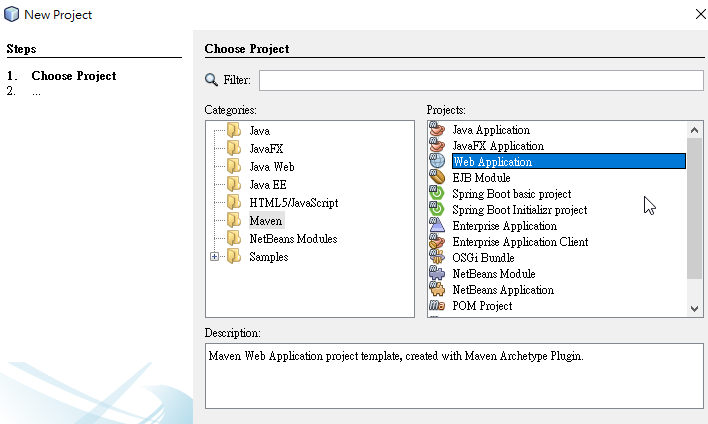
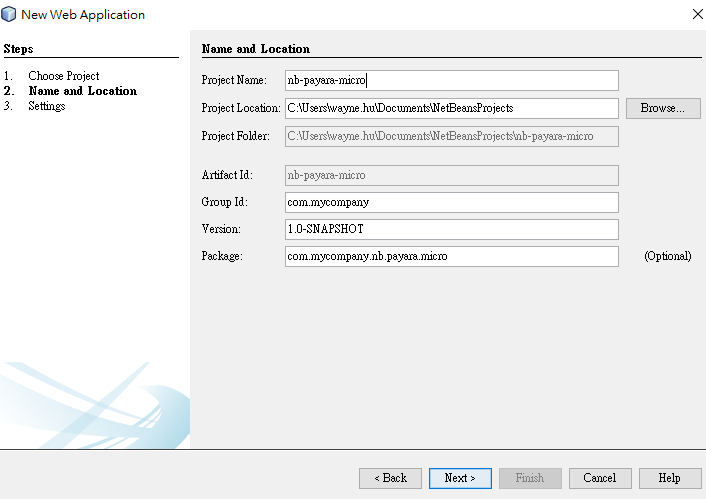
**使用Netbeans & Payara Micro 來開發Micro Service & Deployed on docker (by Wayne Hu)**

1. 建立新Maven專案,

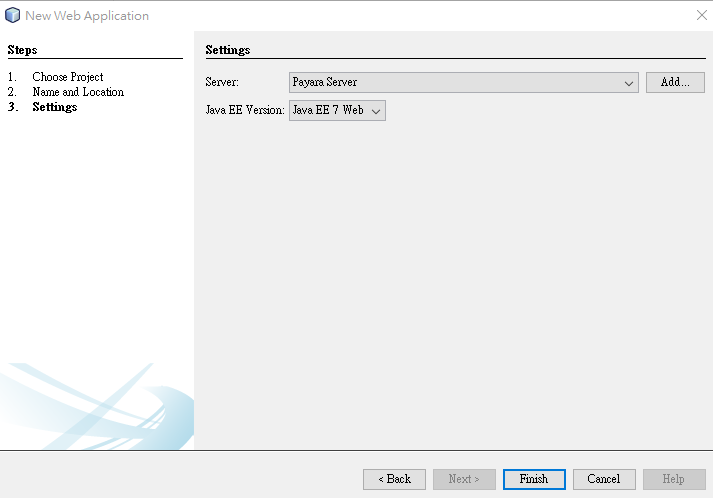
* 於Categories 選擇Maven, Projects 選擇 Web Applications



* 輸入專案名稱

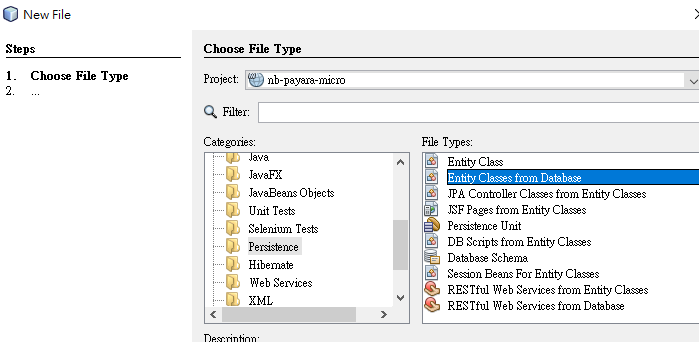


* 於Server 選擇Payara Server.

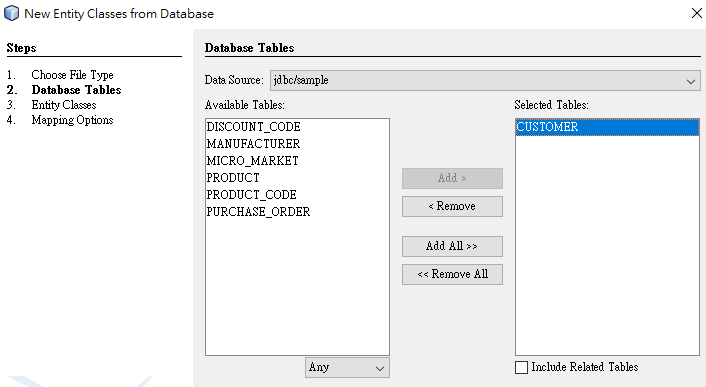


1. 建立JPA Entity

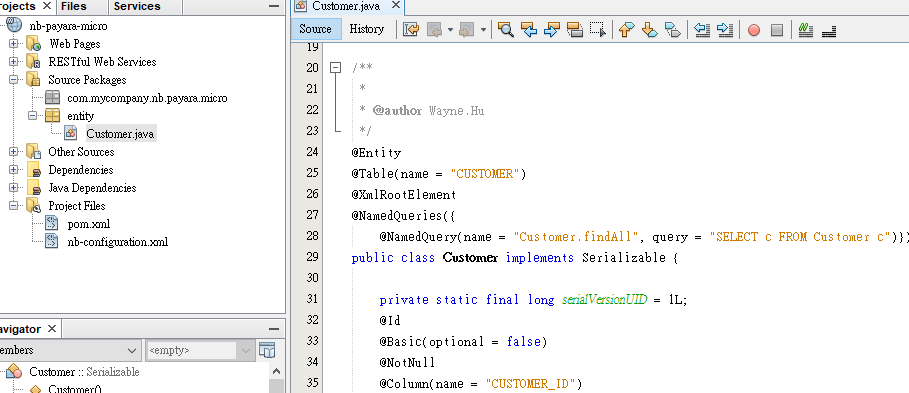
* 選擇New Entity Classes from Database



選擇 Data Source : jdbc/sample, 挑選Customer Table

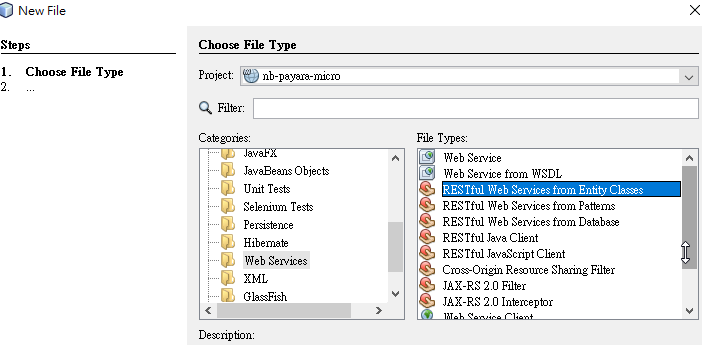


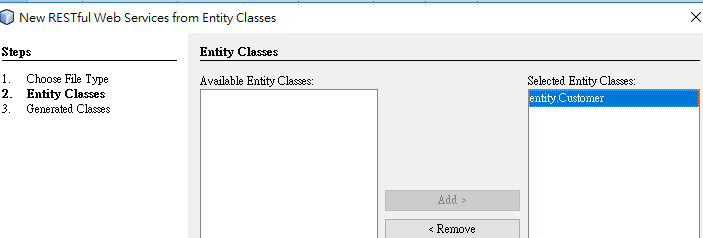
* 建立Entity Customer.java



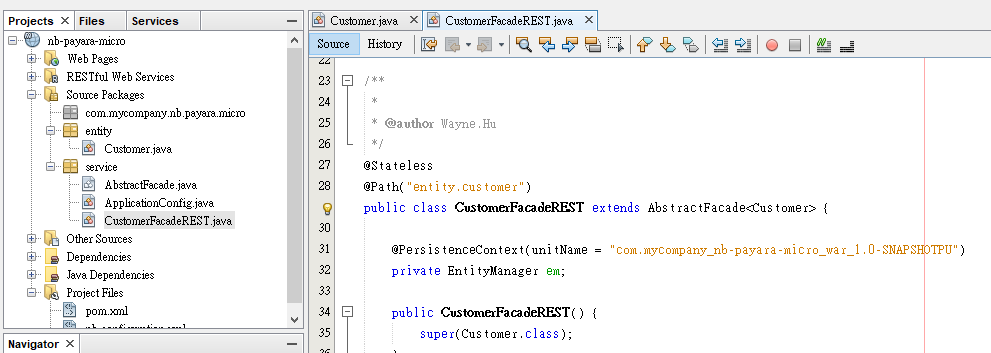
1. 建立Restful Web Service

* 於Categories: Web Services , 選擇建立Restful Web Services from Entity Classes, 並選擇Customer Entity Class.



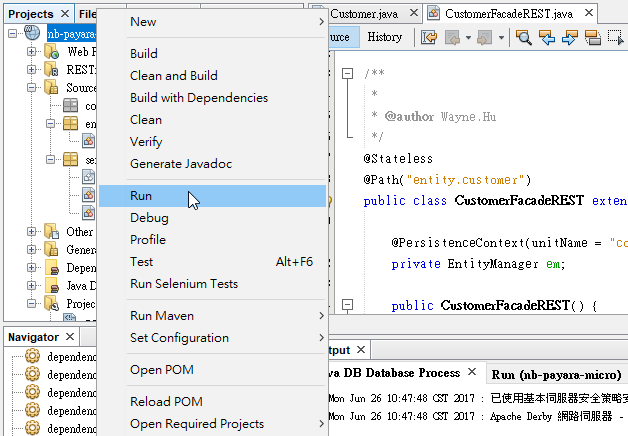


* 成功建立CustomerFacadeREST.java



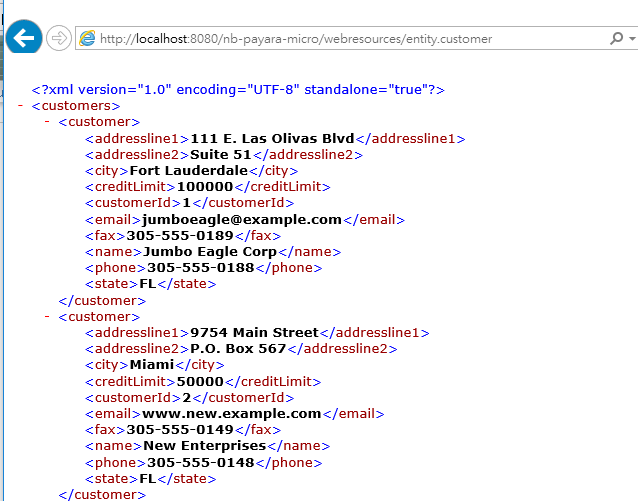
1. 測試Restful Web Service

* 於專案選擇Run



* 並於瀏覽器輸入 <http://localhost:8080/nb-payara-micro/webresources/entity.customer>

得到web service 回傳結果



1. 將專案改成Micro services (Payara Micro) 方式執行

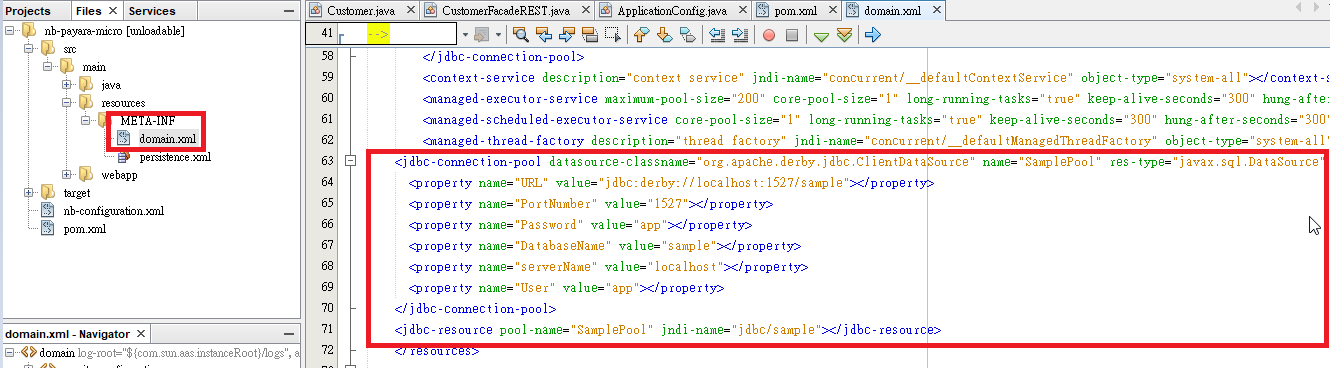
* 於pom.xml 加上payara micro 相關plugin

參考: <https://payara.gitbooks.io/payara-server/documentation/payara-micro/maven/maven.html>

|  |
| --- |
| <dependency>  <groupId>fish.payara.extras</groupId>  <artifactId>payara-micro</artifactId>  <version>4.1.2.172</version>  </dependency> |
| <execution>  <id>copy-payara-micro</id>  <phase>package</phase>  <goals>  <goal>copy</goal>  </goals>  <configuration> <outputDirectory>${project.build.directory}</outputDirectory>  <stripVersion>true</stripVersion>  <silent>true</silent>  <artifactItems>  <artifactItem>  <groupId>fish.payara.extras</groupId>  <artifactId>payara-micro</artifactId>  <type>jar</type>  </artifactItem>  </artifactItems>  </configuration>  </execution> |
| <plugin>  <groupId>org.codehaus.mojo</groupId>  <artifactId>exec-maven-plugin</artifactId>  <version>1.6.0</version>  <executions>  <execution>  <goals>  <goal>exec</goal>  </goals>  </execution>  </executions>  <configuration>  <executable>java</executable>  <arguments>  <argument>-jar</argument>  <argument>${project.build.directory}/payara-micro.jar</argument>  <argument>--deploy</argument>  <argument>${project.build.directory}/${project.build.finalName}.war</argument>  <argument>--domainConfig</argument>  <argument>${project.basedir}/src/main/resources/META-INF/domain.xml</argument>  </arguments>  </configuration>  </plugin> |

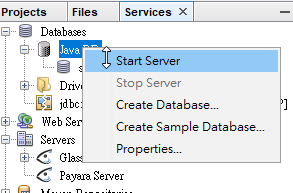
* 因上述執行參數--domainConfig需要一個自己專案的domain.xml , 作法是由Payara Micro 中將domain.xml 抽離出來, 置放於專案目錄中, 然後將前面 Payara Server 中的domain.xml 將 </resources> 上方的jdbc-connction-pool 及 </resources> 複製到專案中的domain.xml

參考文件: <https://stackoverflow.com/questions/32899120/how-do-i-add-jdbc-drivers-and-configure-jdbc-resources-in-payara-micro>

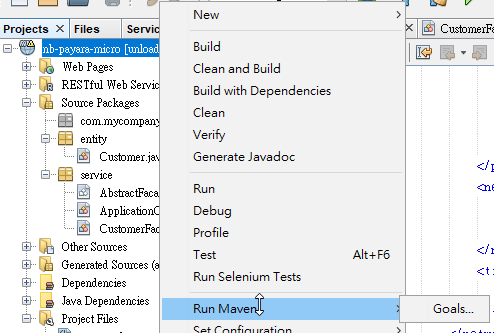


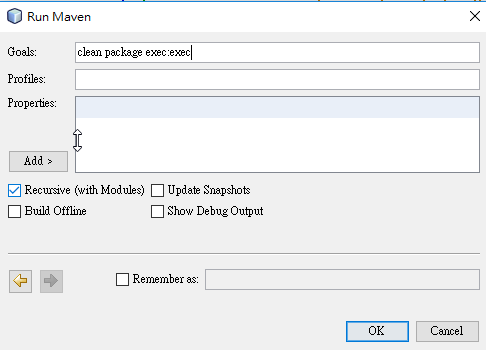
1. 透過Maven 執行Micro service

* 停掉Payara Server 以釋出port #8080, 然後手動啟動 Java DB



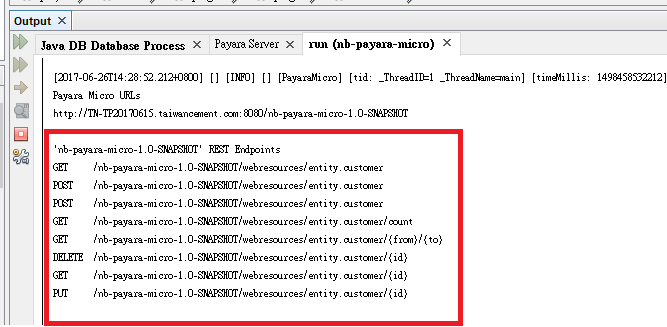
* 於 專案中, 選擇Run Maven, 並輸入Goals : clean package exec:exec



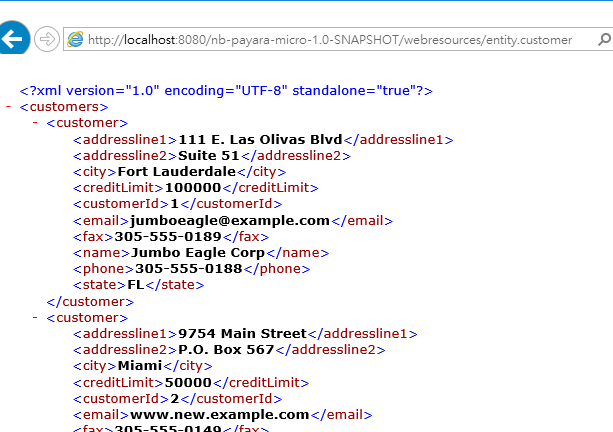


1. 測試結果

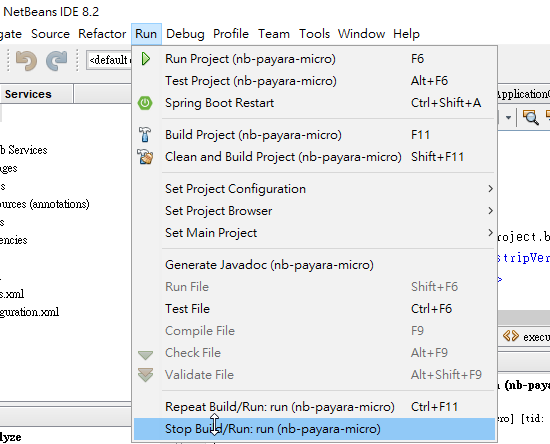
* Payara Micro 執行後, Console 會顯示web service URL (注意: URL 有加上-1.0-SNAPSHOT)



* 於瀏覽器輸入URL: <http://localhost:8080/nb-payara-micro-1.0-SNAPSHOT/webresources/entity.customer>



* 在下次執行前, 需於Netbean/Run 中停掉執行中的Micro service



**Deployed on Docker**

1. 安裝 Docker (略, 請自行參考Docker 安裝手冊)
2. 依Docker 安裝環境, 於Windows 設定環境變數, 範例如下:

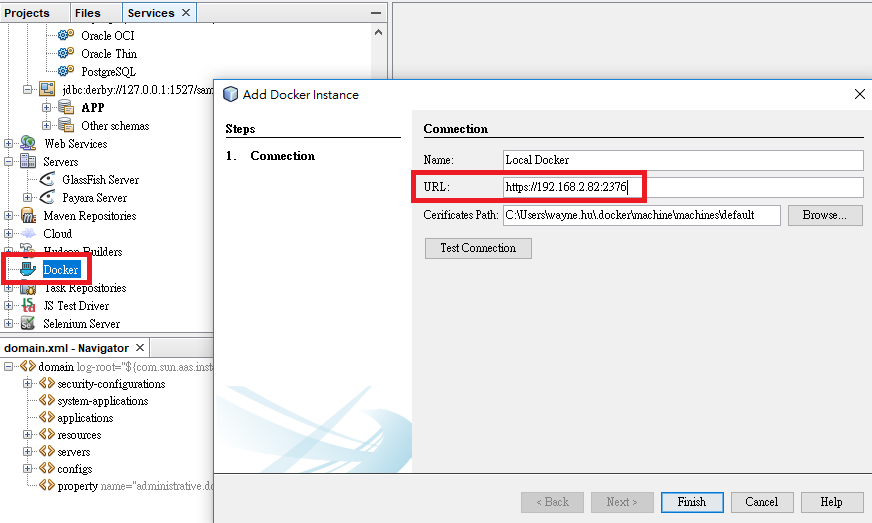
DOCKER\_CERT\_PATH = C:\Users\wayne.hu\.docker\machine\machines\default

DOCKER\_HOST = tcp://your\_docker\_machine\_ip:2376

DOCKER\_MACHINE\_NAME = default (Your docker machine name)

DOCKER\_TLS\_VERIFY = 1

1. 於 Netbeans 加入Docker Instance ( https:// your\_docker\_machine\_ip:2376)



加入Docker Instance 後, 會顯示Docker Images & Containers (如下圖)



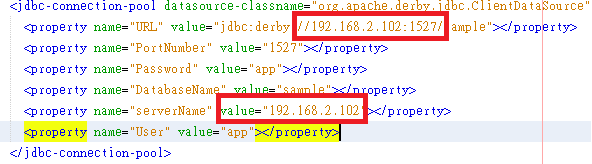
1. 修改前述Maven 專案的 pom.xml, 加入

<phase>package</phase> 以及

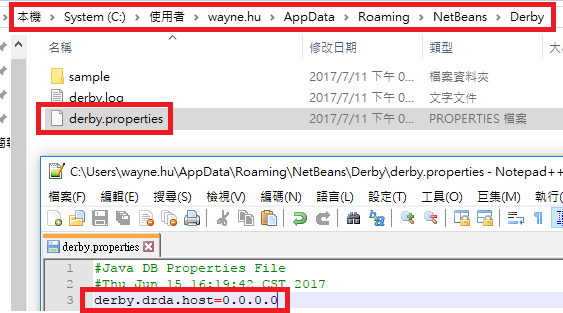
<argument>--outputUberJar</argument> <argument>${basedir}/target/${project.build.finalName}.jar</argument>

|  |
| --- |
| <plugin>  <groupId>org.codehaus.mojo</groupId>  <artifactId>exec-maven-plugin</artifactId>  <version>1.6.0</version>  <executions>  <execution>  <phase>package</phase>  <goals>  <goal>exec</goal>  </goals>  </execution>  </executions>  <configuration>  <executable>java</executable>  <arguments>  <argument>-jar</argument>  <argument>${project.build.directory}/payara-micro.jar</argument>  <argument>--deploy</argument>  <argument>${project.build.directory}/${project.build.finalName}.war</argument>  <argument>--domainConfig</argument>  <argument>${project.basedir}/src/main/resources/META-INF/domain.xml</argument>  <argument>--outputUberJar</argument>  <argument>${basedir}/target/${project.build.finalName}.jar</argument>  </arguments>  </configuration>  </plugin> |

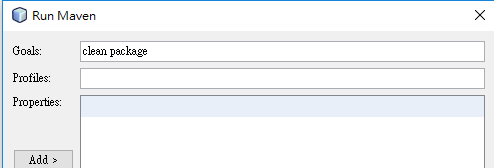
1. 因為我們要建的docker image 內並無Derby DB, 所以我們修改 domain.xml, 連向遠端的現有Derby DB (另一種做法是連向一個Derby DB docker image, 但我找不到一個現成Derby DB docker image 內建有sample db, 所以先連結原本的Derby DB, 但是由localhost連線, 改成遠端連線)



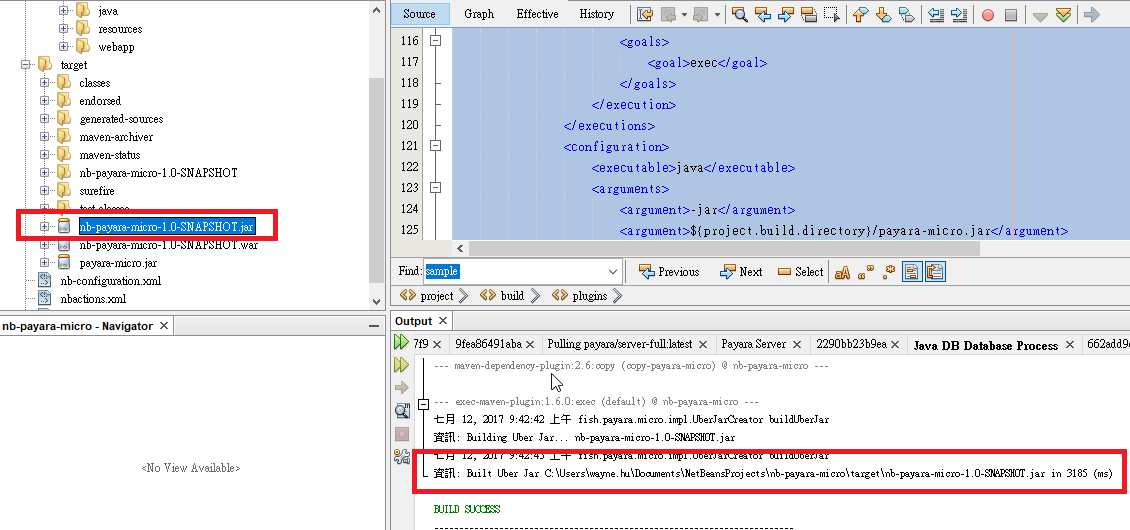
1. 但Derby 預設只能透過localhost 連線, 我們必須修改 Derby DB 的derby.properties, 加入derby.drda.host=0.0.0.0 , 才能透過遠端連線



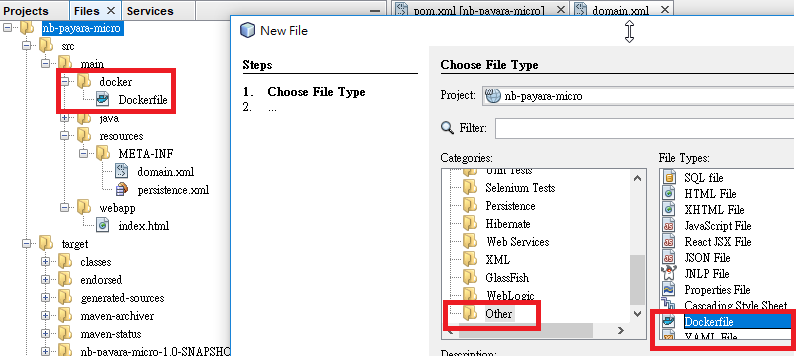
1. 執行 mvn clean package,



1. 執行後會產生一個Uber Jar 檔 nb-payara-micro-1.0-SNAPSHOT.jar, 我們將透過這個Uber Jar檔來建立我們的docker image.



1. 於src/main/docker 目錄, 透過Netbeans 建立一個Dockerfile



* Dockerfile內容如下:

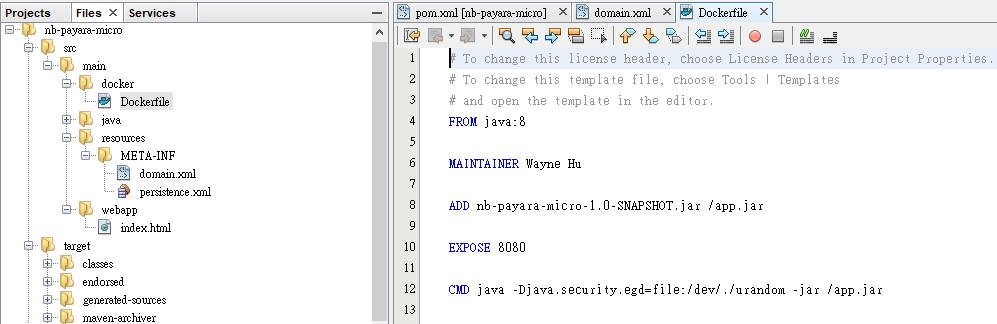
FROM java:8

MAINTAINER Wayne Hu

ADD nb-payara-micro-1.0-SNAPSHOT.jar /app.jar

EXPOSE 8080

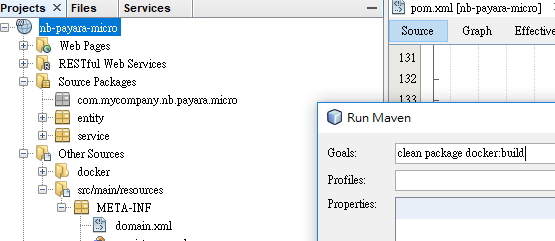
CMD java -Djava.security.egd=file:/dev/./urandom -jar /app.jar



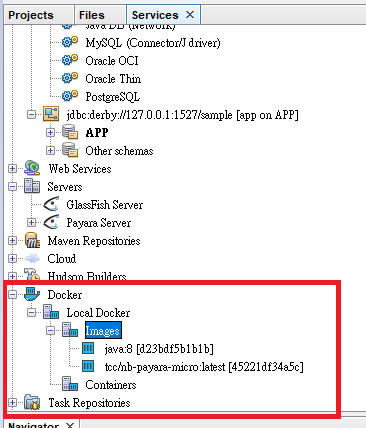
1. 於Maven pom.xml 加入build docker image 的plugin 如下:

|  |
| --- |
| <plugin>  <groupId>com.spotify</groupId>  <artifactId>docker-maven-plugin</artifactId>  <version>0.4.13</version>  <configuration>  <imageName>tcc/${project.artifactId}</imageName>  <dockerDirectory>src/main/docker</dockerDirectory>  <resources>  <resource>  <targetPath>/</targetPath>  <directory>${project.build.directory}</directory>  <include>${project.build.finalName}.jar</include>  </resource>  </resources>  </configuration>  </plugin> |

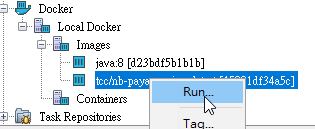
1. 執行 mvn clean package docker:build



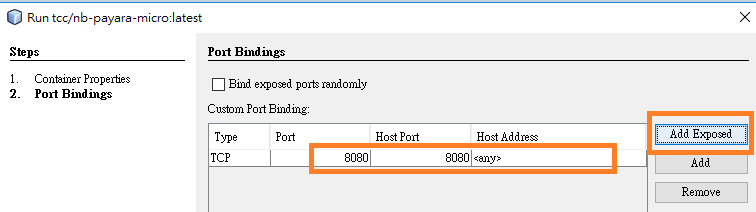
1. 將會於Docker machine 看到我們新建立的 tcc/nb-payara-micro image, 以及它所需要的 Java:8 image.



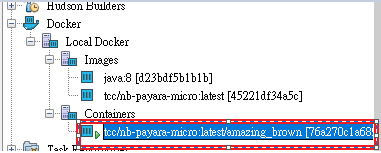
1. 手動執行我們的 docker image



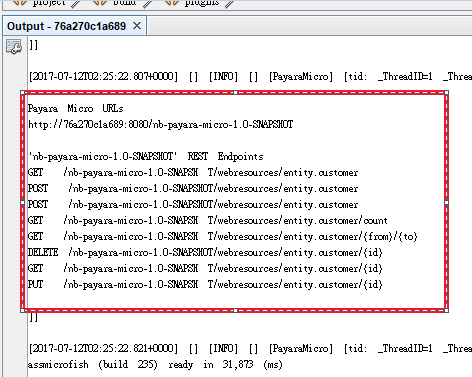
並於Port Bindings 畫面, 按下 <Add Exposed> Button, 帶入預設Port 8080



1. Docker 顯示該image 已於Containers 內執行

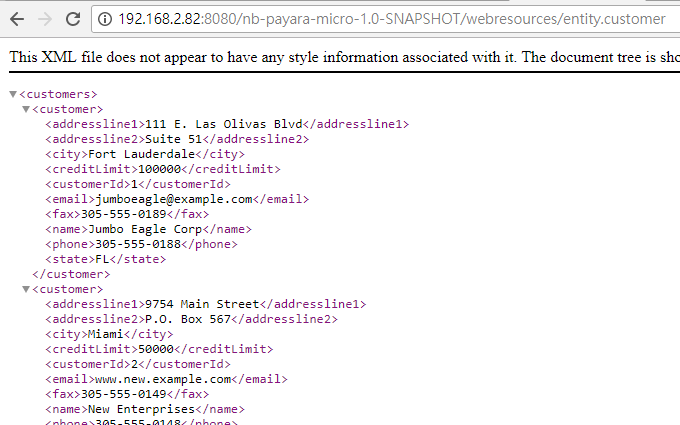


並於Log 內確認Derby DB 連線正常, 並顯示 Rest Endpoints 網址

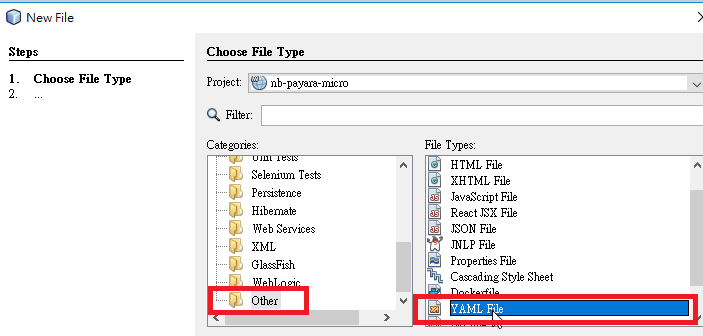


1. 於瀏覽器執行 <http://your_docker_machine_ip:8080/nb-payara-micro-1.0-SNAPSHOT/webresources/entity.customer>

顯示結果如下:



1. 如果想將手動啟動docker image 改成自動化, 可以於專案建立一個 YAML File



並於 MVN exec-maven-plugin 加入 docker-compose 的執行腳本, 最後再執行 mvn clean package exec:exec

|  |
| --- |
| <plugin>  <groupId>org.codehaus.mojo</groupId>  <artifactId>exec-maven-plugin</artifactId>  <executions>  <execution>  <id>DockerRun</id>  <goals>  <goal>exec</goal>  </goals>  </execution>  </executions>  <configuration>  <executable>C:/Program Files/Docker/Docker/resources/bin/docker-compose</executable>  <arguments>  <argument>-f</argument>  <argument>src/main/docker/app.yml</argument>  <argument>up</argument>  </arguments>  </configuration>  </plugin> |