Cookies, Sessions, and Authentication

What is a Cookie

- A cookie is often used to identify a user.
- A cookie is a small file that the server embeds on the user's computer.
- Each time the same computer requests a page with a browser, it will send the cookie too.
- With PHP, you can both create and retrieve cookie values.

How to Create a Cookie? - Syntax

- The setcookie() function is used to set a cookie.

- Note: The setcookie() function must appear BEFORE the <html> tag.

```
setcookie(Name, value, expire, path, domin)
```

How to Create a Cookie? - Example

- In the example below, we will create a cookie named "user“ and assign the value "Alex Porter" to it.
- We also specify that the cookie should expire after one hour:

```
<?php
Setcookie(“user”, “Alex Porter”, time()+3600);
?>
<html>
<body>
</body>
</html>
```

- Note: The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use setrawcookie() instead).

How to Retrieve a Cookie Value?

- The PHP $_COOKIE variable is used to retrieve a cookie value.
- In the example below, we retrieve the value of the cookie named "user" and display it on a page:

```
<?php
//print a cookie
Echo  $_COOKIE[“user”];
//a way to view all cookies
Print_r($_COOKIE);
?>
```

How to Retrieve a Cookie Value?

- In the following example we use the isset() function to find out if a cookie has been set:

```
<html>
<body>
<?php
if (isset($_COOKIE[“user”]))
      echo “Welcome”.$_COOKIE[“user”].”!<br/>”;
Else
      echo “Welcome guest!<br/>;
?>
</body>
</html>
```

How to Delete a Cookie?

- When deleting a cookie you should assure that the expiration date is in the past.
- Delete example:

```
<?php
//set the expiration date to one hour ago
Setcookie("user","",time()-3600);
?>
```

What is a PHP Session

- A session is a way to store information (in variables) to be used across multiple pages.

- Unlike a cookie, the information is not stored on the users computer.

PHP Session Variables

- When you are working with an application, you open it, do some changes and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are and what you do because the HTTP address doesn't maintain state.

PHP Session Variables

- A PHP session solves this problem by allowing you to store user information on the server for later use (i.e. username, shopping items, etc). However, session information is temporary and will be deleted after the user has left the website. If you need a permanent storage you may want to store the data in a database.
- Sessions work by creating a unique id (UID) for each visitor and store variables based on this UID. The UID is either stored in a cookie or is propagated in the URL.

Starting a PHP Session

- Before you can store user information in your PHP session, you must first start up the session.
- Note: The session_start() function must appear BEFORE the <html> tag:

```
<?php session_start();?>
<html>
<body>
</body>
</html>
```

- The code above will register the user's session with the server, allow you to start saving user information, and assign a UID for that user's session

Storing a Session Variable

- The correct way to store and retrieve session variables is to use the PHP $_SESSION variable:

```
<?php
session_start();
// store session data
$_SESSION['views']=1;
?>

<html>
<body>

<?php
//retrieve session data
echo "Pageviews=". $_SESSION['views'];
?>

</body>
</html>
```

- Output:

```
Pageviews=1
```

Storing a Session Variable

- In the example below, we create a simple page-views counter.
- The isset() function checks if the "views" variable has already been set. If "views" has been set, we can increment our counter. If "views" doesn't exist, we create a "views“ variable, and set it to 1:

```
<?php

session_start();

if(isset($_SESSION['views']))
  $_SESSION['views']=$_SESSION['views']+1;

else
  $_SESSION['views']=1;
echo "Views=". $_SESSION['views'];
?>
```

Destroying a Session

- If you wish to delete some session data, you can use the unset() or the session_destroy() function.
- The unset() function is used to free the specified session variable:

```
<?php
unset($_SESSION['views']);
?>
```

Destroying a Session

- You can also completely destroy the session by calling the session_destroy() function:

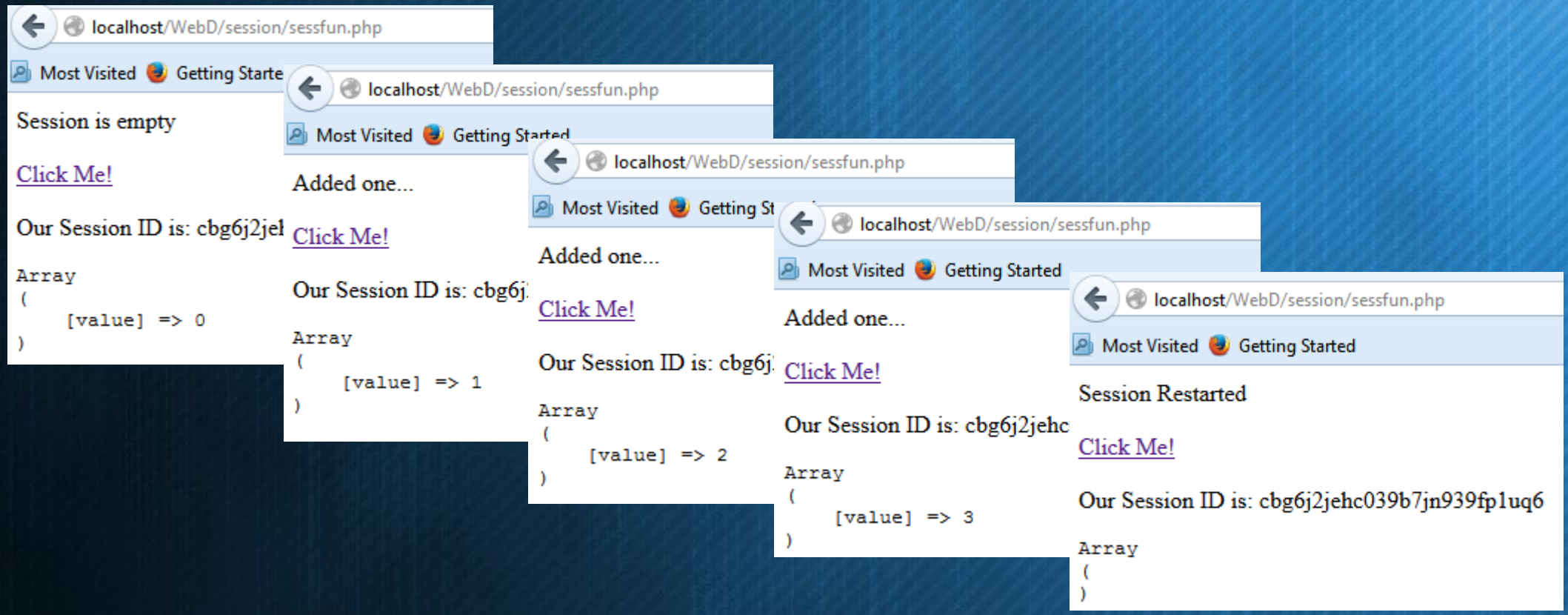
```
<?php
session_destroy();
?>
```

- Note: session_destroy() will reset your session and you will lose all your stored session data.

Session Fun

```
<?php
  // Note - cannot have any output before this
  session_start();
  if ( ! isset($_SESSION['value']) )
  {echo("<p>Session is empty</p>\n");
  $_SESSION['value'] = 0;
} else if ( $_SESSION['value'] < 3 )
{ $_SESSION['value'] = $_SESSION['value'] + 1;
      echo("<p>Added one...</p>\n");
  } else {session_destroy();
      session_start();
      echo("<p>Session Restarted</p>\n");  }  ?>
  <p><a href="sessfun.php">Click Me!</a></p>
  <p>Our Session ID is: <?php echo(session_id()); ?></p>
  <pre><?php print_r($_SESSION); ?> </pre>
```

Session Fun


localhost/WebD/session/sessfun.php
Most Visited
Getting Started
Session is empty
Click Me!
Our Session ID is: cbg6j2jeh
Array
(
[value] => 0
)
localhost/WebD/session/sessfun.php
Most Visited
Getting Started
Added one...
Click Me!
Our Session ID is: cbg6j
Array
(
[value] => 1
)
localhost/WebD/session/sessfun.php
Most Visited
Getting St
Added one...
Click Me!
Our Session ID is: cbg6j
Array
(
[value] => 2
)
localhost/WebD/session/sessfun.php
Most Visited
Getting Started
Added one...
Click Me!
Our Session ID is: cbg6j2jehc
Array
(
[value] => 3
)
localhost/WebD/session/sessfun.php
Most Visited
Getting Started
Session Restarted
Click Me!
Our Session ID is: cbg6j2jehc039b7jn939fp1uq6
Array
(
)

Login / Logout

- Having a session is not the same as being logged in.
- Generally you have a session the instant you connect to a web site
- The Session ID cookie is set when the first page is delivered
- Login puts user information in the session (stored in the server)
- Logout removes user information from the session

```
<?php    session_start();
unset($_SESSION["account"]);
if ( isset($_POST["account"]) && isset($_POST["pw"]) )
      {    if ( $_POST['pw'] == 'dt211' )
             {      $_SESSION["account"] = $_POST["account"];
                    $_SESSION["success"] = "Logged in.";
                    header( 'Location: index.php' ) ;
                    return;
             } else {
                    $_SESSION["error"] = "Incorrect password.";
                    header( 'Location: login.php' ) ;
                    return;
}    } else if ( count($_POST) > 0 )
      {      $_SESSION["error"] = "Missing Required Information";
             header( 'Location: login.php' ) ;
             return;
      }
?>
<
<html>
…
```

login.php

```
<html>
<head>
</head>
<body style="font-family: sans-serif;">
<h1>Please Log In</h1>
<?php
      if ( isset($_SESSION["error"]) ) {
      echo('<p style="color:red">Error:'.
      $_SESSION["error"]."</p>\n");
      unset($_SESSION["error"]);
      }
?>
<form method="post">
<p>Account: <input type="text" name="account" value=""></p>
<p>Password: <input type="text" name="pw" value=""></p>
<p><input type="submit" value="Log In"></p>
</form>
</body>
</html>
```

login.php

HTTP Location Header

- If your application has not yet sent any data, it can send a special header as part of the HTTP Response
- The redirect header includes a URL that the browser is supposed to forward itself to
- It was originally used for web sites that moved from one URL to another

HTTP Location Header

header

(PHP 4, PHP 5)

header — Send a raw HTTP header

Description

Report a bug

void **header** (string *$string* [, bool *$replace* = *true* [, int *$http_response_code*]])

header() is used to send a raw HTTP header. See the » HTTP/1.1 specification for more information on HTTP headers.

Remember that **header()** must be called before any actual output is sent, either by normal HTML tags, blank lines in a file, or from PHP. It is a very common error to read code with include, or require, functions, or another file access function, and have spaces or empty lines that are output before **header()** is called. The same problem exists when using a single PHP/HTML file.

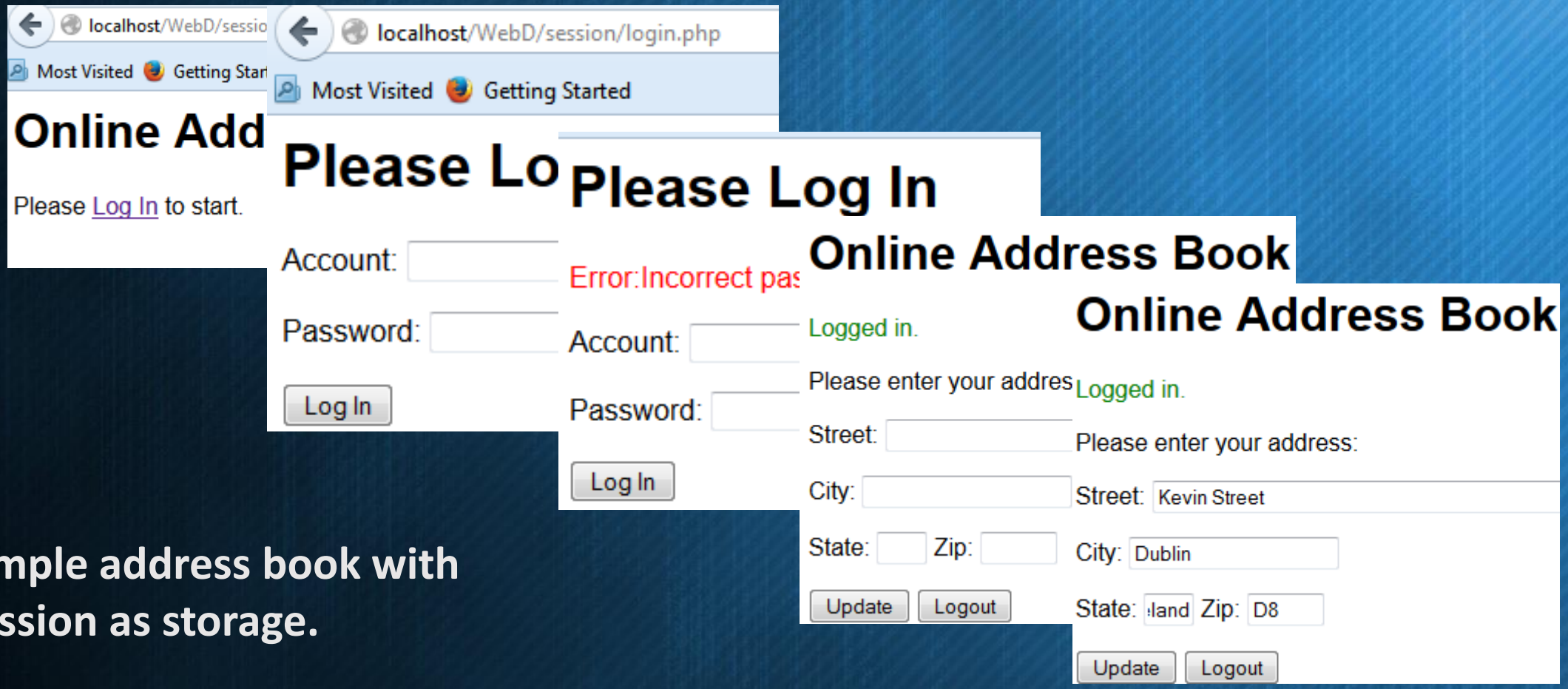
```
<html>
<?php
/* This will give an error. Note the output
 * above, which is before the header() call */
header('Location: http://www.example.com/');
?>
```

Login / Logout

```php
<?php
session_start();
session_destroy();
header("Location: login.php");
```

logout.php

Address Book Example



Simple address book with session as storage.

```
?>
<html>
<head>
</head>
<body style="font-family: sans-serif;">
<h1>Online Address Book</h1>
<?php
if ( isset($_SESSION["error"]) ) {
echo('<p style="color:red">Error:'.$_SESSION["error"]."</p>\n");
unset($_SESSION["error"]);
}
if ( isset($_SESSION["success"]) ) {
echo('<p style="color:green">'.$_SESSION["success"]."</p>\n");
unset($_SESSION["success"]);
}
// Retrieve data from the session for the view
$street = isset($_SESSION['street']) ? $_SESSION['street'] : '';
$city = isset($_SESSION['city']) ? $_SESSION['city'] : '';
$state = isset($_SESSION['state']) ? $_SESSION['state'] : '';
$zip = isset($_SESSION['zip']) ? $_SESSION['zip'] : '';
```

index.php

```
if ( ! isset($_SESSION["account"]) ) { ?>
Please <a href="login.php">Log In</a> to start.
<?php } else { ?>
<p>Please enter your address:
<form method="post">
<p>Street: <input type="text" name="street" size="50"
value="<?php echo(htmlentities($street)); ?> "></p>
<p>City: <input type="text" name="city" size="20"
value="<?php echo(htmlentities($city)); ?>
"></p>
<p>State: <input type="text" name="state" size="2"
value="<?php echo(htmlentities($state)); ?> ">
Zip: <input type="text" name="zip" size="5"
value="<?php echo(htmlentities($zip)); ?> "></p>
<p><input type="submit" value="Update">
<input type="button" value="Logout"
onclick="location.href='logout.php'; return false "></p>
</form>
<?php } ?>
</body>
```

index.php

```php
<?php
session_start();
if ( isset($_POST["street"]) && isset($_POST["city"]) &&
isset($_POST["state"]) && isset($_POST["zip"]) ) {
$_SESSION['street'] = $_POST['street'];
$_SESSION['city'] = $_POST['city'];
$_SESSION['state'] = $_POST['state'];
$_SESSION['zip'] = $_POST['zip'];
header( 'Location: index.php' ) ;
return;
} else if ( count($_POST) > 0 ) {
$_SESSION["error"] = "Missing Required Information";
header( 'Location: index.php' ) ;
return;
}
?>
<html>
```

index.php

PHP: ODBC and Error Processing

Create an ODBC Connection

- ODBC is an Application Programming Interface (API) that allows you to connect to a data source

- With an ODBC connection, you can connect to any database, on any computer in your network, as long as an ODBC connection is available

Create an ODBC Connection

- Here is how to create an ODBC connection to a MS Access Database:
  - Open the Administrative Tools icon in your Control Panel.
  - Double-click on the Data Sources (ODBC) icon inside.
  - Choose the System DSN tab.
  - Click on Add in the System DSN tab.
  - Select the Microsoft Access Driver. Click Finish.
  - In the next screen, click Select to locate the database.
  - Give the database a Data Source Name (DSN).
  - Click OK.

Create an ODBC Connection

- But if you are using 64 bit install Window 10, and you can't find Access driver there, here is how to create an ODBC connection to a MS Access Database:
  - Use the ODBC control panel here: c:\windows\ODBC Data Sources(64-bit)
  - Choose the User DSN tab.
  - Click on Add in the User DSN tab.
  - Select the Microsoft Access Driver. Click Finish.
  - In the next screen, click Select to locate the database.
  - Give the database a Data Source Name (DSN).
  - Click OK.

Connecting to an ODBC

- The odbc_connect() function is used to connect to an ODBC data source. The function takes four parameters: the data source name, username, password, and an optional cursor type.

- The odbc_exec() function is used to execute an SQL statement.

Connecting to an ODBC

- The following example creates a connection to a DSN called userodbc, with no username and no password. It then creates an SQL and executes it:

```
$conn=odbc_connect('userodbc','','');
$sql="SELECT * FROM customers";
$rs=odbc_exec($conn,$sql);
```

- The odbc_fetch_row() function is used to return records from the result-set. This function returns true if it is able to return rows, otherwise false.
- The function takes two parameters: the ODBC result identifier and an optional row number:

```
odbc_fetch_row($rs)
```

Retrieving Fields from a Record

- The odbc_result() function is used to read fields from a record. This function takes two parameters: the ODBC result identifier and a field number or name.
- The code line below returns the value of the second field from the record:

```
$username=odbc_result($rs,2);
```

- The code line below returns the value of a field called “UserName":

```
$username=odbc_result($rs,“Username");
```

- The odbc_close() function is used to close an ODBC connection:

```
odbc_close($conn);
```

An ODBC Example

- The following example shows how to first create a database connection, then a result-set, and then display the data in an HTML table.

```
<html>
<body>
<?php
$conn=odbc_connect('userodbc','','');
if (!$conn)
  {exit("Connection Failed: " . $conn);}
$sql="SELECT * FROM user";
$rs=odbc_exec($conn,$sql);
if (!$rs)
  {exit("Error in SQL");}
echo "<table><tr>";
echo "<th>Username</th>";
echo "<th>Age</th></tr>";
while (odbc_fetch_row($rs))
  {
  $username=odbc_result($rs,"UserName");
  $age=odbc_result($rs,"Age");
  echo "<tr><td>$username</td>";
  echo "<td>$age</td></tr>";
  }
odbc_close($conn);
echo "</table>";
?>
</body>
</html>
```

PHP Error Handling

- When creating scripts and web applications, error handling is an important part. If your code lacks error checking code, your program may look very unprofessional and you may be open to security risks.
- We will show different error handling methods:
  1. Simple "die()" statements
  2. Custom errors and error triggers
  3. Error reporting:

Basic Error Handling: Using the die() function

- The first example shows a simple script that opens a text file:

```
<?php
$file=fopen("welcome.txt","r");
?>
```

- If the file does not exist you might get an error like this:

```
Warning: fopen(welcome.txt) [function.fopen]:
failed to open stream: No such file or
directory in C:\webfolder\test.php on line 2
```

Basic Error Handling: Using the die() function

- To prevent the user from getting an error message like the one above, we test whether the file exist before we try to access it:

```
<?php
if(!file_exists("welcome.txt"))
{
die("File  not  found");
        }
else
{
$file=fopen("welcome.txt","r");
        }
?>
```

- Now if the file does not exist you get an error like this

```
File  not  found
```

Creating a Custom Error Handler

- We simply create a special function Error Handler that can be called when an error occurs in PHP.
- This function must be able to handle a minimum of two parameters (error level and error message) but can accept up to five parameters (optionally: file, line-number, and the error context):
- Syntax:

```
error_function(error_level,error_message,
error_file,error_line,error_context)
```

- Set Error Handler

```
set_error_handler("customError");
```

Example

- Testing the error handler by trying to output variable that does not exist:

```
<?php
//error  handler  function
function  customError($errno,  $errstr)
{
echo  "<b>Error:</b>  [$errno]  $errstr";
}
//set  error  handler
set_error_handler("customError");
//trigger  error
echo($test);
?>
```

Example

- The output of the code above should be something like this:

  Error: [8] Undefined variable: test

Trigger an Error

- In a script where users can input data it is useful to trigger errors when an illegal input occurs. In PHP, this is done by the trigger_error() function
- Example:

```
<?php
$test=2;
if  ($test>1)
{
trigger_error("Value  must  be  1  or  below");
}
?>
```

- The output of the code above should be something like this:

Notice: Value must be 1 or below
in C:\webfolder\test.php on line 6

specify what error level is triggered

- Possible error types:
  - E_USER_ERROR - Fatal user-generated run-time error. Errors that can not be recovered from. Execution of the script is halted
  - E_USER_WARNING - Non-fatal user-generated run-time warning. Execution of the script is not halted E_USER_NOTICE - Default. User-generated run-time
  - notice. The script found something that might be an error, but could also happen when running a script normally

Example

- In this example an E_USER_WARNING occurs if the "test“ variable is bigger than "1". If an E_USER_WARNING occurs we will use our custom error handler and end the script:

```
<?php
//error handler function
function customError($errno, $errstr)
    {
    echo "<b>Error:</b> [$errno] $errstr<br>";
    echo "Ending Script";
    die();
    }
//set error handler
set_error_handler("customError",E_USER_WARNING);
//trigger error
$test=2;
if ($test>1)
    {
    trigger_error("Value must be 1 or below",E_USER_WARNING);
    }
?>
```

Example

- The output of the code above should be something like this:

Error: [512] Value must be 1 or below

Ending Script