Computer Science E-75 Lecture 3: SQL

- **RSS** - Really Simple Syndication: Information in a standardized XML file.
  - Course forum's RSS link
  - "item" elements have standardized elements and attributes
  - Demo (demo.php)
    - Want to have all headlines from the course forum RSS feed
      - Getting an XML file from the web: file_get_contents() supports reading in from a URL e.g., `file_get_contents("http:/ / cs75.net/phorum/feed.php?0,replies=1,type=rss");`
    - Want to list each item element (`foreach ($xml->channel->item as $item) { echo $item->link; echo $item->title; })`
  - RSS spec: http:/ /cyber.law.harvard.edu/rss/rss.html
  - Description has CDATA section - `<![CDATA[ ... ]]>`
    - Parsed character data - Parsed elements, attributes, etc.
    - Character data - XML parser should simply read it in verbatim. (Prevents, say, "Is 2 > 3?" from upsetting the parser.)
  - The course website's main page (or specifically, a component of it, posts.php) loads all the latest posts
    - Gets the current time (we care about the publication data (pubDate) and only want to display announcements that have been posted in the last few days)
    - If the announcement was submitted less than 5 days ago, add it to the $announcements array.
    - `strtotime()` converts strings to a PHP date/time. We can use this function to convert the values of the pubDate elements in the RSS feed into PHP date/times.
    - Wanted to remove the parenthetical expressions at the end of the titles that was present in the RSS feed
      - Regular expressions
        - `preg_replace()`
        - Expression: `/\(.*\)$/`
          - We have to escape the parentheses with a backspace because they have special meaning in Perl regular expressions.
          - The "$" matches the end of the string.
          - What if there's a left parenthesis in the title - greedy vs. non-greedy
- **Authentication and cookies**
  - Cookie name for sessions by default is: PHPSESSID
  - Lifetime: Only as long as the browser window is open
  - What if I still want to be logged in when I reboot, or reopen the browser?
  - => We need to remember user details
    - Remembering usernames in cookies is fine
    - But remembering passwords in cookies? Since the passwords are stored on the user's computer, anyone who has access to the computer can look at the password.
    - How can we store a password securely?
      - We can encrypt (using a bidirectional algorithm) the password, and store it on the user's computer.
        - However, anybody can come along, retrieve the cookie, and perform a brute-force attack on it and uncover the original password.
      - Set a cookie saying "authenticated=true" on the user's browser, and when you receive that cookie, allow the user access
        - Cookies can be spoofed easily using a text editor.
      - Extend the lifetime of the PHPSESSID (which is a pseudorandom string)
        - Since PHP session data is stored in /tmp (a temporary directory), you might not want to store that many files there, since sessions can become stale, etc.
      - People can spoof cookie IDs, even though that may take a lot of times.
        - Disable logins for people who try to login several times (those are probably programs, not people)
        - Store the IP address of the user (But IP addresses can change)
- **login1 - login9: Login modules**
  - home.php checks whether you are logged in or not. If so, show them some information. If not, show them other information.
  - The course website login - When the user authenticated, stores the username in the session, to signify that the user has logged in.
  - login1.php
    - Calls session_start(): Starts a session. It also sends a cookie to the browser with the PHPSESSID.
      - Since this is sending the cookie as a header ("Cookie:"), if you have any output (including whitespace, or error messages) above the function, you may get the "Headers have already been sent" error.
      - This problem also occurs if you are including or requiring a file, and the file has whitespace anywhere outside of the PHP tags.
    - This demo simply defines as a constant one valid set of username and password.
    - If the user field and pass field are both set, then the implication is that the user submitted the form (as opposed to simply viewing the form for the first time).
    - This demo simply sets an "authenticated" boolean inside the session variable to TRUE.
    - Then redirects user to the home page.
    - What it's missing: If the username or password is wrong, it simply presents a blank form.

Computer Science E-75 Lecture 3: SQL

- login2.php
  - Adds a "value" attribute to the input that reads in the value of `$_POST['user']`.
  - We can also display an "Invalid username or password" error by simply counting the number of $_POST variables, and if there are any $_POST variables, assume that the form was submitted but the authentication failed and show an error message.
- login3.php
  - Stores a username in the cookie so that the user doesn't need to type in their username every time.
  - Store a separate cookie on the user's computer using setcookie().
  - `setcookie("user", $_POST["user"], time() + 7 * 24 * 60 * 60)` - sets a cookie named "user" with the value of $_POST["user"] for seven days on the user's computer.
  - When the page is loaded again, the username field is prepopulated with the previously logged in user.
- login4.php
  - Has a "Keep my logged in until I click log out" checkbox
  - Sets both the username *and password* in a cookie, is the checkbox was checked.
    - Anyone with access to the browser/computer can see the password in plain text!
    - Instead, we can store a pseudorandom number that corresponds to a user's password and compare it against a database of usernames vs. pseudorandom numbers.

- **Databases**
  - Similar to Microsoft Excel's tables (rows and columns) (Columns may represent username, phone number, zip code, etc. and Rows may represent users)
  - Traditionally, the MySQL command line prompt was the only way to access SQL databases
  - phpMyAdmin - a graphical interface that we can use to access MySQL databases
  - panel.cs75.net
    - MySQL Management - Can create new, empty databases through this interface.
  - Connecting to MySQL through PHP
    - `mysql_connect("localhost", "malan_test", "password")` -- creates a connection to the MySQL server
    - `mysql_select_db("malan_test")` -- since you can have multiple databases on a single server, this "selects" a specific database for our use.
    - SELECT statement ("`SELECT * FROM <the name of the table> WHERE <a column name> = '<some value>'`") retrieves a certain subset of a table according to our requirements.
    - `mysql_real_escape_string($somevariable)` -- an input validation function to stop users from issuing arbitrary SQL queries in input (preventing SQL injection attacks -- this will be covered later).
    - `mysql_query($sql)`, where $sql is a SQL query, such as a SELECT, returns false when something has gone wrong -- for example, if the database connection dies. If not, it returns a result set (if the SQL query was a SELECT).
    - `mysql_num_rows($result)` (where $result is the result set, similar to SimpleXMLElement's return values) returns the number of rows in a MySQL result set.
    - `mysql_fetch_assoc($result)` returns the next row in a result set.
  - Looking at a table in phpMyAdmin
    - Column types: VARCHAR (takes a number; for example, `VARCHAR(255)` defines a column that is maximum 255 characters long)
    - "not null" means that the column may never be empty.
    - Database keys - a column that should be used to look up data in the database. Keys tell MySQL to optimize queries based on that key.
      - *Primary keys* must be unique. We don't want to say passwords are primary keys because two different people wouldn't be able to have the same password. But we want the username to be a primary key because we don't want two people with the same username.
    - Inserting a row into a table: `INSERT INTO <database name>.<table name> ( <column1>, <column2> ) VALUES ( <column1_data>, <column2_data> );` (The database name is not required if you have already selected a database to use.)
    - You can easily edit column names, etc. using the phpMyAdmin interface (the pencil mark means edit; the red cross is delete)
  - Security implications
    - The password is stored in the MySQL database in plaintext -- anybody with access to the database can retrieve all passwords
      - Solution: Don't store the password in plain text.
- login7.php
  - Instead of retrieving the password for a user and checking for equality in the PHP script, why don't we pass both the username and password to the MySQL server and let it compare the values?
  - "`SELECT 1 FROM users WHERE user=<username> AND pass=PASSWORD(<PASSWORD>)`" -- returns one row with one column with "1" if the username and password is correct. The `PASSWORD()` function is a function in MySQL that creates a one-way hash (takes a string and returns a near-random string based on the original string).
    - We store the password in the first place using the same `PASSWORD()` function; we never keep around a plain text version of the password. On the login form, we hash the user's password input, and check it against the hashed value stored in the database.
- login8.php

Computer Science E-75 Lecture 3: SQL

- We're using a better function caled `AES_ENCRYPT()`. It takes two arguments, the string to encrypt and a key (not to be confused with database keys).
- If we use the same key though, if someone steals our database *and* our code, then they will be able to have access to all our accounts. So never use a constant key, but use something variable (for example, the username concatenated with the password).