Processing Forms with PHP

Using PHP with forms is fairly simple

- When forms are submitted the server executes the php script, returning the resulting html
  - Remember that some of the file is unchanged, since it may not have an embedded php script within it
  - Server can be set so that the form variables can be accessed directly by simply using the $ sign
  - However, it is better to access the variables from the $_POST array (or the $_GET array)
    - The form element name is the key into the array
    - Discuss and see getpost.php

Processing Forms with PHP

- We can also use PHP to create forms
  - However, it is really just HTML that we are using
  - We can "interleave" the PHP and html to get the desired overall result, or we can have PHP output the appropriate HTML tags
  - So if you don't know it yet – learn some HTML
    - See Chapter 2 in Sebesta
- See ex12.php, ex12b.php – note many comments!
  - Note how the script interacts with the data file
    - It will show as many rows in the table as there are lines in the file
    - Note how the PHP and html are interleaved

Maintaining State

- HTTP is a stateless protocol
  - It is simply defines how clients and servers communicate with each other over the Web
  - Yet with many Web applications, maintaining state is important
    - Ex: When a customer logs into a site such as Amazon, he/she may go through multiple pages
      - We may want to keep track of the user (authentication information)
      - We may want to keep track of what the user has been doing

Maintaining State

- State can be maintained in various ways and in various places
  - Ex: We can store information on the server or on the client
  - We will examine several of these throughout the rest of the term
- One way of maintaining state is via **Cookies**
  - http://en.wikipedia.org/wiki/HTTP_cookie

Cookies

- Cookies – what are they?
  - Small pieces of information (up to 4K) initially sent by the server to the client and stored on the **client machine**
  - When client next connects to a server, it sends cookies from that server back to it
  - Information about the client can then be extracted by the server
    - If no cookie, server can create a new cookie for the client and send it with the response
    - However, browsers can disable cookies
      - Can cause problems if server is dependent upon them

Cookies

- Cookie format:
  - **Name:** name of the cookie – typically used to extract / examine the cookie
  - **Value:** contents of the cookie – seems like a simple value but can be an array if generated correctly
  - **Domain:** domain of the server that is to receive the cookie – actual domain of server must match domain stored in the cookie
    - Idea is that other servers cannot look at all of your cookies to see what you have
    - If not explicitly set in the cookie, it is the full domain of the server that created the cookie

Cookies

- **Expires:** When cookie will expire
  - Timestamp: Very specific format is required, but we can use function calls to make it easier
- **Path:** Path in server from which cookie can be sent
  - If not specified it is the full path from where cookie was set
- **Secure:** Does cookie require secure server using https
    - Default is no

Sending Cookies to Client

- Cookies are sent with the HTTP header of an html file:
  - Set-Cookie: oreo=Count Chocula;
    domain=.chocolate.com;
    path=/cgi/bin;
    expires=Thu, 08-Jun-2014, 16:15:00 GMT;
    - Must be set PRIOR to any html tags (since it is sent with the header)
  - If not sent with HTTP header will not be interpreted as a cookie
  - If client does not accept cookies it will just discard them
      - We can send a cookie and test to see if client accepts cookies

Cookies in PHP

- Cookies in PHP are fairly easy to use:
  - setcookie() function is called to create a cookie that will be sent to the client
    - See http://php.net/manual/en/function.setcookie.php
    - As always with cookies, they must be sent with the http header
      - Thus, you should determine and set any cookies in PHP mode prior to using any html (or even simple text)
  - $_COOKIE array contains the cookies received back from the client machine
    - Cookies sent to client by server previously
    - Associative array allows access of cookies by name

Cookies in PHP

- We will look at an example soon
- Thus, to maintain state a server can:
  - Send the client a cookie the first time the client connects to the server
  - Receive and update / modify the cookie as client navigates the site
    - Or send additional cookies
  - Use the presence and / or value of cookies to discern information about the client
    - Ex: A repeat customer – time of last visit
    - Ex: A current customer – last request or last page visited

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

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

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

Session Tracking

- 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

Session Tracking

- During a session, session variables are accessed by scripts through the $_SESSION array
  - Arbitrary values can be stored there
- Implementation
  - Be 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)

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

Mail

- Many web apps require mail to be sent
  - PHP has a built-in mail function:

  mail ($receiver, $subject, $message, $extras)
  - All arguments are strings
    - $extras allows additional information to be passed
      - Ex: From, Cc, Bcc
    - See mail() in the PHP manual
  - This should work in a production server
  - However, it relies on the web server having the ability to actually send the mail
    - This may not be the case for your XAMPP servers

Mail

- There is an alternative that we can use, which will utilize a mail server that you already have access to
  - Ex: Virginia Mail or GMail
- However, it is not standard PHP but rather an add on
  - It is called PHPMailer and is fairly widely used
  - To download it see:
    - https://github.com/Synchro/PHPMailer
  - You will need to install this onto your system
    - Demonstrating this will be Weekly In-class Exercise 3 (Thursday, Jan.29)
  - See mail.php and sendmail.php