Lab 12.1: External Server

In this lab, you will deploy a RESTful service to an external server

Notes:

Lab Synopsis

Lab

- **Overview**: In this lab, you will deploy a REST resource to an external server
  - We'll start with a version using the embedded server
  - With minor changes, we'll deploy to an external server
  - We'll also configure our app to use an embedded Undertow server
- **Builds on previous labs**: None
- **Approximate Time**: 25-35 minutes

Notes:

Lab Preparation

Lab

- The new lab folder where you will do all your work is:
  workspace\Lab12.1

Tasks to Perform

- Close all open files and projects in Eclipse
- **Import** the Lab12.1 project into Eclipse as follows:
  - **File | Import … | Maven | Existing Maven Projects**
  - Click **Next**, Browse to the **workspace\Lab12.1** folder, click **Finish**
- The lab contains the same REST resource we've been using in previous labs
  - Its POM and configuration are the same

Notes:

Run the Application (Embedded Server)

Lab

Tasks to Perform

- Make sure no other app is running
  - Neither external server nor Boot app with embedded server
  - It will prevent the embedded server in this lab from running
- Right click on **BootWebDemo.java** (in Eclipse Package Explorer)
  - Select **Run As | Java Application**
    - We're still running as a regular Java app with embedded server
  - It should run cleanly, and produce output like that below
  - Note the startup of the embedded Tomcat server
    - Which is why we run as a regular Java app - it contains the server
- Browse to *http://localhost:8080/javatunes/rest/items/1*
  - You should see familiar display of an item

```
INFO 82911 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
INFO 82911 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
INFO 82911 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.17]
INFO 82911 --- [           main] o.a.c.c.C.[.[localhost].[/javatunes]     : Initializing Spring embedded WebApplicationContext
```


Notes:

External Deploy - Modify pom.xml

Lab

Tasks to Perform

- **Stop** your running program
  - We're going to deploy now to an external server

pom.xml

- Add a dependency on `spring-boot-starter-`**`tomcat`**
  - With a scope of **`provided`**
  - Because `spring-boot-starter-web` pulls in embedded Tomcat
    - We don't want to bundle that in the deployable app or WAR we'll create later
- Configure packaging as a WAR

```
<packaging>war</packaging>
```

- Update the project (Right click on it, **Maven | Update Project**)

Notes:

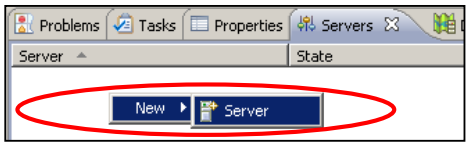
Creating a Server

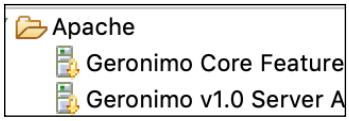
Lab

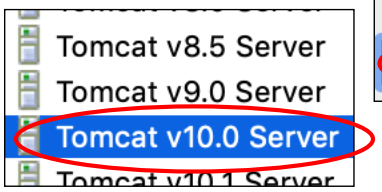
- We will use Tomcat as our external server – so we need to create a server profile in Eclipse

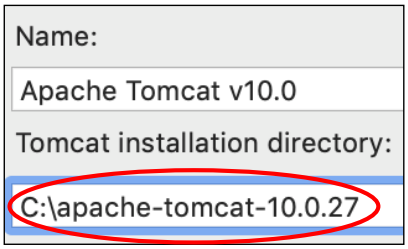
Tasks to Perform

1. Go to the Servers view, right click, and select **New | Server**
2. In the next dialog, select **Apache | Tomcat V10.0**(1), click **Next**
3. In the next dialog, browse to your **Tomcat install directory**(2), click **OK**, and then **Finish**(3)









Notes:

(1) If you use a different Tomcat version then select the appropriate version in the dialog where you choose the server

(2) Tomcat is likely installed in a directory such as **C:\apache-tomcat-10.0.27**

- If its been installed in a different directory, you'll need to modify the instructions in the lab to refer to your install directory
- If it isn't installed, you'll need to download then install it
- Download Tomcat 10.0 from: **https://tomcat.apache.org/download-10.cgi**

(3) If you click Next instead of Finish in Step 3, you'll come to a dialog that lets you configure the project to run on the server

- We are going to do this slightly differently

- Eclipse for Java EE has support for deploying Web applications to a configured server
  - It also has support to start and stop the servers from within Eclipse

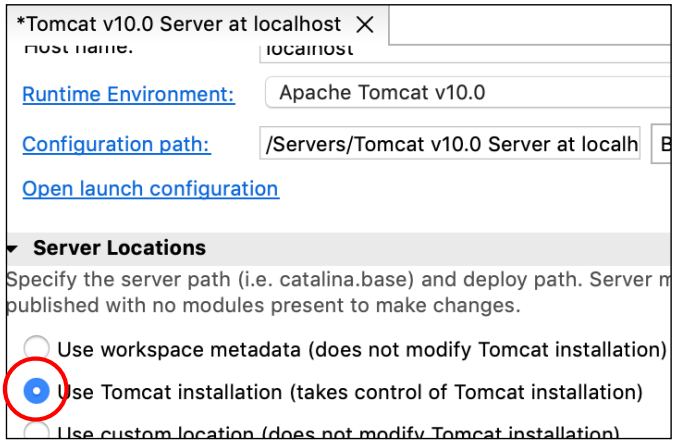
Configure Server

Lab

- We will reconfigure the deploy location for our server
  - To prevent Eclipse deploy issues we've occasionally seen (see notes)

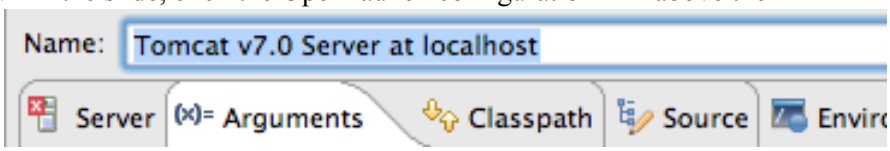
Tasks to Perform

- In **Servers view**, double click on the Tomcat server
  - The server configuration should open in the editor pane
  - Check the "Use Tomcat installation" choice (see below)
  - Save the configuration (See notes about OutOfMemory Exception)

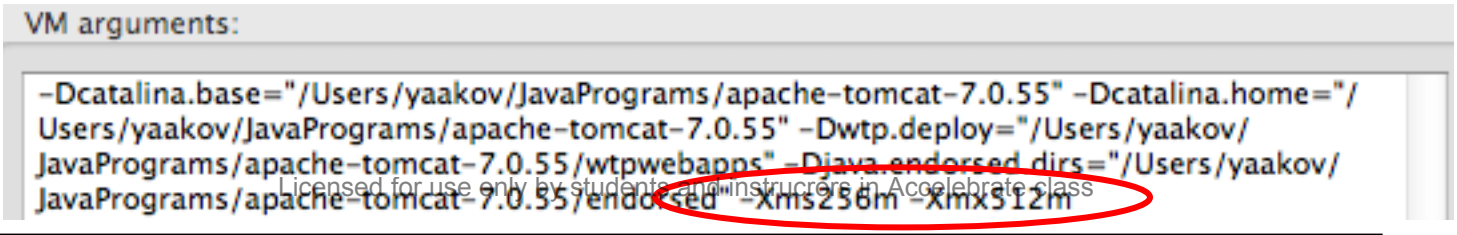


Notes:

- The default Tomcat integration deploys to a location in the workspace (under the .metadata folder)
  - We've sometimes seen issues where this creates intermittent problems in deploying Web apps
  - Changing the deploy location to your Tomcat install folder seems to reduce these problems
  - Accordingly, we tell you here how to change the deploy location to the Tomcat install folder
  - Remember to undeploy the lab apps from Tomcat when done with them
  - So they don't interfere with any other Web apps you need to run on Tomcat
- If you find yourself getting OutOfMemory exceptions periodically when running the Tomcat labs, you can increase the memory allocated to Tomcat
  - In the same configuration editor shown in the slide, click the Open launch configuration link above the Server Locations pane
  - Click the Arguments tab



  - Add VM arguments to increase the heap memory, e.g. -Xms256m -Xmx512m

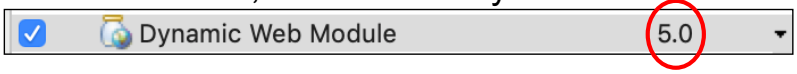


Configure Project for Tomcat

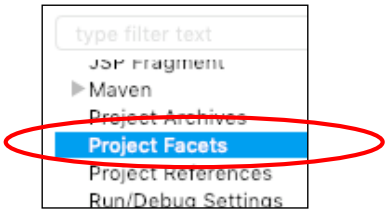
Lab

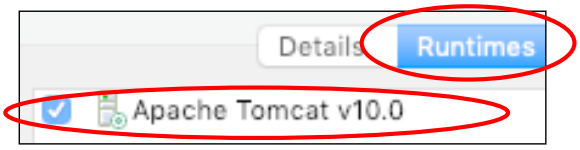
Tasks to Perform

- Right click on Lab12.1 in Package Explorer, select **Properties**
  - Select the **Web Project Settings** node on the left, make sure the Context root is **javatunes** – it should be but correct it if it's not
    - We set it in the POM using `<finalName>`
  - Next, select **Project Facets** on the left, and set the Dynamic Web Module Version to 5.0



  - Still with Project Facets selected, select the **Runtimes** tab on the right
  - Check off your Tomcat server, as shown below
  - Click **Apply and Close**





Notes:

- Note that the configuration on this page is all Eclipse configuration.
  - It's required because we didn't create the project in Eclipse as a Web app associated with a server runtime.
  - This configuration is needed to deploy to a server, so we do it now.

Deploy to Server and Test

Lab

Tasks to Perform

- **Right click** on your server in Servers View
  - Select **Add and Remove…**
  - Select this lab project, click **Add**, then click **Finish**
- **Run** the server - **right click** on the server in Servers View
  - Select **Start**
- Try to view a resource - at *http://localhost:8080/**javatunes**/rest/items/1*
  - **Error**! **404/Not Found** (expected)
  - Why? Because our app is not configured for deployment to an external servlet environment
  - We'll configure it now

Notes:

Configure App for Standalone Deploy

Lab

Tasks to Perform

- Open `BootWebDemo` for editing (`com.javatunes`)
  - Uncomment the `extends SpringBootServletInitializer`
  - Uncomment the `configure()` method
  - This configures the Web app for an external container - other configuration is taken care of by Spring Boot
- **Right click** on your server in Servers View
  - Select **Restart** - so the changes take effect in the server
  - The server console should show the Spring Boot banner/logging
- View a resource again - *http://localhost:8080/**javatunes/rest/items/1***
  - It should work now
  - Optionally, try to change the REST path (**Restart** server, test) (1)
- **Remove the project** from the server, and **stop the server**

Lab 12.1: External Server 237

Notes:

(1) You can change the REST path (the dispatcher servlet mapping) by modifying the appropriate property in *application.properties*

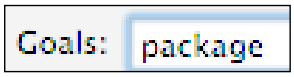
- However, you can't change the Web app context root anymore via a property - that is just usable for the embedded server.
- The Web app context root is now set as part of the deployment process in Eclipse, so must be changed on the Eclipse project (as we did when we set it to javatunes earlier).

Create a Deployable War

Lab

Tasks to Perform

- Open *pom.xml* for review - should have set the packaging to WAR, and excluded the Tomcat starter
- Right click on the project, select **Run As | Maven build …**
  - Name the Run configuration **Lab12.1-package**
  - Enter a goal of **package**, click **Run**



  - Creates ***javatunes.war*** in ***target*** folder (via Spring Boot maven plugin)
  - In Project Explorer view, **right click** on **Lab12.1**, select **Refresh**
  - You should see the WARs that were created (under *target* folder)
    - You can open them with any zip archive tool to examine the contents
  - There should also be a *javatunes* folder with the exploded archive
    - Review its contents - especially *WEB-INF\classes* and *\lib*
    - Contains all the dependency jars, except the Tomcat ones we excluded
- **[Optional]**: You can test it if you want - outside of Eclipse [1]

Notes:

[1] You can deploy the WAR to Tomcat outside of Eclipse to test it.

- Copy the WAR to the *<tomcat-install>/webapps* folder.
- Start the Tomcat server (generally by opening a command window in *<tomcat-install>*/bin and executing *startup.bat* (or startup.sh under *nix).
- Then browse to your REST resource – it should be there.
- Remember to remove the WAR from the webapps folder and shut down the server (shutdown script in bin) when done.

[Optional] Use Embedded Undertow

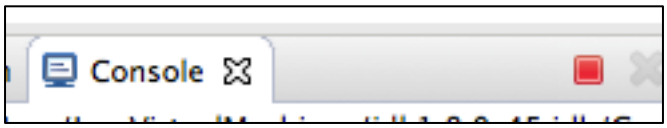
Lab

Tasks to Perform

- **Stop** your Tomcat server and any running applications
- In *pom.xml*
  - Add a dependency to the boot undertow starter
  - Add an exclusion on the boot tomcat starter (see manual slides)
  - Remove the dependency on `spring-boot-starter-tomcat`
- In `SpringBootServletInitializer`, comment out the **`configure()`** method
- Update the project (Right click on it, **Maven | Update Project**)
- Review **Dependency Hierarchy** tab of *pom.xml*
  - Note how Tomcat is gone, and undertow is present
- **Run** the `BootWebDemo` program again (as a Java app)
  - Access your REST resources again - should be no change
  - Look at the console in Eclipse - you should see logging that undertow has started - Easy !

Notes:

- To stop the application, you can click the red rectangle visible when looking at the console view



Summary

Lab

- We started with an embedded server version
  - And with minor changes created a server-deployable version
  - And with more minor changes changed embedded servers
    - From Tomcat to Undertow
- Boot is very flexible in supporting different environments
  - Saving you time!

Notes: