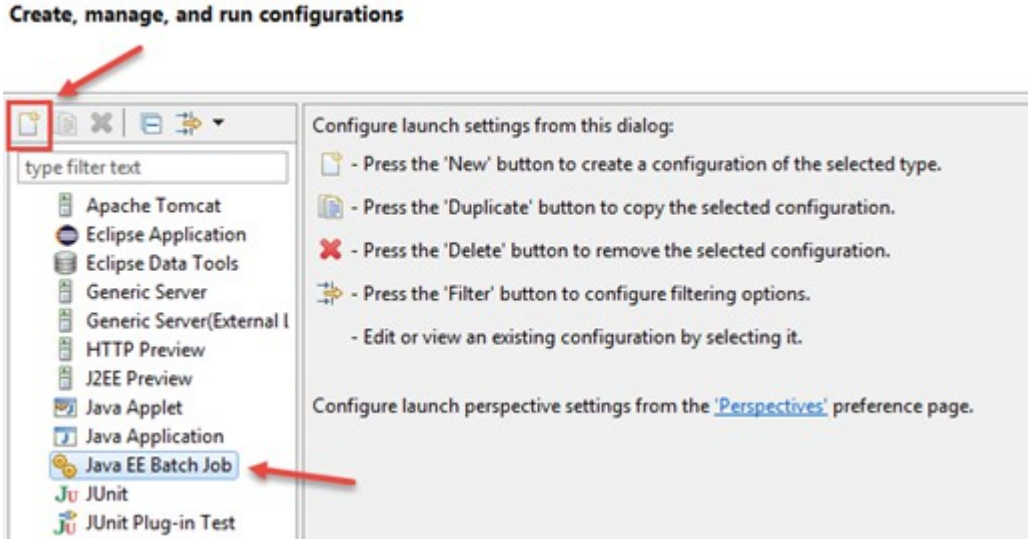
Run Your Batch Job

In this section we will create a "Run Configuration" to tell Eclipse how to use the server's REST interface to submit our batch job. We'll save the configuration to make it easy to run the job whenever we want.

- ☐ From the menu bar, select "Run," and then "Run Configurations...":



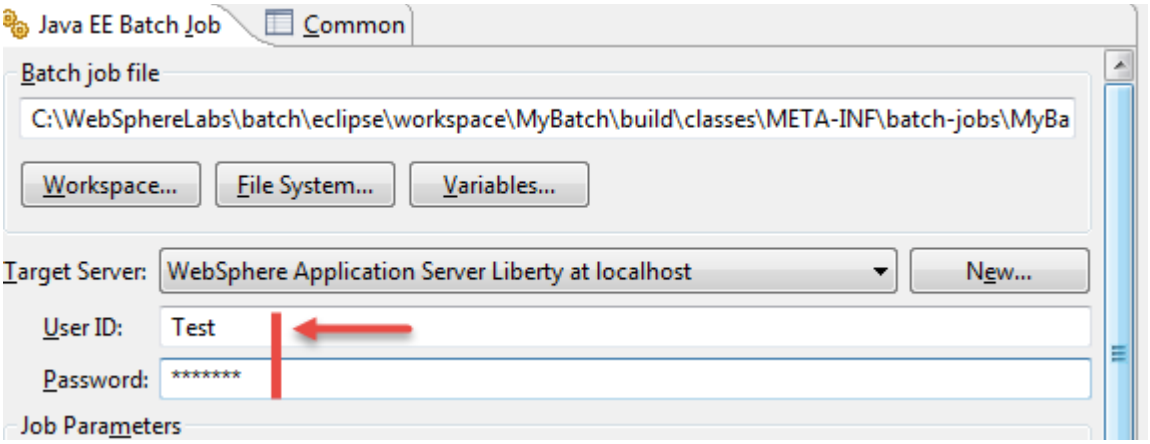
- ☐ In the window that appears, select "Java EE Batch Job," then click the "New" icon in the upper left:



- ☐ Provide a name (example: "JobRun1") and click the "Workspace" button:

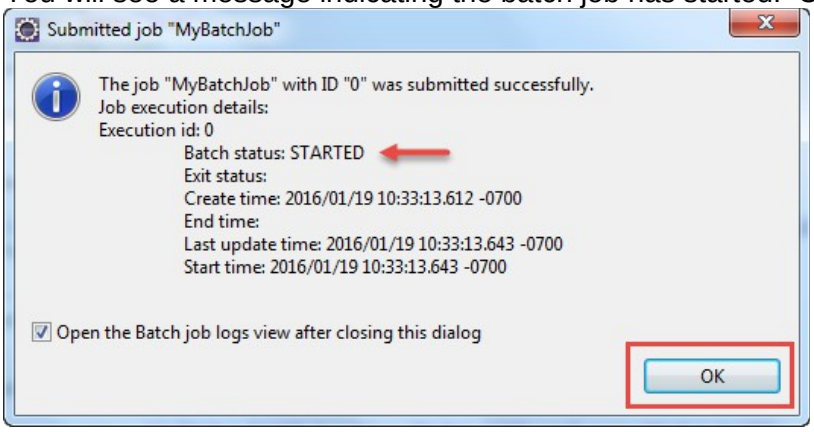
- ☐ The next step qualifies the configuration run to the batch job you created earlier. Expand the tree to expose *MyBatch build classes META-INF batch-jobs* and then select MyBatchJob.xml, then click on "OK":

- ☐ Provide the userid (**`Test`**) and its password (**`testpwd`**) you created earlier. Those are case-sensitive values, so type carefully:

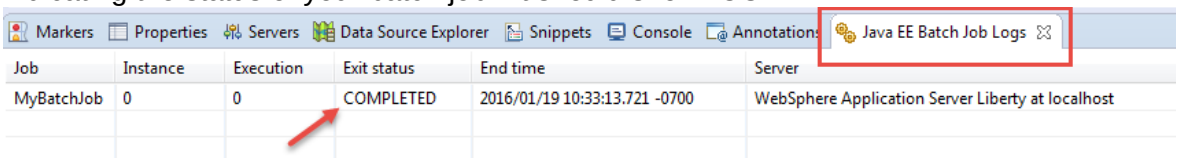


- ☐ Now click the "Apply" and the "Run" buttons:

- ☐ You will see a message indicating the batch job has started. Click "OK":



- ☐ You will see the "Java EE Batch Job Logs" tab come into focus, and there will be a line indicating the status of your batch job. It should show "COMPLETED":



Now that the job as finished we can go look at the log that was written by the job. The log will include messages issued by the batch container about the progress of the job as well as the message written by our batch application.

- ☐ Double-click on that job log line. You will see a security challenge. Click "Yes" to proceed:

  Note: that's because Eclipse is attempting to access the secure port of your Liberty server, and the Liberty server is using a self-signed certificate.

- ☐ It will prompt for a userid and password. Provide the value you used before (**`Test`** and **`testpwd`**), then click "OK":

- ☐ A job log tab will open, showing the contents of the job log:

  Scroll to the right using the scroll bar at the bottom of the screen. You should see the "Hello World!" text you added to the batchlet step:

  And that illustrates the execution of a batchlet step in the JSR 352 Java Batch container of IBM Liberty.

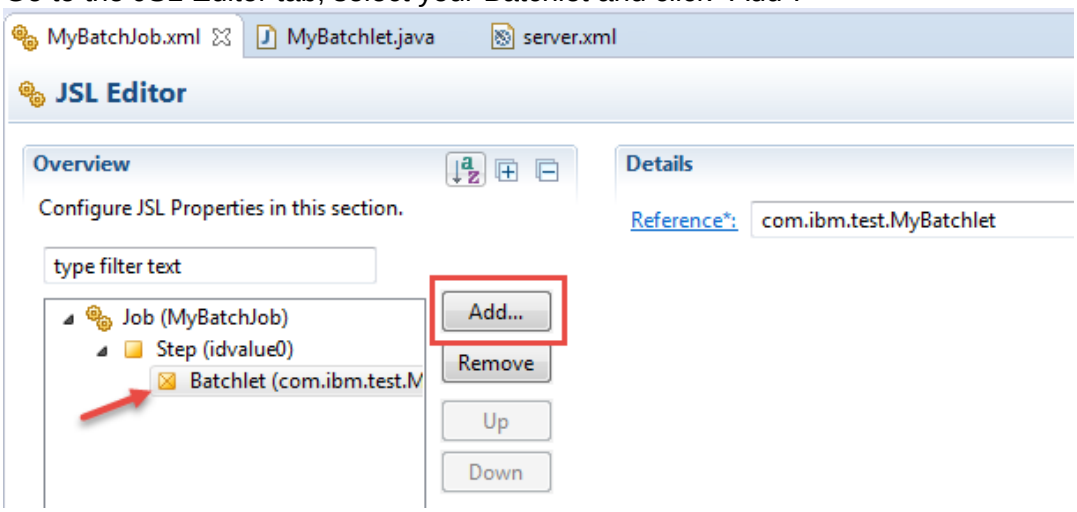
Optional Exercises

In this section we will provide some relatively simple optional exercises you may do.

Job Parameter

In this optional exercise we will change the batchlet you just created to pass in a job parameter so the output "Hello World!" is "Hello *<parameter you pass in>*!" A small thing, but it illustrates how values can be passed in at time of job submission. Do the following:

- ☐ Go to the JSL Editor tab, select your Batchlet and click "Add":

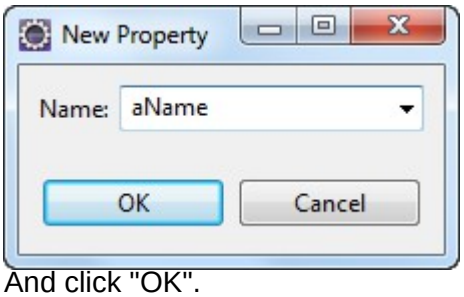


- ☐ "Properties" should be the only thing you see, so click on "OK":

- ☐ Back in the JSL Editor, your new "Properties" element should be highlighted. Click "Add" again:

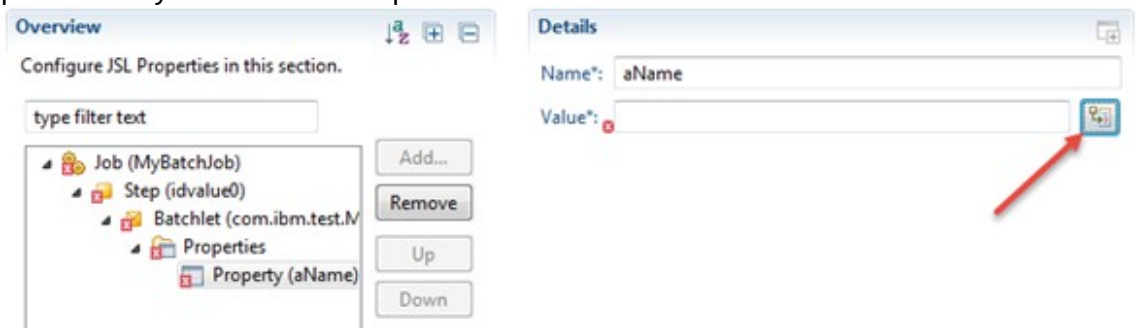
- ☐ You will see a window to "Select item to add to Properties," with only option being "Property":

  Click "OK". It will come back and ask you for a value. Provide a value of **`aName`** as illustrated here:

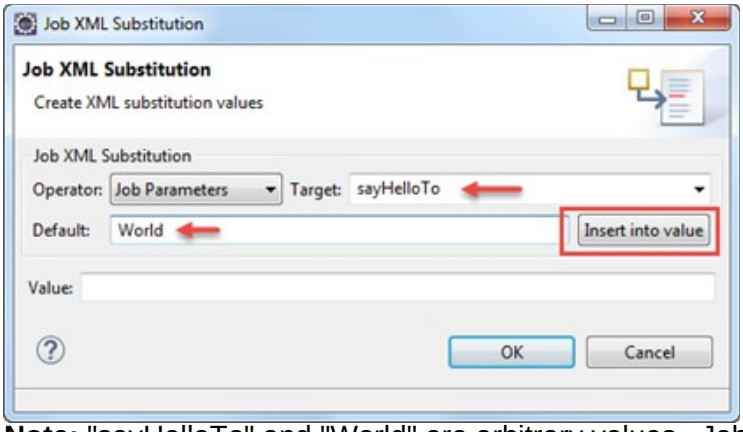


  And click "OK".

- ☐ You should see something like what we show below. Click on the icon to the right as pointed to by the arrow in this picture:

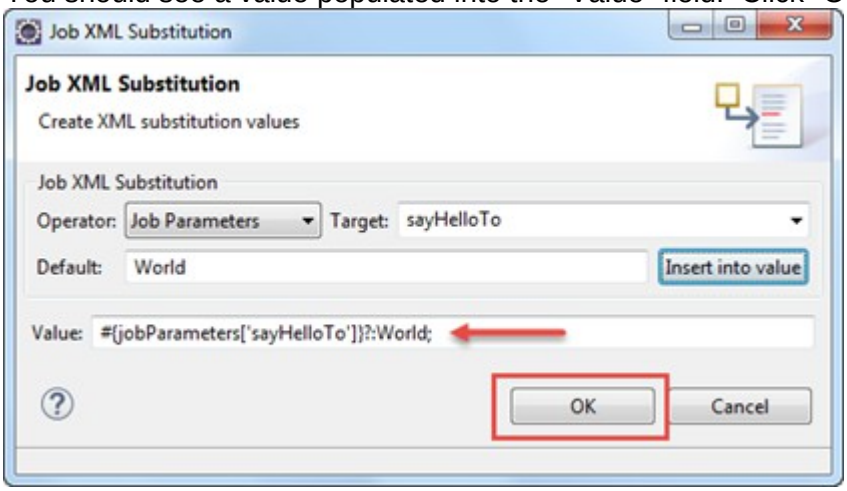


- ☐ On the Job XML Substitution panel, leave the "Operator" set to "Job Parameters," set the target a value of **`sayHelloTo`**, and set the default for that value to **`World`**. Then click "Insert into Value":



Note: "sayHelloTo" and "World" are arbitrary values. Job parameters may be whatever string values you want that are meaningful to you.

- ☐ You should see a value populated into the "Value" field. Click "OK".



- ☐ Select **Ctrl-s** to save the changes.

Now the JSL supports a job parameter called 'sayHelloTo' that defaults to 'World' and will be given to the batchlet as a property called "aName." Next we need to go update our batch application to get the parameter value and use it. We'll do this with Injection.

- ☐ Go back to the batchlet Java source tab:

- ☐ From the Copy-and-Paste file you used earlier, get the three lines provided after the "Optional Labs" comment block and insert them into the source Java as illustrated here:

```
26      */
27     @Inject
28     @BatchProperty(name = "aName")
29     String aNameValue;
30     public String process() {
31         // TODO Auto-generated method stub
32         log.log(Level.INFO, "Hello World!");
33             return null;
34     }
35
36 }
37
```

- ☐ Notice the red-X's on the first two lines. You will need to fix those as you did before – by clicking on the red-X symbol and importing the appropriate item:

`@Inject`	
`@BatchProperty`	

The red-X's should go away. If you look at the top of the source you should see your imports:

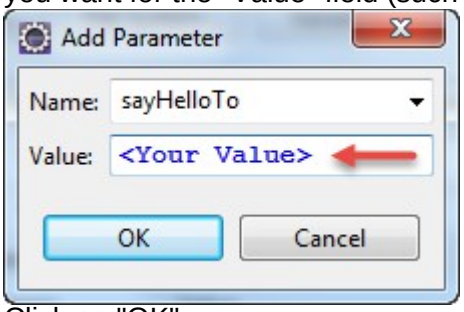
- ☐ Next, go to the Copy-and-Paste file and retrieve the log statement and paste it into the source Java in place of the current log statement:

- ☐ Select **Ctrl-s** to save the changes.

If the Liberty server is still running from before, those changes will be dynamically updated. You do *not* need to re-deploy the application. If the server is stopped, you can simply re-start the server and the updated application will be in place to run.

- ☐ From the menu bar, select *Run* *Run Configurations*.
- ☐ Click the "Add" button for Job Parameters:

And then select "sayHelloTo" from the pulldown menu for "Name," and provide any value you want for the "Value" field (such as your name):



Click on "OK"

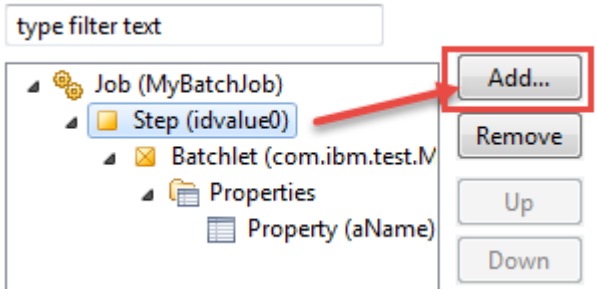
- ☐ Click "Apply" and then "Run." This will submit your job.
- ☐ Click "OK" to the window that pops up giving you the submitted message.
- ☐ In the "Java EE Batch Job Logs" tab, double-click on the line that represents this job submission. It should have an "Instance" value of 1 and "Execution" value of 1.
- ☐ Scroll over as you did before and see the parameter in the log output:

```
CWWKY0018I: Step idvalue0 started for job instance
Hello <your value>
CWWKY0020I: Step idvalue0 ended with batch status C
CWWKY0010I: Job MyBatchJob ended with batch status
```

Step Listener

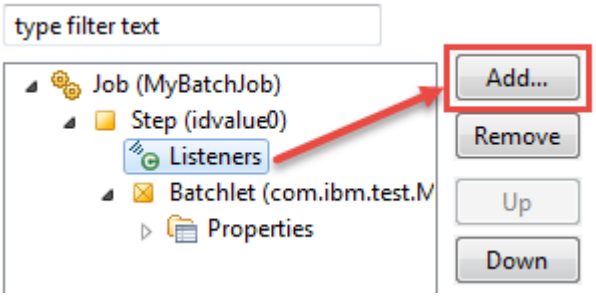
A "step listener" is code that is invoked before and after the step is executed. It is useful when you wish to take some action before and after the step, or to simply log information. Do the following:

- ☐ Go back to the JSL Editor (the "MyBatchJob.xml" tab).
- ☐ Select the job step and click "Add":



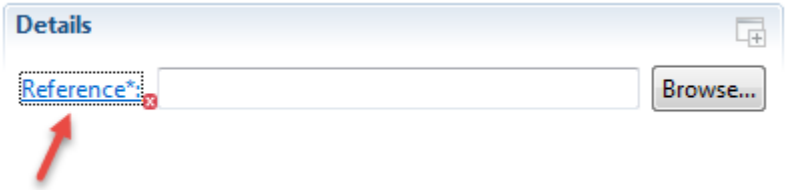
- ☐ Select "Listeners" and click "OK":

- ☐ Select "Listeners" and click "Add":



- ☐ Select the single "Listener" option and click "OK":

- ☐ Over to the right, click on the "Reference" link:



- ☐ Fill in the Java package and Class name fields:

Java package	`com.ibm.test`
Class name	`MyStepListener`

And then click "Finish".

- ☐ This will open up a Java source editor for the listener code. Go to the Copy-and-Paste file and retrieve the "private static final" line for the logger and paste into the source:

Note: again you see the red-X, and again resolve that by clicking on the red-X symbol and importing java.util.logging. The red-X should go away.

- ☐ From the Copy-and-Paste file, retrieve the "Before the Step" log statement and paste into the source editor as shown here:

Note: resolve the red-X by importing java.util.logging. The red-X should go away.

- ☐ From the Copy-and-Paste file, retrieve the "After the Step" log statement and paste into the source editor as shown here:

```
28    public void afterStep() {
29        // TODO Auto-generated method stub
30        log.log(Level.INFO, "After the Step");
31    }
32
```

- ☐ Select **Ctrl-s** to save the source Java *and* **Ctrl-s** to save the JSL.
- ☐ From the menu bar, select *Run* *Run Configurations*. You should see what you saw for the Job Parameter execution; that is, your job input parameters.
- ☐ Click the "Run" button to submit the job.
- ☐ Click "OK" for the message indicating the job has been submitted.
- ☐ In the "Java EE Batch Job Logs" tab, double-click on the line that represents this job. You should see the "before" and "after" messages:

And because they appear on either side of the "Hello *<your value>*" message, you know the listener was run before and after the step.

End of Document