SY306 Lab Ten

MySQL and Python

0. Introduction

This lab will allow you to practice the techniques of indirect interaction with MySQL databases over the web using Python. In this lab you will add a database back-end for your online store and allow a user to buy products using the shopping cart functionality you previously created.

1. Setup - Do This First!

You must create a folder in your `public_html` called "Lab10" (without the quotes) and store your work in that directory.

Copy your files from Lab07 to Lab10. If your Lab 7 is not working properly (create shopping cart, checkout), contact the instructor. (**Note:** A working Lab 7 solution is available on the course calendar).

IMPORTANT: To allow the webserver to later create files when running a Python script, the webserver user needs to have extra permissions on your Lab10 directory (and others as needed). To enable this, open a terminal window on your Unix machine, SSH into **`midn.cyber.usna.edu`**, and type the following:

```
cd public_html

setfacl -R -m u:www-data:rwx Lab10
setfacl -R -dm u:www-data:rwx Lab10
setfacl -R -m u:m21xxxx:rwx Lab10
setfacl -R -dm u:m21xxxx:rwx Lab10
setfacl -R -m g:cfy_gp_scs:rx Lab10
setfacl -R -dm g:cfy_gp_scs:rx Lab10
```

Advice:

Review the song.py and songPage.py available on the course webpage at bottom of class 18 (link). Much of the code required for this lab is similar to the code provided.

2. Preliminary

You should have the files products.html, shoppingCart.js, shoppingCart.html, shoppingCart.py from your lab 7 and the tables PRODUCT, SALE, SALEITEM from lab 9 part A. (**Note:** A working solution to lab 9 part A is available on the course calendar). The tables MUST be appropriate to store the data for your products and sales. This means that they have to contain the below specified columns, and if you add column(s) they must be documented in a file why_more_columns.txt where you explain why they are required and give one example to support the decision. The primary and foreign keys must be defined as specified in Lab 9. The primary keys are underlined, and the foreign keys are *italicized*. Points will be deducted if these requirements are not met.

> **PRODUCT**(ProductID, PName, PDescription, Price) - this table records information about each product. Modify the table as needed to fit your products. The table must have a numeric primary key. If you have images for your products, you should store the image name/path in the database (i.e. the value for the src attribute in the img tag). You shall NOT store the image files in the database (while it is possible to do so, it is not efficient). Choose appropriate data types for the columns. See https://dev.mysql.com/doc/refman/5.7/en/data-types.html for more information on MySQL data types if needed.
>
> **SALE**(SaleID, DeliveryAddress, CreditCard) - this table records information about each sale/transaction. Modify the table as needed to fit your sale information. SaleID must be a surrogate key - created by MySQL. Hint: Declare SaleID int AUTO_INCREMENT in the column declaration for SaleID.
>
> **SALEITEM**(*SaleID, ProductID*, Quantity) - this table records information about which products were sold in each sale/transaction. SaleID and ProductID should be declared as foreign keys. The primary key is the combination of SaleID and ProductID.

3. Requirements - The Actual Lab:

Read the entire lab so you understand the requirements and know what to expect.

A. Display products: [60 points]

1. Insert the information about the products you have in your products.html page into the database. Write and execute SQL INSERT statements for that. You MUST have at least 5 products in your Products table unless prior agreement has been made with your instructor.
2. Write the Python script(s) to dynamically generate the products.html page, including the "add to cart" buttons based on the data in your Products table. Follow these steps:
    a. Create a file named product.py

b. Write a module or class product in product.py with one method, printProducts(cursor) that returns a string with all the products in the Product table as a nice HTML table very similar to the one you had for your products.html page. This should include the code for the "add to cart" buttons.

c. Write a script productPage.py that when executed, will produce an output similar with products.html. Your productPage.py should invoke the printProducts() from the product.py module to print the products. Test the functionality using the URL http://midn.cs.usna.edu/~m21xxxx/Lab10/productPage.py

d. Make sure that you handle any errors that might arise from the database connection gracefully. The user should not be exposed to the raw server errors.

B. Shopping Cart Functionality [25 points]

1. Ensure that your shoppingCart.js script from Lab 7 is used by the new productPage.py and that the addToCart and viewCart functionality still work as expected. Note: if you hard-coded some product information in JavaScript, that information must now be generated dynamically from the database, or must be removed completely.

2. Ensure that the checkout button/link to the shoppingCart.html is visible when productPage.py is loaded by the browser and it works as expected.

3. Ensure that shoppingCart.html displays the list of products in the shopping cart and the check-out form correctly works with action shoppingCart.py

C. Final Steps [15 points]

1. **Documentation**: Ensure you have appropriate comments in your Python script.

2. **Important final steps:** Create two links in your top-level default.html page under the heading "Lab10":

a. Under the name "Products", make a link to your Lab10/productPage.py page

b. Under the name "shoppingCart.js" make a link to your Lab10/shoppingCart.js file

D.1 - EXTRA CREDIT #1: Products Table from the database - JSON/AJAX style

For a nominal amount of extra credit, instead of the requirements in A.2 do the following:

1. Create productJSON.py with a method that will receive a JSON request for the products in the database and return a JSON response of the products. This JSON response must have all the information required to create the products table for display to the user. HINT: The JSON response should be an object with the required information, not a string of the values that has to be parsed.

2. The script productJSON.py will NOT call the method printProducts from product.py as that function already generates the string for the table, and that will now be generated client side.

3. Create productJSON.js that includes functions to send the AJAX request to productJSON.py, receive the response, and create the HTML table for products from that response.

4. Modify products.html such that it calls the methods in productJSON.js to request the products from the database and display the products table generated by productsJSON.js.

5. Make sure all errors are handled gracefully.

6. The "add to cart" buttons must function as expected.

D.2 - EXTRA CREDIT #2: Buy products - database style

For a nominal amount of extra credit, do the following:

1. You should have the SALE and SALEITEM tables from Lab 9. Check that the tables exist, that the SaleID is the primary key and AUTO_INCREMENT in SALE table, and that the foreign key constraints between SALEITEM and Product and SALE are in place. If needed, modify the tables (add extra columns) such that the information from your checkout form and shopping cart can be stored in these tables.

2. Modify your shoppingCart.py script to save the information into the SALE and SALEITEM tables instead of the files, and if everything OK, delete the shopping cart cookie and display a confirmation message, or error message if errors occurred. Here are the steps:

    a. Write a module or class sale.py with at least one method addSale(cursor, any extra information needed) that inserts the data in the SALE and SALEITEM tables and returns either a SaleID as the confirmation number or 0 otherwise. The function should:

        i. Insert general order data into the SALE table in your database. Note that the SaleID is a surrogate key. The surrogate keys were created using AUTO_INCREMENT keyword, and the value for the surrogate key is generated by the database system. When writing the INSERT statement to insert into the table, you should not insert into the SaleID column (see the addSong function in song.py on the course calendar for an example). Since you need the generated SaleID value for subsequent inserts, you should use cursor.lastrowid -this represents the ID created by the previous insert statement.

ii. Insert each product from the shopping cart in the SALEITEM table. Note that the sale id should correspond to the sale id previously created (see above).

iii. If data was saved in the database, return the new SaleID.

iv. If there was any error, return 0.

b. Modify the shoppingCart.py script to invoke the addSale function with the values posted by the user during checkout. If everything is OK, delete the shopping cart cookie and display the confirmation message, including the transaction number (SaleID) and list of products purchased. If errors, display an error message to the user.

D.3 - EXTRA CREDIT #3: SECURITY -- SQL Injection Defense

1. Next week, class 20 will introduce SQL injection and some best practices to defend against these attacks.

2. For additional extra credit, research the use of Python / MySQL parameterized queries and use them instead of string concatenation to implement the multiple INSERT queries required in section **D.2-2.a** above.

3. When complete, ensure that D.2-2.a is not susceptible to common SQL injection attacks.

4. Deliverables

1. New files: product.py, productPage.py.

2. Modified files: shoppingCart.py

3. Files that could have been modified or stayed the same: shoppingCart.html, shoppingCart.js

4. If completed for extra credit: [productJSON.py, productJSON.js, and sale.py]

5. You should have all the pieces working as described in "Requirements" above.

6. You should have the two links in default.html that are described above.

7. All of your files should be in a folder called "Lab10" (without the quotes) in your public_html. Your instructor will assume that your web pages are viewable at http://midn.cs.usna.edu/~m21xxxx/Lab10/productPage.py

8. **Turn in (due before lab on Friday):**

    1. **Paper submission:** turn in the following hardcopy at the beginning of Lab11, stapled together in the following order (coversheet on top):

        1. A completed assignment coversheet. Your comments will help us improve the course.

2. A printout of the source of your product.py file.

3. A printout of the source of your productPage.py file

4. If you added columns to the database tables a printout of why_more_columns.txt

5. If extra credit is done, a printout of sale.py and shoppingCart.py

6. A screen-shot of your browser window showing the productPage.py loaded in the browser window

2. **Electronic submission:** Submit all Lab10 files via the online system: `submit.cs.usna.edu` by by the following deadlines:

- Section **4341**: 23:59, Wednesday, March 27th
- Sections **1151, 3351, 5551**: 23:59, Thursday, March 28th