ISIT307 - WEB SERVER PROGRAMMING

LECTURE 2.2 – PROCESSING USER INPUT

LECTURE PLAN

- Learn about autoglobal variables
- Build HTML Web forms
- Process form data
- Handle submitted form data
- Create an All-in-One form
- Display dynamic data based on a URL token

USING AUTOGLOBALS

- **Autoglobals** are predefined global arrays that provide information about server, environment, and user input

Array	Description
$_COOKIE	An array of values passed to the current script as HTTP cookies
$_ENV	An array of environment information
$_FILES	An array of information about uploaded files
$_GET	An array of values from a form submitted with the "get" method
$_POST	An array of values from a form submitted with the "post" method
$_REQUEST	An array of all the elements in the $_COOKIE, $_GET, and $_POST arrays
$_SERVER	An array of information about the Web server that served the current script
$_SESSION	An array of session variables that are available to the current script
$GLOBALS	An array of references to all variables that are defined with global scope

PHP Programming with MySQL, 2011, Cengage Learning.

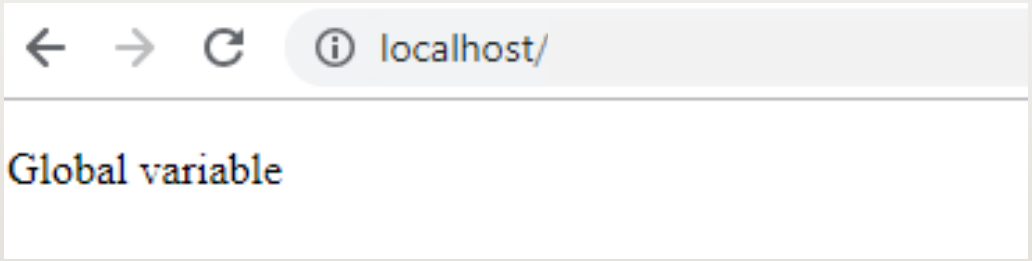
USING AUTOGLOBALS

- Autoglobals are associative arrays
  - To access the values in an associative array, place the element's key in single or double quotation marks inside the array brackets. (the following example displays the SCRIPT_NAME element of the $_SERVER autoglobal)

```
$_SERVER["SCRIPT_NAME"];
//displays the path and name of the current script
```

USING AUTOGLOBALS - EXAMPLE

```
<?php
$GlobalVariable = "Global variable";
function scopeExample() {
      echo "<p>" .  $GLOBALS["GlobalVariable"] . "</p>";
}
scopeExample();
?>
```



BUILDING HTML WEB FORMS

- **Web forms** are interactive controls that allow users to enter and submit data to a processing script
- A Web form is a standard HTML form with two required attributes in the opening `<form>` tag:
  - **Action attribute:** Identifies the program on the Web server that will process the form data when it is submitted
  - **Method attribute:** Specifies how the form data will be sent to the processing script

ADDING AN `ACTION` ATTRIBUTE

- The opening form tag requires an `action` attribute
- The value of the action attribute identifies the program on the Web server that will process the form data when the form is submitted

```
<form action="http://www.example.com/
                        HandleFormInput.php">
```

ADDING THE `METHOD` ATTRIBUTE

- The value of the `method` attribute must be either "post" or "get"
  - The "post" method embeds the form data in the request message
  - The "get" method appends the form data to the URL specified in the form's action attribute
- When a Web form is submitted using the "post" method, PHP automatically creates and populates a `$_POST` array; when the "get" method is used, PHP creates and populates a `$_GET` array

ADDING THE `METHOD` ATTRIBUTE

- Form fields are sent to the Web server as a *name/value* pair
  - The *name* portion of the *name/value* pair becomes the key of an element in the `$_POST` or `$_GET` array, depending on which method was used to submit the data
  - The *value* portion of the name/value pair is populated by the data that the user enters in the input control on the Web form
- When submitting data using the "get" method, form data is appended to the URL specified by the action attribute
- Name/value pairs appended to the URL are called **URL tokens**

ADDING THE METHOD ATTRIBUTE

- The form data is separated from the URL by a question mark (?)
- The individual elements are separated by an ampersand (&)
- The element name is separated from the value by an equal sign (=).
- Spaces in the *name* and *value* fields are encoded as plus signs (+)
  - all other characters except letters, numbers, hyphens (-), underscores (_) and periods (.) are encoded using a percent sign (%) followed by the two-digit hexadecimal representation of the character's ASCII value

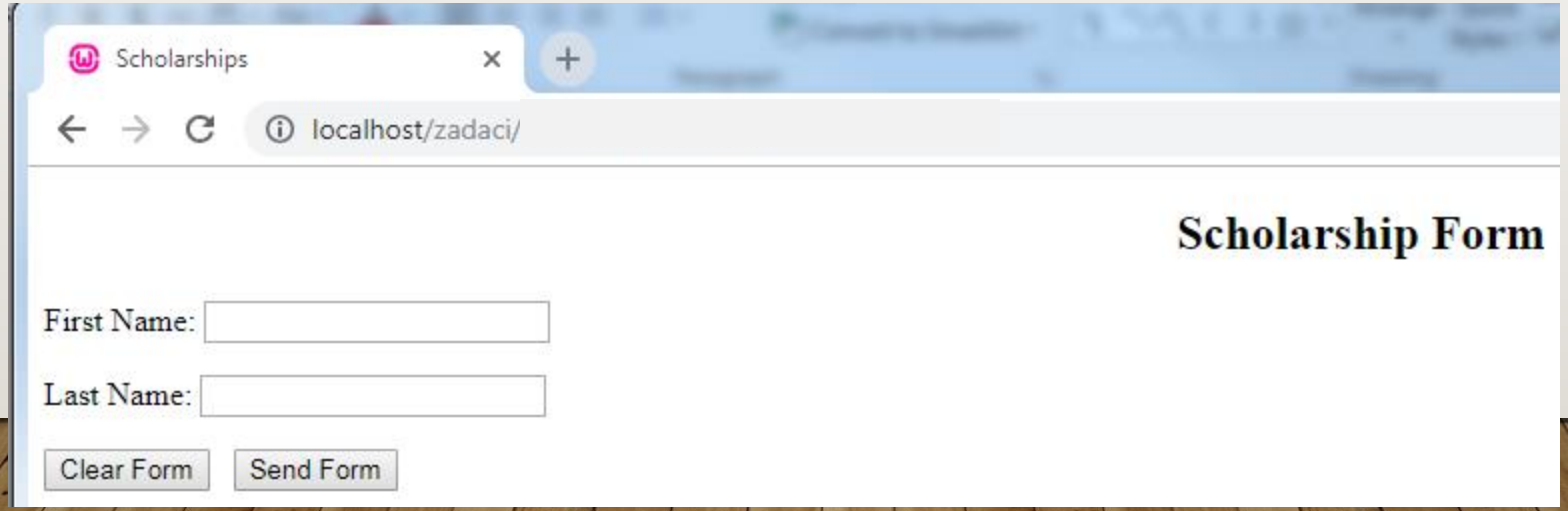
```
http://www.example.net/process_Scholarship.php?
  fName=Elena&lName=Vlahu&Submit=Send+Form
```

ADDING THE `METHOD` ATTRIBUTE

- Limitations of the "get" method for submitting form data
  - The form values are appended to the URL in plain text, making a URL request insecure
  - Restriction on the the length (number of characters) of the URL
- Advantage of the "get" method for submitting form data
  - Passed values are visible in the Address Bar of the browser

WEB FORMS - EXAMPLE

```
<body>
<h2 style="text-align:center">Scholarship Form</h2>
<form name="scholarship" action="process_Scholarship.php" method="post">
<p>First Name: <input type="text" name="fName" /></p>
<p>Last Name: <input type="text" name="lName" /></p>
<p><input type="reset" value="Clear Form" />  
<input type="submit" name="Submit" value="Send Form" /></p>
</form>
</body>
```



PROCESSING FORM DATA

- A **form handler** is a program or script that processes the information submitted from a Web form
- A form handler performs the following:
  - Verifies that the user entered the minimum amount of data to process the form
  - Validates form data
  - Works with the submitted data
  - Returns appropriate output as a Web page

RETRIEVING SUBMITTED DATA

- When the PHP script processes the user-submitted data, it can accessed values stored in the `$_POST/$_GET` array

```
$firstName = $_POST['fName']; //$firstName = $_GET['fName'];
$lastName = $_POST['lName'];  //$lastName = $_GET['lName'];

echo "Thank you for filling out the scholarship form,"
      .$firstName." ".$lastName  . ".";
```

- The `$_REQUEST` autoglobal can be used to access the results from data sent using either the "get" or "post" methods

```
$firstName = $_ REQUEST['fName'];
```

HANDLING SUBMITTED FORM DATA

- It is necessary to validate Web form data to ensure PHP can use the data
- The optimal way to ensure valid form data is only allow the user to enter an acceptable response
- Examples of data validation include verifying that
  - the user did not leave any required fields blank
  - an e-mail address was entered in the correct format
  - the user did not exceed the word limit in a comment box

DETERMINING IF FORM VARIABLES CONTAIN VALUES

- When form data is posted using the "post" or "get" method, all controls except unchecked radio buttons and checkboxes get sent to the server even if they do not contain data
- The `empty()` function is used to determine if a variable contains a value
- The `empty()` function returns `FALSE` if the variable being checked has a nonempty and nonzero value, and a value of `TRUE` if the variable has an empty or zero value

VALIDATING ENTERED DATA

- Validating form data refers to verifying that the value entered in a field is appropriate for the data type that should have been entered
- The best way to ensure valid form data is to build the Web form with controls (such as check boxes, radio buttons, and selection lists) that only allow the user to select valid responses
- Unique information, such as user name, password, or e-mail must be validated

VALIDATING NUMERIC DATA

- All data in a Web form is string data and PHP automatically converts string data to numeric data if the string is a number
  - The `is_numeric()` or `is_*()` functions are used to determine if a variable contains a number
  - The `round()` function can be used to a numeric variable with an appropriate number of decimal places

VALIDATING STRING DATA

- Regular expression functions are some of the best tools for verifying that string data meets the strict formatting required for e-mail addresses, Web page URLs, or date values
  - The `stripslashes()` function removes the leading slashes for escape sequences
  - The `trim()` function removes any leading or trailing white space from a string

HANDLING MULTIPLE ERRORS

- When processing a Web form, it is best to track any errors on the form during processing and then redisplay the form for the user to correct all the errors at one time

- Example
  - process_scholarship1.php

REDISPLAYING THE WEB FORM

- A good option is to redisplay the form with the controls set to the values that the user entered the last time the form was submitted – so, the user only has to enter data for fields that were left empty or did not contain a valid response
- A **sticky form** is used to redisplay the form with the controls set to the values the user entered the last time the form was submitted
- To redisplay the Web form, we need to add the HTML form elements to the output of the PHP script - the code to output the Web form should be part of the error-handling section of the script

REDISPLAYING THE WEB FORM

- The most convenient way to embed large portions of HTML code within a PHP script is to use **advanced escaping** from HTML
- With advanced escaping, we close one PHP script section, insert some HTML elements, and then open another PHP script section to continue the script
- Any HTML code between the two PHP script sections is considered output.

REDISPLAYING THE WEB FORM - EXAMPLE

- process_scholarship2.php

CREATING AN ALL-IN-ONE FORM

- A **two-part form** has one page that displays the form and one page that processes the form data
- For simple forms that require only minimal processing, it's often easier to use an **All-in-One form**—a single script used display a Web form and process its data
- When the user click the submit button, the script submit data to the current script

VALIDATING AN ALL-IN-ONE FORM

- Conditionals:
  - if the data has been submitted and needs to be validated
  - If the form needs to be redisplay (for the first time, or because of error)

VALIDATING AN ALL-IN-ONE FORM

- The `isset()` function is used to determine if the $Submit variable has been set (the form is submitted)

```
if (isset($Submit)) {
    // Validate the data
}
```

- The argument of the `isset()` function is the name assigned to the Submit button in the Web form

REDISPLAYING THE WEB FORM

- If the submitted data did not pass all validation checks or no data has been entered, the All-in-One form will display the Web form, for the user to enter data for the first time or re-enter data that did not pass validation

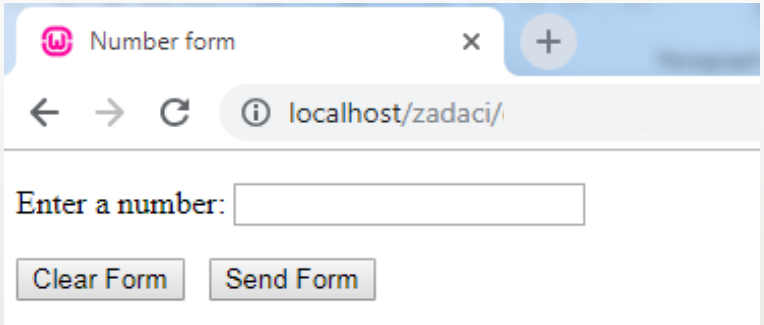
```
if (isset ($_POST['Submit'])) {
  // Process the data
}
else {
  // Display the Web form
}
```

REDISPLAYING THE WEB FORM - EXAMPLE

```
<?php
$DisplayForm = TRUE;   $Number = "";
if (isset($_POST['Submit'])) {
        $Number = $_POST['Number'];
        if (is_numeric($Number)) {
           $DisplayForm = FALSE;
        } else {
           echo "<p>You need to enter a numeric value.</p>\n";
           $DisplayForm = TRUE;}
}
if ($DisplayForm) {
     ?>
     <form name="NumberForm" action="example4-28.php" method="post">
     <p>Enter a number: <input type="text" name="Number" value="<?php echo $Number; ?>" /></p>

     <p><input type="reset" value="Clear Form" />   
     <input type="submit" name="Submit" value="Send Form" /></p>
     </form>
     <?php
}else {
     echo "<p>Thank you for entering a number.</p>\n";
     echo "<p>Your number, $Number, squared is " . ($Number*$Number) . ".</p>\n ";
     echo "<p><a href=\"example4-28.php\">Try again?</a></p>\n";   }
?>
```

REDISPLAYING THE WEB FORM – EXAMPLE OUTPUT



DISPLAYING DYNAMIC CONTENT BASED ON A URL TOKEN

- By passing URL tokens to a PHP script, many different types of information can be displayed from the same script
- By using a Web page template with static sections and a dynamic content section, a single PHP script can produce the same content as multiple static HTML pages

USING A WEB PAGE TEMPLATE

- A **Web template** is a single Web page that is divided into separate sections such as
  - Header
  - Button/Hyperlink Navigation
  - Dynamic Content
  - Footer
- The contents of the individual sections are populated using include files
- The navigation within the Web page template is by using hyperlinks and buttons

USING TEXT HYPERLINKS FOR NAVIGATION

- When the user clicks on a text hyperlink the contents that display in the dynamic data section are replaced by the contents referenced by the `href` attribute
- A *name/value* pair is appended to the index URL
  (this attribute and value will be referenced in the dynamic data section)
  - The name is user defined
  - The value is user defined

```
<a href = "index.php?content=Home">Home</a>
```

- Buttons must be enclosed by a opening and closing `<form>` tag

```
<input type="submit" name="content" value="Home" />
```

DISPLAYING THE DYNAMIC CONTENT

- The dynamic content section of the index.php file will contain the code to determine which content page to display

EMAILING THE WEB FORM DATA

- The `mail()` function is used to send an e-mail message containing form data in PHP
- The basic syntax for this function is

```
mail(recipient(s), subject, message)
```

- The **Address Specifier** defines the format of the e-mail addresses that can be entered as the recipient argument
  - Plain e-mail address: `jdoe@example.net`
  - Recipients name and e-mail address: `Mary Smith <mary.smith@example.com>`

EMAILING THE WEB FORM

- The `subject` argument of the `mail()` function must include only plain text with no HTML tags or character entities unless a special MIME format is used
- The `message` argument of the `mail()` function is a text string that must also be in plain text
- A fourth, optional `additional_headers` argument can include headers that are standard in most e-mail editors – From, Cc, Bcc and Date
- A successful e-mail message returns a value of `TRUE`