Slide 1.13: The LAMP stack (cont.)
Slide 2.2: MySQL set-up
Home

PRINT VERSION

Programming Exercise I: A Simple Online Bookstore Using LAMP Technologies

(An Industry-Level, Second-to-None Comprehensive Specification)

Absolutely no copying others' works

Development Requirements

When start developing the exercise, follow the two requirements below:

- Have to use the LAMP technologies consisting of Linux, Apache, MySQL, and PHP/Perl/Python.
- The system entry page must be located at `http://people.aero.und.edu/~userid/457/1/` and all pages must be hosted by `http://people.aero.und.edu/~userid/` .

Soft Due Date† and Submission Methods

On or before Wednesday, February 22, 2017 in class and have completed the following tasks:

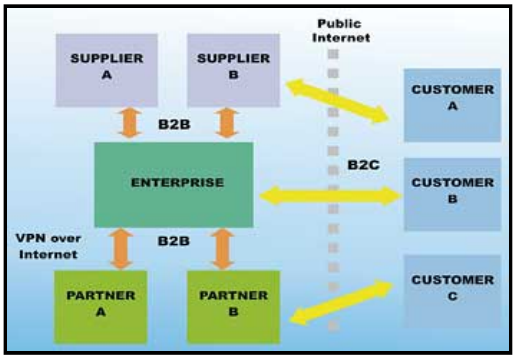
- Turn in
  - printouts of all source code (no documentation needed) including CSS, DTD, (X)HTML, JavaScript, Perl, PHP, Python, Unix shell, SQL (especially `create` commands), XML, and whatever languages‡, and
  - the password for displaying the source code online (only one password for all interfaces and all exercises). The code will be examined during the demonstration too.
- Send an email to the instructor at wenchen@cs.und.edu to set up an appointment to demonstrate your exercise to the instructor individually, so misunderstanding would be minimized. The mail lists the times you will be available. The instructor will prepare a set of test data to be used by all students.

†Since related topics may not be covered completely by the due date, no penalty will be applied if submitted after the due date. However, you may lag behind if you are not able to submit it by then.

‡Note that you are allowed to use any languages and tools for this exercise, but the exams will focus on LAMP and AJAX technologies unless otherwise specified.

Background

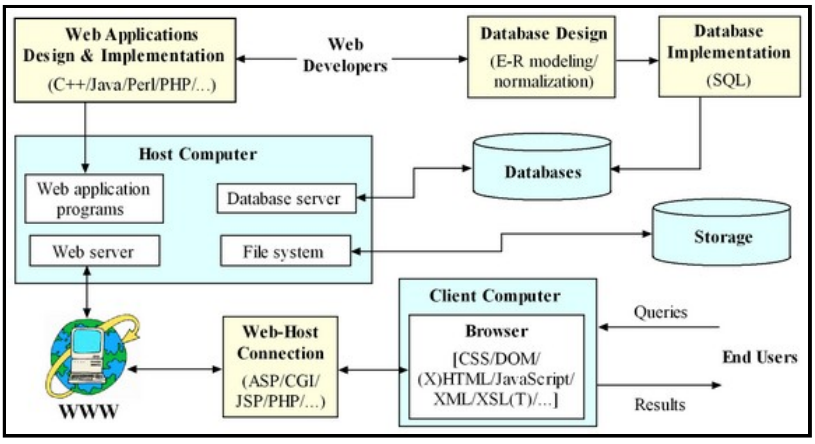
This exercise includes activities of business-to-business (B2B) and business-to-consumer (B2C). B2B describes commerce transactions between businesses, such as between a manufacturer and a wholesaler, or between a wholesaler and a retailer. XML (eXtensible Markup Language) defines a standard way of encoding the structure of information in plain text format. With XML, financial information can be exchanged over the Internet conveniently. Whereas B2C refers to transactions conducted directly between a company and consumers.



Objectives and Procedures

This exercise is to design and implement a simple online database-driven bookstore based on a B2B business model in addition to the B2C model. The bookstore has an agreement with the suppliers about the format of the book list in XML and together they come up with a DTD (Document Type Definition). The bookstore implements the system based on the DTD and uploads book lists from the suppliers to its system. The owners then manage their books via their store website. In addition, customers can purchase books via the website. This exercise has students use the LAMP technologies and learn how to integrate XML into their systems and databases. It includes the following features: XML data to MySQL database, data searching and ranking, data processing and management, etc. The online bookstore includes the following three kinds of users:

- the suppliers (manufacturers), who have agreed with the bookstore owners on the DTD in advance and supply book lists in XML according to the DTD to the bookstore,
- the bookstore owners (owners), who uploads the book lists into the database via the bookstore website and manages the books and customer accounts via the website, and
- the customers (users), who can purchase books and manage their own accounts via the bookstore website.



Exercise Requirements

The bookstore includes the following features:

- The data of a book includes

  - a unique ISBN (10 characters),
  - a unique title, and
  - a price.

- The data of a customer includes
  - a unique ID assigned by the system automatically after signing up,
  - a name,
  - the customer's account includes the following information:
    - the information of the all purchased books (one accumulative quantity for each book and no duplicate books) and
    - the total amount spent on the purchased books (one amount per customer).

- Each customer has his/her own shopping cart, which is available only after the customer signs in and the cart contents remain even after the customer signs out. The data of a shopping cart for each customer includes
  - the information of the all selected books (one quantity for each book and no duplicate books) and
  - the total amount of the all selected books (one amount per cart).

- **(System administrator: 30% total)**
  For the one and only one system administrator:

  - **(XML upload: 10%)** Upload a book list in XML to your database rather than files. This function will be performed at the very beginning of the demo/grading and will be used one-time only. The DTD books.dtd, which defines the legal building blocks of the book list such as books.xml, is given as follows:

books.dtd

```
<!ELEMENT booklist (book+)>
<!ELEMENT book     (ISBN, title, price)>
<!ELEMENT ISBN     (#PCDATA)>
<!ELEMENT title    (#PCDATA)>
<!ELEMENT price    (#PCDATA)>
```

An example of a book list in XML such as books.xml is given as follows:

books.xml

```
<?xml version="1.0"?>
<booklist>
  <book>
    <ISBN>0134291255</ISBN>
    <title>MySQL and PHP for the Web</title>
    <price>24.54</price>
  </book>
  <book>
    <ISBN>0596157134</ISBN>
    <title>Learning PHP, MySQL, and LAMP</title>
    <price>5.86</price>
  </book>
```

```
  <book>
    <ISBN>1449325572</ISBN>
    <title>LAMP: Linux, Apache, MySQL, and PHP</title>
    <price>22.16</price>
  </book>
  <book>
    <ISBN>020177061X</ISBN>
    <title>Web Development with LAMP</title>
    <price>14.40</price>
  </book>
  <book>
    <ISBN>0321833899</ISBN>
    <title>Linux, MySQL, and PHP Web Development</title>
    <price>17.27</price>
  </book>
</booklist>
```

The actual book list used for demo/grading will be different from the above XML and include more books. XML (eXtensible Markup Language) and DTD (Document Type Definition) will be covered when we discuss AJAX. The XML and DTD used here are the simplest and self-explanatory. For details, check the XML and DTD tutorials. The get node/attribute values of PHP SimpleXML can be used to facilitate the implementation of this step.

- **(Listing all books sorted: 05%)** List the data of all books sorted by title ascendingly, each including (i) an ISBN, (ii) a hyperlinked title, and (iii) a price.
- **(Showing a book's customers: 05%)** List (i) IDs and (ii) names of all customers having purchased the book by clicking a book's hyperlinked title.
- **(Listing all customers sorted: 05%)** List the data of all customers sorted by name ascendingly, each including (i) an ID and (ii) a hyperlinked name, and (iii) a total amount.
- **(Showing a customer's books: 05%)** List (i) ISBNs, (ii) titles, and (ii) quantities of the books purchased by the customer by clicking his/her hyperlinked name.

- **(Customers: 50% total)**
  For each customer:
  - **(Signing up, in, and out: 10%)** Customer can sign up, in, and out. The instructor will use multiple browsers or tabs to test your exercise at the same time.
  - **(Searching for books: 10%)** List a book if its title includes any of the case-insensitive keywords in a query, where the keywords are separated by spaces. If the query is empty, list all books. The search results are a list of books, each including (i) a checkbox (for putting the book in the shopping cart) and (ii) a hyperlinked title.
  - **[Ranking: 10% (graduate students only)]** The search results are sorted based on the length of LCS (Longest Common Subsequence). For example, assuming the query is

linux mysql php, the ranked search results for the above book list are:

1. LAMP: Linux, Apache, MySQL, and PHP (|LCS| = |Linux MySQL PHP| = 3)
1. Linux, MySQL, and PHP Web Development (|LCS| = |Linux MySQL PHP| = 3)
2. MySQL and PHP for the Web (|LCS| = |MySQL PHP| = 2)
3. Learning PHP, MySQL, and LAMP (|LCS| = |MySQL| = |PHP| = 1)

- **[Managing the shopping cart: 10% (undergraduate) and 05% (graduate)]** After searching, can put several books in the shopping cart at the same time. Each entry in the cart including two items: (i) a quantity to purchase (in an input form) and (ii) a hyperlinked title. If the book is in the cart before, no action is taken. If the quantity is zero, remove the book from the cart. Additionally, always show the up-to-date total amount of the books in the cart.

- **[Purchasing books: 10% (undergraduate) and 05% (graduate)]** Purchase all books in the shopping cart. Clear the shopping cart and update the customer's account accordingly after purchasing. Each book has only one entry in the account and its quantity is accumulative.

- **(Checking his/her own account: 05%)** List the data of a customer including (i) an ID, (ii) a name, (iii) hyperlinked titles and quantities of purchased books, and (iv) a total amount spent on the books.

- **(Showing the details of a book: 05%)** List the book's (i) ISBN, (ii) title, and (iii) price by clicking its hyperlinked title.

- **(Instructor's requirements: 20% total)**
  Other than the above system requirements, the instructor has the following requirements:

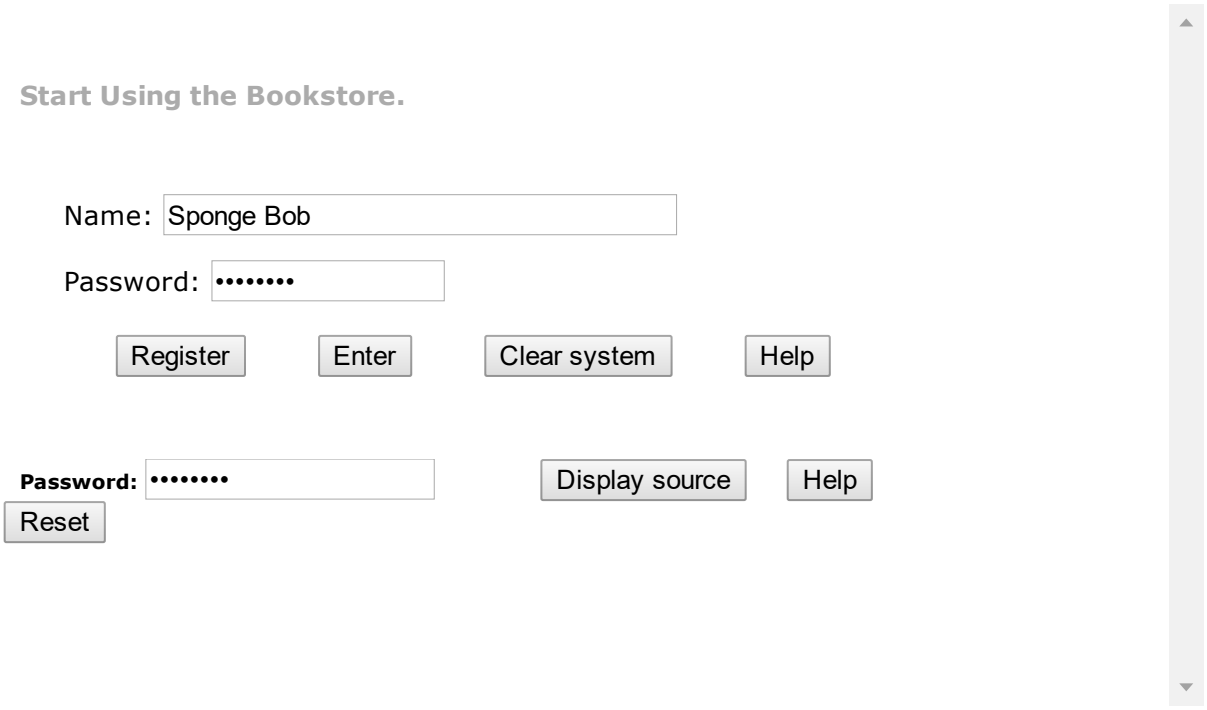
  - **(User friendliness: 10%)** User-friendliness will be heavily considered when grading. In the past, some exercises were awkward, which made the grading or browsing difficult. For example, it is considered not user-friendly if the system repeatedly asks users to enter their names/IDs/passwords.

  - **(System reset: 05%)** The system can be reset, which is to clear all data stored in the database and files, so the instructor can test the system by using only his own test data. That is the system has to include a button such as "Clear system" at the system entry page.

  - **(Plagiarism-proof: 05%)** It is for the instructor to find any plagiarism. Each interface includes a button "Display source," which is to list ALL the source code for implementing the functions of THIS interface. Only one password is for all interfaces and all exercises. The system will be highly suspected if fail to implement this button.

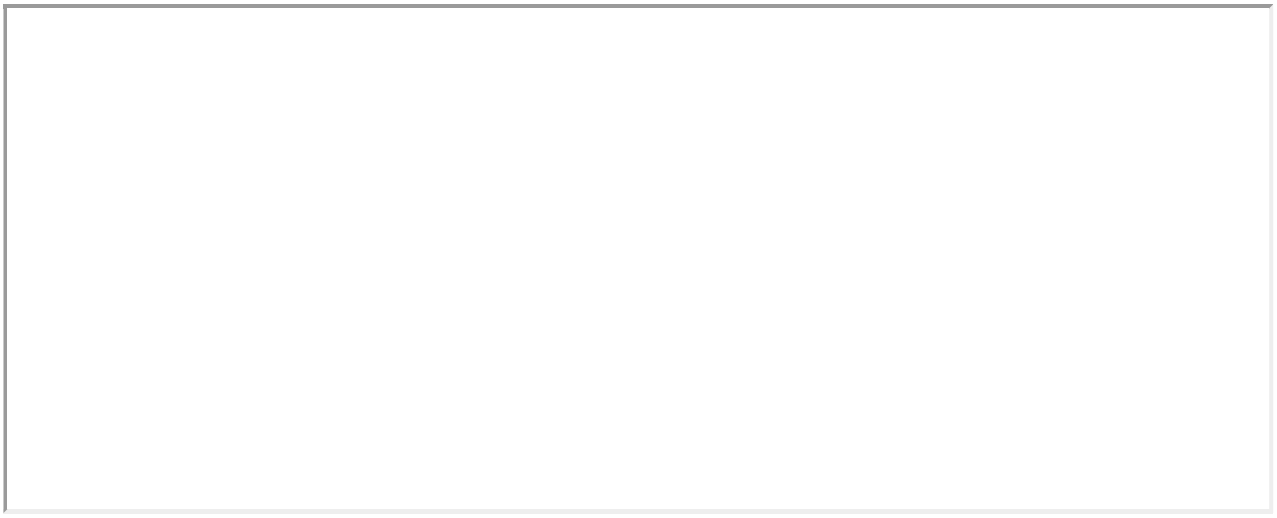
An Example of Web System Interfaces

Note that this example is not related to this exercise. It is only to show how to display web pages.



An Example of File Uploading

The following interface, `upload.html` in html or txt, is used to upload an XML file:



Most web languages allow file uploading. The above example uses the PHP script, upload.php, which requires a directory at "`/home/wenchen/public_html/course/457/exercise/1/upload/`" to store the uploaded files. Note that you may need to open the folder by using the following command:

```
chmod 777 upload/
```

so the Web can write the files to the folder.

/home/wenchen/public_html/course/457/exercise/1/upload.php

```
01 <html><body>
02   <?php
03     if ( $_POST['act'] == "Upload" ) {
04       header( "Content-type: text/html" );
05       if ( ( $_FILES[file][type] == "text/xml" ) &&
06            ( $_FILES[file][size] < 50000 ) ) {
07         if ( $_FILES[file][error] <= 0 ) {
08           echo "Upload: <em>" . $_FILES[file][name] . "</em>";
09           echo "Type: <em>" . $_FILES[file][type] . "</em>";
10           echo "Size: <em>" . ceil( $_FILES[file][size] / 1024 ) . " Kb</em>";
11           move_uploaded_file( $_FILES[file][tmp_name], "upload/books.xml" );
12           echo "Stored in: <em>upload/books.xml</em>";
13           chmod( "upload/books.xml", 0755 );
14         }
15         else {
16           echo "Error: " . $_FILES[file][error];
17           print_r( $_FILES[file] );
18         }
19       }
20       else {
21         echo "Invalid file";
22         print_r( $_FILES[file] );
23       }
24     }
25     elseif ( $_POST['act'] == "Check the upload" ) {
26       header( "Location: upload/books.xml" );
27     }
28
29     elseif ( $_POST['act'] == "Help" ) {
30       header( "Content-type: text/html" );
31       system( "cat help.html" );
32     }
33   ?>
34 </body></html>
```

Evaluations

The following features will be considered when grading:

- **Specification:**
  - The instructor (or your assumed client) has given the exercise specification as detailedly as possible. If you are confused about the specification, you should ask in advance. Study the specification very carefully. No excuses for misunderstanding or missing parts of the specification after grading.
  - The specification is not possible to cover every detail. You are free to implement the issues not mentioned in the specification, but the implementations should make sense. Implemented functions lacking of common sense may cause the instructor to grade your exercise mistakenly, and thus lower your grade.
  - The exercise must meet the specification. However, exercises with functions exceeding the specification will not receive extra credits.

- **Grading:**
  - This exercise will not be graded if the submission methods are not met. Students take full responsibility if the website is not working.
  - A set of test data will be used by all students. The grades are primarily based on the results of testing. Other factors such as performance, programming styles, algorithms, and data structures will be only considered minimally.
  - Before submitting the exercise, test it comprehensively. Absolutely no extra points will be given after grading.
  - The total weight of exercises is 36% of final grade and all three exercises have the same weight, 12% each.
  - Multiple browser tabs, browsers, or computers will be used to test the systems at the same time to make sure the multi-processing is working well.
  - If not specified, no error checking is required; i.e., you may assume the input is always correct for that case. For example, the price entered will always be a rational number.
  - Feel free to design your own interfaces; user-friendliness will be heavily considered; each function/button will be tested extensively; and from the source code submitted, the database design and programs will be examined.
  - Firefox 50.1 or above browser will be used to grade exercises. Note that Internet Explorer, Chrome, and Firefox are not compatible. That is your exercises may work on the IE or Chrome but not Firefox.
  - The systems have to be active and the exercise printouts will be kept until the end of this semester. They will be re-checked for plagiarism from time to time. The instructor will inform you the exercise evaluations by emails after grading.

- **Databases:**
  - A database has to be used and try to perform the tasks by using SQL as much as possible because SQL, a non-procedural language, can save you a great deal of programming effort.

- The SQL DDL commands such as "`create table`" have to be submitted, where SQL is Structured Query Language and DDL is Data Definition Language.
- From the source code submitted, the database design and programs will be examined. Poor database design or uses will result in a lower grade.
- **(-05%)** if the database design is NOT optimal.
- **(-05%)** if the SQL `create` commands of database implementation are NOT submitted.
- There are many advantages of using databases. If database is not used, the problems caused by not-using-transaction must be considered. For example, if two customers are enrolled at the same time, an ID may be assigned to different customers if databases are not used.

- **Comments:**
  - Make the exercise work first. Do not include extra features, such as user passwords, in the beginning. By the way, you will not receive credits for the extra features.
  - One way to build a complex web system from scratch is to design the user interfaces first and then implement the system button by button. By doing this way, it could simplify the construction. The recommended construction steps are

    i. Database design (E-R modeling or normalization),
    ii. Database implementation (SQL),
    iii. Interface building (HTML and CSS), and
    iv. Button-by-button implementation (PHP).

  - Many times, simplicity is the same as user-friendliness.
  - Security concerns are mainly ignored here. For example, a customer may be able to check others' account information if protection is not enforced. Small or medium -size businesses do not usually implement their own secure payment schemes. Instead they use a third-party payment system such as PayPal or purchase software from company like Cayan and integrate it with their websites.
  - The function of automatically sending emails is important for e-commerce systems, but will not be used here since companies complained our students sending out numerous mails because of faulty programs.

Slide 1.13: The LAMP stack (cont.)
Slide 2.2: MySQL set-up
Home