

School of Information and Computer Technology
Sirindhorn International Institute of Technology
Thammasat University
ITS351 Database Programming Laboratory

Laboratory #3: HTML Forms and PHP

HTML Forms

Is there a way that users can provide information to servers and get a personalized response return? The answer, of course, is “yes”, and the way to do it is with World Wide Web forms. Users enter information into form fields and click a button to submit the data. The browser then packages the data, opens an HTTP connection, and sends the data to a server. The server then processes the information and creates a reply, usually in HTML. This hand-off occurs with the help of the Common Gateway Interface or CGI—a set of standards by which servers communicate with external programs.

The program that processes the form data is typically called a CGI script or a CGI program. The script or program performs some manipulations of the data and composes a response – typically an HTML page. The response page is handed back to the server (via CGI), which, in turn, passes it along to the browser that initiated the request. The server-side data-processing aspects of forms are not part of HTML standard; they are defined by the server’s software.

<FORM> Tag

HTML’s form support is simple and complete. All form-related tags occur between the <FORM> and </FORM> container tags. Each HTML form has three main components: the form header, one or more input fields, and one or more action buttons. The form header and the <FORM> tag are actually one and the same. The <FORM> tag takes the three attributes shown below. Only the ACTION is mandatory.

<FORM> begin a form
<INPUT> ask for information by one of several different ways...
<INPUT> ...there can be as many input areas as you wish
</FORM> end a form

Table 1: A list of FORM attributes

Attribute	Purpose
Accept	Specifies a list of MIME types that the server will process correctly
Accept-Charset	Provides a list of character sets that are acceptable to the server
Action	Specifies the URL of the processing script.
Enctype	Supplies the MIME type of a file used as form input.
Method=Get Post	Tells the browser how it should send the form data to server.
Target	Gives the name of the frame where the response from the form submission is to appear

For the attribute method, with the post method, the browser sends the data in two steps: the browser first contacts the form-processing server specified in the action attribute, and contact is made, sends the data to the server in a separate transaction. Forms can contain a wide range of HTML markup including several

kinds of form field such as single and multi-line text fields, radio button groups, checkboxes, and menus.

Example of FORM:

```
<HTML>
  <BODY>
    <FORM action="form2.php" method="POST">
      <!-- All Form Fields must locate here !!!-->
    </FORM>
  </BODY>
</HTML>
```

FORM fields

INPUT, SELECT and TEXTAREA are only allowed within FORM elements. INPUT can be used for a variety of form fields including single line text fields, password fields, checkboxes, radio buttons, submit and reset buttons, hidden fields, file upload, and image buttons. SELECT elements are used for single or multiple choice menus. TEXTAREA elements are used to define multi-line text fields. The content of the element is used to initialize the field.

INPUT

INPUT elements are not containers and so the end tag is forbidden.

Table 2: INPUT attributes

Attribute	Purpose
Name	Used to define the property name that will be used to identify this field's content when it is submitted to the server.
Value	Used to initialize the field, or to provide a textual label for submit and reset buttons.
Checked	The presence of this attribute is used to initialize checkboxes and radio buttons to their checked state.
Size	Used to set the visible size of text fields to a given number of average character widths, e.g. size=20
Maxlength	Sets the maximum number of characters permitted in a text field.
Src	Specifies a URL for the image to use with a graphical submit button.
Align	Used to specify image alignment for graphical submit buttons. It is defined just like the <u>IMG</u> align attribute.
Type	A single line text field whose visible size can be set. The attribute of type is on next table.

Table 3: Type attributes

Attribute	Purpose	Example
Type=password	Input Password	<input type=password size=12 name=pw>
Type=checkbox	Check Box	<input type=checkbox checked name=nm value=yes>
Type=radio	Radio buttons	<input type=radio name=age value="26-35" checked>
Type=image	Graphical submit buttons	<input type=image src=partyon.gif value="Party on ...">
Type=reset	reset button	<input type=reset value="Start over ...">
Type=file	Attach a file to the form's contents.	<input type=file name=photo size=20 accept="image/*">
Type=hidden	Allow servers to store state info. with a form.	<input type=hidden name=customerid value="c2415-345-8563">

Example of Input Tag:

```

<HTML>
  <BODY>
    <FORM action="form2.php" method="POST">
      <!--Example of Text -->
      TEXT1:<INPUT TYPE="text" NAME="text1" SIZE=20><BR>
      TEXT2:<INPUT TYPE="text" NAME="text2" SIZE=30 VALUE="Enter item
2 here">
      <!--Example of Password Box -->
      PASSWORD: <INPUT TYPE="password" NAME="passwd" SIZE=20> <BR>
      <!--Example of Radio Button -->
      RADIO:<INPUT TYPE="radio" NAME="radioname" VALUE="better"
CHECKED> Better
      <INPUT TYPE="radio" NAME="radioname" VALUE="best"> Best <BR>
      <!--Example of Check Box -->
      CHECKBOX: <INPUT TYPE="checkbox" NAME="checkbox1"
VALUE="database"> Database Books
      <INPUT TYPE="checkbox" NAME="checkbox2" VALUE="network" CHECKED>
Network Book<BR>
      <!-- Example of Hidden -->
      <INPUT NAME="hiddenvalue" TYPE="hidden" VALUE="secret"><BR>
      <!-- Upload File -->
      FILE: <INPUT TYPE="file" NAME="filename"> <BR>
    </FORM>
  </BODY>
</HTML>

```

SELECT

SELECT is used to define select one from many or many from many menus. SELECT elements require start and end tags and contain one or more OPTION elements that define menu items. One from many menus is generally rendered as drop-down menus while many from many menus are generally shown as list boxes.

Table 4: SELECT attributes

Attribute	Purpose
Name	This specifies a property name that is used to identify the menu choice when the form is submitted to the server.
Disabled	Deactivates the field
Multiple	The presence of this attribute signifies that the users can make multiple selections. By default only one selection is allowed.
Size	This sets the number of visible choices for many from many menus.

Table 5: OPTION attributes

Attribute	Purpose
Selected	When this attribute is present, the option is selected when the document is initially loaded.
Disabled	Makes the option unavailable
Label	Provides a short label for the menu option. If specified, this label is used in place of the option text itself.
value	Specifies the property value to be used when submitting the form's content.

Example of Select Tag:

```
<HTML>
  <BODY>
    <FORM action="form2.php" method="POST">
      <!--Example of Combo Box -->
      COMBO BOX:<SELECT NAME="combo">
        <OPTION VALUE="lions">Lions</OPTION>
        <OPTION VALUE="monkey">Monkeys</OPTION>
        <OPTION VALUE="tiger">Tigers</OPTION>
      </SELECT><BR>
      <!--Example of List Box -->
      LIST BOX:<SELECT NAME="list" SIZE=3>
        <OPTION VALUE="people">People</OPTION>
        <OPTION VALUE="who">who</OPTION>
        <OPTION VALUE="feed">feed</OPTION>
      </SELECT><BR>
    </FORM>
  </BODY>
</HTML>
```

TEXTAREA

TEXTAREA elements require start and end tags. The content of the element is restricted to text and character entities. It is used to initialize the text that is shown when the document is first loaded.

Table 6: TEXTAREA attribute

Attribute	Purpose
Name	This specifies a property name that is used to identify the TEXTAREA field when the form is submitted to the server.
Rows	Specifies the number of visible text lines.
Cols	Specifies the visible width in average character widths.

Example of Text Area Tag:

```
<HTML>
  <BODY>
    <FORM action="form2.php" method="POST">
      TEXT AREA:<TEXTAREA NAME="comment" ROWS=7 COLS=56>This is an
      Example of Text area so you can view that
    </TEXTAREA><BR>
  </FORM>
</BODY>
</HTML>
```

BUTTON

The BUTTON tag places a button on the form. This type of button is different from the one rendered by <INPUT> because it has improved presentation features, such as 3-D rendering and up/down movement when clicked.

Table 7: BUTTON attribute

Attribute	Purpose
Name	Gives the button a unique name
Disabled	Disables the button
Type	Set to SUBMIT, RESET, or BUTTON, depending on the type of button you are defining. TYPE="BUTTON" is typically used for defining a scripted button.
Value	Specifies what is passed to the server when the button is clicked.

Example of Button Tag:

```
<HTML>
  <BODY>
    <FORM action="form2.php" method="POST">
      <!-- Example Submit & Reset Button -->
      BUTTON: <INPUT TYPE="submit" name="submit" VALUE="Submit my
      form">
      <INPUT TYPE="reset" VALUE="Reset my form">
      <!-- Example of submit with picture -->
      <INPUT TYPE="image" SRC="images/save.png" WIDTH=48 HEIGHT=48
      BORDER=0 ALT="Image Button">

    </FORM>
  </BODY>
</HTML>
```

Example of form1.html:

```

<HTML>
<BODY>
  <FORM action="form2.php" method="POST">
    TEXT1:<INPUT TYPE="text" NAME="text1" SIZE=20><BR>
    TEXT2:<INPUT TYPE="text" NAME="text2" SIZE=30 VALUE="Enter item 2 here">
    PASSWORD: <INPUT TYPE="password" NAME="passwd" SIZE=20> <BR>
    RADIO:<INPUT TYPE="radio" NAME="radioname" VALUE="better" CHECKED> Better
    <INPUT TYPE="radio" NAME="radioname" VALUE="best"> Best <BR>
    CHECKBOX: <INPUT TYPE="checkbox" NAME="checkbox1" VALUE="database">
Database Books
  <INPUT TYPE="checkbox" NAME="checkbox2" VALUE="network" CHECKED> Network
Book<BR>
  <INPUT NAME="hiddenvalue" TYPE="hidden" VALUE="secret">
  FILE: <INPUT TYPE="file" NAME="filename"> <BR>
  COMBO BOX:<SELECT NAME="combo">
    <OPTION VALUE="lions">Lions
    <OPTION VALUE="monkey">Monkeys
    <OPTION VALUE="tiger">Tigers
  </SELECT><BR>
  LIST BOX:<SELECT NAME="list" SIZE=3>
    <OPTION VALUE="people">People
    <OPTION VALUE="who">who
    <OPTION VALUE="feed">feed
  </SELECT><BR>
  TEXT AREA:<TEXTAREA NAME="comment" ROWS=7 COLS=56>This is an Example of
Text area so you can view that
  </TEXTAREA><BR>
  BUTTON: <INPUT TYPE="submit" name="submit" VALUE="Submit my form">
  <INPUT TYPE="reset" VALUE="Reset my form">
  <INPUT TYPE="image" SRC="images/save.png" WIDTH=48 HEIGHT=48 BORDER=0
ALT="Image Button">
  </FORM>
</BODY>
</HTML>

```

TEXT1:

TEXT2: Enter item 2 here PASSWORD:

RADIO: Better Best


CHECKBOX: Database Books Network Book

FILE:

COMBO BOX: ▼

LIST BOX:
 People
 who
 feed

TEXT AREA:

BUTTON: 

Form Processing Using PHP

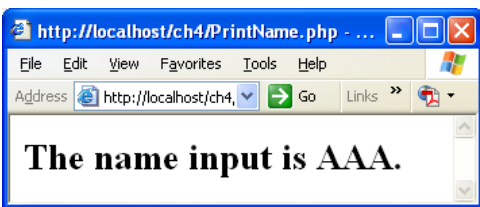
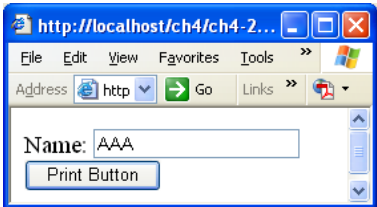
When a form is submitted to a PHP script, any variables from that form will be automatically made available to the script by PHP.

example1.html

```
<!-- This is a test. -->
<FORM ACTION="printname1.php" METHOD="post">
Name: <INPUT TYPE="text" NAME="name"><br>
      <INPUT TYPE="submit" NAME="submit" VALUE="Print Button">
</FORM>
```

printname1.php

```
<?
$name=$_POST["name"];
echo("<H2>The name input is $name.<BR></H2></n>");
?>
```



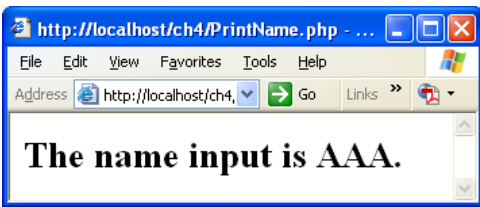
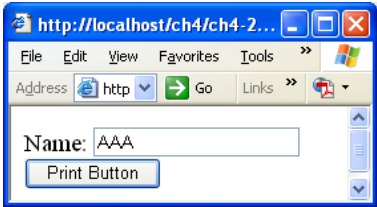
When submitted, PHP will create the variable `$name`, which will contain whatever what entered into the *Name:* field on the form. PHP also understands arrays in the context of form variables, but only in one dimension. You may, for example, group related variables together, or use this feature to retrieve values from a multiple select input.

example2.html

```
<!-- This is a test. -->
<FORM ACTION="printname2.php" METHOD="get">
Name: <INPUT TYPE="text" NAME="name"><br>
      <INPUT TYPE="submit" NAME="submit" VALUE="Print Button">
</FORM>
```

printname2.php

```
<?
$name=$_GET["name"];
echo("<H2>The name input is $name.<BR></H2></n>");
?>
```



We're still using the post method. The action is now "printname3.php", since this is a new

example, and we've added a new input: a "select" box, also known as a "drop-down" or "pull-down" box. A select box contains one or more "options". Each option has a "value", just like other inputs, and also a string of text between the option tags. **This means when a user selects "Male", the "gender" value when accessed by PHP will be "M".**

example3.html

```

<FORM ACTION="printname3.php" METHOD="post">
Name: <input type="text" name="name"><br>
Email: <input type="text" name="email"><br>
Please choose your gender? <br>
  <select name="gender">
    <option value="">Select...</option>
    <option value="M">Male</option>
    <option value="F">Female</option>
  </select><br>
<input type="submit">
</FORM>

```

printname3.php

```

<?
  $name=$_POST["name"];
  $email=$_POST["email"];
  $gender = $_POST["gender"];
  echo("<H2>The name is $name.<BR></H2></n>");
  echo("<H2>The e-mail is $email.<BR></H2></n>");
  echo("<H2>The gender is $gender.<BR></H2></n>");
?>

```

Name:

Email:

Please choose your gender?

Male ▾

The name is aaa.

The e-mail is aaa@aaa.com.

The gender is M.

Next, we are going to create our PHP file that will process the HTML form of "form1.html". A lot of things were going on in this example. Let step through it to be sure you understand what was going on.

1. We first identify an HTML form "form1.html" that points to "form2.php" and set the method to "post" in the form tag.
2. We create "form2.php" which gets the information that was posted by setting new variables equal to the values in the \$_POST associative array.
3. We used the PHP echo function to output the values from HTML form.


form2.php

```
<?
    $text1=$_POST["text1"];
    $text2=$_POST["text2"];
    $passwd = $_POST["passwd"];
    $radioname = $_POST["radioname"];
    $checkbox1 = $_POST["checkbox1"];
    $checkbox2 = $_POST["checkbox2"];
    $hiddenvalue = $_POST["hiddenvalue"];
    $filename = $_POST["filename"];
    $combo = $_POST["combo"];
    $list = $_POST["list"];
    $comment = $_POST["comment"];

    echo("The text1 is $text1.<BR></n>");
    echo ("The text2 is $text2.<BR></n>");
    echo ("The passwd is $passwd.<BR></n>");
    echo ("The radioname is $radioname.<BR></n>");
    echo ("The checkbox1 is $checkbox1.<BR></n>");
    echo ("The checkbox2 is $checkbox2.<BR></n>");
    echo ("The hiddenvalue is $hiddenvalue.<BR></n>");
    echo ("The filename is $filename.<BR></n>");
    echo ("The combo is $combo.<BR></n>");
    echo ("The list is $list.<BR></n>");
    echo ("The comment is $comment.<BR></n>");
?>
```

Output will be shown as in the following figure.

The screenshot shows a web form with the following elements:

- TEXT1:
- TEXT2: PASSWORD:
- RADIO: Better Best
- CHECKBOX: Database Books Network Book
- FILE:
- COMBO BOX: (Dropdown menu with options: People, who, feed)
- LIST BOX: (Text area with vertical scrollbar)
- TEXT AREA:
- BUTTON: 

OUTPUT IS:

The text1 is aaa.
The text2 is bbb.
The passwd is 1234.
The radioname is better.
The checkbox1 is .
The checkbox2 is network.
The hiddenvalue is secret.
The filename is C:\\2.xps.
The combo is tiger.
The list is people.
The comment is This is Comment.

Worksheet 1

Create a set of web pages for user management project with HTML forms as follows. Design the size of each frame and the contents used in the web by yourself. The MENU frame (left frame) should contain the links to contents in the MAIN PAGE frame (right frame).

add_group.html

Steps of creating HTML form

1. You need to create form in html file, add_group.html, in folder "AppServ\www\lab3\"
2. Open file, add_group.html, using editor to write HTML
3. Start with <HTML> and </HTML> Tag
4. Write Title "ITS331 Sample Website" between <TITLE> and </TITLE> Tag inside <HEAD> Tag
5. In <BODY> Tag, create table using <TABLE> Tag. Set width="620", border="1", align="center", cellpadding="0", and cellspacing="0"
6. Inside <TABLE> Tag (from step 5) in the first row, create new table using <TABLE> Tag to contain page title. Separate it into 2 columns. First column contains "ITS331 System" and second column contains "HTML Forms and PHP". Close Tag to this table.
7. Inside <TABLE> Tag (from step 5) in the second row, create new table using <TABLE> Tag to contain menus and HTML forms. Separate this table into 4 columns.
8. Inside first column of <TABLE> Tag in step 7, create new table using <TABLE> Tag to contain list of menus as followed:
 - User Profile (link to user.php)

- Add User (link to add_user.html)
 - User Group (link to group.php)
 - Add User Group (link to add_group.html)
- Close Tag to this table.
9. Inside second column of <TABLE> Tag in step 7, leave it blank.
 10. Inside third column of <TABLE> Tag in step 7, create new table using <TABLE> Tag to contain form fields as followed:
 - Group Code as textbox
 - Group Name as textbox
 - Remark as text area
 - URL as textbox
 - Submit and Cancel Button
 Don't forget to add form name and close Tag to this table.
 11. Inside fourth column of <TABLE> Tag in step 7, leave it blank. Close tag to this table.
 12. Inside <TABLE> Tag (from step 5) in the third row, leave it blank.
 13. Inside <TABLE> Tag (from step 5) in the fourth row, write down the copyright. Close tag to this table.
 14. Submit button in step 10 must send user input data to "group.php" using "POST" method.

group.php

Group Code	Group Name	Remark	URL	Edit	Del
1	Admin	Administrator Account	admin.php		

Steps of creating PHP for receiving input data

1. You need to create form in PHP file, group.php, in folder "AppServ\www\lab3\"
2. Open file, group.php, using editor to write PHP and HTML
3. Start with <HTML> and </HTML> Tag
4. Write Title "ITS331 Sample Website" between <TITLE> and </TITLE> Tag inside <HEAD> Tag
5. In <BODY> Tag, create table using <TABLE> Tag. Set width="650", border="1", align="center", cellpadding="0", and cellspacing="0"
6. Inside <TABLE> Tag (from step 5) in the first row, create new table using <TABLE> Tag to contain page title. Separate it into 2 columns. First column contains "ITS331 System" and second column contains "HTML Forms and PHP". Close Tag to this table.
7. Inside <TABLE> Tag (from step 5) in the second row, create new table using <TABLE> Tag to contain menus and HTML forms. Separate this table into 4 columns.
8. Inside first column of <TABLE> Tag in step 7, create new table using <TABLE> Tag to contain list of menus as followed:
 - User Profile (link to user.php)

- Add User (link to add_user.html)
 - User Group (link to group.php)
 - Add User Group (link to add_group.html)
- Close Tag to this table.
9. Inside second column of <TABLE> Tag in step 7, leave it blank.
 10. Go to the top of the page (above <HTML> Tag). Write PHP code in order to store value of input data in the set of variables by using \$_POST as followed:
 - Group Code stores in \$groupcode
 - Group Name stores in \$groupname
 - Remark stores in \$remark
 - URL stores in \$url
 11. Inside third column of <TABLE> Tag in step 7, create new table using <TABLE> Tag to display the input data from step 10. In this case, you need to apply PHP code together with HTML. Last two columns must display modify icon and delete icon consequently. Close Tag to this table.
 12. Inside fourth column of <TABLE> Tag in step 7, leave it blank. Close tag to this table.
 13. Inside <TABLE> Tag (from step 5) in the third row, leave it blank.
 14. Inside <TABLE> Tag (from step 5) in the fourth row, write down the copyright. Close tag to this table.

Worksheet 2

Extend the user management project (from worksheet) with HTML forms as follows.

add_user.html

Note that you must create form fields in this page as followed:

- Title contains Mr., Mrs., and Ms. as combobox
- First Name as textbox
- Last Name as textbox
- Gender contains Male and Female as radio button
- Email as textbox
- User Name as textbox
- Password as password box
- Confirmed Password as password box
- User Group as Admin, Staff, and Member as combobox
- Disabled as checkbox
- Submit and Cancel Button

Submit button must send user input data to "user.php" using "POST" method.

Create PHP file, user.php, for receiving input data from previous HTML form and display them in a form of table. Note that you must check correctness of input data and display error message if error as followed:

1. If password input does not match with confirmed password. You must display the following error message. Keyword "back" must link to add_user.html.

Password Mismatch. Please go [back](#) to edit !!!

2. If title and gender not match, for example, Mr. must select Male while Ms. and Mrs. must select Female, you must display the following error message. Keyword "back" must link to add_user.html.

Gender Mismatch. Please go [back](#) to edit !!!

3. Username must greater than 6 characters. If error, you must display the following error message. Keyword "back" must link to add_user.html.

Username must greater than 6 characters. Please go [back](#) to edit !!!

If all conditions pass, please display input data as shown in the figure.

The screenshot shows a web application interface. At the top, there are two tabs: "ITS331 System" and "HTML Forms and PHP". Below the tabs is a navigation menu with four items: "User Profile", "Add User", "User Group", and "Add User Group". The "User Profile" item is circled in red, and an arrow points from it to a table titled "User Profile". The table has five columns: "Title", "Name", "Email", "User Group", and "Disabled". The first row of data shows "Mr.", "John Smith", "js@mail.com", "Admin", and a checked checkbox. At the bottom of the page, there is a copyright notice: "Copyright © 2011 Sitenam.com."