Advanced PHP Forms

Data Validation

and

Making forms “sticky”

What is data validation?

- Data validation is the process of checking that user or system entered data is valid to the problem that being solved or the business process being satisfied
- Can be achieved both client-side or server-side
  - Client-side (using something like JavaScript) has the advantage of taking some of the workload off of the server (important on high traffic sites) but requires web developers to compensate for browser issues
  - Server-side (PHP is one solution) though uses server resources (processing and memory) has the advantage of creating simple HTML output that does not worry about browser type/version

Why worry about valid data?

- Garbage in – garbage out
  - Invalid data is worse than no data
  - Takes up space and resources but usually does not satisfy any business function
- Assume users are “foolish”
  - Though the majority of users when presented with an input text box labelled “Age:” would enter “23”, but the one in a thousand person that enters “twenty-three” is out there
  - You as a web developer have to compensate for that user.

What are sticky forms?

- Sticky form is a form that saves data that has been entered into a form, so a user does not have to re-type out the information if there is an error.
- It is achieved by using a technology that can handle decision making (like PHP).
- HTML cannot perform this task, though the inputted data is placed (i.e. echo'ed) into the HTML elements *value* attribute
- Utilizes the method in which the page loads as part of the process

Why use sticky forms?

- Have you ever tried to register for a site that requires multiple pieces of information and mistyped in one field? Having to re-enter all the information again is a good way of losing potential users on your site.
- Though, most technologies will allow you to use “Back” button on your browser that is a horrible solution to the problem (and in this course it is unacceptable)
- Instead, give a specific message to the problem, and leave the correct/valid data on the page.

PHP Provide Tools

- PHP comes pre-loaded with thousands of pre-defined functions (~3000)
  - Means that web developers do not have to code/create this functionality themselves
- Additionally the language provides several pre-defined arrays that capture data automatically
- To check out the scope check out the PHP manual at: http://php.net/manual/en/index.php

Examples of Some Useful PHP functions

- In this course we will use a very small portion of the functions available
- These include:

`trim()`

  - This function returns a string with whitespace stripped from its beginning and end

`isset()`

  - Determine if a variable is set and is not **NULL**

`is_numeric()`

  - Finds whether the given variable is numeric. Numeric strings consist of optional sign, any number of digits, optional decimal part and optional exponential part

Some Useful PHP Arrays

- In addition to functions, PHP also provided some useful arrays:
  - `$_SERVER[ ]` server and execution environment information
  - `$_GET[ ]` contains form information that was submitted using the “GET” method
  - `$_POST[ ]` contains form information that was submitted using the “POST” method
  - `$_FILES[ ]` contains HTTP File Upload variables used for
  - `$_COOKIE[ ]` used to access locally stored website info
  - `$_SESSION[ ]` used to maintain virtual states between client and server while a user is navigating a site

$_SERVER[ ] Array Elements

- The $_SERVER[ ] array contains several elements (approx. 35), the ones most useful for this course are:

  `$_SERVER['REQUEST_METHOD' ]`

  - Which request method was used to access the page; i.e. '*GET*', '*POST*', '*PUT*'.

  `$_SERVER['PHP_SELF' ]`

  - stores the filename of the currently executing script

  `$_SERVER['REQUEST_URI' ]`

  - stores the URI (*Uniform Resource Identifier*) which was given in order to access the current page

  NOTE: For this course, the last two will work the same, echo'ing the element in the *action* attribute in a form will make the page self-referring

Self-referring/Data validation Example

- Create a new text file named *lab6lecture.php*, add into this the contents of the file found at:

http://opentech2.durhamcollege.org/pufferd/webd2201/lab6_lecture.txt

- If you save this new file into your working (where your *header.php* and *footer.php* files exist) directory, and change its file extension to **.php*, it should work.