Lecture 4: Session Tracking

- Cookies allow us to maintain state, but are somewhat clumsy to program
- To keep detailed state information we probably need many cookies and we must store a lot of information within them
    - Each cookie is only 4K and Value field is simple
- Cookies are good for keeping track of return visitors
- For keeping state within a "current" visit, there are better ways
    - PHP allows session tracking which can simplify and streamline the process of maintaining state

Lecture 4: Session Tracking

- Idea:
- When user first logs into (or simply visits) a site, a session is started and a unique, random ID is assigned to that user
- ID is stored in a cookie (on client) or on the URL, but state information (session variables) is stored on the server
- Any accesses by the same client with the same session ID are recognized and the session variables can be retrieved and used
    - From any .php script – multiple scripts can be used in the same session

Lecture 4: Session Tracking

- In other words, the session variables are a pool of semi-permanent data stored on the server
    - A separate pool is associated with each client
    - Through the session ids the pools can be distinguished and accessed appropriately
    - Arbitrary information can be stored for each client
- When session is finished (client logs out or browser is closed) the session variables are cleared and the session ID is disposed of

- Syntax
- Session tracking can be automatically turned on (with a server setting)
- If not the programmer must explicitly start a session in each script using session_start()
    - This should be done at the beginning of the script, prior to any regular html tags
    - It must be done in any script in which the session variables are to be accessed
    - See ex5.php to note that session tracking is NOT automatically on in our server

Lecture 4: Session Tracking

- During a session, session variables are accessed by scripts through the $_SESSION array
    - – Arbitrary values can be stored there
- Implementation
- By default PHP uses cookies to implement sessions
    - However, they are used behind the scenes, so programmer does not have to deal with the particulars
    - Session ID is embedded within a cookie
- Can also insert the session ID into the URL if you prefer (ex: client doesn’t accept cookies)

Lecture 4: Session Tracking

- Issues:
- Session tracking in itself is not a secure process
    - Session id is the key to obtaining the information, so it must be protected
    - If we use a secure server (using SSL) we ensure that the ids are not sent as plain text
- For more information:
    - See: http://www.php.net/manual/en/intro.session.php
- For example of using session tracking and cookies, see
    - ex13.php for simple example
    - usesession.php for a bit more complex handout

Lecture 4: OOP PHP

- PHP is an object-oriented language
    - See: http://us.php.net/manual/en/language.oop5.php
- Has classes + objects
- Has inheritance and method overriding
    - However, the dynamic typing of PHP variables does not give it quite the same type of polymorphism as Java
        - The reference type always matches the object type
- Object syntax is more like C++ than Java
    - Uses the scope resolution operator for parent class access
    - Uses the "arrow" operator for field / method access

Lecture 4: OOP PHP

- PHP objects can have instance variables and instance methods
  - Like Java (more or less) we can restrict visibility by using
    - private
      - > Only visible within class of variable's declaration
    - protected
      - > Visible within class of variable's declaration, plus any subclasses
    - public
      - > Visible anywhere
  - Unlike Java we do not have implicit access to instance variables from within objects
    - To access we must use "this" for explicit access

```
class Foo {
  private $x;
  public function setX($data) {
    $this->x = $data;
  }
  public function getX() {
    return $this->x;
  }
  ...
}
```

- See what happens if you just use $x

Lecture 4: OOP PHP

- PHP also has a lot of functions to help with OOP
    - Some are particularly useful for the Web environment in which PHP is used
    - Ex: __autoload()
        - Can automatically include class files for any classes used in a PHP script
            - > We don’t have to explicitly include each file
            - > We don’t have to worry about including a file multiple times
        - Note the name: prefixed with two underscores
            - > There are several useful functions with this notation
            - > Ex: __construct(), __destruct(), __toString()

- These are called "magic methods"
  - Mostly because they are called implicitly in some way or another
  - PHP programmer may define the method bodies but does not explicitly call them
  - For more information see:
    - > http://php.net/manual/en/language.oop5.magic.php
  - See ex14.php, Foo.php and SubFoo.php
  - Ex: serialize(), unserialize()
    - Allow serialization and deserialization of PHP objects
      - > This is good if we want to save an object into a file or a cookie and then later restore it
      - > See usesession-oop.php and User.php

Lecture 4: OOP PHP

- PHP OOP definitely has differences from Java OOP
    - However, there is extensive documentation on it so avail yourselves of it
- Ex: Interfaces and Polymorphism
    - Since PHP variables are dynamically typed, we never have to cast objects to store them
    - See ex15.php and class files
- Why use it (or when to use it)?
    - When scripts get larger / more complex
    - To interact with some predefined resources
        - Ex: a MySQL database

Lecture 4: Sorting Instability

- As we mentioned previously, the default sort() method in PHP is unstable
- This does not really matter when sorting simple types
- However, when sorting complex types such as objects, we can have issues:
    - Original data is in order on Field A
    - We sort the data on Field B
    - Objects which are equal on Field B, may not have the original order based on Field A
- To obtain stability we will have to write our own sort method

Lecture 4: Sorting Stability

- Or, more likely, use code that someone else has written!
- (This is different than just copying and pasting source code)
- See unstable.php (in the OOP dir, as the example is written in an OO fashion)

Lecture 4: Flat Files vs. DB Files

- So far, our PHP examples have used regular text files
    - Often called FLAT FILES
- These have a certain advantage, since we can edit the files easily and can read them without any special software
- However, they have many disadvantages as well
    - It is difficult to "update" the data in a file without rewriting the entire file
        - How to change data in the middle of the file?

Lecture 4: Flat Files vs. DB Files

- Concurrent access of the file is tricky
  - We use FLOCK to lock out the file, but even that only works when used consistently
  - We also often FLOCK a file for a long period of time to prevent corruption – limiting access to the file for that time
    - Even if we really need to lock only part of the file
- Access can be slow, especially if the data is large
- Access privileges must be implemented by the programmer

Lecture 4: Flat Files vs. DB Files

- An alternative is to use a DATABASE to store our data
  - Most common databases now are relational databases
    - We have data stored in tables and relate the data from one table to that of another
  - Access is faster than flat files
  - Queries to obtain specific sets of data can be done using a well-defined query language
  - User has random access to data
  - Concurrent access handling is built in
  - Access privileges are built-in

Lecture 4: Database Definitions

- Some definitions / notions we will be using
- Database
    - The overall collection of data – may consist of many tables
- Table
    - An individual "relation" in the relational database
        - Relates keys to values
    - Table Column
        - An attribute in the table
    - Table Row
        - An entity in the table
        - Typically has a value for each column

Lecture 4: Database Definitions

- Key
    - An attribute that uniquely identifies an entity
        - Ex: SSN for a student at Pitt
- Foreign Key
    - Key used to relate data in one table with data in another table
        - Ex: PSID may be key to a student table
        - May also be a foreign key in a table for a given course
- Schema
    - A set of table designs that determine a database
    - Does not yet include the data – simply shows how it will be structured in the database

Lecture 4: Database Definitions

- Relationships -- how do data in different tables relate?
  - One to one
    - An entity in a table corresponds to a single entity in another table
    - The relationship is typically established using a foreign key for one or both entities
      - Ex: If we have a table for Student_Info and a table for Academic_History, there is a one-to-one relationship between them
  - One to many
    - An entity in a table corresponds to 1 or more entities in another table

Lecture 4: Database Definitions

  - Ex: If the table for Academic_History has an entry for each term, the relationship now becomes one student to many terms
- Many to many
- Multiple entities in one table correspond to multiple entities in another table
- This relationship is often defined by a separate table, which in fact changes it into 2 one to many relationships
- Ex: Tables Student_Info and Courses_Taken have a many to many relationship, since a student can take many courses and each course can be taken by many students
- However, if we create a new table Student_Courses, we can have each entity be a pair: Student_Id, Course_id
- Now Student_Info has a one to many relationship with Student_Courses, and so does Courses_Taken

Lecture 4: Getting Started with MySQL

- First we need access
- Typically this is done by the site administrator
- Since you are using your own servers, you are the administrators!
- This can be done in several ways:
    - XAMPP comes with phpMyAdmin
        - This is accessed via the Web Server and allows you to access / modify the DB through a Web interface
    - XAMPP also comes with a mysql program
        - This allows command line access to the MySQL server
    - You can use either of these to initialize your DB

Lecture 4: Getting Started with MySQL

  - The mysql program also allows you to run batch commands using the “source” command
    - Be careful if you do this!
- Note: The root MySQL account in XAMPP initially has no password
  - If others will have access to your site, you need to add one!
  - There are other security holes in XAMPP that you should address if it will be shared
  - To address / fix these, go to the XAMPP home page on your system:

    http://localhost/
    - Click on "English" and then click on the Security link for more information

Lecture 4: Getting Started with MySQL

- There is a lot of info available for MySQL
- For MySQL home page, see: http://www.mysql.com
- For docs, see: http://dev.mysql.com/doc/refman/5.0/en/
    - There are many versions in the doc site – depends on your version of MySQL
- Googling various issues you may have is also helpful

Lecture 4: Setting up a DB

- As mentioned before, first we should develop a schema for our database
- Once that is done satisfactorily, we need to set up our tables and initialize the DB
- Can be done fairly simply using a script in conjunction with one or more flat files
    - The nice thing here is that the flat files are needed only for initialization
    - Once the database is set up, we use it exclusively

Lecture 4: PHP MySQL Basics

- To set up our DB, we need to issue some MySQL commands
    - Allow us to create tables and add data to them
    - Some valid MySQL commands are shown in Chapter 13 of the Sebesta text
        - Ex: create table, insert into, select
- We will use these MySQL commands through PHP scripts
    - In PHP, there are several ways of accessing a MySQL DB
    - Older installations used procedural access with the mysql_xxx() suite of functions

Lecture 4: PHP MySQL Basics

- Newer versions of PHP allow object-oriented access using mysqli
  - This is how we will be accessing our DB
  - Note:
    - Text uses mysqli commands but via the older procedural access
    - This works but oo access is nicer so we will be accessing it that way
  - We will initially create a mysqli object which connects us to the server
  - Then we access the db via methods in the object

Lecture 4: Setting up a DB

- Thus it is important to know both
  - The correct MySQL syntax to manipulate / access the database
    - More info in Chapter 13 of Sebesta and http://dev.mysql.com/doc/refman/5.0/en/index.html
  - The correct PHP syntax to connect to and query using its MySQL objects and functions
    - More info in Section 13.5 and
    - http://www.php.net/manual/en/mysql.php
      - > Overview
    - http://www.php.net/manual/en/book.mysqli.php
      - > Object-oriented access using mysqli

Lecture 4: MySQL Commands to Set Up a DB

- Some example commands
- DROP TABLE
  CREATE TABLE
    - Allows us to delete a table or create a new table in our DB schema
    - We can specify the name and properties of each of the fields (columns) within the new table
- INSERT INTO
    - Allows us to insert entries (rows) into a table
    - Column values are provided positionally based on the order established when the table was created

Lecture 4: PHP MySQL Commands

- First we must connect to the DB (via PHP)
- We do this by creating a new mysqli object:

```
$db = new mysqli(<server>,<user>,<pass>,<db>);
```

  - Where <server> is the mysql server we are connecting to
  - <user> is the account on the server
  - <pass> is the password for the account
  - <db> is the database that we want to access

- Since our Webserver and PHP server are on the same host, we can use 'localhost' (or 127.0.0.1) for the host
- For simplicity, in our accounts the <user> and <db> values will always be the same
  - Not required though

Lecture 4: PHP MySQL Commands

- Once we have connected to our database and created our mysqli object we can operate on it
- There are many methods available in the mysqli class
- The method we will use the most is query()
    - This will pass an arbitrary query string to the database
    - The return value depends on the nature of the query
    - If the query fails
        - the boolean false is returned

Lecture 4: PHP MySQL Commands

- If the query succeeds
- Commands such as INSERT, UPDATE, CREATE, etc will return true to indicate that they succeeded
    - Ex: The new row was added to the table
    - Ex: The current row was modified as specified
- Commands such as SELECT and SHOW will return a mysqli_result object
    - Idea in this case is that there is data that is returned from the query
    - The mysqli_result object gives us access to this data through its methods
    - We will look more at this class and its methods soon when we look at some queries

Lecture 4: PHP Commands to Set Up a DB

- For example, let’s create a new table called “CD” (what the heck is that?) with a field for an id number, a title and an artist
- The id number will be the **primary key**
    - Must be unique for each row / entry
- We will then insert a few CDs into our table
- See setCDs.php
- Then see carsdata.html
    - Don’t let the name fool you – you can do an arbitrary query from this web form