

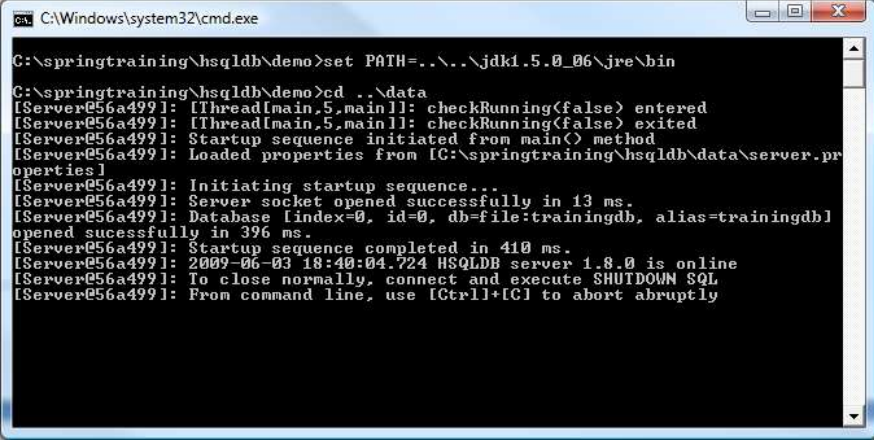
## Lab 3

### Part A: JDBC

For this exercise, we will be using the HyperSQL database (HSQLDB)..

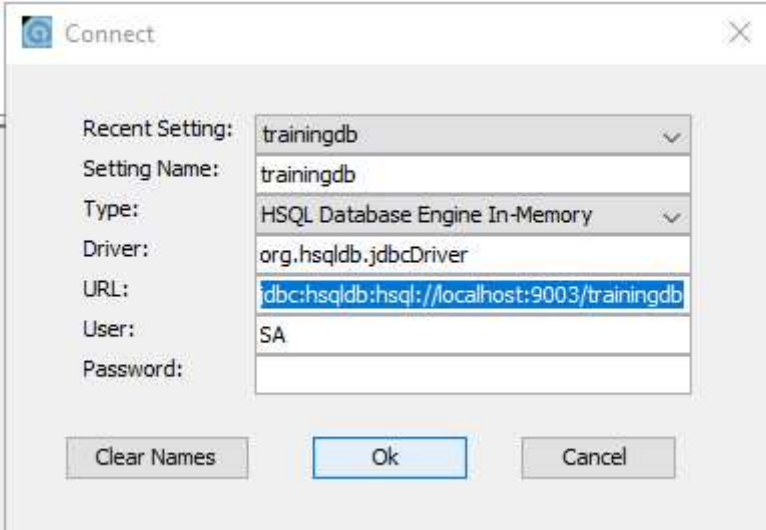
You can start the training database by double-clicking on

**C:\EnterpriseArchitecture\hsqldb\bin\runServerTrainingdb.bat**. This should open the following window to indicate that the database is running.



```
C:\Windows\system32\cmd.exe
C:\springtraining\hsqldb\demo>set PATH=..\..\jdk1.5.0_06\jre\bin
C:\springtraining\hsqldb\demo>cd ..\data
[Server@56a499]: [Thread[main,5,main]]: checkRunning(false) entered
[Server@56a499]: [Thread[main,5,main]]: checkRunning(false) exited
[Server@56a499]: Startup sequence initiated from main() method
[Server@56a499]: Loaded properties from [C:\springtraining\hsqldb\data\server.pr
operties]
[Server@56a499]: Initiating startup sequence...
[Server@56a499]: Server socket opened successfully in 13 ms.
[Server@56a499]: Database [index=0, id=0, db=file:trainingdb, alias=trainingdb]
opened successfully in 396 ms.
[Server@56a499]: Startup sequence completed in 410 ms.
[Server@56a499]: 2009-06-03 18:40:04.724 HSQLDB server 1.8.0 is online
[Server@56a499]: To close normally, connect and execute SHUTDOWN SQL
[Server@56a499]: From command line, use [Ctrl]+[C] to abort abruptly
```

Then, start the database manager by double-clicking on **runManagerSwing.bat** in the same directory.



Connect

|                 |  |
|-----------------|--|
| Recent Setting: | trainingdb                                     |
| Setting Name:   | trainingdb                                     |
| Type:           | HSQL Database Engine In-Memory                 |
| Driver:         | org.hsqldb.jdbcDriver                          |
| URL:            | jdbc:hsqldb:hsqldb://localhost:9003/trainingdb |
| User:           | SA   |
| Password:       |  |

Clear Names    Ok    Cancel

When the manager asks for connection settings, and if this is the first time you start the database manager, then fill in the following information:

Setting Name: **trainingdb**

Type: **HSQL Database Engine Standalone**

Driver: **org.hsqldb.jdbcDriver**

URL: **jdbc:hsqldb:hsqldb://localhost:9003/trainingdb**

User: **SA**

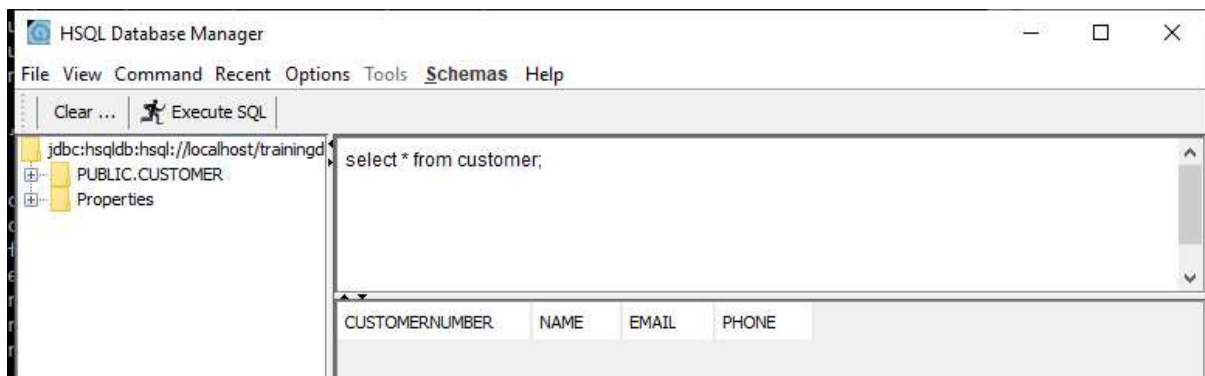
Password:  
And click the **OK** button.

Once the manager is open, enter the following SQL, and then click the **Execute SQL** button to create a product table:

```
create table customer (customernumber integer primary key not null,  
                        name varchar(100) ,  
                        email varchar(100) ,  
                        phone varchar(100) );
```



A customer table is created. You can now enter SQL statements to see the content of the customer table:



In the same way create the creditcard table:

```
create table creditcard (cardnumber varchar(100) primary key not null,  
                          type varchar(100) ,  
                          validationDate varchar(100),  
                          customernumber int  
                          );
```

Now open the given project **Lesson3SpringBootJBDBCCustomerWithCreditCard** that is given in the demo code into your favorite Java IDE and run the file `Application.java`.

The code inserts 2 customers with a corresponding credit card in the database.

| select * from customer |               |                |            |
|------------------------|---------------|----------------|------------|
| ▲▼                     |               |                |            |
| CUSTOMERNUMBER         | NAME          | EMAIL          | PHONE      |
| 66                     | James Johnson | jj123@acme.com | 068633452  |
| 101                    | John doe      | johnd@acme.com | 0622341678 |

| select * from creditcard |            |                |                |
|--------------------------|------------|----------------|----------------|
| ▲▼                       |            |                |                |
| CARDNUMBER               | TYPE       | VALIDATIONDATE | CUSTOMERNUMBER |
| 12324564321              | Visa       | 11/23          | 101            |
| 99876549876              | MasterCard | 01/24          | 66             |

Check the database.

Study the code and now write the Java code to save products in the database.

A product has the following attributes: productNumber, name, price

Make sure to create a new table in the database first.

Write a ProductDAO with the following methods:

- save()** Saves a product in the database
- findByProductNumber()** Return the product with this productNumber
- getAllProducts()** Return the list with all products
- findByProductName()** Return the list with products with this name
- removeProduct** Remove the product with the given productNumber

**In your application test if your methods work.**

Now add a new class Supplier with the attributes name and phone. Assume a 1 to 1 relation between product and supplier.

Modify your code so that every product also has a supplier. Modify all the methods in the ProductDAO so that they work properly with the new Supplier class.

## Part B: JPA

Open the given application **Lesson3SpringJPACustomer** that is given in the demo code. The application inserts a number of customers in the database, and then performs some queries on the customers in the database.

In the same project, create the following entity class

```
public class Book {  
    private int id;  
    private String title;  
    private String ISBN;  
    private String author;  
    private double price;  
}
```

Annotate the class with the correct JPA annotations, and write a Book repository.

Save some books, and retrieve all books from the database.

- Create 3 books, and save them in the database
- Retrieve all books from the database and display them in the console
- Update a book
- Delete a book (not the book that was just updated)
- Retrieve all books from the database again and display them in the console

### What to hand in:

1. A separate zip file with the solution of part A
2. A separate zip file with the solution of part B