Software Architecture

Week 13 - Micro services

Microservices are a software development technique —a variant of the service-oriented architecture (SOA) structural style— that arranges an application as a collection of loosely coupled services. Typically, microservices are implemented as RESTful web services by invoking HTTP methods (GET, POST, PUT or DELETE) on each component.

During this exercise session, we present a simple application used to save a new customer in a remote database. The application is splitted into 3 modules according to the MVC (Model-View-Controller) design pattern:

- a microservice client
- a microservice implementation of a controller
- an implementation of the Persistence

For this exercise session, we will use Payara Micro which is a JAR file that enables you to run WAR files from the command line without any installation.
It's designed to run Java EE applications in a modern containerized/virtualized infrastructure in the cloud.
To run a WAR file, you only have to run the following command:
`java -jar payara-micro.jar --deploy <Path-of-your-war>`
The generated WAR file is under `<your-project>/target`.
By using this command, you will run your application using the default configuration.

Download the following files:
Exercise files
Payara Micro 5

(Optional)
Demystifying Microservices For Java Developers by Payara

Make a new folder and copy `war` files of each project there. You can find `war` files in the `target` folder of your projects (after building):

1. `CrudController\target\CrudController.war`
2. `CrudPersistence\target\CrudPersistence.war`
3. `CrudView\target\CrudView.war`

Put `payara-micro-5.jar` file to the folder as well.

Creating Database

1. Open *CrudDatabase* application on NetBeans.
2. Run the *H2DatabaseWrapper.java* class and run it.
3. Check the Output to make sure that the database has been successfully created. You should see the following message:

   ```
   Deleting existing database file at
   <PATH-HERE>
   Connection Established: H2/CUSTOMERDB
   JDBC URL: <URL-HERE>
   Login as user "sa" with no password
   ```

4. If the application is still running after this point, you can stop it manually.
5. Create a connection to your database by following the steps mentionned on the following link

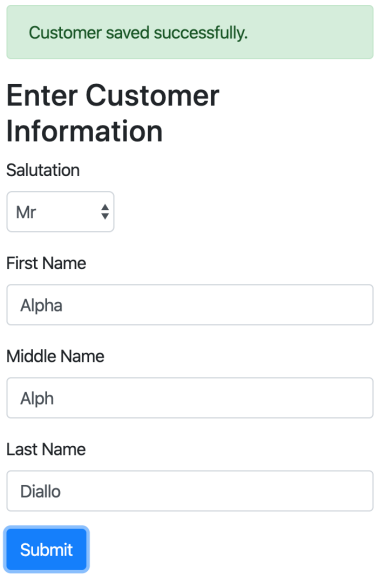
Running the applications

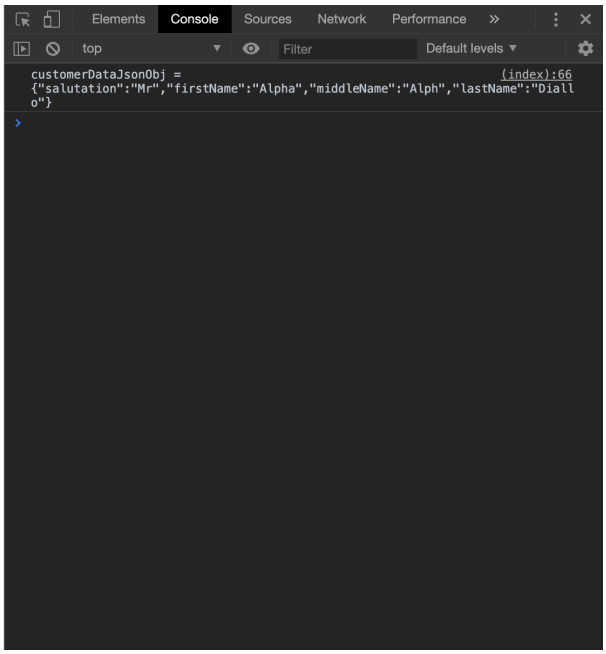
Before you start this part, make sure that Payara Server is **NOT** running! If so, stop it.

In the folder, open the terminal/console and start the microservices respectively (open different sessions and don't close until you're done):

1. Run CrudView application using the script: `java -jar payara-micro-5.jar --noCluster --port 8080 CrudView.war`

2. Run CrudController application using the script: `java -jar payara-micro-5.jar --noCluster --port 8180 CrudController.war`
3. Before running CrudPersistence persistence, we should start the database:
   3.1. Go to Payara Server's installation folder. (If you don't remember where you installed it, you can right-click on Payara Server on NetBeans and see the **Installation location**)
   3.2. Under Payara Server's, go to `bin` folder, then run terminal/console in the `bin` folder.
   3.3. Type `asadmin`, when it starts type `start-database` (Windows) or `./asadmin start-database` if you are using MacOS or Linux.
4. Run CrudPersistence application using the following script: `java -jar payara-micro-5.jar --noCluster --port 8280 CrudPersistence.war`
5. Now you can go to `http://localhost:8080/CrudView/` (View) to insert the Customer's information.





Checking the Health of the Applications

Once you run your applications, you can check if they are *UP* and running. To do so, you can change the URL as shown below:
Before: `localhost:8080/CrudView/`
After: `localhost:8080/health`

The text in the body of the page should say either `state: DOWN`, which means it doesn't work or `state: UP`, which means it works.

Tips

To generate your WAR file using Maven, you can simply right-click on your project > Run Maven > Goals. In the `Goals`' field, type `compile war:war`, set a name (`Remember as:`) and save.