1. Web Data Management

152-155

PHP Functions and Include Files

| Notes | Activity |
| --- | --- |
| 1. Quick Links & Text References    * [Locations](#locations) Pages 390 – 391    * [Function Libraries](#library) Pages 390 – 393    * [Namespaces](#namespace) Pages 394 – 395    * [Including Files](#include) Pages 76 – 77   Pages 164 – 165   Pages 390 – 393 [Defining a Function](#define) Pages 162 – 163  380 – 381    * [Returning Multiple Values](#multiple) Pages 382 – 383    * [Global Variables](#global) Pages 384 – 385    * [Optional Parameters](#optional) Pages 386 – 387    * [Variable Parameter Lists](#variable) Pages 388 – 389 |  |
| 1. Locations    * Can actually be placed anywhere (inside php tags)      + Even below the </html> tag    * Best stored in external files (libraries, most common) |  |
| 1. Function Libraries    * PHP functions stored in a separate file (.php extension)      + File must include begin and end php tags      + BIG IMPORTANT NOTE: Do not include **any** whitespace before the <?php or after the ?>.        - Localhost deals with them OK, but real web servers get confused by the extra spaces.    * Functions only. No HTML unless created by PHP (echo)    * To *include* in your web page: require('model/myLibrary.php');      + There is also a require\_once statement. Larger web sites may include libraries where the libraries include other libraries. A function may only be included once. If you try to include (require) it again, you’ll get an error.      + Murach recommends using require unless you run into trouble. | Create a weights.php  Model file  Add require statement to weightCases  Remove from cases.  Note errors (require?) |
| 1. Namespaces    * Occasionally, you’ll need to include two (or more) libraries that include functions with the same names, which of course is not allowed.    * Namespaces allow you to accomplish this without making the function names unique, by defining each function’s *namespace*.    * See the text (394-395) if you need additional information | Info Only |
| 1. Including Files    * As mentioned above require( )can be used to *include* PHP libraries (don’t include HTML)    * If you want to *include* external files that contain HTML and PHP, use the include( ) function.  include('views/errorPage.php');       + This must be included where you want the HTML injected. |  |
| * + In some circumstances (see examples in book) the controller needs to include a copy of itself (maybe with a slightly different display criteria).   + Unfortunately, a page cannot include itself (endless loop)   + To avoid this, use a header("Location: .") command     - This command actually sends a signal to the browser to request this same page again.       * The dot (period) represents the home directory of this web site, where index should be located       * Since we didn’t specify a file name, the default file (index) is automatically opened     - header("Location: .?action=$newAction");       * This version of the command sets a form variable (in index) to some value and sends it to the new version of the form.   + Often follows a save or delete action that doesn’t cause additional output on its own, so there’s no button or link to return to this page and display the main menu   + Use this command to basically link from one case to another in the controller     - Don’t try to include any other information (variables). Page is completely reloaded.   + I believe this command should only be used in the *cases* files   + Important note: header seems to ignore any SQL warnings (overwriting them before you can read them). | Example below |
| case isset($\_REQUEST['btnCancel']):  header("Location: ?action=employeeList");  break;  case isset($\_REQUEST['btnDelete']):  deleteEmployee($\_REQUEST['empID']);  header("Location: ?action=employeeList");  break;  case isset($\_REQUEST['btnTrips']):  header("Location: ?action=tripList&employeeId=" . $\_REQUEST['empID']);  break; | |
| * + BIG IMPORTANT NOTE Don’t include blank lines at the beginning or end of include files. These can cause problems when you transfer PHP pages to an actual server. See PHP Journal |  |
| 1. Basic Function Definitions   function funcName($parm1, $parm2 ...) {  return $value;  }//end function   * + return statement is optional     - Some functions do things, but don’t return a value   + Parameters ($parm1, $parm2, etc) are also optional   + Unlike some languages, **you cannot overload function names** (same name, different parameters) | Create:  calcTotalOunces( )  extract here  Create convertToOunces($lb) |
| 1. Arrays and Functions    * In PHP (unlike most other languages), arrays can be sent to or returned from functions like any other type of variable.      + Can be passed as parameter either by value or by reference      + Can be returned from a function via return statement | FYI |
| 1. Returning Multiple Values    * Return statement can only return one variable    * If you need a function to return multiple values, use *pass-by-reference* parameters   function funcName(&$parm1, &$parm2)   * + Function changes memory location (variable) passed to it instead of making its own copy   + & instructs PHP to treat parameters as pass by reference (instead of pass by value)     - & represents the *address* of the variable   + Inside the function, you don’t include the & in variable name (only in the parameter)     - Also don’t include in call   + Alternatively, could use global variables | Create calcWeightResults  Call totalOz  Calcs both endLbs and endOz  Use two PBR output parameters  Call from cases file |
| 1. Global Variables    * Any variable declared **outside a function** is considered a global variable      + All the variables we’ve used so far are global      + They are available in all PHP tags throughout a page, **except in functions**    * Any variable declared in a function is local to the function    * A function may access a global variable. The function must *declare* that it is using a global variable  global $varName;       + This example assumes the variable $varName is declared outside the function, but not in another function (in the global PHP code) | Use globals |
| 1. Optional Parameters    * Normal parameters are required    * If a parameter is optional, the function call does not need to include it (though it may)    * To define an optional parameter, you include it in the parameter list but also assign it a default value   function showAddress($addr, $city, $state='WI')   * + When calling this function, the call must include an address and city, but the state is optional. If it is not included, WI is used. Examples:   showAddress('123 Main St', 'New York', 'NY');  showAddress('933 Michigan Ave', 'Stevens Point');   * + If some parameters are required and others are optional (like the example above), the optional parameters must come last in the parameter list     - A required *zip code* parameter could not be included after the optional *state* parameter in the example above. | Create calcDeliveryDate with optional time frame parameter (default + 2 wks).  Use REQUEST element  Test with no parameter.  Test with parameter. |
| 1. Variable Parameter Lists    * Some functions accept a variable number of parameters    * These parameters differ from optional parameters      + No parameters are listed in the function declaration      + All parameters are optional      + You can force the call to include one parameter, but including a parameter in the ( )    * Inside the function you can list a list of the values sent to the function: $parms = func\_get\_args();    * func\_get\_args is a built in PHP function      + Creates an array of values sent to the function | FYI Only |