(DSDS E2012)

A11: "Last Year's Exam" :-)

48-hour take-home exam:

Introduction to Scripting, Databases, and System Architecture (aka., DSDS)

(E 2011)

IT University of Copenhagen

January 18, 2012

by Claus Brabrand

[brabrand@itu.dk]

(8 pages)

--- General Information ---

Start date: Wednesday, January 18, 2012 at 13:00 on The DSDS-E2011 Course Blog
Hand-in deadline: Friday, January 20, 2012 at 13:00 at The Exam Office, 2E08, ITU

General information about this exam

Exam exercise solving

The students attending the exam are to follow the same procedure they have used when solving the mandatory assignments in the course, except for the delivery (hand-in) of their solutions (see details below). This means that students are recommended to set up their own database on the ITU MySQL server (`mysql.itu.dk`) or use the one they have already set up as part of solving the assignments. Details for setting up a MySQL database on the ITU server can be found in assignment A6 available on the course blog. You are then to develop and execute the exam solution scripts on the ITU PHP-Server by placing the PHP and HTML files in your personal course directory (`W:\e2011\DSDS\`*`username`*`\`). Students who want to develop their exam solution using a different server are welcome to do so, but need to make sure that the code is accessible to the teacher and external examiner (censor) by including a web address in the hand-in package to where the source files can both be inspected and executed (see exam hand-in below) during four weeks following the exam delivery deadline.

Solving the exam exercises involves both the writing of HTML, PHP, and SQL code as well as describing parts of the design in natural language, similar to how solutions for assignments A6-A11 in the course were done. The natural language used can be either Danish or English.

Students are welcome to re-use and include files that they have used themselves throughout the DSDS course (e.g., `fn_headerfooter.php`, `fn_input_validation.php`, and `fn_mydb_connect.php`). Of course, any such files should also be included on the CD-ROM and located in the appropriate place on the web server. Information and code examples freely available on the web (e.g. in PHP programmer's online forums) are also valid resources if successfully adapted to the exam exercise and suitably commented. *Under no circumstances,* however, are students on the DSDS course allowed to copy and/or share information with each other as part of solving the exam (see "Check for Plagiarism" below)!

Exam hand-in

The solutions are to be delivered in the shape of `.html`, `.php`, `.sql`, `.txt`, and `.png/gif/jpg` files all to be burnt onto three (3x) *identical* CD-ROMs and handed in to the exam office (2E08 at ITU) no later than Friday, January 20, 2012 at 13:00. A web address to a folder containing a directly executable version of the designed web service[1] is to be included: a) on a paper note accompanying the CD-ROMs; and: b) as a clickable link from a very simple HTML file called "*username*.html" included on the CD-ROMs. (In both

[1] E.g., [`http://www.itu.dk/stud/f2011/DSDS/`*`username`*`/exam/`].

cases mentioned above, "*username*" refers to the specific student's ITU *username*; e.g. "`brabrand`".) You are not allowed to modify your online service after the deadline. When you hand in, you also have to print out, fill in, and hand in the standard ITU *front cover*[2]. Beyond this front cover, you should not hand in anything on paper.

Hint: Prepare your delivery in good time (e.g. you should ideally have made sure to have a working procedure for burning CD-ROMs *before* the exam starts) because the hand-in deadline is hard. If you miss it, your next chance is not until the next (re-) exam occasion.

Grading

The exam will be graded according to the Danish 7-point grade scale relative to the course's ***intended learning outcomes*** (ILOs) as stated in the ITU course base:

- **Plan** and **develop** medium sized web applications using the scripting language, PHP;
- **Design** small MySQL databases;
- **Construct** PHP scripts that interact with databases using SQL;
- **Describe** the techniques behind database-driven web applications; *and*
- **Describe** the fundamental system architectural considerations behind web applications so as to be able to communicate and collaborate with programmers and technologists.

Example of factors that influence the grade:

- **Functionality:** do the solutions execute without errors and meet the requirements specified in the exam?
- **System robustness:** do the solutions handle bad user input gracefully and handle a non-responding MySQL server gracefully?
- **Code quality:** is the code well structured and commented so that a fellow system developer (in particular, the teacher and censor) easily can understand your program and what it does?
- **Overall system design quality:** is the overall design well thought-through so that the redundancy in the MySQL database is minimized and that the functionality of the web service is logically distributed over a set of HTML and PHP files?

Check for plagiarism

Due to the nature of this exam, extra careful checks for plagiarism will be performed on the handed-in exam solutions. Students are *under no circumstances allowed to collaborate* in solving the exam. Thus, students handing in identical or near-identical solutions will be reported as part of the normal handling of misconduct and fraud at ITU. For details, see:
- [`http://intranet.itu.dk/en/Studiehaandbogen/Eksamen/Regler-og-retningslinjer`]

Note: An oral plagiarism check will take place immediately following the hand-in deadline (in 2A18) on Friday, January 20, 2012 at 13:00 for the 20% of students who are randomly selected for it by the Exam Office. (You will be informed of this when you hand in.)

---

[2] I.e., [`http://intranet.itu.dk/en/Studiehaandbogen/Eksamen/Standardforside`].

--- Web Service Specification ---

Design and implement a Pizza Web Shop: "*La Pizzeria*"

Introduction

In this exam assignment you will design and implement a **Pizza Web Shop:** "*La Pizzeria*" (à la Just-Eat.dk). A registered customer should be able to *log in* to the Pizza Web Shop. Once the customer's credentials (i.e., username and password) are *verified*, the customer should see a list of pizzas that can be *ordered* via appropriate clicking. When a pizza is ordered, it should be added to an *order basket* the contents of which the customer should also be able to somehow see as well as appropriately delete pizzas from. Further, there should be "*proceed to checkout*" link that should take the customer to a page that states what pizzas have now been ordered, the total price, and the address of the customer. As a last step in the pizza purchasing process, the service should then finally collect the customer's credit card number and expiration information and validate that the format of this information conforms to appropriate specifications (see below for details). Finally, there should be two administration pages (without password protection). First, there should be a page for showing the *history of all purchases*. Second, there should be a page for showing a *top five list* of the best customers.

General requirements for your "*La Pizzeria*" Web Service:

1. **Logging in:** Already registered customers should be able to log in to the service by entering their username and password. The actual registering of customers is *not* part of this assignment (i.e., it is perfectly fine to just manually insert a few customers into some appropriate MySQL database table of registered customers). If the log in is successful (i.e., the information entered by the customer matches the contents of the database), the customer should then be able to order pizzas (see further below). If the log in fails, the customer should get some appropriate error message. Your service only needs to perform the password validation once upon first entry (i.e., it does not have to validate users that "jump" into the middle of the service by "guessing" the name of a subsequent PHP script).
2. **Pizza list:** The customer should be presented with a list of pizzas (name, ingredients, and price) that can be ordered, sorted by price (see the top part of Screen Shot 1 on last page of the exam). Each pizza should come with a link for adding it to the order basket the contents of which the customer should also somehow be able to see (see below).
3. **Order basket:** The customer should also be presented with the contents of the order basket (see the bottom part of Screen Shot 1). In the order basket, a pizza should be listed just with its name and price (i.e., without its ingredients). Each pizza should come with a link for removing it from the order basket. The basket should be stored as normal "persistent state" (i.e., in a suitable table in the database) and not as "session state".
4. **Notes** (about requirements 2. and 3.)**:** The "Pizza list" and the "Order basket" do not have to be on the same page (as is the case on Screen Shot 1); they can just as easily reside on separate pages connected by appropriate links (e.g., "show my basket" and "order more pizzas"). This is up to you. There also needs to be a way of taking the customer from these page(s) to the checkout (e.g., a "Proceed to checkout" link as in Screen Shot 1).

5. **Checkout:** When the customer is done ordering pizzas and has clicked "proceed to checkout", he/she should be presented with an alphabetically ordered list of the pizzas indeed ordered, the total price, and a delivery address as originally registered in the customer database table (cf. requirement 1. and Screen Shot 2). You are also very welcome to present the list of pizzas in a slightly more fancy version (not shown in Screen Shot 2) by collecting pizzas with the same name (e.g., 3x Margherita: 135 kr.; 2x Vegetariana: 110 kr.; 5x pizzas in total: 245 kr.). However, this is not a mandatory requirement (i.e., you can still get a 12 for a "perfect solution" without this feature).

6. **Credit card validation:** Before a purchase is finalized, the customer needs to enter a valid *credit card number* (a 16-digit number where each 4 consecutive digits are separated by a hyphen; e.g., "`1234-5678-9012-3456`") and a legal *month* (two digits, possibly with a leading zero as in "`09`") and two-digit *year* (e.g., "`15`"). If the credit card information is valid according to the above formats, the purchase is considered *completed* which means that the pizzas should be removed from the order basket and an entry should be made in a purchase table (an entry should consist of a customer, the total price and the date of purchase). If the credit card information does not comply with the formats, an appropriate error message should be given.

7. **Purchase history:** There should be an administration page (without password protection) that will show a list of all the purchases made (name, purchase price, and date of purchase), ordered by the purchase date.

8. **Best customers:** There should also be an administration page (without password protection) that will show the top five best customers (i.e., the five customers who have spent the most money, in total, over time, buying pizzas), sorted by total amount spent. It should state the name of the customer, how many purchases he/she has made, the average amount spent per purchase, and, of course, the total sum spent.

9. **index.html**: Finally, there should be a simple "`index.html`" file with links to use the service. It should contain a link to start the service (i.e., to the customer "log in" page) and links to each of the two administration pages.

Your "*La Pizzeria*" Web Service does *not* have to:

1. **Register customers and pizzas:** You do not have to make a way of registering a new customer (you can just create a few customers directly in some appropriate MySQL database customer table) nor do you need any way of adding new pizza kinds in your service (again, you can just add some directly in the database beforehand).

2. **Administrator password:** You don't have to protect the two administration pages ("purchase history" and "best customers") by passwords in any way.

3. **Bank transactions and pizza delivery:** You, quite obviously, don't need to perform any real bank transactions. Once the credit card information is validated, we'll just assume a bank transaction is conducted. Also, you obviously don't need to deliver any pizzas. :-)

Note I: The exam is deliberately loosely defined in order to force you to make use of many of the design skills acquired during the DSDS course. Use your common sense to resolve any ambiguities or under-specifications. Whenever you find the requirements specification above unclear, choose an interpretation and state that interpretation clearly in the file where you describe your design (`description.txt`).

Note II: If you run out of time, just simplify or downscale the project yourself and hand in a "partial solution". There are many possibilities for simplification (e.g., skip an admin page).

--- The Exam Exercises ---

Not surprisingly, the exercises proceed according to the *"Four-step Web Service Design and Implementation Process"* you have seen and worked with throughout the course:

Exercise 1: Design of Data Model (20%)

Exercise 1a): MySQL Database Design (8%):

`exam/1a_table_design.png/gif/jpg`	*(destination file)*
`exam/description.txt`	*(destination file)*

Design the tables you need in your service. Remember to do your design so that data redundancy is kept at a minimum. Draw your tables with a few rows of sample data in them and show how the tables and columns relate to each other (as in the To-Do-List Web Service: http://www.itu.dk/people/brabrand/DSDS-12.pdf on slide number 10). Write a couple of sentences describing your design under the heading "Exercise 1a Table Design" in `description.txt`.

Exercise 1b): MySQL Table Definitions (8%):

`exam/1b_table_definitions.sql`	*(destination file)*
`exam/description.txt`	*(destination file)*

Create the MySQL tables that you designed in Exercise 1a) above. Remember to make use of features such as; e.g.: ids, primary keys, auto incrementing, and foreign keys (as in the To-Do-List Web Service: http://www.itu.dk/people/brabrand/DSDS-12.pdf on slide number 10). Write a couple of sentences describing important aspects of your tables under the heading "Exercise 1b Table Definitions" in `description.txt`.

Exercise 1c): MySQL Data Insertion (4%):

`exam/1c_data_insertion.sql`	*(destination file)*

Create the table entries in your database using appropriate MySQL INSERT commands to the extent specified in the requirements specification so that your service can be properly used.

Exercise 2: Design of Database Transactions (20%)

`exam/2_database_transactions.sql`	*(destination file)*
`exam/description.txt`	*(destination file)*

Construct a set of SQL queries that you believe your PHP scripts will need to make use of in order for your service to provide the functionalities specified in the requirements specification earlier. This includes retrieval of selected data from your tables, deletion, modification, and insertion of new data into them. For each of the MySQL queries you present, write a sentence or two explaining what it actually does under the heading "Exercise 2 Database Transactions" in `description.txt`.

Exercise 3: Design of Sitemap and Web Structure (20%)

```
exam/3_sitemap.png/gif/jpg                    (destination file)
exam/description.txt                          (destination file)
```

Now the time has come to design the overall sitemap of the web service. What web pages and PHP scripts are needed and what variables need to be sent between them? Draw a diagram of your suggested structure of your web service that includes information about the name of each script/page/file and information about the passing of form variables from one script/page/file to another. Name the files and variables as you see fit as long as it all conforms to the constraints outlined in the requirement specification. Describe the general flow of interaction and the role of the different files in your web service in a couple of sentences under the heading "Exercise 3 Sitemap" in `description.txt`.

Exercise 4: Construction of HTML and PHP scripts (40%)

```
exam/*.html                                   (destination file)
exam/*.php                                    (destination file)
exam/description.txt                          (destination file)
```

Now you are finally asked to program the Web Service (i.e., construct the PHP and HTML files defined in Exercise 3 some of which will make use of the database transactions defined in Exercise 2). Keep in mind the code quality factors listed in the "Grading" section earlier in this exam. (You don't have to put include-files in a special directory; they can just be placed as `.php` files in the exam directory along with all the other files.) Any comments you might have about your program you can write in `description.txt` under "Exercise 4 HTML and PHP".

Appendix

Checklist

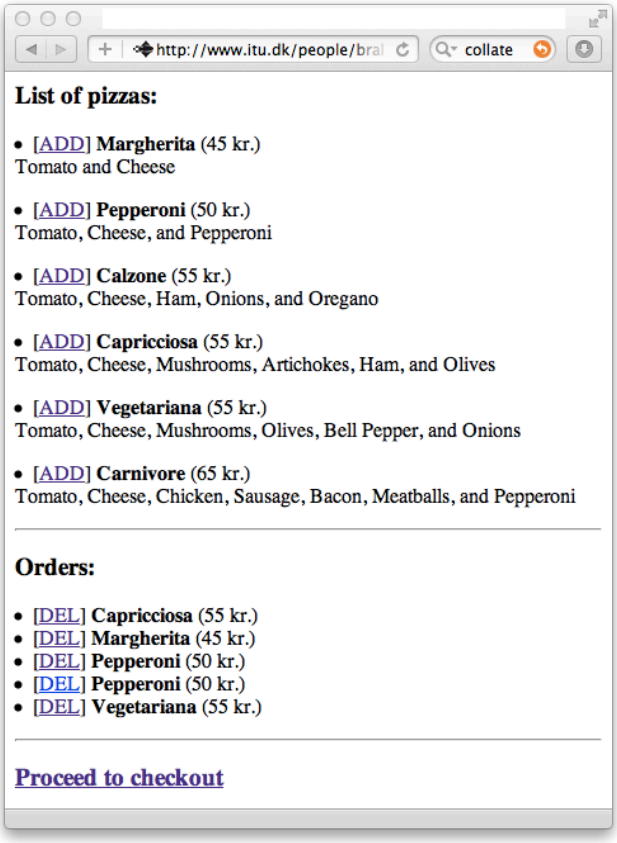
If you have done all four exercises of this exam, you should have the following files in your online folder (which you may not modify after the exam deadline) and on your CD-ROMs:

```
exam/1a_table_design.png/gif/jpg              (destination file)
exam/1b_table_definitions.sql                 (destination file)
exam/1c_data_insertion.sql                    (destination file)
exam/2_database_transactions.sql              (destination file)
exam/3_sitemap.png/gif/jpg                    (destination file)
exam/*.html                                   (destination file)
exam/*.php                                    (destination file)
exam/description.txt                          (destination file)
exam/username.html          (full URL link to your online Web Service)
```

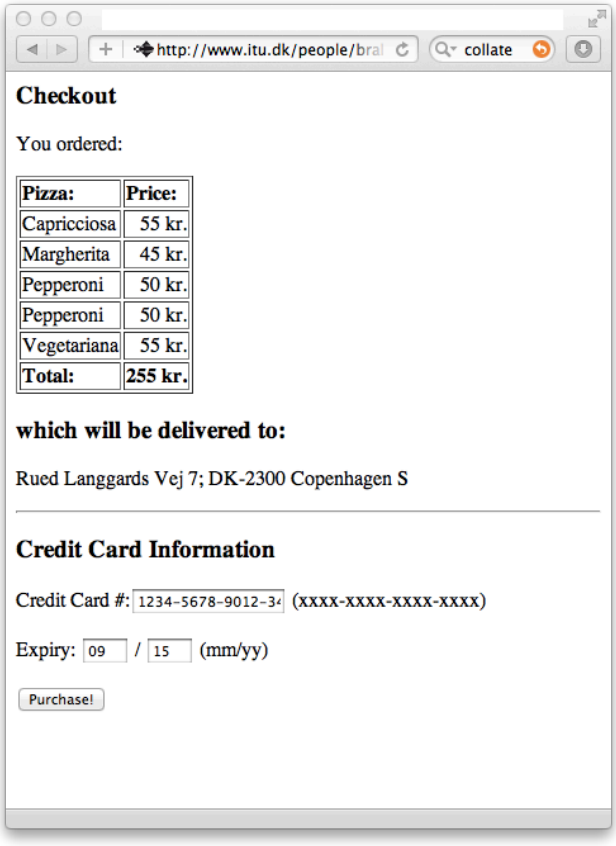
Pizzas

Here are a few common pizzas you may cut'n'paste, if you don't want to add your own:

Name	Ingredients	Price
Margherita	Tomato and Cheese	45
Pepperoni	Tomato, Cheese, and Pepperoni	50
Calzone	Tomato, Cheese, Ham, Onions, and Oregano	55
Capricciosa	Tomato, Cheese, Mushrooms, Artichokes, Ham, and Olives	55
Vegetariana	Tomato, Cheese, Mushrooms, Olives, Bell Pepper, and Onions	55
Carnivore	Tomato, Cheese, Chicken, Sausage, Bacon, Meatballs, and Pepperoni	65



Screen Shot 1: One possible design of a "*La Pizzeria*" Pizza Web Shop.
(Note that this is just one of many possibilities of how to meet the requirements.)



Screen Shot 2: one possible design of a "*La Pizzeria*" Pizza Web Shop.
(Note that this is just one of many possibilities of how to meet the requirements.)