

PHP Form Handling

Prof. Jim Whitehead

CMPS 183 – Spring 2006

May 3, 2006

Importance

- A web application receives input from the user via form input
- Handling form input is the cornerstone of a successful web application – everything else builds on it

Overview

- The browser interprets the HTML source for a particular page
 - Result is a combination of text, images, and entry fields
 - Each entry field has a specific name
- User fills in these fields, (with potentially some client-side input checking via JavaScript) and then selects a submission button
- The browser reads the input fields, and creates a message that is sent to the server
 - A series of name, value pairs

HTML Form Creation

- **FORM**
 - Encloses all input fields
 - Defines where and how to submit the form data
- **INPUT**
 - Defines a specific input field
- **TEXTAREA**
 - Creates a free-form text fill-in box
- **SELECT**
 - Creates a menu
 - **OPTION** defines options within the menu

FORM

- FORM attributes
 - action
 - URL of the resource that receives the filled-in form
 - This is the URL of your PHP code that receives the input
 - method
 - Choices are “get” or “post” – you should choose “post”
 - enctype
 - MIME type used to send results. By default is application/x-www-form-urlencoded
 - Would use multipart/form-data if submitting a file (INPUT, type=file)

```
<FORM action="MyHandler.php" method="post">
```

INPUT

- INPUT attributes
 - **type**: the kind of user input control
 - **name**: the name of the control
 - This gets passed through to the handling code
 - In PHP: `$_POST['name']`
 - **value**: initial value of the control
 - **size**: initial width of the control
 - in pixels, except for text and password controls
 - **maxlength**: for text/password, maximum number of characters allowed
 - **checked**: for radio/checkbox, specifies that button is on
 - **src**: for image types, specifies location of image used to decorate input button

INPUT Control Types

- **text**: single input line
- **password**: single input line, with input characters obfuscated
- **checkbox**: creates a check list
- **radio**: creates a radio button list (checkbox, where inputs are mutually exclusive – only one input at a time)
- **button**: push button
- **hidden**: a hidden control. No input field is visible, but value is submitted as part of the form

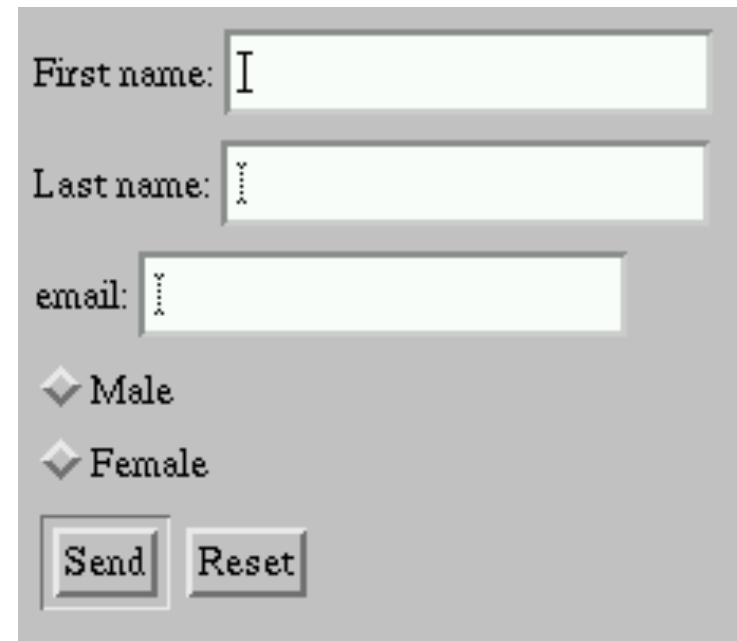
INPUT control types

- Special buttons
 - **submit**: the submit button. Causes input to be sent to the server for processing
 - **reset**: the reset button. Causes all input fields to be reset to their initial values
- File upload
 - **file**: creates a file upload control

Example

- From HTML 4.1 specification

```
<FORM
  action="http://people.ucsc.edu/~ejw/m
  yhandler.php" method="post">
  <P>
  First name: <INPUT type="text"
    name="firstname"><BR>
  Last name: <INPUT type="text"
    name="lastname"><BR>
  email: <INPUT type="text"
    name="email"><BR>
  <INPUT type="radio" name="sex"
    value="Male"> Male<BR>
  <INPUT type="radio" name="sex"
    value="Female"> Female<BR>
  <INPUT type="submit" value="Send">
  <INPUT type="reset">
  </P>
</FORM>
```



First name:

Last name:

email:

Male

Female

Receiving form input in PHP

- Upon receiving a form submission, PHP automatically creates and populates two arrays with the form input data
 - Either: `_POST[]` or `_GET[]`, depending on the FORM method type (post or get)
 - Also: `HTTP_POST_VARS[]` or `HTTP_GET_VARS[]`
 - Also: `$name`
 - Requires `register_globals` option, and is generally a bad idea due to cross-site attacks
 - Additionally, `_REQUEST[]` is also created
- The array indices are the names of the form variables (INPUT name=...)
- The array value is the user entry data

Validating Data

- All web applications must validate user input data
 - Ensure the data is syntactically correct, and meets input constraints
- Ideally want *one* PHP page to create form, validate input, and handle correct input
 - This is a little tricky

All-in-one Input Handling Page

- Overall logic:
 - If *first time displaying page* Then
 - Create form with initial default values
 - Else *have just received user input* Then
 - Validate input
 - If *input not valid* Then
 - Redisplay page, with just-received user input data as default values (so as not to lose the user's input)
 - Else
 - Handle the input (typically by writing it to a database)
 - Proceed to next screen in the web application

How to determine if first time

- Can check if the `$_POST[]` array is empty
 - Will be empty first time through
 - `if (empty($_POST)) { create initial form }`
 - `if (!empty($_POST)) { validate input }`

All-in-one structure

- Functions for input validation
- Function(s) that creates form
 - Default values set from contents of `_POST[]`
- HTML preamble
- Create initial form
- Validate input
- Recreate form
- Code to handle valid input
- HTML end

Libraries

- There are many libraries that exist for handling input, and performing validation
- These libraries can also create client-side Javascript for client-side value checking
- PEAR HTML_quickform is one example