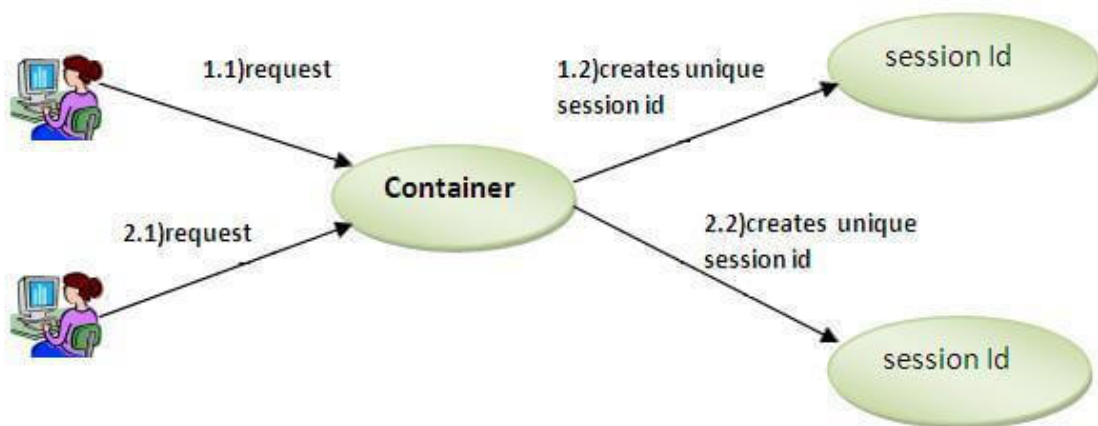


# PHP Session

PHP session is used to store and pass information from one page to another temporarily (until user close the website).

PHP session technique is widely used in shopping websites where we need to store and pass cart information e.g. username, product code, product name, product price etc from one page to another.

PHP session creates unique user id for each browser to recognize the user and avoid conflict between multiple browsers.



## PHP session\_start() function

PHP session\_start() function is used to start the session. It starts a new or resumes existing session. It returns existing session if session is created already. If session is not available, it creates and returns new session.

### Syntax

1. `bool session_start ( void )`

### Example

1. `session_start();`

## PHP \$\_SESSION

PHP \$\_SESSION is an associative array that contains all session variables. It is used to set and get session variable values.

### Example: Store information

1. `$_SESSION["user"] = "Sachin";`

### Example: Get information

1. `echo $_SESSION["user"];`

## PHP Session Example

*File: session1.php*

1. `<?php`
2. `session_start();`
3. `?>`
4. `<html>`
5. `<body>`
6. `<?php`
7. `$_SESSION["user"] = "Sachin";`
8. `echo "Session information are set successfully.<br/>";`
9. `?>`
10. `<a href="session2.php">Visit next page</a>`
11. `</body>`
12. `</html>`

*File: session2.php*

1. `<?php`
2. `session_start();`
3. `?>`
4. `<html>`
5. `<body>`
6. `<?php`
7. `echo "User is: ".$_SESSION["user"];`

8. ?>
9. </body>
- 10.</html>

## PHP Session Counter Example

*File: sessioncounter.php*

1. <?php
2. session\_start();
- 3.
4. **if** (!isset(\$\_SESSION['counter'])) {
5.     \$\_SESSION['counter'] = 1;
6. } **else** {
7.     \$\_SESSION['counter']++;
8. }
9. echo ("Page Views: ".\$\_SESSION['counter']);
10. ?>

## PHP Destroying Session

PHP session\_destroy() function is used to destroy all session variables completely.

*File: session3.php*

1. <?php
2. session\_start();
3. session\_destroy();
4. ?>

## Generating image in PHP

## PHP | imagecreate() Function

The **imagecreate()** function is an inbuilt function in PHP which is used to create a new image. This function returns the blank image of given size. In general **imagecreatetruecolor()** function is used instead of **imagecreate()** function because **imagecreatetruecolor()** function creates high quality images.

**Syntax:**

```
imagecreate( $width, $height )
```

**Parameters:** This function accepts two parameters as mentioned above and described below:

- **\$width:** It is mandatory parameter which is used to specify the image width.
- **\$height:** It is mandatory parameter which is used to specify the image height.

**Return Value:** This function returns an image resource identifier on success, FALSE on errors.

Below programs illustrate the **imagecreate()** function in PHP:

**Program 1:**

```
<?php

// Create the size of image or blank image
$image= imagecreate(500, 300);

// Set the background color of image
$background_color= imagecolorallocate($image, 0, 153, 0);

// Set the text color of image
$text_color= imagecolorallocate($image, 255, 255, 255);

// Function to create image which contains string.
imagestring($image, 5, 180, 100, "GeeksforGeeks", $text_color);
imagestring($image, 3, 160, 120, "A computer science portal", $text_color);

header("Content-Type: image/png");

imagepng($image);
imagedestroy($image);

?>
```

**Output:**

GeeksforGeeks  
A computer science portal

# MySQL

MySQL and PHP - interacting  
with a database

# Stack System

- It is assumed a stack system has been created with a testing server running, and a database linked to a web page, e.g.

<b>PHP</b>	scripting language
<b>MySQL</b>	database
<b>Apache</b>	web server
<b>Windows 7</b>	operating system

WAMP

# Can log on to MySQL with administrator rights

- `http://localhost/phpmyadmin`
- Username is **root**
- If your database is for a live web project use a password and different username for security
- As this is a testing server used locally, there is no password, just the username **admin**

The screenshot shows the phpMyAdmin interface. At the top, the browser address bar displays `localhost/phpmyadmin/`, which is circled in red. Below the browser bar is the phpMyAdmin logo, featuring a sailboat icon and the text "phpMyAdmin". Underneath the logo, it says "Welcome to phpMyAdmin".

The main content area contains a "Language" section with a dropdown menu set to "English". Below this is a "Log in" section with a "Log in" button. The "Username:" field contains the text "root" and is circled in red. A handwritten red arrow points to this field with the word "default" written next to it. The "Password:" field is empty, and a handwritten red slash is drawn over it with the text "leave blank" written next to it. At the bottom right of the form is a "Go" button.



# Database `simpledata` created earlier (example)

Table `feedback` created inside `simpledata`

The screenshot shows the phpMyAdmin interface. The left sidebar displays a tree view of databases, with 'simpledata' circled in red. The main content area shows the 'feedback' table structure, with the table name 'feedback' also circled in red. The table has one column named 'Sum'.

Database: localhost » simpledata » feedback

Table	Action	Rows	Type	Collation
<input type="checkbox"/> feedback	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	~0	InnoDB	latin1_swedish_
1 table	Sum	0	InnoDB	latin1_swedish_

Check All    With selected:

[Print view](#) [Data Dictionary](#)

[Create table](#)

Name:     Number of columns:

# PHPMyAdmin (MySQL) settings

- Name of host: **localhost** (default name for most testing servers)
- Login: MySQL database (PHPMyAdmin)  
login name: **root** (default name which you would change if using the website live)
- Password: leave blank
- Database name: **simpledata**
- Table name (inside database): **feedback**.

If you experience problems you can go back to the other PowerPoint presentation showing you how to set up a MySQL database in PHPMyAdmin.

Access PHPMyAdmin from a web page in Dreamweaver.





# Accessing the database using PHP

- PHP script created in Dreamweaver.

```
1 <?php
2 // Create connection
3 $con=mysqli_connect("localhost","root","",
4 "simpdata");
5
6 // Check connection
7 if (mysqli_connect_errno())
8 {
9     echo "Failed to connect to MySQL: " .
10    mysqli_connect_error();
11 }
12 ?>
```

Connection to server

No error messages

# Presenting a database using PHP

- The SELECT statement is used to select data from a database:
- Add the following PHP

```
<?php
$result = mysqli_query($con, "SELECT * FROM feedback");

while($row = mysqli_fetch_array($result))
{
    echo $row['name'] . " " . $row['rating'];
    echo "<br>";
}
mysqli_close($con);
?>
```

- The data from the table **feedback** is returned by the `mysqli_query()` function and stored in the `$result` variable.
- The `mysqli_fetch_array()` function returns the first row from the table as an **array** variable `&row`.
- The while loop allows `mysqli_fetch_array()` to return the next row in the recordset. The while loop loops through all the rows.
- To **echo** the value of each row, we use the PHP `$row` array variable (`$row['FirstName']` and `$row['LastName']`).

# Presenting a database using PHP

- The source code and live view will look similar to this

The screenshot shows a web browser window with the address `http://localhost/php/php-examples/databases/connect_to_simpdata.php`. The browser's developer tools are open, showing the source code of the page. The code is a PHP script that connects to a MySQL database, queries the `feedback` table, and displays the results. The output of the script is visible in the right-hand pane of the developer tools, showing the names and ratings of two users: Alan Sebrill (7) and Josef K (9). Red handwritten annotations are present: an arrow points from the text "server connection" to the `mysqli_connect` function call in the code; another arrow points from the text "output" to the output pane.

```
1 <?php
2 // Create connection
3 $con=mysqli_connect("localhost","root","","simpdata");
4
5 // Check connection
6 if (mysqli_connect_errno())
7 {
8     echo "Failed to connect to MySQL: " . mysqli_connect_error();
9 }
10 ?>
11
12
13 <?php
14 $result = mysqli_query($con,"SELECT * FROM feedback");
15
16 while($row = mysqli_fetch_array($result))
17 {
18     echo $row['name'] . " " . $row['rating'];
19     echo "<br>";
20 }
21
22 mysqli_close($con);
23 ?>
24
25
```

Alan Sebrill 7  
Josef K 9

server connection

output

# Selecting and changing a database

PHPMyAdmin MySQL and  
Dreamweaver

# Accessing using MySQL/PHP

- Specify a range of MySQL commands to interact with a database **simpledata** from PHP (in Dreamweaver)
- Add new records to the table **feedback**
- Delete a record from the table **feedback**
- List records from the database **feedback**
- Amend data from the database **feedback** and list
- Select a range of data using comparitors (e.g. < less than)

## International Cheese (new database)

- Create a new **database** called **cheese**, using PHP
- Create a new table **international\_cheese.php**
- Add records to the table  
(e.g. name, country, strength, image, in\_stock)
- Output data from table using PHP.
- Save the web page as **cheese.php**





# Create a new database

- If the database **cheese** does not exist you can create it as shown below.



The screenshot shows a web browser window with the address bar displaying `http://localhost/php/php-examples/databases/MySQL/create_cheese_databas`. Below the address bar, a status message reads "No syntax errors." The main content area displays PHP code for creating a database. The code uses `mysqli_connect` to connect to the MySQL server on localhost as the root user. It then checks the connection and attempts to create a database named `$database` (which is "cheese"). If the creation is successful, it echoes a confirmation message. The browser's output area on the right shows the message "Database cheese created successfully".

```
<?php
/*Connection name*/
$con=mysqli_connect("localhost","root","");
$database="cheese";
/*Check connection/mysqli = mysql(improved)*/
if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

// Create database
$sql="CREATE DATABASE $database";
if (mysqli_query($con,$sql))
{
    echo "Database ".$database." created successfully";
}
else
{
    echo "Error creating database: " . mysqli_error($con);
}
?>
```

Database cheese created successfully

# Editing MySQL

- You can change the PHP and edit the MySQL code to be more selective in the process.
- For more info on MySQL, check [http://www.w3schools.com/php/php\\_mysql\\_intro.asp](http://www.w3schools.com/php/php_mysql_intro.asp)

E.g. to select from database only records where the name is "Sam Lowry" change the line

```
$result = mysqli_query($con,"SELECT* FROM feedback");
```

To

```
$result = mysqli_query($con,"SELECT* FROM feedback WHERE name='Sam Lowry'");
```

# Editing MySQL



```
<?php
// Create connection
$con=mysqli_connect("localhost","root","","simplifiedata");

// Check connection
if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}
?>

<?php
$result = mysqli_query($con,"SELECT * FROM feedback WHERE name='Sam
Lowry'");
while($row = mysqli_fetch_array($result))
{
    echo $row['name'] . " " . $row['rating'];
    echo "<br>";
}
mysqli_close($con);
?>
```

Sam Lowry 10

# PHP Date and Time

we will see how to get the date & time using the date() & time() function in PHP, we will also see the various formatting options available with these functions & understand their implementation through the examples.

Date and time are some of the most frequently used operations in PHP while executing SQL queries or designing a website etc. PHP serves us with predefined functions for these tasks. Some of the predefined functions in PHP for date and time are discussed below.

**PHP date() Function:** The PHP date() function converts timestamp to a more readable date and time format.

## Why do we need the date() function?

The computer stores dates and times in a format called UNIX Timestamp, which measures time as a number of seconds since the beginning of the Unix epoch (midnight Greenwich Mean Time on January 1, 1970, i.e. January 1, 1970, 00:00:00 GMT ). Since this is an impractical format for humans to read, PHP converts timestamp to a format that is readable and more understandable to humans.

## Syntax:

```
date(format, timestamp)
```

## Explanation:

- The format parameter in the date() function specifies the format of returned date and time.
- The timestamp is an optional parameter, if it is not included then the current date and time will be used.

**Example:** The below program explains the usage of the date() function in PHP.

- PHP

```
<?php
echo"Today's date is :";
$today= date("d/m/Y");
echo$today;
?>
```

## Output:

```
Today's date is :05/12/2017
```

**Formatting options available in date() function:** The format parameter of the date() function is a string that can contain multiple characters allowing to

generate the dates in various formats. Date-related formatting characters that are commonly used in the format string:

- d: Represents day of the month; two digits with leading zeros (01 or 31).
- D: Represents day of the week in the text as an abbreviation (Mon to Sun).
- m: Represents month in numbers with leading zeros (01 or 12).
- M: Represents month in text, abbreviated (Jan to Dec).
- y: Represents year in two digits (08 or 14).
- Y: Represents year in four digits (2008 or 2014).

The parts of the date can be separated by inserting other characters, like hyphens (-), dots (.), slashes (/), or spaces to add additional visual formatting.

**Example:** The below example explains the usage of the date() function in PHP.

- PHP

```
<?php
echo "Today's date in various formats:". "\n";
echodate("d/m/Y") . "\n";
echodate("d-m-Y") . "\n";
echodate("d.m.Y") . "\n";
echodate("d.M.Y/D");
?>
```

### Output:

Today's date in various formats:

05/12/2017

05-12-2017

05.12.2017

05.Dec.2017/Tue

The following characters can be used along with the date() function to format the time string:

- h: Represents hour in 12-hour format with leading zeros (01 to 12).
- H: Represents hour in 24-hour format with leading zeros (00 to 23).
- i: Represents minutes with leading zeros (00 to 59).
- s: Represents seconds with leading zeros (00 to 59).
- a: Represents lowercase antemeridian and post meridian (am or pm).
- A: Represents uppercase antemeridian and post meridian (AM or PM).

**Example:** The below example explains the usage of the date() function in PHP.

- PHP

```
<?php
    echodate("h:i:s") . "\n";
    echodate("M,d,Y h:i:s A") . "\n";
    echodate("h:i a");
?>
```

**Output:**

03:04:17

Dec,05,2017 03:04:17 PM

03:04 pm

**PHP [time\(\)](#) Function:** The time() function is used to get the current time as a Unix timestamp (the number of seconds since the beginning of the Unix epoch: January 1, 1970, 00:00:00 GMT).

The following characters can be used to format the time string:

- h: Represents hour in 12-hour format with leading zeros (01 to 12).
- H: Represents hour in 24-hour format with leading zeros (00 to 23).
- i: Represents minutes with leading zeros (00 to 59).
- s: Represents seconds with leading zeros (00 to 59).
- a: Represents lowercase antemeridian and post meridian (am or pm).
- A: Represents uppercase antemeridian and post meridian (AM or PM).

**Example:** The below example explains the usage of the time() function in PHP.

- PHP

```
<?php
    $timestamp= time();
    echo($timestamp);
    echo"\n";
    echo(date("F d, Y h:i:s A", $timestamp));
?>
```

**Output:**

1512486297

December 05, 2017 03:04:57 PM

**PHP [mktime\(\)](#) Function:** The mktime() function is used to create the timestamp for a specific date and time. If no date and time are provided, the timestamp for the current date and time is returned.

**Syntax:**

mktime(hour, minute, second, month, day, year)

**Example:** The below example explains the usage of the mktime() function in PHP.

- PHP

```
<?php  
    echomktime(23, 21, 50, 11, 25, 2017);  
?>
```

**Output:**  
1511652110

The above code creates a time stamp for 25th Nov 2017,23 hrs 21mins 50secs.

# PHP File System

In this tutorial you will learn how to create, access (or read) and manipulate files dynamically using the PHP's file system functions.

## Working with Files in PHP

Since PHP is a server side programming language, it allows you to work with files and directories stored on the web server. In this tutorial you will learn how to create, access, and manipulate files on your web server using the PHP file system functions.

## Opening a File with PHP `fopen()` Function

To work with a file you first need to open the file. The PHP `fopen()` function is used to open a file. The basic syntax of this function can be given with:

```
fopen(filename, mode)
```

The first parameter passed to `fopen()` specifies the name of the file you want to open, and the second parameter specifies in which mode the file should be opened. For example:

### Example

[Run this code »](#)

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

The file may be opened in one of the following modes:

Modes	What it does
r	Open the file for reading only.
r+	Open the file for reading and writing.
w	Open the file for writing only and clears the contents of file. If the file does not exist, PHP will attempt to create it.
w+	Open the file for reading and writing and clears the contents of file. If the file does not exist, PHP will attempt to create it.
a	Append. Opens the file for writing only. Preserves file content by writing to the end of the file. If the file does not exist, PHP will attempt to create it.



a+	Read/Append. Opens the file for reading and writing. Preserves file content by writing to the end of the file. If the file does not exist, PHP will attempt to create it.
x	Open the file for writing only. Return FALSE and generates an error if the file already exists. If the file does not exist, PHP will attempt to create it.
x+	Open the file for reading and writing; otherwise it has the same behavior as 'x'.

If you try to open a file that doesn't exist, PHP will generate a warning message. So, to avoid these error messages you should always implement a simple check whether a file or directory exists or not before trying to access it, with the PHP `file_exists()` function.

### Example

[Run this code »](#)

```
<?php
$file = "data.txt";

// Check the existence of file
if(file_exists($file)){
    // Attempt to open the file
    $handle = fopen($file, "r");
} else{
    echo "ERROR: File does not exist.";
}
?>
```

**Tip:** Operations on files and directories are prone to errors. So it's a good practice to implement some form of error checking so that if an error occurs your script will handle the error gracefully. See the tutorial on [PHP error handling](#).

## Closing a File with PHP `fclose()` Function

Once you've finished working with a file, it needs to be closed. The `fclose()` function is used to close the file, as shown in the following example:

### Example

[Run this code »](#)

```
<?php
$file = "data.txt";
```

```

// Check the existence of file
if(file_exists($file)){
    // Open the file for reading
    $handle = fopen($file, "r") or die("ERROR: Cannot open the
file.");

    /* Some code to be executed */

    // Closing the file handle
    fclose($handle);
} else{
    echo "ERROR: File does not exist.";
}
?>

```

**Note:** Although PHP automatically closes all open files when script terminates, but it's a good practice to close a file after performing all the operations.

## Reading from Files with PHP `fread()` Function

Now that you have understood how to open and close files. In the following section you will learn how to read data from a file. PHP has several functions for reading data from a file. You can read from just one character to the entire file with a single operation.

### Reading Fixed Number of Characters

The `fread()` function can be used to read a specified number of characters from a file. The basic syntax of this function can be given with.

```
fread(file handle, length in bytes)
```

This function takes two parameter — A file handle and the number of bytes to read. The following example reads 20 bytes from the "data.txt" file including spaces. Let's suppose the file "data.txt" contains a paragraph of text "The quick brown fox jumps over the lazy dog."

#### Example

**Run this code »**

```

<?php
$file = "data.txt";

// Check the existence of file

```

```

if(file_exists($file)){
    // Open the file for reading
    $handle = fopen($file, "r") or die("ERROR: Cannot open the
file.");

    // Read fixed number of bytes from the file
    $content = fread($handle, "20");

    // Closing the file handle
    fclose($handle);

    // Display the file content
    echo $content;
} else{
    echo "ERROR: File does not exist.";
}
?>

```

The above example will produce the following output:

The quick brown fox

## Reading the Entire Contents of a File

The `fread()` function can be used in conjunction with the `filesize()` function to read the entire file at once. The `filesize()` function returns the size of the file in bytes.

### Example

[Run this code »](#)

```

<?php
$file = "data.txt";

// Check the existence of file
if(file_exists($file)){
    // Open the file for reading
    $handle = fopen($file, "r") or die("ERROR: Cannot open the
file.");

    // Reading the entire file
    $content = fread($handle, filesize($file));

    // Closing the file handle
    fclose($handle);

    // Display the file content
    echo $content;
} else{

```

```
    echo "ERROR: File does not exist.";
}
?>
```

The above example will produce the following output:

The quick brown fox jumps over the lazy dog.

The easiest way to read the entire contents of a file in PHP is with the `readfile()` function. This function allows you to read the contents of a file without needing to open it. The following example will generate the same output as above example:

## Example

[Run this code »](#)

```
<?php
$file = "data.txt";

// Check the existence of file
if(file_exists($file)){
    // Reads and outputs the entire file
    readfile($file) or die("ERROR: Cannot open the file.");
} else{
    echo "ERROR: File does not exist.";
}
?>
```

The above example will produce the following output:

The quick brown fox jumps over the lazy dog.

Another way to read the whole contents of a file without needing to open it is with the `file_get_contents()` function. This function accepts the name and path to a file, and reads the entire file into a string variable. Here's an example:

## Example

[Run this code »](#)

```
<?php
$file = "data.txt";

// Check the existence of file
if(file_exists($file)){
    // Reading the entire file into a string
    $content = file_get_contents($file) or die("ERROR: Cannot open
the file.");

    // Display the file content
    echo $content;
```

```
} else{
    echo "ERROR: File does not exist.";
}
?>
```

One more method of reading the whole data from a file is the PHP's `file()` function. It does a similar job to `file_get_contents()` function, but it returns the file contents as an array of lines, rather than a single string. Each element of the returned array corresponds to a line in the file.

To process the file data, you need to iterate over the array using a [foreach loop](#). Here's an example, which reads a file into an array and then displays it using the loop:

### Example

[Run this code »](#)

```
<?php
$file = "data.txt";

// Check the existence of file
if(file_exists($file)){
    // Reading the entire file into an array
    $arr = file($file) or die("ERROR: Cannot open the file.");
    foreach($arr as $line){
        echo $line;
    }
} else{
    echo "ERROR: File does not exist.";
}
?>
```

## Writing the Files Using PHP `fwrite()` Function

Similarly, you can write data to a file or append to an existing file using the PHP `fwrite()` function. The basic syntax of this function can be given with:

```
fwrite(file handle, string)
```

The `fwrite()` function takes two parameter — A file handle and the string of data that is to be written, as demonstrated in the following example:

### Example

[Run this code »](#)

```
<?php
```

```

$file = "note.txt";

// String of data to be written
$data = "The quick brown fox jumps over the lazy dog.";

// Open the file for writing
$handle = fopen($file, "w") or die("ERROR: Cannot open the file.");

// Write data to the file
fwrite($handle, $data) or die ("ERROR: Cannot write the file.");

// Closing the file handle
fclose($handle);

echo "Data written to the file successfully.";
?>

```

In the above example, if the "note.txt" file doesn't exist PHP will automatically create it and write the data. But, if the "note.txt" file already exist, PHP will erase the contents of this file, if it has any, before writing the new data, however if you just want to append the file and preserve existing contents just use the [mode](#) a instead of w in the above example.

An alternative way is using the `file_put_contents()` function. It is counterpart of `file_get_contents()` function and provides an easy method of writing the data to a file without needing to open it. This function accepts the name and path to a file together with the data to be written to the file. Here's an example:

## Example

[Run this code »](#)

```

<?php
$file = "note.txt";

// String of data to be written
$data = "The quick brown fox jumps over the lazy dog.";

// Write data to the file
file_put_contents($file, $data) or die("ERROR: Cannot write the
file.");

echo "Data written to the file successfully.";
?>

```

If the file specified in the `file_put_contents()` function already exists, PHP will overwrite it by default. If you would like to preserve the file's contents you can pass the special `FILE_APPEND` flag as a third parameter to the `file_put_contents()` function. It will simply append the new data to the file instead of overwriting it. Here's an example:

## Example

[Run this code »](#)

```
<?php
$file = "note.txt";

// String of data to be written
$data = "The quick brown fox jumps over the lazy dog.";

// Write data to the file
file_put_contents($file, $data, FILE_APPEND) or die("ERROR: Cannot
write the file.");

echo "Data written to the file successfully.";
?>
```

## Renaming Files with PHP `rename()` Function

You can rename a file or directory using the PHP's `rename()` function, like this:

## Example

[Run this code »](#)

```
<?php
$file = "file.txt";

// Check the existence of file
if(file_exists($file)){
    // Attempt to rename the file
    if(rename($file, "newfile.txt")){
        echo "File renamed successfully.";
    } else{
        echo "ERROR: File cannot be renamed.";
    }
} else{
    echo "ERROR: File does not exist.";
}
?>
```

## Removing Files with PHP `unlink()` Function

You can delete files or directories using the PHP's `unlink()` function, like this:

## Example

[Run this code »](#)

```
<?php
$file = "note.txt";

// Check the existence of file
if(file_exists($file)){
    // Attempt to delete the file
    if(unlink($file)){
        echo "File removed successfully.";
    } else{
        echo "ERROR: File cannot be removed.";
    }
} else{
    echo "ERROR: File does not exist.";
}
?>
```

In the next chapter we will learn more about [parsing directories or folders](#) in PHP.

# PHP Filesystem Functions

The following table provides the overview of some other useful PHP filesystem functions that can be used for reading and writing the files dynamically.

Function	Description
<code>fgetc()</code>	Reads a single character at a time.
<code>fgets()</code>	Reads a single line at a time.
<code>fgetcsv()</code>	Reads a line of comma-separated values.
<code>filetype()</code>	Returns the type of the file.
<code>feof()</code>	Checks whether the end of the file has been reached.
<code>is_file()</code>	Checks whether the file is a regular file.
<code>is_dir()</code>	Checks whether the file is a directory.
<code>is_executable()</code>	Checks whether the file is executable.



<code>realpath()</code>	Returns canonicalized absolute pathname.
<code>rmdir()</code>	Removes an empty directory.

Please check out the [PHP filesystem reference](#) for other useful PHP filesystem functions.

# Unit-5

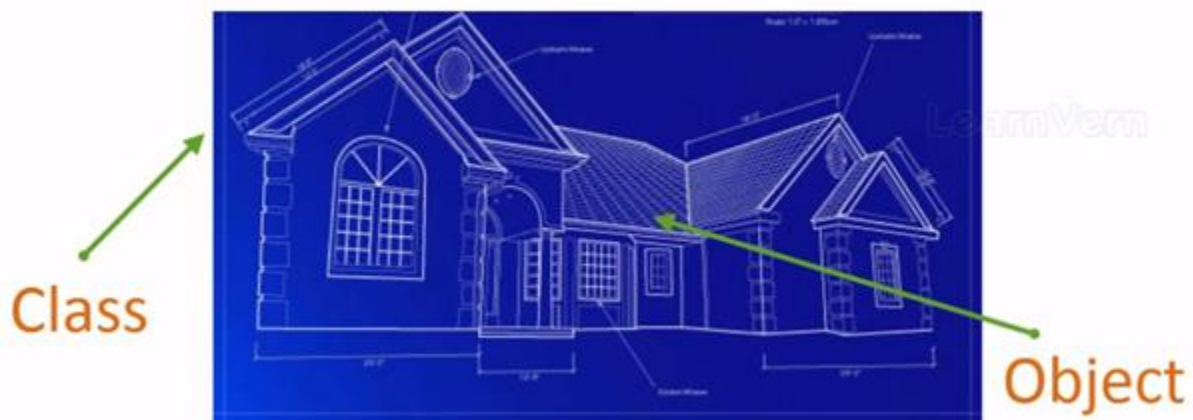
## PHP WITH OOPS CONCEPT

Object-oriented programming is a programming model organized around **Object rather than the actions** and **data rather than logic**.

### Class:

A class is an entity that determines how an object will behave and what the object will contain. In other words, it is a blueprint or a set of instruction to build a specific type of object.

In PHP, declare a class using the class keyword, followed by the name of the class and a set of curly braces ({}).



This is the blueprint of the construction work that is class, and the houses and apartments made by this blueprint are the objects.

### Syntax to Create Class in PHP

1. <?php
2. **class** MyClass
3. {
4.     // Class properties and methods go here
5. }
6. ?>

### Important note:

In PHP, to see the contents of the class, use `var_dump()`. The `var_dump()` function is used to display the structured information (type and value) about one or more variables.

### Syntax:

1. `var_dump($obj);`

### Object:

A class defines an individual instance of the data structure. We define a class once and then make many objects that belong to it. Objects are also known as an instance.

An object is something that can perform a set of related activities.

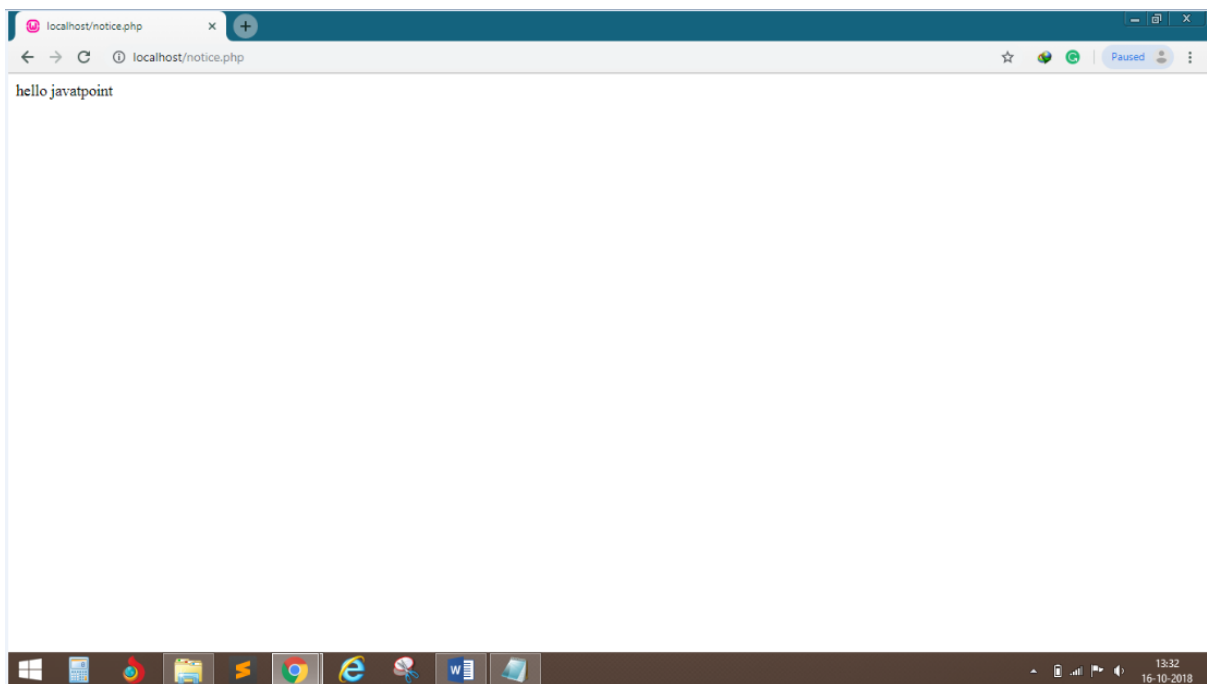
### Syntax:

1. <?php
2. **class** MyClass
3. {
4.     // Class properties and methods go here
5. }
6. `$obj = new` MyClass;
7. `var_dump($obj);`
8. ?>

### Example of class and object:

1. `<?php`
2. **class** demo
3. `{`
4. **private** `$a= "hello javatpoint";`
5. **public function** `display()`
6. `{`
7. `echo $this->a;`
8. `}`
9. `}`
10. `$obj = new demo();`
11. `$obj->display();`
12. `?>`

### Output:

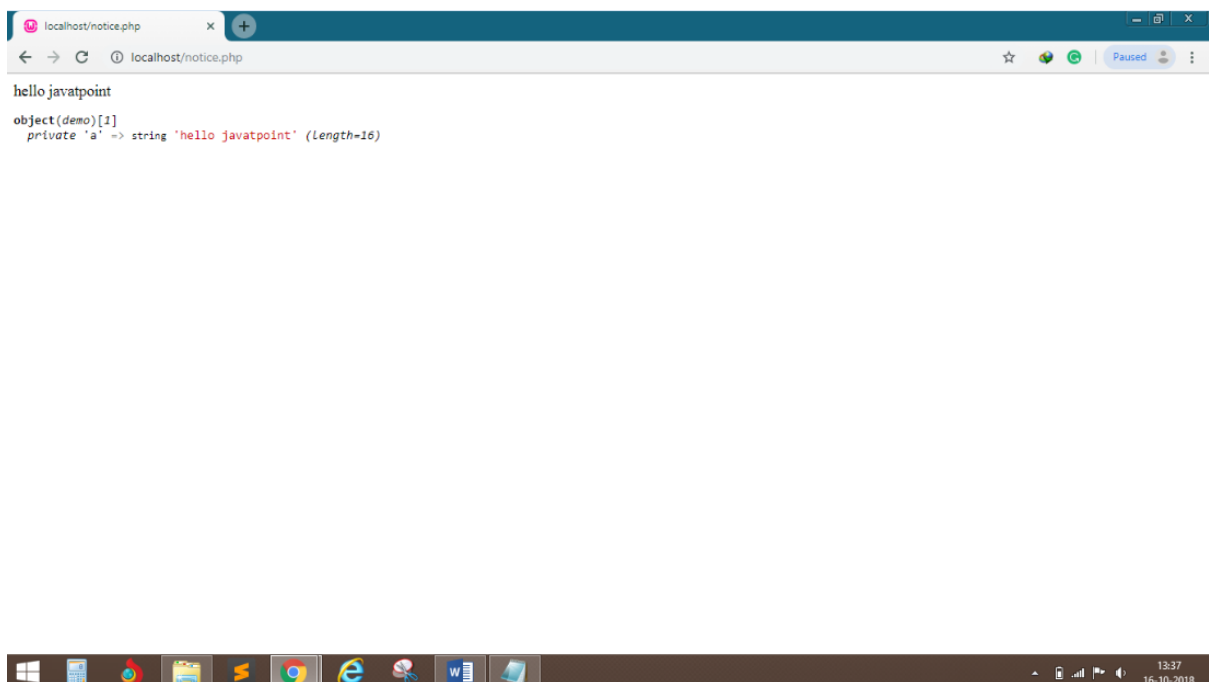


### Example 2: Use of `var_dump($obj);`

1. `<?php`
2. **class** demo
3. `{`
4. **private** `$a= "hello javatpoint";`
5. **public function** `display()`

```
6.     {
7.     echo $this->a;
8.     }
9. }
10. $obj = new demo();
11. $obj->display();
12. var_dump($obj);
13. ?>
```

## Output:



```
localhost/notice.php
hello javatpoint
object(demo)[1]
  private 'a' => string 'hello javatpoint' (Length=16)
```

## Inheritance

It is a concept of accessing the features of one class from another class. If we inherit the class features into another class, we can access both class properties. We can extend the features of a class by using 'extends' keyword.

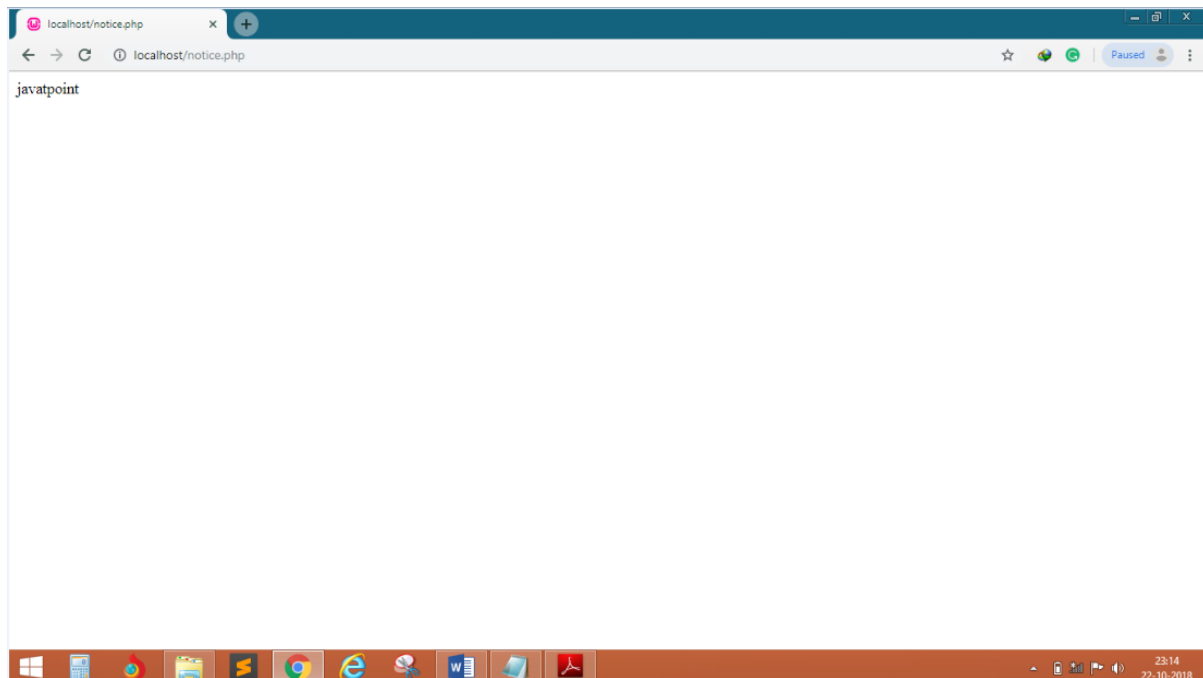
- It supports the concept of **hierarchical classification**.
- Inheritance has three types, **single, multiple and multilevel Inheritance**.
- **PHP** supports only **single inheritance**, where only one class can be **derived from single parent class**.

- We can simulate multiple inheritance by using **interfaces**.

## Example 1

```
1. <?php
2. class a
3. {
4.     function fun1()
5.     {
6.         echo "javatpoint";
7.     }
8. }
9. class b extends a
10. {
11.     function fun2()
12.     {
13.         echo "SSSIT";
14.     }
15. }
16. $obj= new b();
17. $obj->fun1();
18. ?>
```

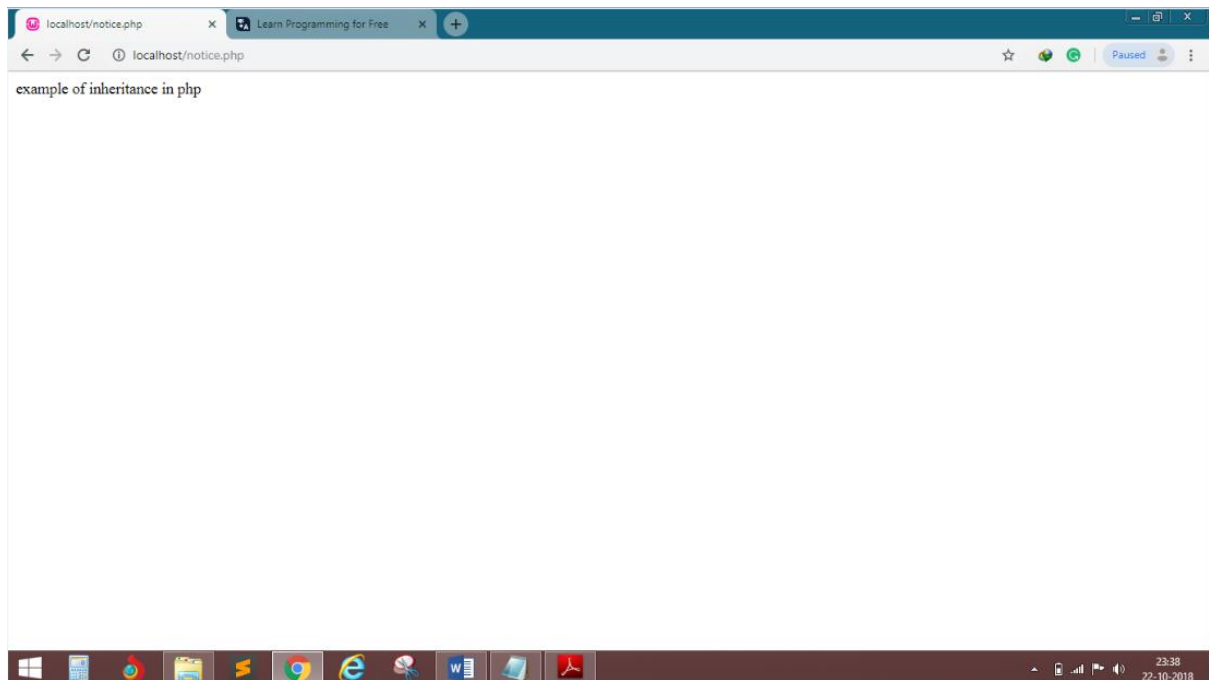
**Output:**



## Example 2

```
1. <?php
2.     class demo
3.     {
4.         public function display()
5.         {
6.             echo "example of inheritance ";
7.         }
8.     }
9.     class demo1 extends demo
10.    {
11.        public function view()
12.        {
13.            echo "in php";
14.        }
15.    }
16.    $obj= new demo1();
17.    $obj->display();
18.    $obj->view();
19. ?>
```

**Output:**



## PHP Form Handling

We can create and use forms in PHP. To get form data, we need to use PHP superglobals `$_GET` and `$_POST`.

The form request may be get or post. To retrieve data from get request, we need to use `$_GET`, for post request `$_POST`.

### PHP Get Form

Get request is the default form request. The data passed through get request is visible on the URL browser so it is not secured. You can send limited amount of data through get request.

Let's see a simple example to receive data from get request in PHP.

File: *form1.html*

```
<form action="welcome.php" method="get">  
Name: <input type="text" name="name"/>  
<input type="submit" value="visit"/>  
</form>
```



File: *welcome.php*

```
<?php
$name=$_GET["name");//receiving name field value in $name variable
echo "Welcome, $name";
?>
```

## PHP Post Form

Post request is widely used to submit form that have large amount of data such as file upload, image upload, login form, registration form etc.

The data passed through post request is not visible on the URL browser so it is secured. You can send large amount of data through post request.

Let's see a simple example to receive data from post request in PHP.

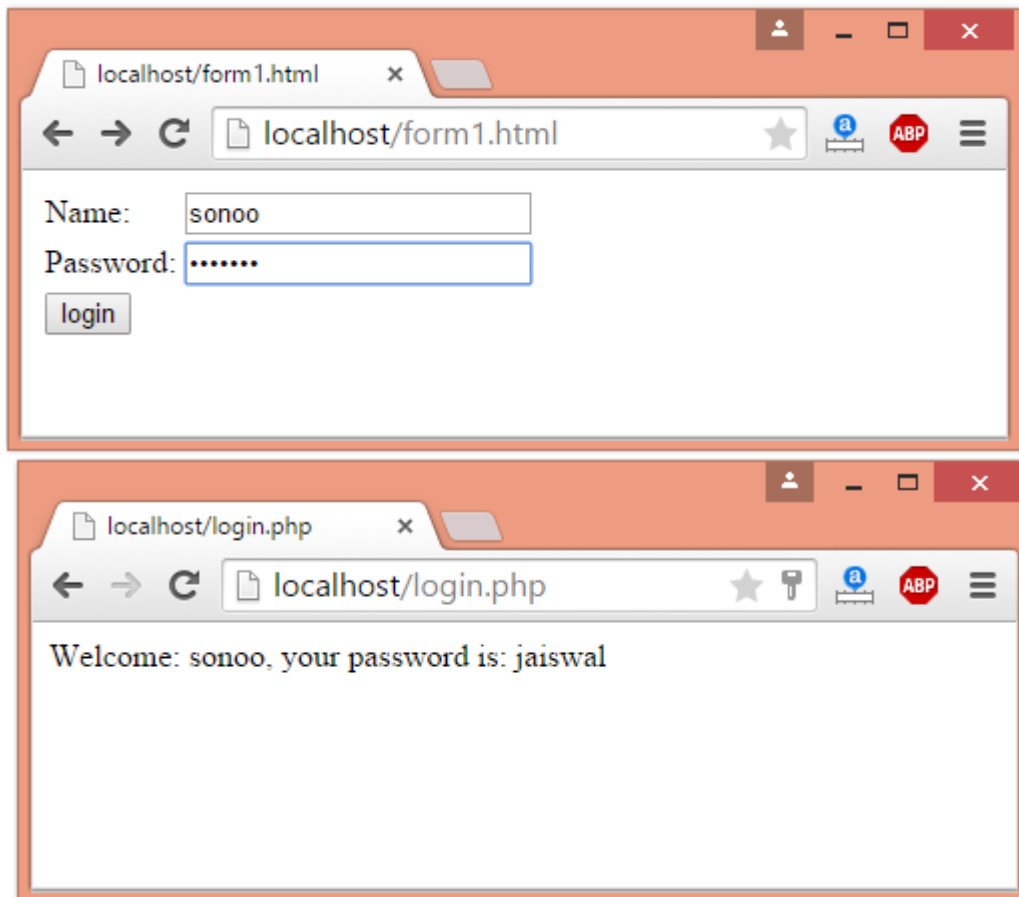
File: *form1.html*

```
<form action="login.php" method="post">
<table>
<tr><td>Name:</td><td> <input type="text" name="name"/></td></tr>
<tr><td>Password:</td><td> <input type="password" name="password"/></td></tr>
>
<tr><td colspan="2"><input type="submit" value="login"/> </td></tr>
</table>
</form>
```

File: *login.php*

```
<?php
$name=$_POST["name");//receiving name field value in $name variable
$password=$_POST["password");//receiving password field value in $password variable
echo "Welcome: $name, your password is: $password";
?>
```

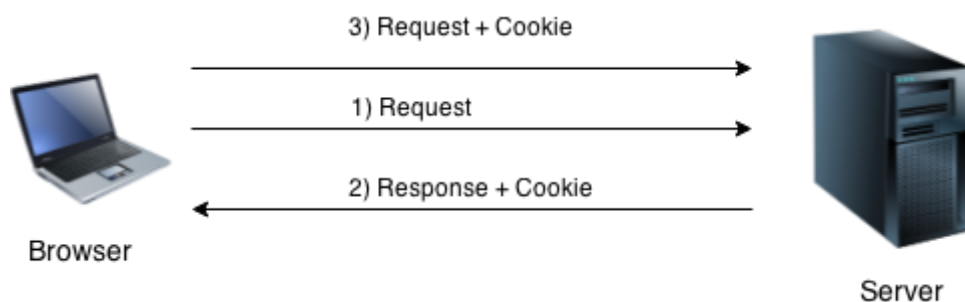
Output:



## PHP Cookie

PHP cookie is a small piece of information which is stored at client browser. It is used to recognize the user.

Cookie is created at server side and saved to client browser. Each time when client sends request to the server, cookie is embedded with request. Such way, cookie can be received at the server side.



In short, cookie can be created, sent and received at server end.

# Exception Handling in PHP

Exception handling is a powerful mechanism of PHP, which is used to handle runtime errors (runtime errors are called exceptions). So that the normal flow of the application can be maintained.

The main purpose of using exception handling is **to maintain the normal execution of the application**.

## What is an Exception?

An exception is an unexcepted outcome of a program, which can be handled by the program itself. Basically, an exception disrupts the normal flow of the program. But it is different from an error because an exception can be handled, whereas an error cannot be handled by the program itself.

In other words, - "An unexpected result of a program is an exception, which can be handled by the program itself." Exceptions can be thrown and caught in PHP.

## Why needs Exception Handling?

PHP provides a powerful mechanism, exception handling. It allows you to handle runtime errors such as IOException, SQLException, ClassNotFoundException, and more. A most popular example of exception handling is - divide by zero exception, which is an arithmetic exception.

**Note: Exception handling is required when an exception interrupts the normal execution of the program or application.**

Exception handling is almost similar in all programming languages. It changes the normal flow of the program when a specified error condition occurs, and this condition is known as exception. PHP offers the following keywords for this purpose:

**try** -

The try block contains the code that may have an exception or where an exception can arise. When an exception occurs inside the try block during runtime of code, it is caught and resolved in catch block. The try block must be followed by catch or finally block. A try block can be followed by minimum one and maximum any number of catch blocks.

### **catch -**

The catch block contains the code that executes when a specified exception is thrown. It is always used with a try block, not alone. When an exception occurs, PHP finds the matching catch block.

### **throw -**

It is a keyword used to throw an exception. It also helps to list all the exceptions that a function throws but does not handle itself.

Remember that each throw must have at least one "catch".

### **finally -**

The finally block contains a code, which is used for clean-up activity in PHP. Basically, it executes the essential code of the program.

## **What happens when an exception is triggered -**

- The current state of code is saved.
- The execution of the code is switched to a predefined exception handler function.
- Depending on the situation, the handler can halt the execution of program, resume the execution from the saved code state, or continue the execution of the code from another location in the code.

## **Advantage of Exception Handling over Error Handling**

Exception handling is an important mechanism in PHP, which have the following advantages over error handling -

### **Grouping of error types -**

In PHP, both basic and objects can be thrown as exception. It can create a hierarchy of exception objects and group exceptions in classes and also classify them according to their types.

### **Keep error handling and normal code separate -**

In traditional error handling, if-else block is used to handle errors. It makes the code unreadable because the code for handling errors and conditions got mixed. Within the try-catch block, exception keeps separate from the code and code become readable.

# FRONT END AND BACK END DATABASE CONNECTIVITY.

## Front end code:

```
<html>
```

```
<head>
```

```
<title>Registration</title>
```

```
</head>
```

```
<body bgcolor="orange">
```

```
<center><h1>Registration</h1>
```

```
<form action="welcome.php"  
method="post">
```

```
ENTER NAME:<input type="text"  
name="Name"></br>
```

```
ENTER PASSWORD:<input type="text"  
name="Password"></br>
```

```
ENTER EMAIL:<input type="text"  
name="Email"></br>
```

```
ENTER COUNTRY:<input type="text"
name="Country"></br>
<input type="submit" value="REGISTER">
<input type="reset" value="RESET">
</form>
</center>
</body>
</html>
```

### Backend Code: (PHP)

```
<?php
$name=$_POST['Name'];
$password=$_POST['Password'];
$email=$_POST['Email'];
$country=$_POST['Country'];

$conn = new
mysqli('localhost:3306','root','','nd');

if($conn->connect_error)
```

```
{  
  
die('Connection failed :!.$conn-  
>connect_error);  
  
}  
  
else  
  
{  
  
$stmt=$conn->prepare("insert into nt(Name,  
Password, Email, Country) value(?,?,?,?)");  
  
$stmt->bind_param('ssss',$Name,  
$Password, $Email, $Country);  
  
$stmt->execute();  
  
echo "Welcome: $Name Registration  
Successfully....";  
  
$stmt->close();  
  
$conn->close();  
  
}  
  
?>
```