Hibernate Persistence API (HPA)

Exercise HPA.1 – Basic Hibernate

The Setup:

The main objectives of this exercise are to guide you through creating your first hibernate application.

To get started download the W1D4-Hibernate-1 from Sakai, and add the following dependencies to the pom.xml:

```
<dependency>
      <groupId>org.hibernate</groupId>
      <artifactId>hibernate-core</artifactId>
      <version>6.2.3.Final</version>
</dependency>
<dependency>
      <groupId>com.mysql</groupId>
      <artifactId>mysql-connector-j</artifactId>
      <version>8.0.33</version>
</dependency>
<dependency>
      <groupId>org.apache.logging.log4j</groupId>
      <artifactId>log4j-core</artifactId>
      <version>2.17.2</version>
</dependency>
```

Next create a META-INF directory (should be all capital letters) inside the resources directory and create the following persistence.xml file inside META-INF (you may want to change the MySQL username and password inside this file):

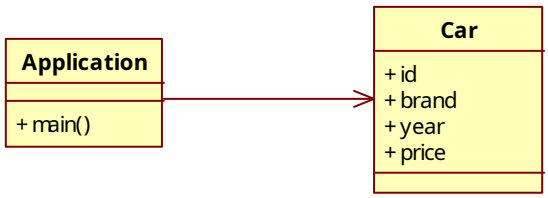
```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
      http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
      version="2.1">

  <persistence-unit name="cs544">
    <description>Persistence unit for Hibernate </description>
    <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
    <properties>
      <property name="javax.persistence.jdbc.url"
            value="jdbc:mysql://localhost:3306/cs544?useSSL=false"/>
      <property name="javax.persistence.jdbc.driver"
value="com.mysql.cj.jdbc.Driver"/>
      <property name="javax.persistence.jdbc.user" value="root"/>
      <property name="javax.persistence.jdbc.password" value="root"/>
```

```
      <property name="hibernate.dialect"
value="org.hibernate.dialect.MySQLDialect" />
      <property name="hibernate.show_sql" value="true" />
      <property name="hibernate.format_sql" value="true" />
      <property name="javax.persistence.schema-generation.database.action"
value="drop-and-create"/>
    </properties>
  </persistence-unit>
</persistence>
```

The Application:

The provided application is very straight forward. The AppCar class creates several Car objects, and uses JPA to persist and then retrieve them.



Running AppCar should create the following output:

```
08:14:05.510 [main] WARN  org.hibernate.orm.connections.pooling - HHH10001002:
Using Hibernate built-in connection pool (not for production use!)
08:14:05.510 [main] WARN  org.hibernate.orm.connections.pooling - HHH10001002:
Using Hibernate built-in connection pool (not for production use!)
Hibernate: drop table if exists Car
Hibernate: create table Car (id bigint not null auto_increment, brand
varchar(255), price double precision not null, year varchar(255), primary key
(id)) engine=MyISAM
Hibernate: insert into Car (brand, price, year) values (?, ?, ?)
Hibernate: insert into Car (brand, price, year) values (?, ?, ?)
Hibernate: select car0_.id as id1_0_, car0_.brand as brand2_0_, car0_.price as
price3_0_, car0_.year as year4_0_ from Car car0_
brand= BMW, year= SDA231, price= 30221.0
brand= Mercedes, year= HOO100, price= 4088.0
```

How much output is generated is controlled by the show SQL properties in **peristence.xml** file, and the log level settings in the **log4j2.xml** file (see example log4j2.xml below).

Try changing the the word **warn** to **debug** on line 12 and running AppCar again. You can run your Hibernate application without a log4j.xml file, but it will generate a warning.

The Database:

Hibernate will automatically generate the database schema needed for the application. To view the database schema and the objects that were persisted by the application, you can use the MySQL plugin for Visual Studio Code, or MySQL Workbench, or any other tool that you are comfortable using for MySQL.

It is important that you know how to drop and create the database schema. Hibernate generates tables for us, but sometimes when there are tables from a previous exercise still present it gets stuck and throws errors.

In Visual Studio Code with the MySQL plugin you can do this by right clicking on the cs544 and selecting **New Query**. Then write the following SQL code to drop and create:

```
drop database cs544;
create database cs544;
```

Right click on the SQL editor and select **Run MySQL Query** to execute it.

The Exercise:

Study the code, and once you feel comfortable with what it does, create a similar but slightly more involved book application existing of a Book class and a AppBook class.

The Book class should have the following attributes and should be annotated for persistence:

```
public class Book {
	private Integer id;
	private String title;
	private String ISBN;
	private String author;
	private double price;
	private java.util.Date publish_date;
}
```

You can use your IDE to automatically generate the Constructors and Getter/Setter methods for your Book class. Hibernate requires that all entity classes have a default (empty) constructor, and a getter and a setter method for each attribute.

See the next page for the actions you should take with this Book class.

Create an **AppBook** class that performs the following operations:

- Open a EntityManager
- Create 3 books save them to the database
- Close the EntityManager

- Open a EntityManager
- Retrieve all books and output them to the console
- Close the EntityManager

- Open a EntityManager
- Retrieve all books from the database, and then:
  - Change the title and price of one book
  - Use em.remove() to delete a different book (not the one that was just updated)
- Close the EntityManager

- Open a EntityManager
- Retrieve all books and output them to the console
- Close the EntityManager

Submitting:

Once you're finished with this exercise you can delete the target directory in your project and create a zip file of the project. Then upload it to Sakai, and write a brief summary of your experiences with this exercise in the textbox.

You can submit right away, as you have unlimited resubmits and can simply add to it when submitting your results for the next exercise.

Exercise HPA.2 – Entities from Schema

The Setup:

The goal of this exercise is to practice generating Entity classes from an existing schema.

Start this project by downloading **W1D4-Hibernate-2** from Sakai. Add the hibernate dependencies to the project (as shown in the previous exercise) and also add the following plugin to the plugins section of the **pom.xml** (note that it requires an older MySQL driver)

```
<plugin>
  <groupId>com.smartnews</groupId>
  <artifactId>maven-jpa-entity-generator-plugin</artifactId>
  <version>0.99.8</version>
  <dependencies>
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>5.1.47</version>
    </dependency>
  </dependencies>
</plugin>
```

Note that this plugin depends on an old mysql-connector version that has a known vulnerability. I have not been able to make it work with newer versions.

The Database:

Inside the resources directory of the project you'll find a **simpsons.sql** file. You will want to execute this on MySQL to create the simpsons database.

On the command line you can do the following to execute a SQL file in MySQL. Start by opening a terminal (cmd.exe).

- change the directory to where the file is
- run mysql with the following arguments

```
mysql -u root -p < simpsons.sql
```

In Visual Studio Code you can do this by going to the MySQL area in the file explorer, right clicking on localhost and selecting **New Query**. Paste the contents of simpsons.sql into the query window, then right click on the query window and select **Run MySQL Query**.

To see the result, right click on localhost again in the MySQL area and select Refresh. You should now see a **simpsons** database under localhost. Opening it should show you the four tables: courses, greades, students, and teachers.

Creating Entities:

To generate entities from this schema first create a **entityGenConfig.yml** file inside the resources directory with the following contents (update the username and password as needed):

```
jdbcSettings:
 url: "jdbc:mysql://localhost:3306/simpsons?useSSL=false"
 username: "root"
 password: "root"
 driverClassName: "com.mysql.cj.jdbc.Driver"

packageName: "cs544"
```

Important: spaces are a ciritical part of a ,yml file. If you copy and paste it make sure that you give it correct indentation.

Once you've made the configuration file you can run the following maven command inside the project directory:

```
mvn jpa-entity-generator:generateAll
```

If everything goes well you should see the following output:

```
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.google.inject.internal.cglib.core.
$ReflectUtils$1 (file:/usr/share/maven/lib/guice.jar) to method
java.lang.ClassLoader.defineClass(java.lang.String,byte[],int,int,java.securit
y.ProtectionDomain)
WARNING: Please consider reporting this to the maintainers of
com.google.inject.internal.cglib.core.$ReflectUtils$1
WARNING: Use --illegal-access=warn to enable warnings of further illegal
reflective access operations
WARNING: All illegal access operations will be denied in a future release
[INFO] Scanning for projects...
[INFO]
[INFO] -------------------< edu.mum.cs544:W1D4-Hibernate-2
>-------------------
[INFO] Building W1D4-Hibernate-2 1.0-SNAPSHOT
[INFO] --------------------------------
[ jar ]---------------------------------
[INFO]
[INFO] --- maven-jpa-entity-generator-plugin:0.99.7:generateAll (default-cli)
@ W1D4-Hibernate-2 ---
[INFO]
------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO]
------------------------------------------------------------------------
[INFO] Total time:  2.288 s
[INFO] Finished at: 2019-05-30T09:41:28-05:00
[INFO]
------------------------------------------------------------------------
```

If you check the cs544 package / directory you should see that it generated four java files: Courses.java, Grades.java, Students.java, and Teachers.java. Unfortunately it generates import statements for the old javax.persistence.* instead of the newer jakarta.persistence.*

Notice that it adds the Lombok @Data annotation on top of each class. Be sure to install support for Lombok in your IDE (Lombok Extension), and add the following dependency to your POM

```
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <version>1.18.28</version>
  <scope>provided</scope>
</dependency>
```

The Exercise:

We want to perform the actions written below on our database. But before we can do this we will have to create (copy) a **persistence.xml** file inside the META-INF directory of resources.

If you copy the persistence.xml from the previous exercise there are two important change you have to make.

- On line 12 inside the url change the database name from cs544 to simpsons
- On line 24 of there is a property called: `javax.persistence.schema-generation.database.action` you need to set its value to: **none** (otherwise it will drop and recreate the database in the process of which it delete all the simpsons data).

Create a **App** class that performs the following operations:

- Open a EntityManager
- Retrieve all students from the database and display their names
  important: your query needs to be: from edu.mum.cs544.Students
- Close the EntityManager

- Open a EntityManager
- Add an extra student to the database (you can choose his / her name)
- Close the EntityManager

- Open a EntityManager
- Retrieve all students again from the database and display their names
- Close the EntityManager

Submitting:

Once you're finished with this exercise you can delete the target directory in your project and create a zip file of the project. Then upload it to Sakai, and write a brief summary of your experiences with this exercise in the textbox.