

UNIT – IV

USER DATA INPUT THROUGH FORMS

4.1 Input through Form Controls

- When you are designing the web page, that accepts the input from the user and process the information entered by the user, at that time you need to design a form in your application.
- In PHP, you can design the form which contains various input controls that allows user to input information using the concept of form element.

- **Syntax:**

`<form>`

`</form >`

4.1 Input through Form Controls

- Two important attributes of FORM element are:
 1. **Method** : This attribute specifies which method will be used by the form to transfer the contents of the various controls to the specified file.
 - It can have one of the two values : GET or POST
 2. **Action** : This attribute specifies the name of the file to which the contents of the various controls are sent for the processing.
- Thus the general syntax of Form element should be like:

```
<form method = "GET/POST" action = "filename">
```

```
</form >
```

4.1 Input through Form Controls

- **INPUT Elements**

- Once you define the Form in your PHP page, your next task is to place various controls that accepts input from the user in different ways.
- **INPUT element allows you to place various controls in the Form to accept input from the user.**
- INPUT element must be contained in the FORM element as shown below:

```
<FORM method = "GET/POST" action = "filename">  
  <INPUT type = "text" name = "username">  
  <INPUT type = "password" name = "password">  
</FORM>
```

- **There is no need to close the INPUT tag.**

4.1 Input through Form Controls

- Following are the important attributes that are used with INPUT element.

1. **Name** : This attribute specifies the name of the input control.

- The name is used to retrieve the value of the input control in the destination file.

2. **Type** : This attribute specifies the type of the input control.

- You can specify different values for this attribute as:

Type	Control
Text	Text Box
Password	Password Field
Checkbox	Check Box
Radio	Radio Button
Hidden	Hidden Field
Submit	Submit Button

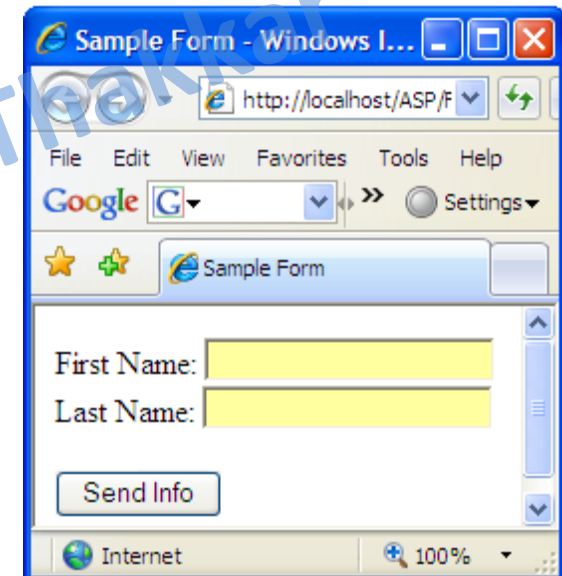
4.1 Input through Form Controls

3. **Value** : This attribute specifies the value associated with the input control.

Form1.php

```
<HTML>
<HEAD>
  <TITLE>Sample Form</TITLE>
</HEAD>
<BODY>
  <FORM method = "POST" action="Form2.php" >
    First Name: <INPUT TYPE="TEXT" NAME="fname" >
    <BR>
    Last Name: <INPUT TYPE="TEXT" NAME="lname" >
    <BR>
    <INPUT TYPE="SUBMIT" NAME="Submit1"
      VALUE="SendInfo">
  </FORM>
</BODY>
</HTML>
```

Output



4.1 Input through Form Controls

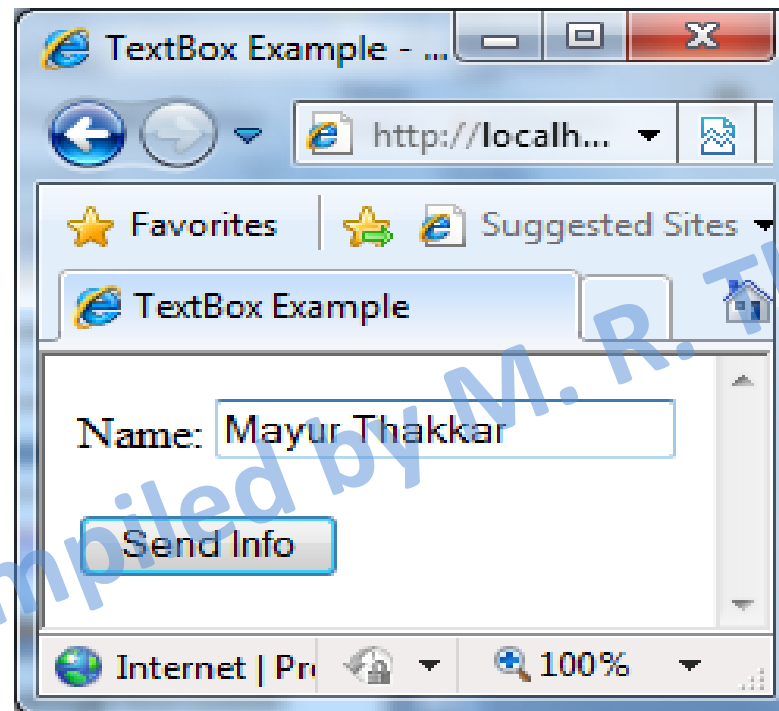
4.1.1 Text Box : Text Box can be used to enter single line of text.

- **Syntax :** <INPUT TYPE="TEXT" NAME="fname" >
- **Example:**

```
Form1.php
<HTML>
<HEAD>
  <TITLE>Sample Form</TITLE>
</HEAD>
<BODY>
  <FORM method = "POST" action="Form2.php" >
    Name: <INPUT TYPE="TEXT" NAME="name" >
    <BR>
    <INPUT TYPE="SUBMIT" NAME="submit" VALUE="Send Info">
  </FORM>
</BODY>
</HTML>
```

4.1 Input through Form Controls

4.1.1 Text Box



4.1 Input through Form Controls

4.1.2 Password : Password Field is used to enter Password.

- When user enter any text within this control, it is displayed in the form of asterisk (*).

- Syntax : <INPUT TYPE="PASSWORD" NAME="pass" >

4.1 Input through Form Controls

4.1.2 Password

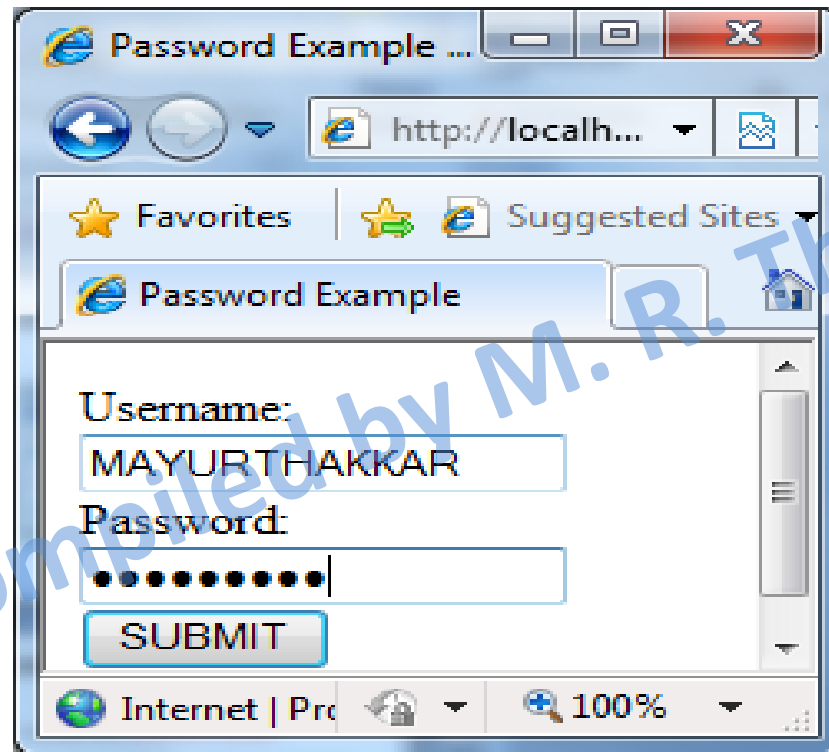
- Example:

Form1.php

```
<HTML>
<HEAD>
  <TITLE>Password Example</TITLE>
</HEAD>
<BODY>
  <FORM method = "POST" action="Form2.php" >
    Username: <INPUT TYPE="TEXT" NAME="user" >
    <BR>
    Password: <INPUT TYPE="PASSWORD" NAME="pass" >
    <BR>
    <INPUT TYPE="SUBMIT" NAME="Submit" VALUE="SUBMIT">
  </FORM>
</BODY>
</HTML>
```

4.1 Input through Form Controls

4.1.2 Password



4.1 Input through Form Controls

4.1.3 CheckBox : CheckBox allow the user to select more than one option from given options.

- **Syntax :** `<INPUT TYPE="CHECKBOX" NAME = "CheckBoxName"
VALUE = "DefaultValue" CHECKED > TEXT TO DISPLAY
</INPUT>`

4.1 Input through Form Controls

4.1.3 CheckBox

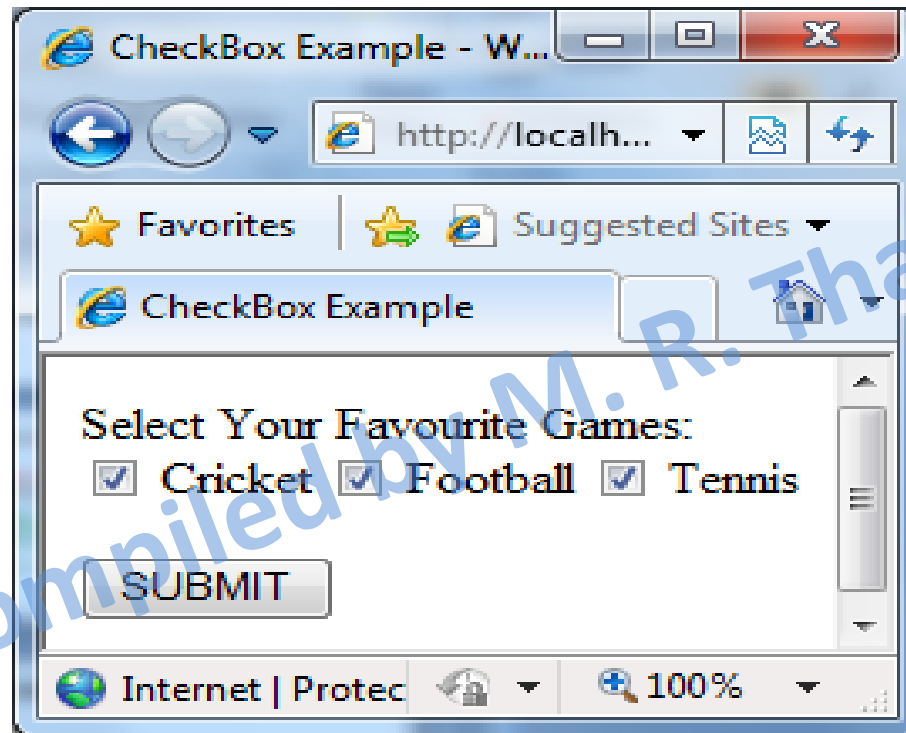
- Example:

Form1.php

```
<HTML>
<HEAD>
  <TITLE>CheckBox Example</TITLE>
</HEAD>
<BODY>
  <FORM method = "POST" action="Form2.php" >
    Select Your Favourite Games:
    <BR>
    <INPUT TYPE="CHECKBOX" NAME = "Game[]" VALUE = "Cricket" CHECKED > Cricket </INPUT>
    <INPUT TYPE="CHECKBOX" NAME = "Game[]" VALUE = "Football" > Football </INPUT>
    <INPUT TYPE="CHECKBOX" NAME = "Game[]" VALUE = "Tennis" > Tennis </INPUT>
    <BR>
    <INPUT TYPE="SUBMIT" NAME="submit" VALUE="SUBMIT">
  </FORM>
</BODY>
</HTML>
```

4.1 Input through Form Controls

4.1.3 CheckBox



4.1 Input through Form Controls

4.1.4 Radio Button : Radio Buttons allow the user to select only one option from given options.

- **Syntax :** `<INPUT TYPE="radio" NAME = "RadioName"
VALUE = "DefaultValue" CHECKED > TEXT TO DISPLAY
</INPUT>`

4.1 Input through Form Controls

4.1.4 Radio Button

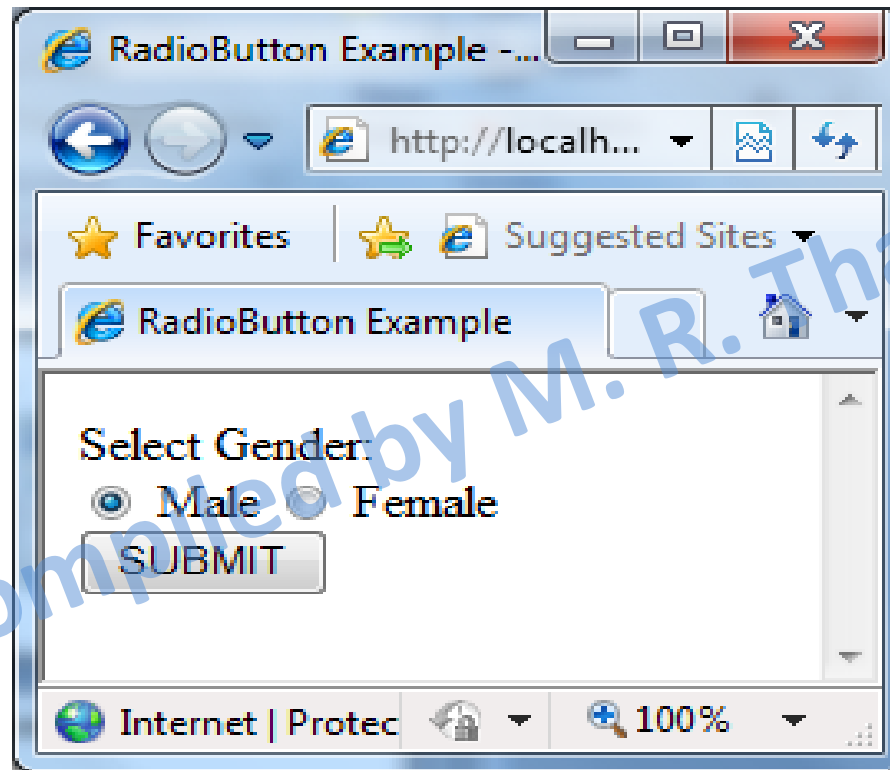
- Example:

Form1.php

```
<HTML>
<HEAD>
  <TITLE>RadioButton Example</TITLE>
</HEAD>
<BODY>
  <FORM method = "POST" action="Form2.php" >
    Select Gender:
    <BR>
    <INPUT TYPE="radio" NAME = "Gender" VALUE = "Male" CHECKED > Male </INPUT>
    <INPUT TYPE="radio" NAME = "Gender" VALUE = "Female" > Female </INPUT>
    <BR>
    <INPUT TYPE="SUBMIT" NAME="submit" VALUE="SUBMIT">
  </FORM>
</BODY>
</HTML>
```


4.1 Input through Form Controls

4.1.4 Radio Button



4.1 Input through Form Controls

4.1.5 Hidden Field : Radio Hidden field contains a value, but it is not visible on the form.

- **Syntax :** `<INPUT TYPE="hidden" NAME = "TextName"
VALUE = "DefaultValue" >`

Compiled by M. R. Thakkar

4.1 Input through Form Controls

4.1.5 Hidden Field

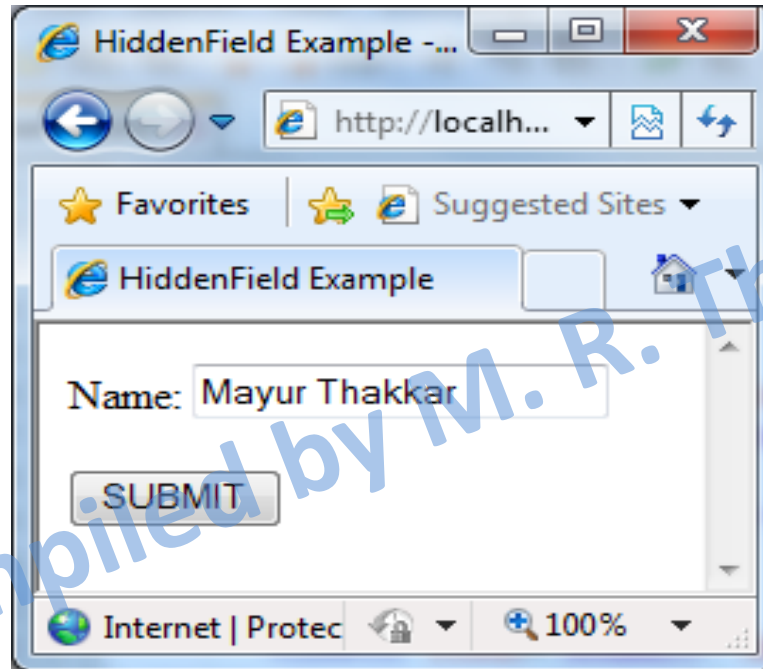
- Example:

Form1.php

```
<HTML>
<HEAD>
  <TITLE>HiddenField Example</TITLE>
</HEAD>
<BODY>
  <FORM method = "POST" action="Form2.php" >
    Name: <INPUT TYPE="text" NAME = "fname" >
    <BR>
      <INPUT TYPE="hidden" NAME = "mobile" VALUE = "9999999999" >
    <BR>
      <INPUT TYPE="SUBMIT" NAME="submit" VALUE="SUBMIT">
  </FORM>
</BODY>
</HTML>
```

4.1 Input through Form Controls

4.1.5 Hidden Field



4.1 Input through Form Controls

4.1.6 Submit Button : Every Form requires a Submit button.

- This is the element that causes the browser to send the names and values of the INPUT elements to the file specified by the ACTION attribute of the FORM element.

- Syntax :

<INPUT TYPE="Submit" NAME = "ButtonName" VALUE = "ButtonValue">

4.1 Input through Form Controls

4.1.6 Submit Button

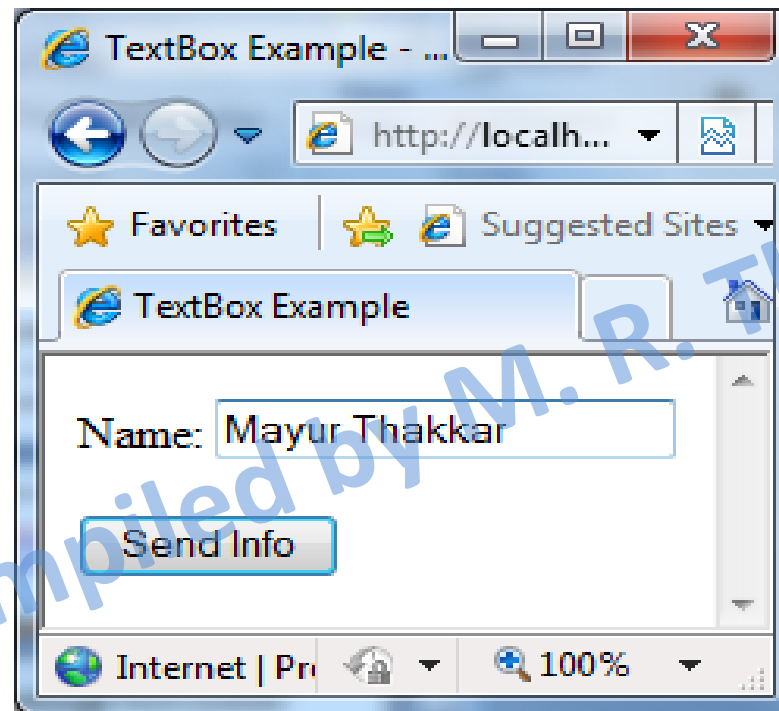
- Example:

Form1.php

```
<HTML>
<HEAD>
  <TITLE>Sample Form</TITLE>
</HEAD>
<BODY>
  <FORM method = "POST" action="Form2.php" >
    Name: <INPUT TYPE="TEXT" NAME="name" >
    <BR>
    <INPUT TYPE="SUBMIT" NAME="submit" VALUE="Send Info">
  </FORM>
</BODY>
</HTML>
```

4.1 Input through Form Controls

4.1.6 Submit Button



4.1 Input through Form Controls

4.1.7 TextArea : TextArea can be used to enter Multiple line of Text.

- It is useful for entering information like Address, Feedback etc.
- Syntax :

```
<TEXTAREA NAME="name" rows = "row_size"  
cols = "column_size" >  
</TEXTAREA>
```


4.1 Input through Form Controls

4.1.7 TextArea

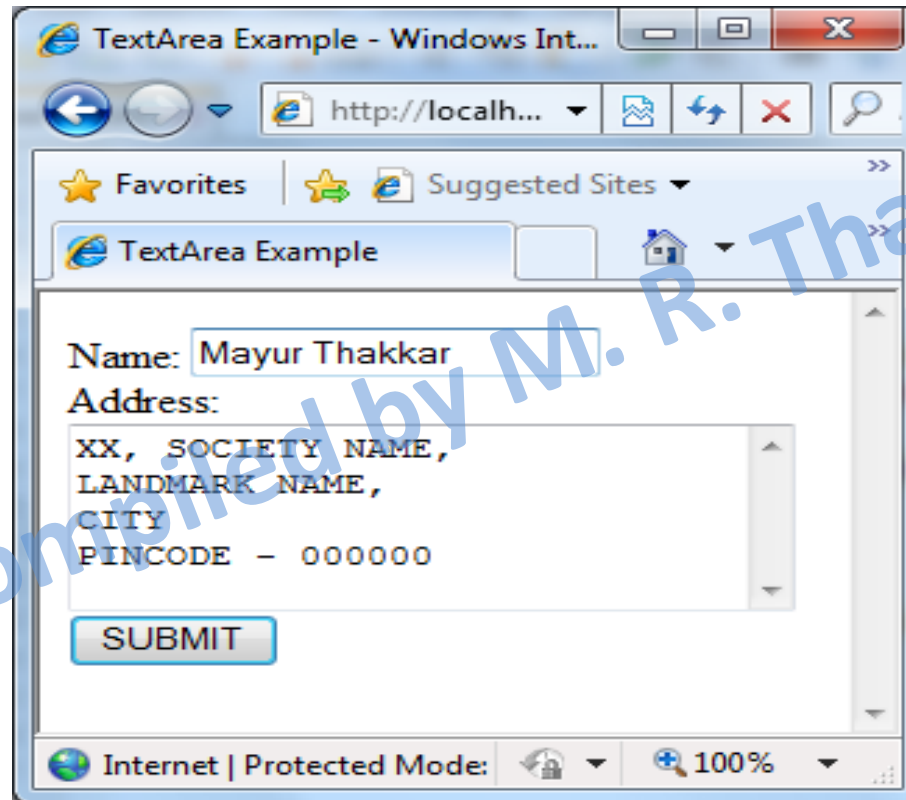
- Example:

Form1.php

```
<HTML>
<HEAD>
  <TITLE>TextArea Example</TITLE>
</HEAD>
<BODY>
  <FORM method = "POST" action="Form2.php" >
    Name: <INPUT TYPE="TEXT" NAME="name" >
    <BR>
    Address: <TEXTAREA NAME="Address" ROWS = "4" COLS = "30" >    </TEXTAREA>
    <BR>
    <INPUT TYPE="SUBMIT" NAME="submit" VALUE="SUBMIT">
  </FORM>
</BODY>
</HTML>
```

4.1 Input through Form Controls

4.1.7 TextArea



4.1 Input through Form Controls

4.1.8 List Box : <SELECT> element is used to display list of the options, from which any one option can be selected.

- **Syntax :** <SELECT NAME = “ListName“ >
 <OPTION> Value1 </ OPTION >
 <OPTION> Value2 </ OPTION >
 <OPTION> ValueN </ OPTION >
 </SELECT>

4.1 Input through Form Controls

4.1.8 List Box

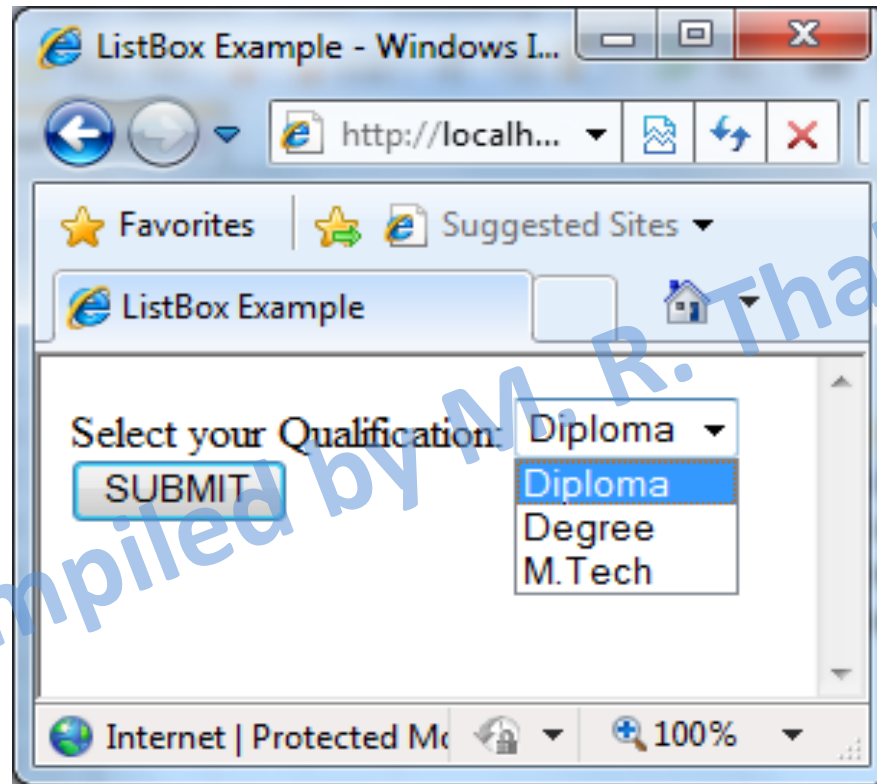
- Example:

Form1.php

```
<HTML>
<HEAD>
  <TITLE>HiddenField Example</TITLE>
</HEAD>
<BODY>
  <FORM method = "POST" action="Form2.php" >
    Select your Qualification: <SELECT NAME = "Qualification" >
      <OPTION> Diploma </ OPTION >
      <OPTION> Degree </ OPTION >
      <OPTION> M.Tech </ OPTION >
    </SELECT>
    <BR>
    <INPUT TYPE="SUBMIT" NAME="submit" VALUE="SUBMIT">
  </FORM>
</BODY>
</HTML>
```

4.1 Input through Form Controls

4.1.8 List Box



4.2 Submitting Form Values

- Using GET method

- When you are using GET method in the FORM element, the information will be send to the destination file through URL using the concept of Query String.
- A query string is collection of Name – Value pairs separated by & sign.
- Every query string starts with ? In the URL.
- When you are passing the information through URL, following difficulties are there:
- The information you are passing to the destination file, is visible in the URL. Thus it is not suitable for transferring secure information.
- You can transfer limited amount of information using this method.

4.2 Submitting Form Values

- Using GET method

- When you are passing the information through URL, following difficulties are there:
 - The information you are passing to the destination file, is visible in the URL. Thus it is not suitable for transferring secure information.
 - You can transfer limited amount of information using this method.

Compiled by M. R. Thakkar

4.2 Submitting Form Values

- Using GET method

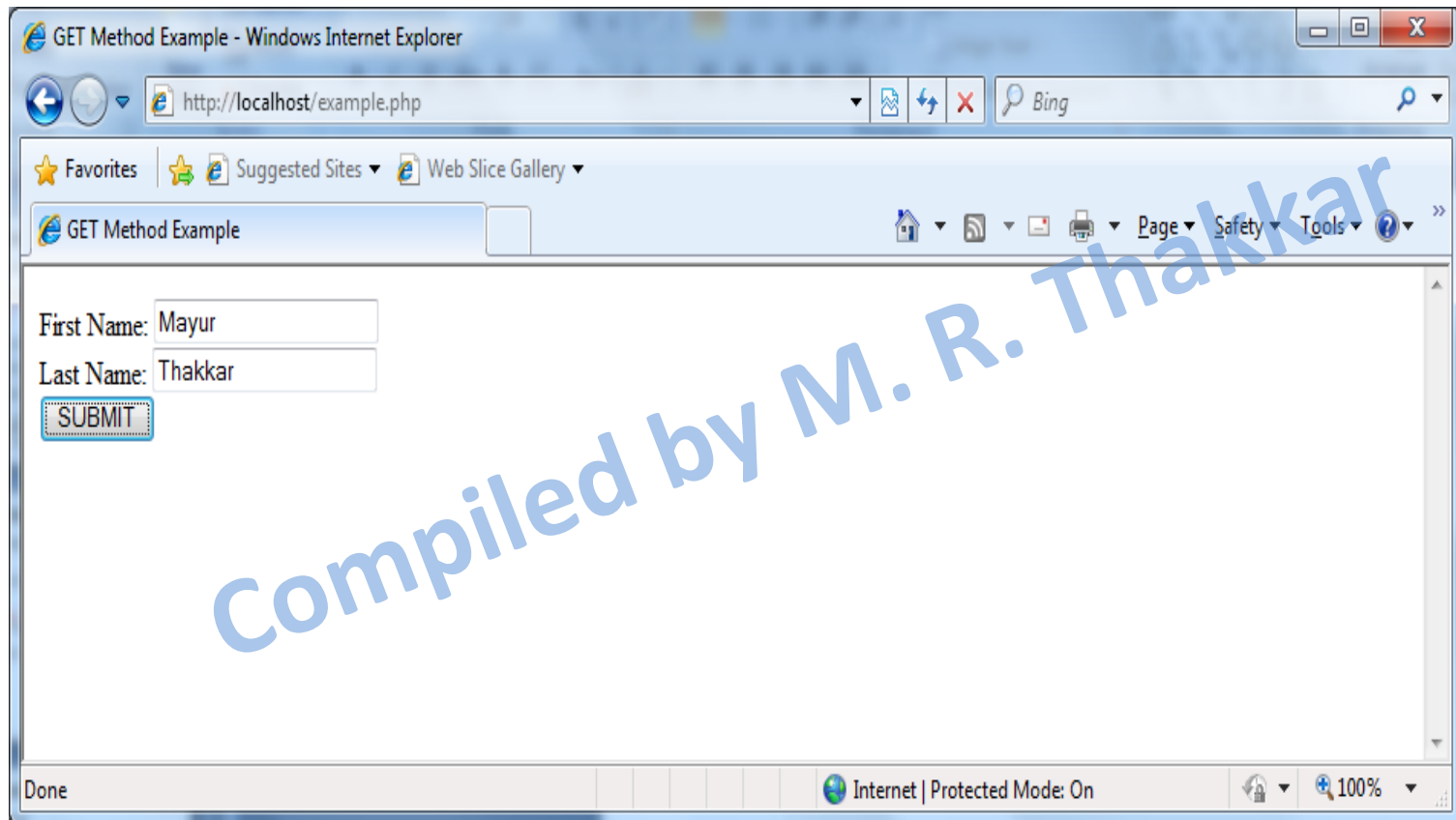
Form1.php

```
<HTML>
<HEAD>
  <TITLE> GET Method Example </TITLE>
</HEAD>
<BODY>
  <FORM method = "GET" action="Form2.php" >
    First Name: <INPUT TYPE="TEXT" NAME="fname" >
    <BR>
    Last Name: <INPUT TYPE="TEXT" NAME="lname" >
    <BR>

    <INPUT TYPE="SUBMIT" NAME="submit" VALUE="SUBMIT">
  </FORM>
</BODY>
</HTML>
```


4.2 Submitting Form Values

- Using GET method



4.2 Submitting Form Values

- Using GET method

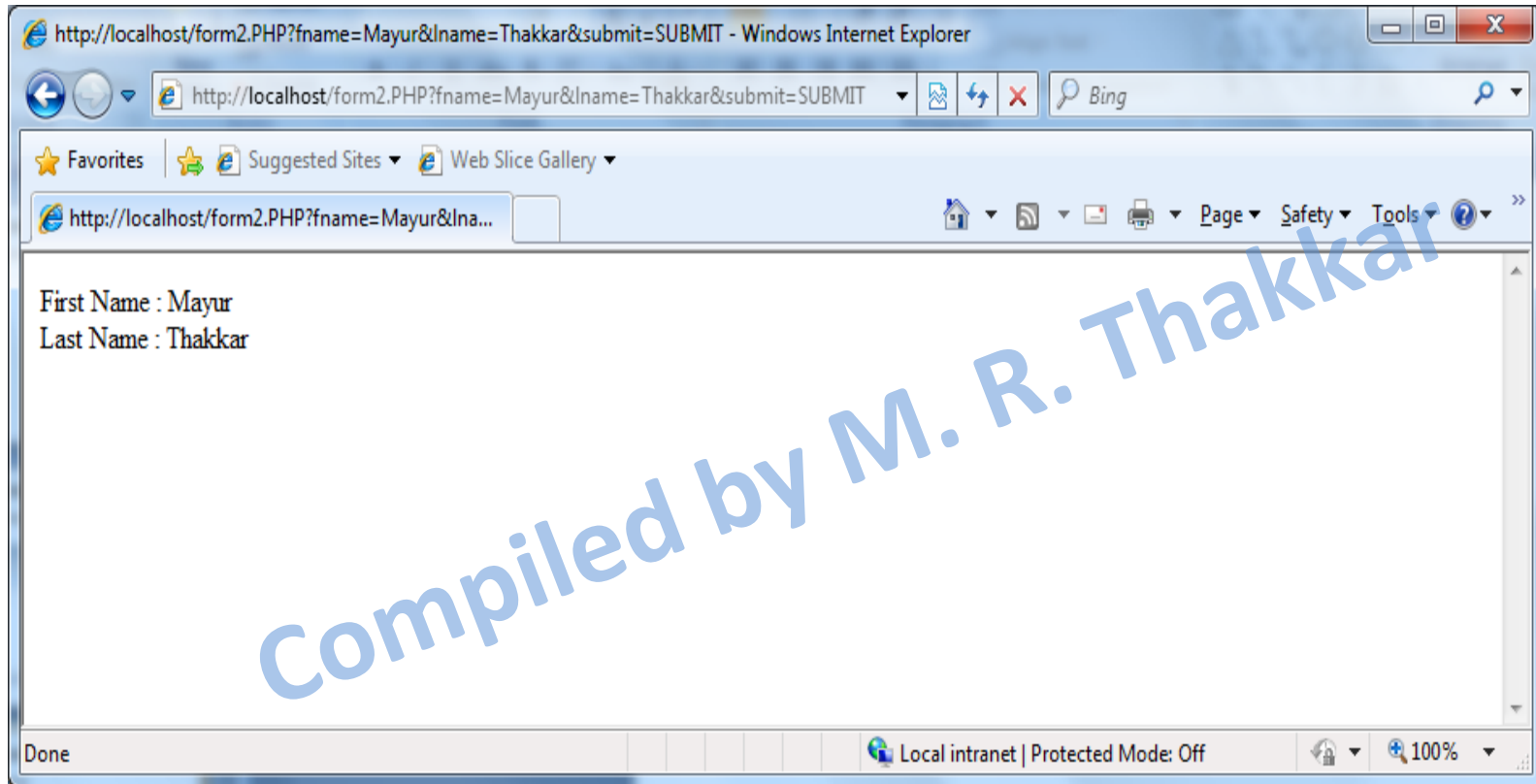
Form2.php

```
<?php  
  
echo "First Name : $_GET[fname] " . "<BR/>";  
echo "Last Name : $_GET[lname] " . "<BR/>";  
  
?>
```

Compiled by M. R. Thakkar

4.2 Submitting Form Values

- Using GET method



4.3 Submitting Form Values

- Using POST method

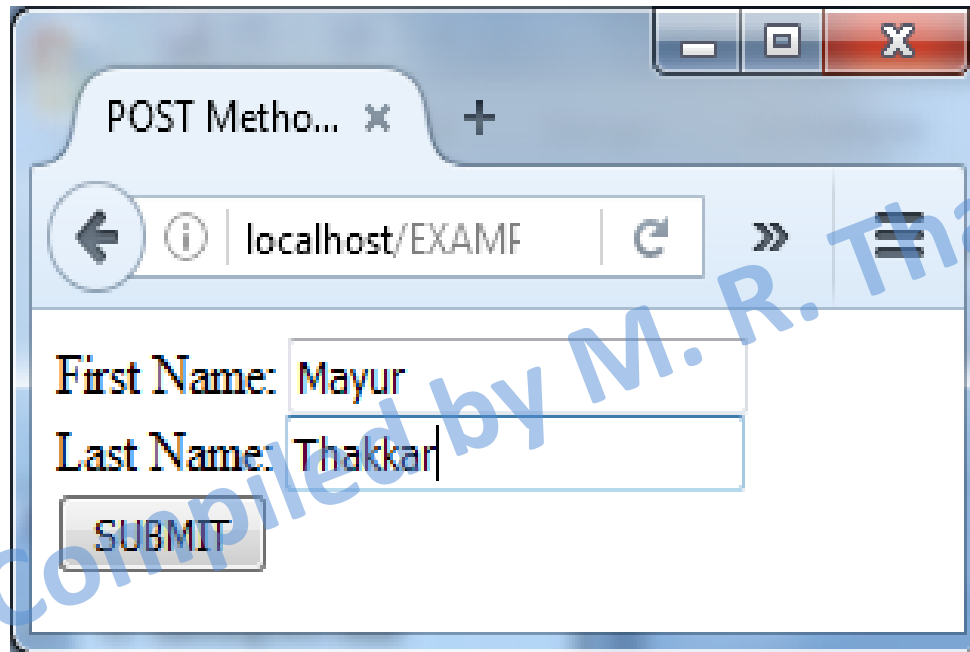
Form1.php

```
<HTML>
<HEAD>
  <TITLE> POST Method Example </TITLE>
</HEAD>
<BODY>
  <FORM method = "POST" action="Form2.php" >
    First Name: <INPUT TYPE="TEXT" NAME="fname" >
    <BR>
    Last Name: <INPUT TYPE="TEXT" NAME="lname" >
    <BR>

    <INPUT TYPE="SUBMIT" NAME="submit" VALUE="SUBMIT">
  </FORM>
</BODY>
</HTML>
```

4.3 Submitting Form Values

- Using POST method



A screenshot of a web browser window. The address bar shows 'localhost/EXAMF'. The page content includes a form with two text input fields: 'First Name: Mayur' and 'Last Name: Thakkar'. Below these fields is a 'SUBMIT' button. The browser window has a tab titled 'POST Metho...' and standard window controls (minimize, maximize, close). A large, diagonal watermark reading 'Compiled by M. R. Thakkar' is overlaid across the entire image.

4.3 Submitting Form Values

- Using POST method

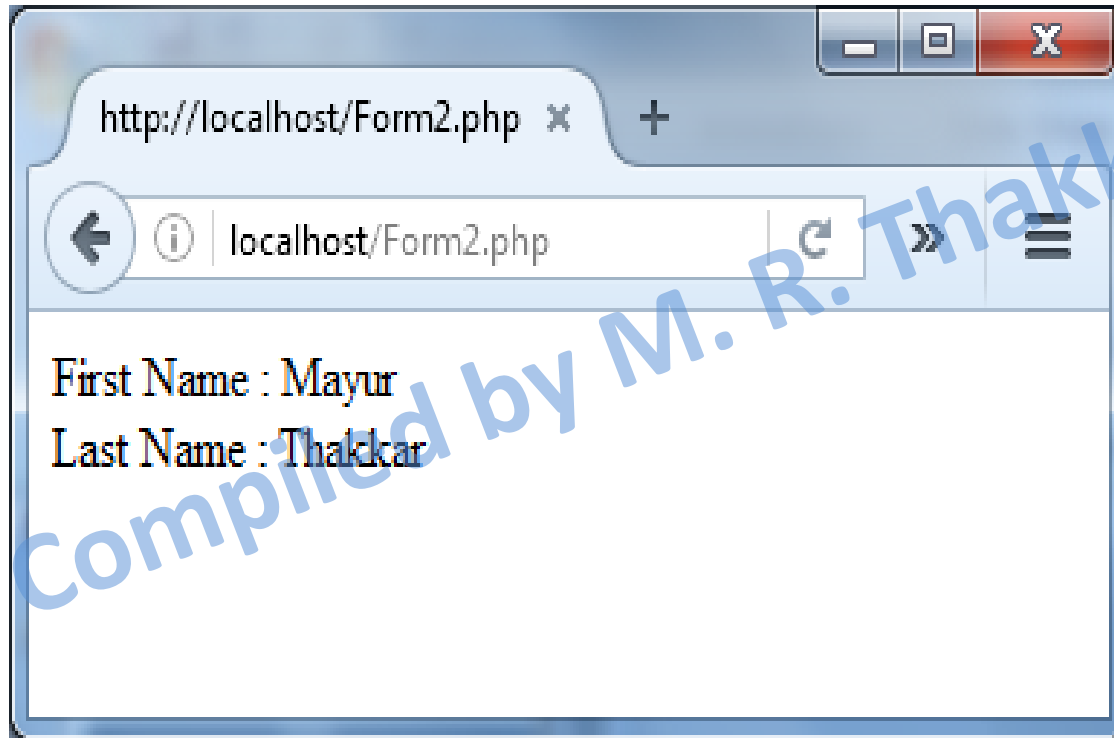
Form2.php

```
<?php  
  
echo "First Name : $_POST[fname] " . "<BR/>";  
echo "Last Name : $_POST[lname] " . "<BR/>";  
  
?>
```

Compiled by M. R. Thakkar

4.3 Submitting Form Values

- Using POST method



- **Difference Between GET/ POST method**

GET Method	POST Method
Using GET Method data is sent from one page to other in the URL	Using POST Method data is sent from one page to other within the body of the HTTP request
The GET method, appends name/value pairs to the URL.	POST method packages the name/value pairs inside the body of the HTTP request, which makes for a cleaner URL.
Unfortunately, the length of a URL is limited, so GET method only works if there are only a few parameters. The GET method is restricted to send at most 1024 characters only.	POST method imposes no size limitations on the forms output.

- **Difference Between GET/ POST method**

GET Method	POST Method
Using GET method is insecure because parameters passed on the URL are visible in the address field of the browser.	Using POST method is more secure because no data is visible in the URL.
GET method can't be used to send binary data, like images or word documents, to the server.	The POST method can be used to send ASCII as well as binary data.
The data sent by GET method can be accessed using \$_GET superglobal variable.	The data sent by POST method can be accessed using \$_POST superglobal variable.

4.4 Combining HTML & PHP code together

- In order to develop an application that accepts input from user, process that input and display the output, we generally need to design two separate files.
- One file that contains code to design FORM that accepts input from user and another file that contains code to process the input.
- It is also possible to combine FORM designing code and FORM processing code on single file.

4.4 Combining HTML & PHP code together

- Form.php

```
<?php
```

```
if(!isset($_POST['fname']))
{
    $msg = "";
}
else
{
    $name = $_POST['fname'];
    $msg="Welcome: " . $name;
}

?>
```

4.4 Combining HTML & PHP code together

```
<HTML>
<HEAD>
    <TITLE>Form Example</TITLE>
</HEAD>
<BODY>

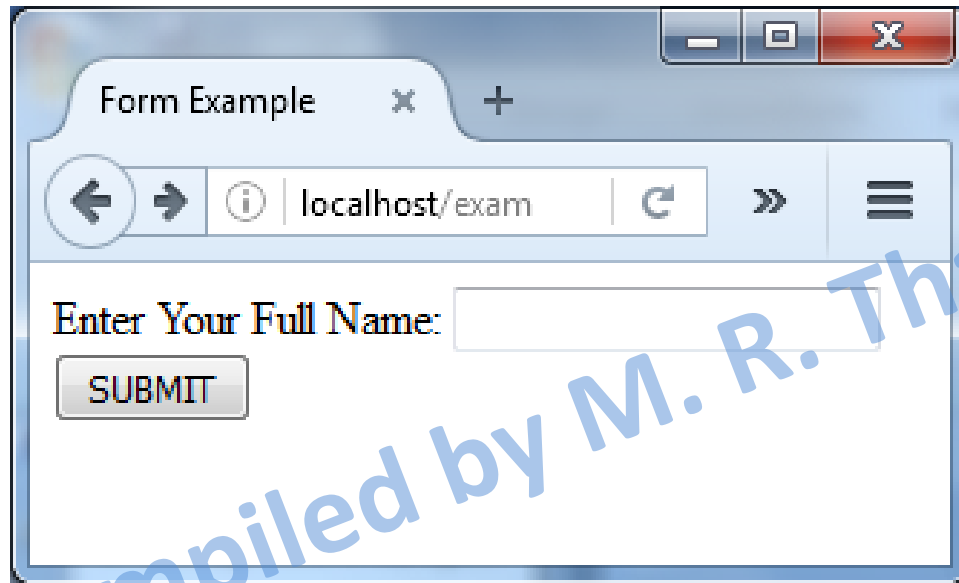
<FORM method = "POST" action="<?php echo $_SERVER["PHP_SELF"]; ?>" >
    Enter Your Full Name: <INPUT TYPE="TEXT" NAME="fname" >
    <BR>

    <INPUT TYPE="SUBMIT" NAME="submit" VALUE="SUBMIT">
</FORM>

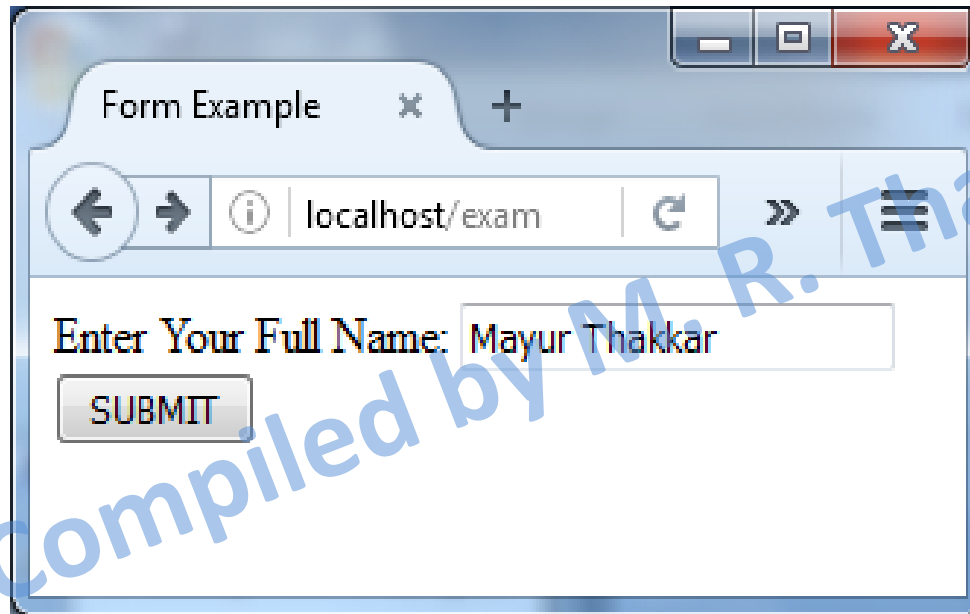
<?php echo $msg; ?>

</BODY>
</HTML>
```

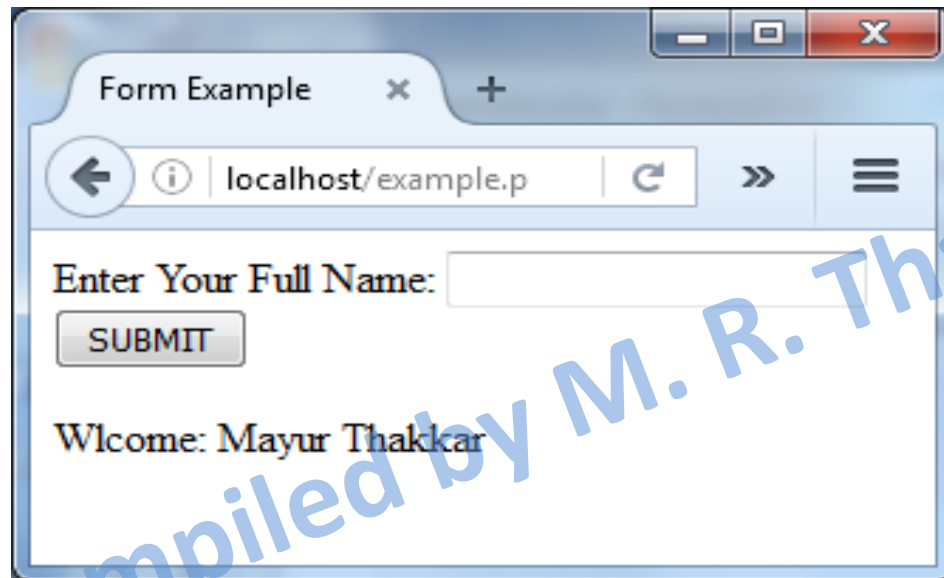
4.4 Combining HTML & PHP code together



4.4 Combining HTML & PHP code together



4.4 Combining HTML & PHP code together



A screenshot of a web browser window titled "Form Example". The address bar shows "localhost/example.p". The page content includes a form with the label "Enter Your Full Name:" followed by a text input field. Below the input field is a "SUBMIT" button. Below the button, the text "Welcome: Mayur Thakkar" is displayed. A large, diagonal watermark reading "Compiled by M. R. Thakkar" is overlaid across the entire browser window.

4.5 Basics of Cookies, Using Cookies and maintaining Session

- A cookie is a small piece of information that is stored on the client computer, either in the browser's memory or in a file on the hard disk.
- A cookie having name and value.
- A cookie is useful for storing user specific information such as username, password, last visit etc.

Compiled by M. R. Thakkar

4.5 Basics of Cookies

- A cookie is a small piece of information that is stored on the client computer, either in the browser's memory or in a file on the hard disk.
- A cookie having name and value.
- A cookie is useful for storing user specific information such as username, password, last visit etc.

Compiled by M. R. Thakkar

4.6 Using Cookie Variable, Using Cookies with Authentication

- **Creating Cookie**

- In PHP, you can create cookie using setcookie() function.

- **Syntax:** Setcookie ('Name', Value, Expire Time)

- Name: is the name of the cookie that you want to create.
- Value: is the value that is associated with the cookie created by you.
- Expire Time: is the time when the cookie will expire and deleted.

It is optional , if you don't provide the ExpireTime then it will be removed when the browser is closed.

4.6 Using Cookie Variable, Using Cookies with Authentication

- Creating Cookie

- Example:

- $\$ExpireTime = 60 * 2 + time();$
Setcookie ('user', 'Mayur Thakkar', \$ExpireTime)
 - It will create a cookie with name: user , value: Mayur Thakkar.
 - It will set expire time of 2 minutes.

4.6 Using Cookie Variable, Using Cookies with Authentication

- **Retriving Cookie**
- You can retrieve cookies in your PHP script using following statement:
- **Syntax:** `$_COOKIE['cookie_name']`
- **Example:** `$_COOKIE['user']`

4.6 Using Cookie Variable, Using Cookies with Authentication

- **Deleting Cookie**

- You can delete the cookie using the `setcookie()` function.
- You can use `setcookie()` function with empty value for cookie name to delete the cookie.
- **Syntax:** `setcookie('cookie_name' , "");`
- **Example:** `setcookie('user' , "");`

4.6 Using Cookie Variable, Using Cookies with Authentication

- Using Cookie With Authentication

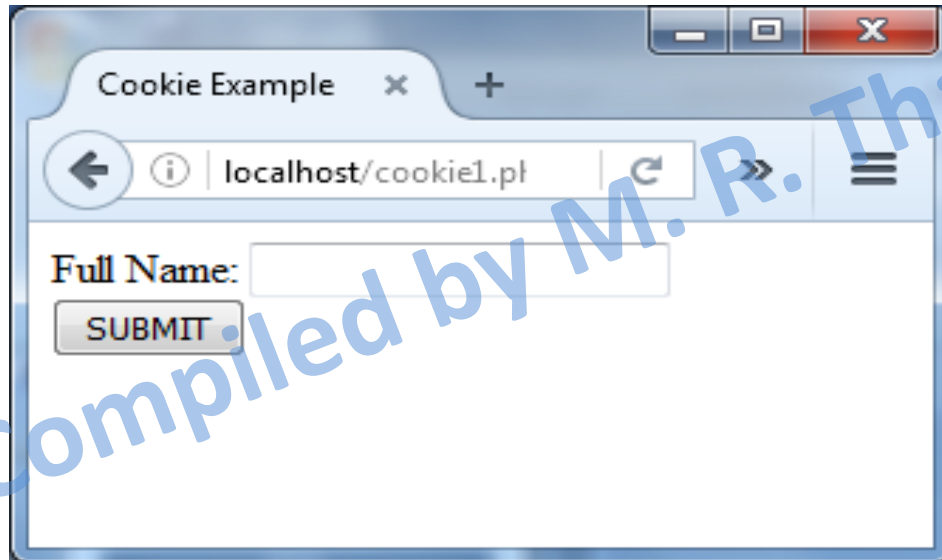
Cookie1.php

```
<HTML>
<HEAD>
  <TITLE> Cookie Example </TITLE>
</HEAD>
<BODY>
  <FORM method = "POST" action="cookie2.php" >
    Full Name: <INPUT TYPE="TEXT" NAME="fname" >
    <BR>

    <INPUT TYPE="SUBMIT" NAME="submit" VALUE="SUBMIT">
  </FORM>
</BODY>
</HTML>
```

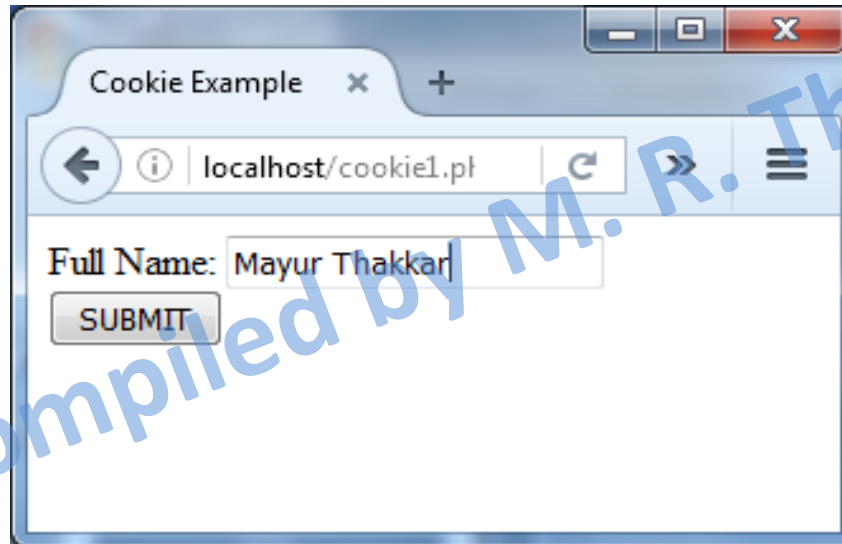
4.6 Using Cookie Variable, Using Cookies with Authentication

- Using Cookie With Authentication



4.6 Using Cookie Variable, Using Cookies with Authentication

- Using Cookie With Authentication



4.6 Using Cookie Variable, Using Cookies with Authentication

- Using Cookie With Authentication

Cookie2.php

```
<?php

$name = $_POST['fname'];
$ExpireTime = 60 * 2 + time();

setcookie('user',$name,$ExpireTime);

header("location: cookie3.php");

?>
```

4.6 Using Cookie Variable, Using Cookies with Authentication

- Using Cookie With Authentication

Cookie3.php

```
<?php
```

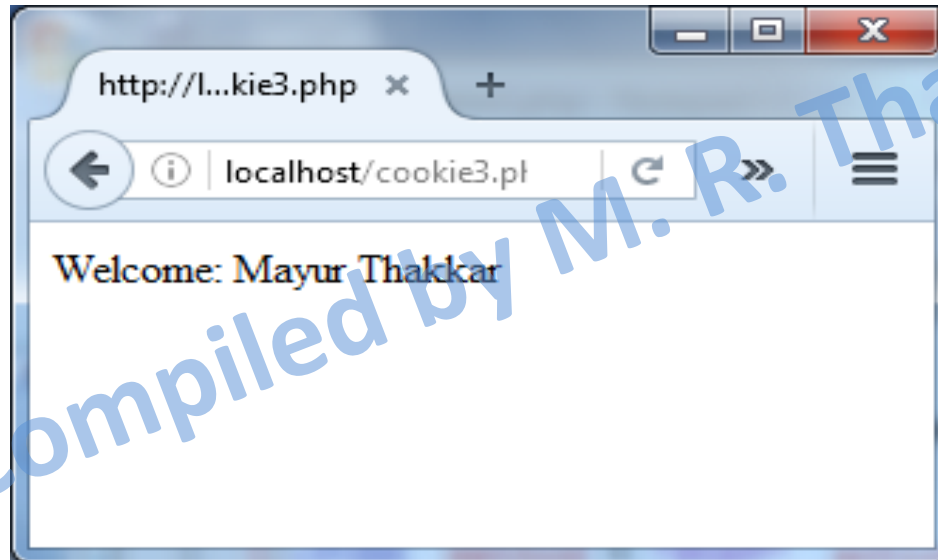
```
    echo "Welcome: ". $_COOKIE["user"];
```

```
?>
```

Compiled by M. R. Thakkar

4.6 Using Cookie Variable, Using Cookies with Authentication

- Using Cookie With Authentication



4.7 Understanding Session & Session Variable

- A session variable is a temporary variable that is created and stored at server side to uniquely identify each user on website.
- A session is created for each user when he or she logs in to the web application and it remain in existence until he or she logs out.

Compiled by M. R. Thakkar

4.8 Starting a Session , Registering and Modifying a session variable

- **Starting Session**

- In order to create of access session variable, you need to start the session at the starting of script using session_start() function.
- As you start session using session_start() function, a unique identification number UID is generated for that user to uniquely identify the user.

Compiled by M. R. Thakker

4.8 Starting a Session , Registering and Modifying a session variable

- **Creating Session Variable:**

- A session variable can be created using Following Syntax:

- **Syntax:** `$_SESSION['VariableName'] = Value`

- **Example:** `$_SESSION['user'] = "Mayur Thakkar"`

4.8 Starting a Session , Registering and Modifying a session variable

- **Destroying Session:**

- A session can be destroyed using session_destroy() function.

- **Example:** session_destroy()

Compiled by M. R. Thakkar

4.8 Starting a Session , Registering and Modifying a session variable

- Using Session With Authentication

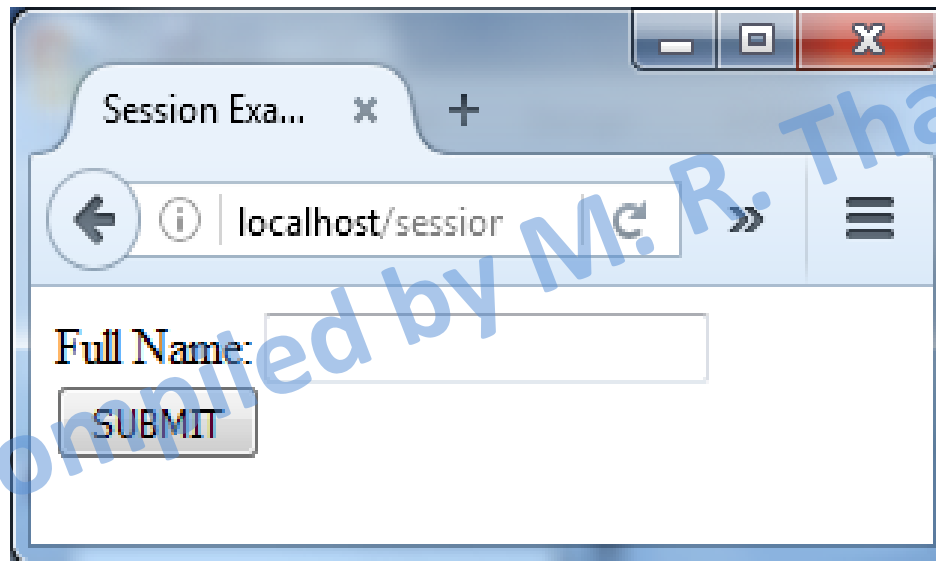
session1.php

```
<HTML>
<HEAD>
  <TITLE> Session Example </TITLE>
</HEAD>
<BODY>
  <FORM method = "POST" action="session2.php" >
    Full Name: <INPUT TYPE="TEXT" NAME="fname" >
    <BR>

    <INPUT TYPE="SUBMIT" NAME="submit" VALUE="SUBMIT">
  </FORM>
</BODY>
</HTML>
```

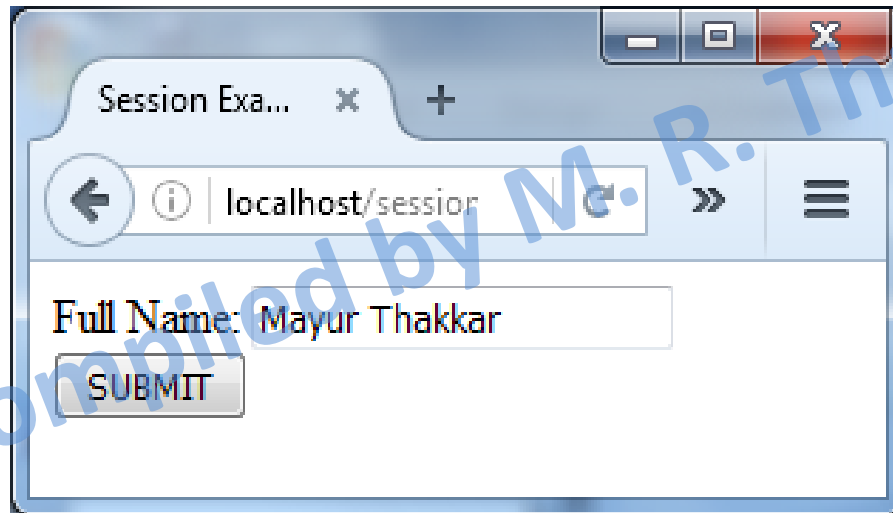

4.8 Starting a Session , Registering and Modifying a session variable

- Using Session With Authentication



4.8 Starting a Session , Registering and Modifying a session variable

- Using Session With Authentication



4.8 Starting a Session , Registering and Modifying a session variable

- Using Session With Authentication

session2.php

```
<?php  
  
    session_start();  
  
    $name = $_POST['fname'];  
    $_SESSION['user'] = $name;  
  
    header("location: session3.php");  
?>
```

4.8 Starting a Session , Registering and Modifying a session variable

- Using Session With Authentication

session3.php

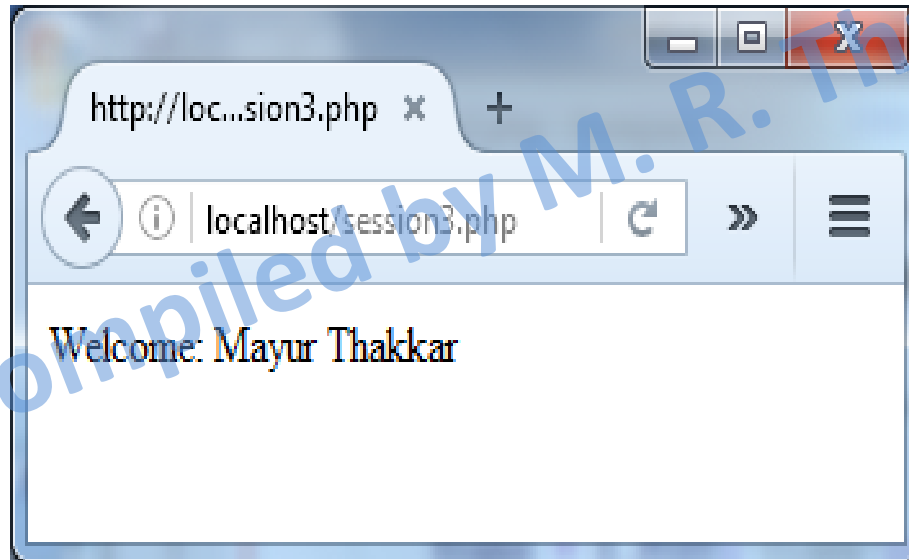
```
<?php
    session_start();

    echo "Welcome: ". $_SESSION['user'];

    session_destroy();
?>
```

4.8 Starting a Session , Registering and Modifying a session variable

- Using Session With Authentication



4.9 Managing user preference with session

- You'll find the following Session related preferences in your **application/config/config.php** file:

Preference	Default	Options	Description
sess_cookie_name	ci_session	None	The name you want the session cookie saved as.
sess_expiration	7200	None	The number of seconds you would like the session to last. The default value is 2 hours (7200 seconds). If you would like a non-expiring session set the value to zero: 0
sess_expire_on_close	FALSE	TRUE/FALSE (boolean)	Whether to cause the session to expire automatically when the browser window is closed.
sess_encrypt_cookie	FALSE	TRUE/FALSE (boolean)	Whether to encrypt the session data.
sess_use_database	FALSE	TRUE/FALSE (boolean)	Whether to save the session data to a database. You must create the table before enabling this option.

4.9 Managing user preference with session

Preference	Default	Options	Description
sess_table_name	ci_sessions	Any valid SQL table name	The name of the session database table.
sess_time_to_update	300	Time in seconds	This options controls how often the session class will regenerate itself and create a new session id.
sess_match_ip	FALSE	TRUE/FALSE (boolean)	Whether to match the user's IP address when reading the session data. Note that some ISPs dynamically changes the IP, so if you want a non-expiring session you will likely set this to FALSE.
sess_match_useragent	TRUE	TRUE/FALSE (boolean)	Whether to match the User Agent when reading the session data.

Compiled by M. R. Thakkar