Assignment Databases - part 3:
database implementation and front-end

In this third part, we will implement the database and develop a front-end web application to let users interact with the database. For this part, you will start from the data definition file 'shipping_database.sql' and the provided PHP source code which you can find on Toledo.
At this end of this assignment, a short report (with a maximum of 2 pages) needs to be written regarding your intermediary findings. A template containing questions can be found on Toledo.

Hand-in

The report (in PDF) as well as a ZIP-file containing the code needs to be uploaded to your group folder on Toledo. The deadline is Monday 30/03/2015, 10:00 (AM). Please make sure that the code is sufficiently documented to show what you want to accomplish in each part of the code.

Guidelines

We now provide some basic guidelines to setup the database and write PHP scripts that can access and query the database.

Editing .PHP files

To edit .PHP files (and possibly .SQL files), you will need a good text editor. You can use any editor you like. Editors supporting syntax highlighting for PHP are recommended. The computers in the PC classes have Notepad++ installed.

Database creation

To develop the front-end application, we will again use XAMPP. The procedure for creating and loading the database is identical to that in the exercises. For this you will use shipping_database.sql, which is available in the accompanying ZIP file.

Remarks: the size of the files that can be uploaded in PHPMyAdmin in XAMPP is limited to 2MB. Loading larger file is possible by changing the *upload_max_filesize* variable in the file 'php.ini' under '[XAMP install dir]/php'. Remark that uploading larger files might take quite some time.

After setting up the shipping database, you can start writing PHP scripts to query the database.

Connection string

To connect to your database server, you need to specify a connection string, including, a data source name, username, password. This information is stored in a configuration file, which you can find in the file ‘configuration.php'.

PDO-based querying

You will use PHP Data Objects (PDO) to access the MySQL database from PHP. To demonstrate how to use PDO, let's look at the following two simple examples.

Example 1: *Execute select statements*

```
<?php
// create a PDO object
include(configuration.php');
$PDO = new \PDO( $config["dsn"], $config["username"], $config["password"] );
$PDO->setAttribute(\PDO::ATTR_ERRMODE, \PDO::ERRMODE_EXCEPTION);

// execute a query using PDO
$stmt = $PDO->prepare(“ SELECT * FROM CUSTOMER WHERE ssn = ?”);
$myssn = 1;
$stmt->execute(array($myssn));
$rows = $stmt->fetchAll ( \PDO::FETCH_ASSOC );
?>
```

Example 2: *Execute update statement*

```
<?php
// create a PDO object
include('configuration.php');
$PDO = new \PDO( $config["dsn"], $config["username"], $config["password"] );
$PDO->setAttribute(\PDO::ATTR_ERRMODE, \PDO::ERRMODE_EXCEPTION);

// execute a query using PDO
$stmt = $PDO->prepare(“ UPDATE CUSTOMER SET first_name = ?, last_name = ? where ssn = ?”);
$newFirstName = “first name”;
$newLastName = “last name”;
$ssn = 1;
$stmt->execute(array($newFirstName, $newLastName, $ssn));
$nAffectedRows = $stmt->rowCount();
?>
```

ConnectionManager class

To avoid the tedious work of managing PDO, we need a way to abstract the procedures of calling PDO. We demonstrate this aim by introducing a ConnectionManager class, which you can find in 'be/kuleuven/cs/gb/connection/ConnectionManager.php'. Now let's look at how to use ConnectionManager to perform the two queries in the previous examples.

Example 3: *Execute select statements*

```
<?php
require_once("be/kuleuven/cs/gb/connection/ConnectionManager.php");

// create a ConnectionManager object
$con = new \be\kuleuven\cs\gb\connection\ConnectionManager();
$ssn = 1;

// execute the query
$customers = $con->executeSelectStatement("SELECT * FROM CUSTOMER WHERE ssn = ?", array($ssn));
?>
```

Using the Data Mapper design pattern

We also demonstrate how to implement the Data Mapper pattern, which was described during the lectures. For a concrete example, you can have a look at the Mapper class: CustomerMapper in the folder 'be/kuleuven/cs/gb/mapper/' and their usage in the file 'customers.php'.

A web-based front-end for the database

You can now start developing a web-based front-end for the database. Download the starter code from Toledo, decompress it and place this in a directory 'scheepsvaart' under 'C:\Workdir\xampp_183\htdocs\, if you are working on a PC in the PC-rooms, or look for the htdocs directory in your XAMPP installation directory. The index page can now be viewed in a browser via 'localhost/scheepsvaart' (or, equivalently, 'localhost/scheepsvaart/index.php').

Exercise 1: Getting used to PHP

Print the current date on the homepage 'index.php'.

Exercise 2: Experimenting with PHP

1. Inspect the source code to see how the list of customers is printed using the Data Mapping pattern and the PDO interface, which are described in the lecture slides.
2. Print a list of all shipments on the 'shipments.php' page. Present the results in the table that is already specified on that page.
3. Similarly, print a list of all orders on the 'orders.php' page. Present the results in the table that is already specified on that page.

Exercise 3: Querying the database

1. List all customers that live in a certain city, in the following steps:
    1. Load all cities of customers in the dropdown box on the 'customers_in_city.php' page.
    2. Based on the user's specified city, list all customers living in that city. Use the table that is already specified on page 'customers_in_city.php' to present the list.
2. List the revenue of each ship broker, grouped by routes in the last month on the page 'ship_broker_revenue.php'. Present your results using the table specified on that page.

Exercise 4: Altering the database

1. Implement the code necessary to update a ship's information in 'update_ship.php'. Using the web interface, a user can retrieve a list of all ships from the system and then choose a specific ship to perform the update on, by clicking on the ship id. This will lead the user to another page, where he/she can actually do the update for the chosen ship.

2. Implement the code to create a new customer in the database using the form in 'create_customer.php'.

Exercise 5: Ordering shipments

Implement a function that allows users to place shipment orders of customers, according to the following procedure:

1. First, users have to check whether a customer exists or not using the form on page 'order_shipment.php'. This feature is already implemented. Hence, you can input the ssn number of the customer and click the button 'Look up'. Then, the customer's information will be displayed on the form as long as the customer exists. If not, the users are requested to go to the page 'create_customer.php' that you implemented in Exercise 4.2 to create the new customer.
2. Second, users are requested to input the information of the shipment, for example, its volume and weight. Then, the user can click the button 'Order shipment' to save the transaction. This feature has not been implemented yet. You should implement the code to store the shipment in the database.