Querying JPA with JPQL

Ex-1)

This exercise uses the database introduced here as a starting point: http://www.mysqltutorial.org/mysql-sample-database.aspx.

So the purpose of the exercise is twofold:

- Show how to create a JPA based design from an existing database
- Use this database and its data to get experience with JPQL (Java Persistence Query Language)

Before you start you should investigate the ER-diagram given in this link to an overview of the tables involved in the design.

Getting started:

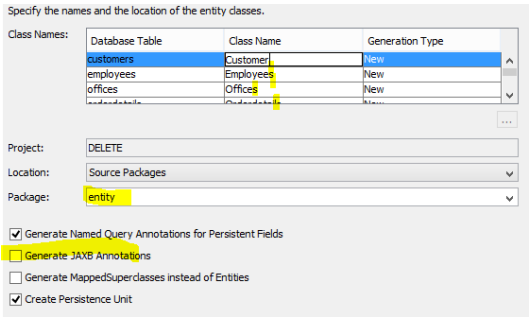
- Create a new plain java project "classicmodels".

Set up the Database

- Create a folder *scripts* and unzip and copy the script from *classicCarsEdited.zip* into the folder.
- Navigate to the services tab, select your local MySQL Server, right click and create a new Database "classicmodels".
- Go back to the script, right click and select "Run File". When prompted for the database to use, select the classicmodels database and execute the script.
- Verify that the tables has been created and populated with data.

Creating the Entity Classes

- Use the wizard "New Entity Classes From Database" to create Entity classes matching the tables in the classicCars database.
    - On the first wizard page, select the database and "Add all" to include all tables
    - On the second wizard page write ***entity*** for Package and deselect "Generate JAXB Annotations"
      The original designers of the database went for the plural form of naming tables (employee**s** and not employee). This is something we definitely not want to be reflected in our class names. Go to the Class Name column as sketched below, and rename all class names to their singular form[1].



    - On the last wizard, de-select the "Use Column Names in Relationships" since this typically ends up with some very unintuitive names

[1] Order is a reserved word (in JPQL) so rename this class to `ClassicOrder`

Tasks:

Explain the generated classes, especially the two (extra) classes generated, ending with PK.

Provide a single façade class, with the following methods:

Creating/Editing Entities:

- `createEmploye(..)` →Creates and return a new Employee[2]
- `updateCustomer(Customer cust)` Updates and returns the updated Customer

Querying

- `getEmployeCount()` → return total employees
- `getCustomerInCity(String city)` → Return all customers living in a given city (Barcelona = 1)
- `getEmployeeMaxCustomers()` → Return the employee with most customers.
- `getOrdersOnHold()` → Return all orders where status is "On Hold"
- `getOrdersOnHold(int customerNumber)` → Return all orders on hold for a given customer (try Customer 144)

[2] Just manually query the database, before you execute the statement to get the next free employee number. A new employee must have an office, assign the office with officeCode = "1" to all new Employees