Faculté des Sciences et Techniques Marrakech

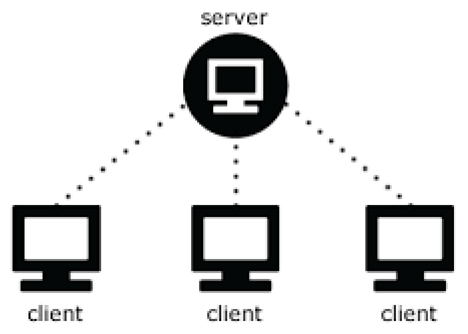
جامعة القاضي عياض
كلية العلوم و التقنيات مراكش
UNIVERSITE CADI AYYAD
FACULTE DES SCIENCES ET TECHNIQUES - MARRAKECH

Module Programmation web

Filière LPU SIR

Prof. Sara Qassimi

25 novembre 2022



Chapitre 3: Développement Back-End

- Objectifs:
  - PHP Fundamentals
  - Database Access
  - PHP Security best practices
  - Cookies and Sessions
  - Object-Oriented Programming (OOP)
  - Model-View-Controller (MVC) framework

Basics

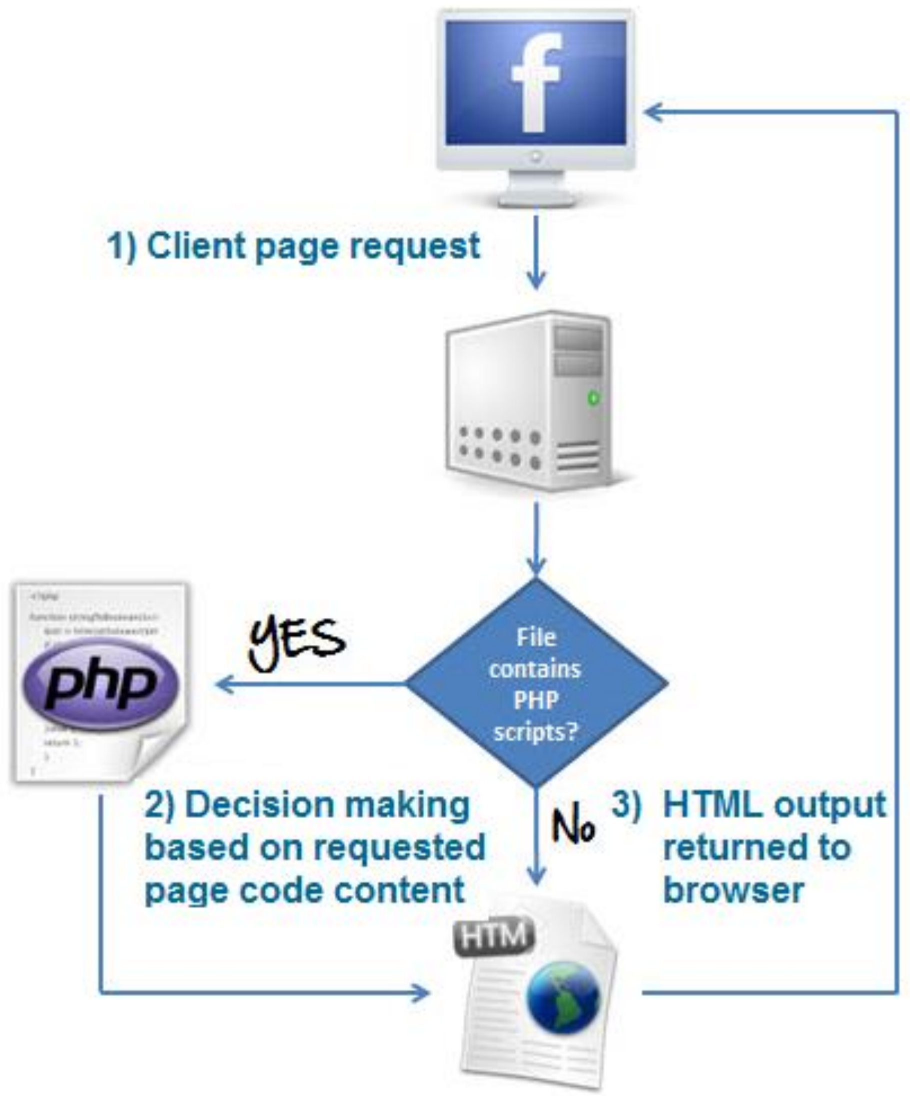
- PHP stands for Hypertext Pre-Processor
- PHP is a server side scripting language. This means that it is executed on the server.

Programming language	Scripting language
The code has to be compiled before it can be executed	The code is usually executed without compiling
Does not need to be embedded into other languages	Is usually embedded into other software environments.

- PHP is a server side script that is interpreted on the server while JavaScript is an example of a client side script that is interpreted by the client browser.
- Both PHP and JavaScript can be embedded into HTML pages.
- PHP files are saved with the ".php" file extension, and the PHP development code is enclosed in tags.
- PHP is open source and cross platform
- **Syntax:**

```
<?php
    echo 'Hello World';
?>
```

- flowchart diagram shown below illustrates the basic architecture of a PHP web application and how the server handles the requests.


1) Client page request
File contains PHP scripts?
YES
No
2) Decision making based on requested page code content
3) HTML output returned to browser
php
HTM

PHP Data Types

- A Data type is the classification of data into a category according to its attributes;
  - Alphanumeric characters are classified as strings
  - Whole numbers are classified integers
  - Numbers with decimal points are classified as floating points.
  - True or false values are classified as Boolean.
- PHP is a loosely typed language; it does not have explicit defined data types.
- PHP determines the data types by analyzing the attributes of data supplied.
- PHP implicitly supports the following data types:
  - Integer : The constant PHP_INT_MAX is used to determine the maximum value

```
<?php echo PHP_INT_MAX; ?>
```

  - Floating point number – decimal numbers e.g. 3.14. Floating point numbers are larger than integers.
  - Character string – e.g. Hello World
  - Boolean – e.g. True or false.

PHP Variable

- A variable is a name given to a memory location that stores data at runtime.
- The scope of a variable determines its visibility.
- A Php global variable is accessible to all the scripts in an application.
- A local variable is only accessible to the script that it was defined in.
- Arithmetic operators are used to manipulate numeric data
- Assignment operators are used to assign data to variables
- Comparison operators are used to compare variables or values
- Logical operators are used to compare conditions or values

PHP Variable

- All variable names must start with the dollar sign e.g. **$my_var**
- Variable names are case sensitive **$my_var ≠ $MY_VAR**
- All variables names must start with a letter follow other characters e.g.

  ✓ **$my_var1;** ✗ ~~**$1my_var;**~~
- Variable names must not contain any spaces ✓ **$my_var;** ✗ **$my var**

```
<?php

 $my_var = 1;
$my_float_var = 3.14;
$my_string_var ="Hypertext Pre Processor";
echo $my_var+ ” ”+ $my_float_var + ” ”+ $my_string_var;

?>
```

PHP Variable

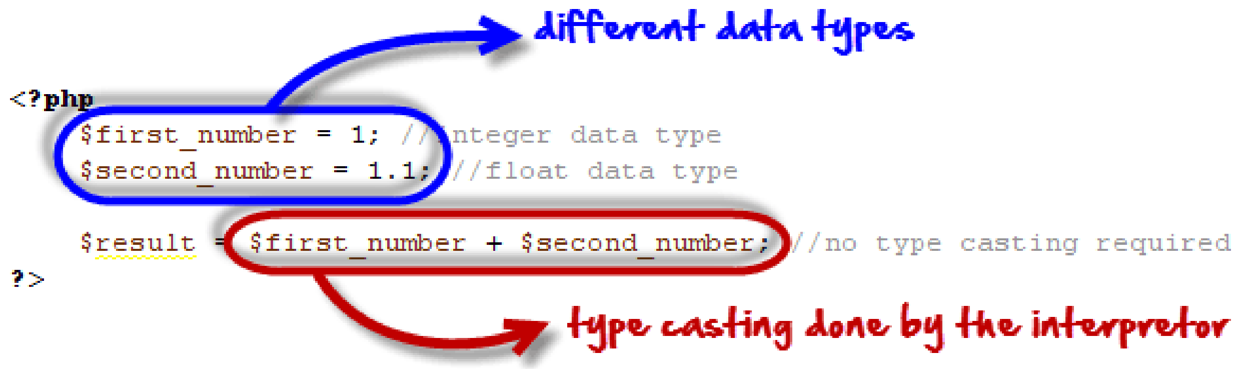
- The var_dump function is used to determine the data type

```
<?php
$a = 1; var_dump($a);
$b = 1.5; var_dump($b);
 $c = "PHP"; var_dump($c);
$d = true;var_dump($d); ?>
```

Output:

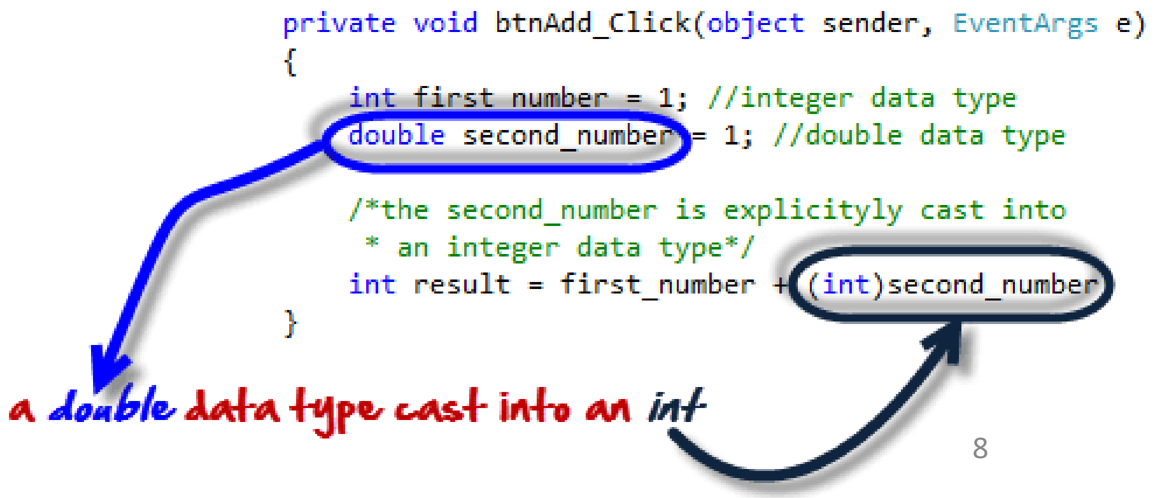
```
int(1) float(1.5) string(10) "PHP" bool(true)
```

- **Define constant**- A constant is a variable whose value cannot be changed at runtime. define('PI',3.14);
- Type casting is used to convert a value or variable into a desired data type



```
<?php
    $first_number = 1; //integer data type
    $second_number = 1.1; //float data type

    $result = $first_number + $second_number; //no type casting required
?>
```



```
private void btnAdd_Click(object sender, EventArgs e)
{
    int first_number = 1; //integer data type
    double second_number = 1; //double data type

    /*the second_number is explicitly cast into
     * an integer data type*/
    int result = first_number + (int)second_number
}
```

PHP Include & Require

- Single HTML code such as headers, footers, side bars etc. can be shared across many pages. This makes it easy to update the website by just updating a single file.
- PHP code such as database configuration settings, custom functions etc. can be shared across many pages ensuring the website/application uses the same settings.
- The "include" php statement is used to include other files into a PHP file.

index.php

```
<?php
 include 'header.php’;
?>
```

header.php

```
<a href="/index.php">Home</a> <a
href="/aboutus.php">About us</a> <a
href="/services.php">Services</a> <a
href="/contactus.php">Contact Us</a>
```

- The require statement is used to include file.
- We can create a configuration file that we can include in all pages that connect to the database using the require statement.

config.php

```
<?php $config['host'] = 'localhost';
$config['db'] = 'my_database';
$config['uid'] = 'root';
$config['password'] = ‘’;  ?>
```

Pages_model.php

```
<?php
 require 'config.php'; //require the config file
//other code for connecting to the database
?>
```

PHP Include vs Require

- The difference between include and require

Include	Require
Issues a warning when an error occurs	Does not issue a warning
Execution of the script continues when an error occurs	Execution of the script stops when an error occurs.

- The “include” and “require” statements can be used at any line in the source codes where you want the code to appear.
- The “include” statement issues a warning and continues with the execution if the requested file has not been found.
- The “ require“ statement raises a fatal error and stops the script execution.
- The “include” statement should be in most cases except in situations where without the requested file to be include, the entire script cannot run.

PHP Array: Associative, Multidimensional

- Arrays are special variables with the capacity to store multi values.
- Arrays are flexibility and can be easily stretched to accommodate more values

Numeric Arrays

- Numeric arrays use number as access keys.
- An access key is a reference to a memory slot in an array variable.
- when dealing with multiple values of the same nature is very easy and flexible.

```
<?php
$movie[0]="Shaolin Monk";
$movie[1]="Drunken Master";
$movie[2]="American Ninja";
$movie[3]="Once upon a time in
China"; $movie[4]="Replacement
Killers"; echo $movie[3]; $movie[3] =
" Eastern Condors"; echo $movie[3];
?>
```

```
<?php
$movie = array(0 => "Shaolin Monk",
                  1 => "Drunken Master",
                  2 => "American Ninja",
                  3 => "Once upon a time in China",
                  4 =>"Replacement Killers" );
echo $movie[4];
 ?>
```

PHP Array: Associative, Multidimensional

PHP Associative Array

- Associative array differ from numeric array in the sense that associative arrays use descriptive names for id keys.

```
<?php
$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");
print_r($persons);
echo "";
echo "Mary is a " . $persons["Mary"];
?>
```

Output

```
Array ( [Mary] => Female [John] => Male [Mirriam] => Female ) Mary is a Female
```

- Associative array are also very useful when retrieving data from the database.

PHP Array: Associative, Multidimensional

PHP Multidimensional Array

- Multidimensional arrays contain other arrays inside them
- The advantage of multidimensional arrays is that they allow to group related data together.

```
<?php
 $movies = array(
             "comedy"    =>    array("Pink Panther", "John English", "See no evil hear no evil"),
             "action"    =>    array("Die Hard", "Expendables"),
             "epic"      =>    array("The Lord of the rings"),
            "Romance"    =>    array("Romeo and Juliet")
              );
print_r($movies); ?>
```

Outer Array defined as an Assoviative Array

Array values defined as Numeric Arrays

Output:

```
Array ( [comedy] => Array ( [0] => Pink Panther [1] => John English [2] => See no evil hear no evil )
[action] => Array ( [0] => Die Hard [1] => Expendables ) [epic] => Array ( [0] => The Lord of the rings )
[Romance] => Array ( [0] => Romeo and Juliet ) )
```

PHP Array Functions

Count function

- The count function is used to count the number of elements that an php array contains.

```
<?php $lecturers = array("Mr. Jones", "Mr. Banda", "Mrs. Smith");
 echo count($lecturers); ?>
```

is_array function

- The is_array function is used to determine if a variable is an array or not.

```
<?php  $lecturers = array("Mr. Jones", "Mr. Banda", "Mrs. Smith");
echo is_array($lecturers); ?>
```

Sort function

- Used to sort arrays by the values. If the values are alphanumeric, it sorts them in alphabetical order. If the values are numeric, it sorts them in ascending order. It removes the existing access keys and add new numeric keys. The output of this function is a numeric array

Output:

```
Array ( [0] => Female [1] => Female [2] => Male )
```

```
<?php $persons = array("Mary" => "Female", "John" => "Male",
"Mirriam" => "Female"); sort($persons); print_r($persons); ?>
```

PHP Array Functions

Ksort function

- Used to sort the array using the key.

```
<?php $persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");
ksort($persons); print_r($persons); ?>
```

Output:

```
Array ( [John] => Male
[Mary] => Female
[Mirriam] => Female )
```

Asort function

- Used to sort the array using the values.

```
<?php $persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");
asort($persons); print_r($persons); ?>
```

Output:

```
Array ( [Mary] => Female
 [Mirriam] => Female
[John] => Male )
```

PHP Loop: ForEach

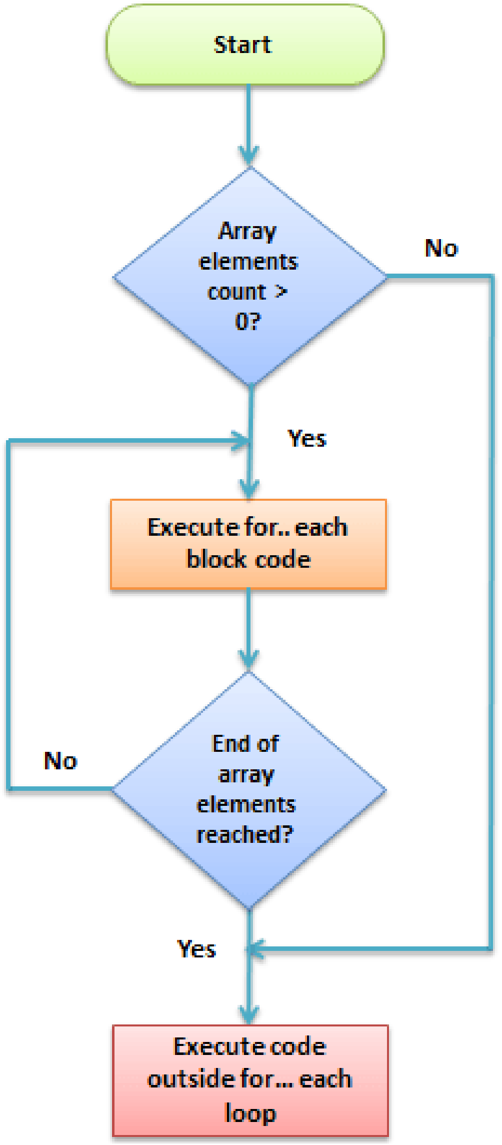
- A Loop is an Iterative Control Structure that involves executing the same number of code a number of times until a certain condition is met, ex. For, While, Do While

PHP ForEach loop

- Used to iterate through array values

```
<?php
$animals_list = array("Lion","Wolf","Dog","Leopard","Tiger");
foreach($animals_list as $array_values)
{ echo $array_values . "<br>"; }
?>
```

```
<?php
// Associative Array
 $persons = array("Mary" => "Female", "John" => "Male",
"Mirriam" => "Female");
foreach($persons as $key => $value)
{ echo "$key is $value"."<br>"; } ?>
```


Start
Array elements count > 0?
No
Yes
Execute for.. each block code
No
End of array elements reached?
Yes
Execute code outside for... each loop

PHP Strings

- The string variables can contain alphanumeric characters.
- The var_dump() function dumps information about one or more variables.
- Creating PHP Strings Using Single quotes

```
<?php    var_dump('You need to be logged in to view this page'); ?>
```

Output:

```
string(42) "You need to be logged in to view this page"
```

- If the single quote is part of the string value, it can be escaped using the backslash.

```
<?php echo 'I \'ll be back after 20 minutes'; ?>
```

Output:

```
I'll be back after 20 minutes
```

- The double quotes are used to create relatively complex strings compared to single quotes.

```
<?php $name='Alicia’;
 echo "$name is friends with kalinda"; ?>
```

- The dollar sign to be treated as a literal value

```
<?php $word="word";
$pwd = "pas\$word";
echo $pwd; ?>
```

Define function

- Function is a block of code that performs specific task.

```
<?php //define a function that displays hello function
function add_numbers(){    echo 1 + 2; }
add_numbers (); ?>
```

Output:
3

```
<?php
function display_name($name) { echo "Hello " . $name; }
display_name("Martin Luther King"); ?>
```

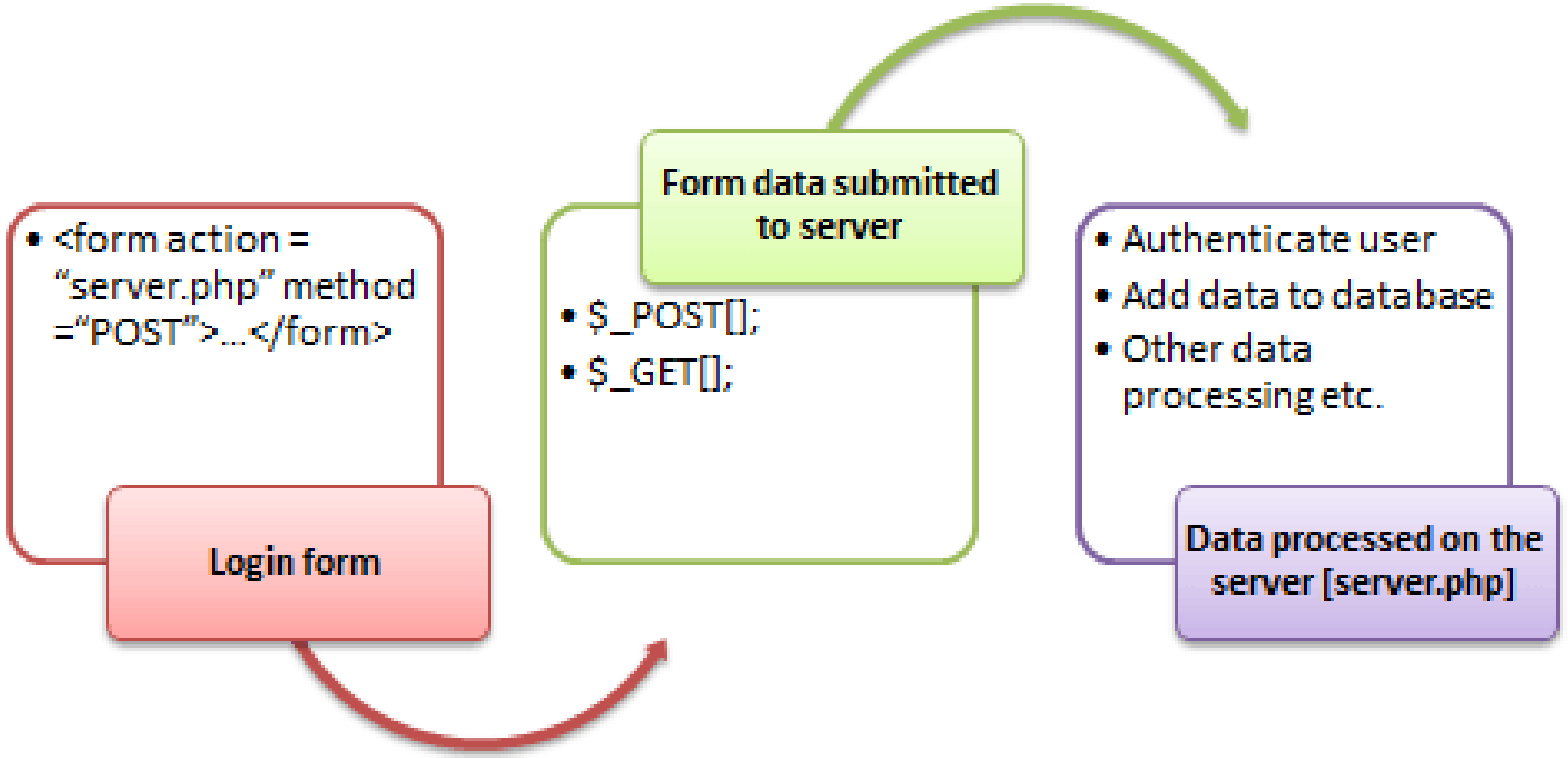
Output:
Hello Martin Luther King

```
<?php
function kilometers_to_miles($kilometers = 0)
{ $miles_scale = 0.62;
 return $kilometers * $miles_scale;
}
echo kilometers_to_miles(100); ?>
```

Output:
62

PHP Registration Form : POST & GET

- Forms -defined using the <form>...</form> tags-are used to get input from the user and submit it to the web server for processing.

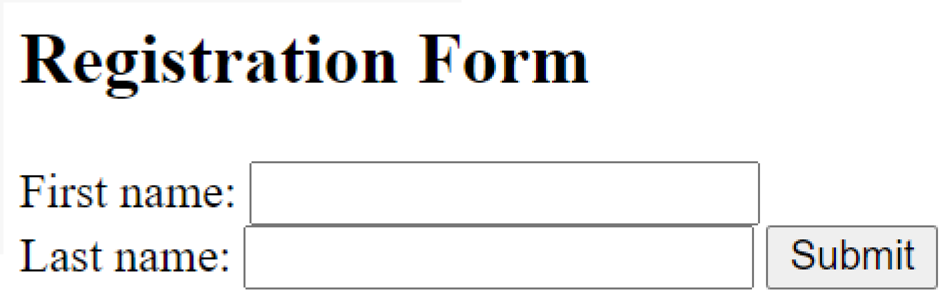


• <form action = "server.php" method ="POST">...</form>
Login form
Form data submitted to server
• $_POST[];
• $_GET[];
• Authenticate user
• Add data to database
• Other data processing etc.
Data processed on the server [server.php]

PHP Registration Form : POST & GET

- **Registration Form**

```
<html>
 <head> <title>Registration Form</title>
              <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
 <h2>Registration Form</h2>
 <form action="registration_form.php" method="POST">
First name: <input type="text" name="firstname"> <br>
Last name: <input type="text" name="lastname">
<input type="hidden" name="form_submitted" value="1" />
<input type="submit" value="Submit">
</form>
</body>
</html>
```



PHP Registration Form : POST & GET

- **Submitting the form data to the server**
- The action attribute of the form specifies the submission URL that processes the data. The method attribute specifies the submission type.

PHP POST method

- Built in PHP super global array variable that is used to get values submitted via HTTP POST method.

```
<?php
$_POST['variable_name’];
 ?>
```

•“$_POST[…]” is the PHP array
•“$_GET[…]” is the PHP array
•The array variable can be accessed from any script in the program; it has a global scope.
•“'variable_name'” is the URL variable name

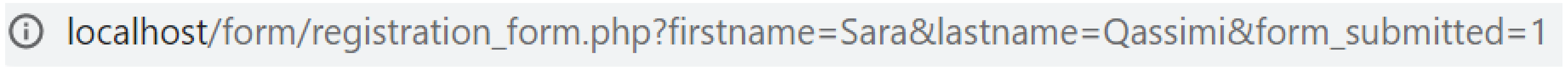
PHP GET method

- Built in PHP super global array variable that is used to get values submitted via HTTP GET method.

```
<?php
$_GET['variable_name'];
?>
```

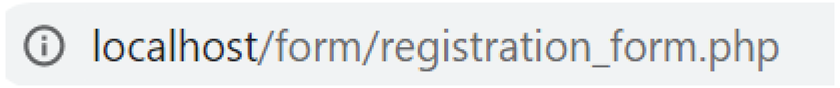
PHP Registration Form : POST & GET

POST	**GET**
Values not visible in the URL	Values visible in the URL
Has not limitation of the length of the values since they are submitted via the body of HTTP	Has limitation on the length of the values usually 255 characters. This is because the values are displayed in the URL. Note the upper limit of the characters is dependent on the browser.
Has lower performance compared to Php_GET method due to time spent encapsulation the Php_POST values in the HTTP body	Has high performance compared to POST method dues to the simple nature of appending the values in the URL.
Supports many different data types such as string, numeric, binary etc.	Supports only string data types because the values are displayed in the URL
Results cannot be book marked	Results can be book marked due to the visibility of the values in the URL


localhost/form/registration_form.php?firstname=Sara&lastname=Qassimi&form_submitted=1

Method GET

Method POST


localhost/form/registration_form.php

PHP Registration Form : POST & GET

- **Processing the registration form data:** Using the PHP isset function to check if the form values have been filled in the $_POST array and process the data.

```
<html> <head> <title>Registration Form</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"> </head>
 <body>
<?php if (isset($_POST['form_submitted'])): //this code is executed when the form is submit
ted ?>
 <h2>Thank You <?php echo $_POST['firstname']; ?> </h2>
 <p>You have been registered as <?php echo $_POST['firstname'] . ' ' . $_POST['lastname'];
?> </p>
<p>Go <a href="registration_form.php">back</a> to the form</p>
 <?php else: ?>
<h2>Registration Form</h2>
 <form action="registration_form.php" method="POST">
  First name: <input type="text" name="firstname"> <br>
   Last name: <input type="text" name="lastname">
    <input type="hidden" name="form_submitted" value="1" />
    <input type="submit" value="Submit">
    </form>
     <?php endif;?>
</body> </html>
```

PHP File() Function

- Files are usually used to store information such as;Configuration settings of a program

file exists	<?php if (file_exists('my_settings.txt')) { echo 'file found!'; } else { echo 'my_settings.txt does not exist'; } ?>
Open a file	<?php $fh = fopen("my_settings.txt", 'w') or die("Failed to create file"); ?>
Closing a file	<?php fclose($fh); ?>
Create File	<?php $fh = fopen("my_settings.txt", 'w') or die("Failed to create file"); $text = " Hello Irisi 1"; fwrite($fh, $text) or die("Could not write to file"); fclose($fh); echo "File 'my_settings.txt' written successfully"; ?>
Read File	<?php $fh = fopen("my_settings.txt", 'r') or die("File does not exist or you lack permission to open it"); $line = fgets($fh); echo $line; fclose($fh); ?>
Copy file	<?php copy('my_settings.txt', 'my_settings_backup.txt') or die("Could not copy file"); echo "File successfully copied to 'my_settings_backup.txt'"; ?>

Mode	Description
r	•Read file from beginning. •Returns false if the file doesn’t exist. •Read only
r+	•Read file from beginning •Returns false if the file doesn’t exist. •Read and write
w	•Write to file at beginning •truncate file to zero length •If the file doesn’t exist attempt to create it. •Write only
w+	•Write to file at beginning, truncate file to zero length •If the file doesn’t exist attempt to create it. •Read and Write
a	•Append to file at end •If the file doesn’t exist attempt to create it. •Write only
a+	•Php append to file at end •If the file doesn’t exist attempt to create it •Read and write

PHP Try Catch : Error & Exception Handling

- An error is an unexpected program result that cannot be handled by the program itself.

  PHP Error handling

- When an error occurs, depending on your configuration settings, PHP displays the error message in the web browser with information relating to the error that occurred.

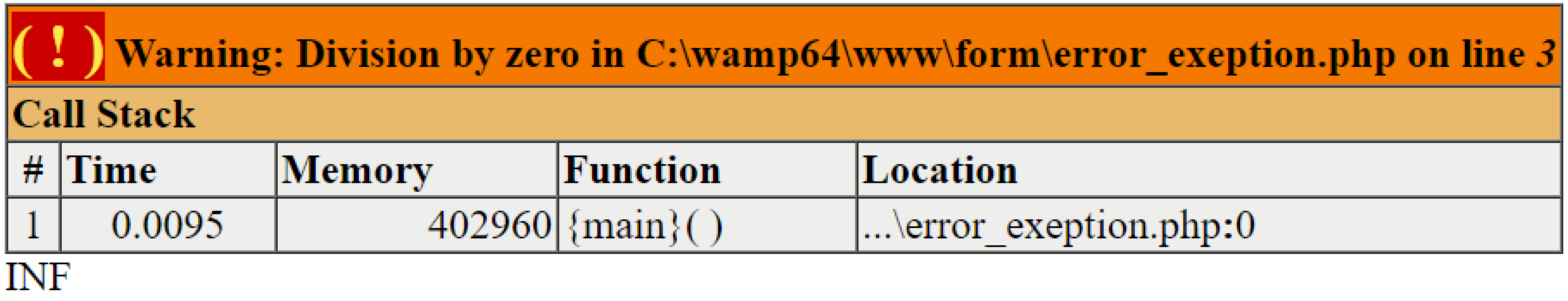
- (3) commonly used methods;

1. **Die statements**– the die function combines the echo and exit function in one. It is very useful when we want to output a message and stop the script execution when an error occurs.
2. **Custom error handlers** – these are user defined functions that are called whenever an error occurs.
3. **PHP error reporting** – the error message depending on your PHP error reporting settings. This method is very useful in development environment when you have no idea what caused the error. The information displayed can help you debug your application.

PHP Try Catch : Error & Exception Handling

- **Die statements**

```
<?php $denominator = 0; echo 2 / $denominator; ?>
```



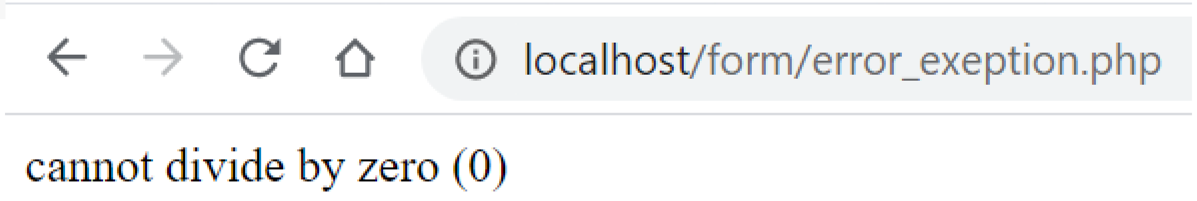
(!) Warning: Division by zero in C:\wamp64\www\form\error_exeption.php on line 3

Call Stack

#	Time	Memory	Function	Location
1	0.0095	402960	{main}( )	...\error_exeption.php:0

```
INF
```

```
<?php $denominator = 0;
if ($denominator != 0) {
 echo 2 / $denominator;
} else { echo "cannot divide by zero (0)"; } ? >
```

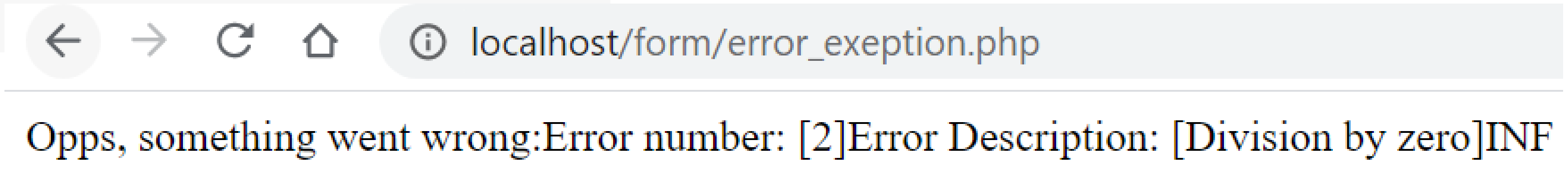


```
cannot divide by zero (0)
```

PHP Try Catch : Error & Exception Handling

- **Custom error handlers**

```
<?php
 function my_error_handler($error_no, $error_msg)
 {  echo "Opps, something went wrong:";
    echo "Error number: [$error_no]";
    echo "Error Description: [$error_msg]";
 }
    set_error_handler("my_error_handler");
 echo (5 / 0); ?>
```



localhost/form/error_exeption.php

```
Opps, something went wrong:Error number: [2]Error Description: [Division by zero]INF
```

- Custom error handlers are powerful in the sense that:
  - They allow to customize the error messages.
  - The custom error handler can also include error logging in a file/database, emailing the developer etc.

PHP Try Catch : Error & Exception Handling

- **PHP error reporting:** “error_reporting” is the PHP error reporting function

Reporting Level	Description	Example
E_WARNING	Displays warning messages only. Does not halt the execution of the script	error_reporting(E_WARNING);
E_NOTICE	Displays notices that can occur during normal execution of a program or could be an error.	error_reporting(E_ NOTICE);
E_USER_ERROR	Displays user generated errors i.e. custom error handler	error_reporting(E_ USER_ERROR);
E_USER_WARNING	Displays user generated warning messages	error_reporting(E_USER_WARNING);
E_USER_NOTICE	Displays user generated notices	error_reporting(E_USER_NOTICE);
E_RECOVERABLE_ERROR	Displays error that are not fatal and can be handled using custom error handlers	error_reporting(E_RECOVERABLE_ERROR);
E_ALL	Displays all errors and warnings	error_reporting(E_ ALL);

PHP Try Catch : Error & Exception Handling

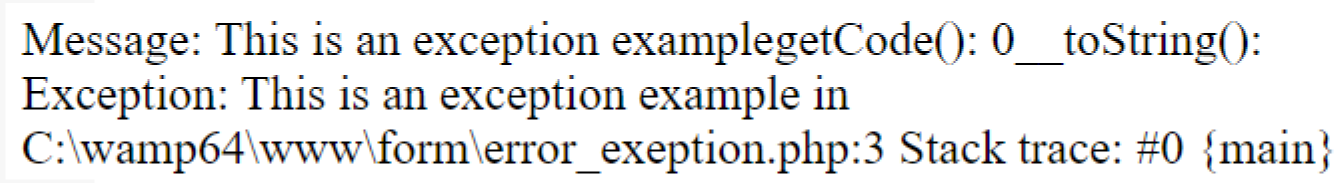
- Exceptions are thrown and intended to be caught while errors are generally irrecoverable.

Method	Description	Example
getMessage()	Displays the exception’s message	<?php echo $e->getMessage(); ?>
getCode()	Displays the numeric code that represents the exception	<?php echo $e->getCode(); ?>
getFile()	Displays the file name and path where the exception occurred	<?php echo $e->getFile(); ?>
getLine()	Displays the line number where the exception occurred	<?php echo $e->getLine(); ?>
getTrace()	Displays an array of the backtrace before the exception	<?php print_r($e->getTrace()); ?>
getPrevious()	Displays the previous exception before the current one	<?php echo $e->getPrevious(); ?>
getTraceAsString()	Displays the backtrace of the exception as a string instead of an array	<?php echo $e->getTraceAsString(); ?>
__toString()	Displays the entire exception as a string	<?php echo $e->__toString(); ?>

PHP Try Catch : Error & Exception Handling

- The throw and catch exceptions

```
<?php
 try { $var_msg = "This is an exception example";
throw new Exception($var_msg); }
 catch (Exception $e) {
echo "Message: " . $e->getMessage();
 echo ""; echo "getCode(): " . $e->getCode();
echo ""; echo "__toString(): " . $e->__toString(); } ?>
```



```
Message: This is an exception examplegetCode(): 0__toString():
Exception: This is an exception example in
C:\wamp64\www\form\error_exeption.php:3 Stack trace: #0 {main}
```

•“try{…}” is the block of code to be executed that could potentially raise an exception
•“catch(Exception $e){…}” is the block of code that catches the thrown exception and assigns the exception object to the variable $e.

PHP Mail

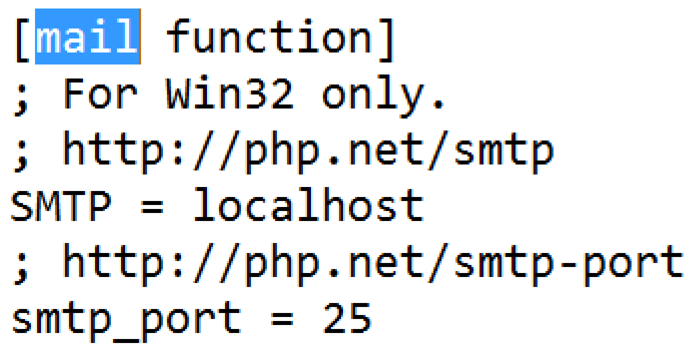
- The mail function accepts the following parameters: Email address , Subject, Message, CC or BC email addresses
- **Sending mail using PHP**

```
<?php mail($to_email_address, $subject, $message, [$headers], [$parameters]); ?>
```

- "$to_email_address" is the email address of the mail recipient
- "$subject" is the email subject
- "$message" is the message to be sent.
- "[$headers]" is optional, it can be used to include information such as CC, BCC
    - CC is the acronym for carbon copy. It's used when you want to send a copy to an interested person i.e. a complaint email sent to a company can also be sent as CC to the complaints board.
    - BCC is the acronym for blind carbon copy. It is similar to CC. The email addresses included in the BCC section will not be shown to the other recipients.

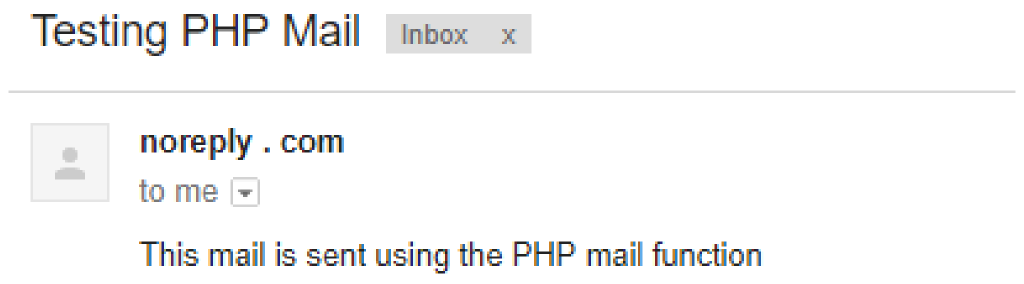
PHP Mail

- **Simple Mail Transmission Protocol (SMTP)**
- PHP mailer uses Simple Mail Transmission Protocol (SMTP) to send mail.
- The SMTP mail settings can be configured from "**php.ini**" file in the PHP installation folder.



```
[mail function]
; For Win32 only.
; http://php.net/smtp
SMTP = localhost
; http://php.net/smtp-port
smtp_port = 25
```

```
<?php
 $to_email = 'name @ company . com';
 $subject = 'Testing PHP Mail';
$message = 'This mail is sent using the PHP mail
function';
$headers = 'From: noreply @ company . com';
mail($to_email,$subject,$message,$headers);
?>
```



Testing PHP Mail Inbox x

noreply . com
to me

This mail is sent using the PHP mail function

PHP Mail

- **Sanitizing email user inputs**
- Users can accidently or intentional inject code in the headers which can result in sending spam mail
- Filter_var function : used to sanitize and validate the user input data.

```
<?php filter_var($field, SANITIZATION TYPE); ?>
```

•"filter_var(...)" is the validation and sanitization function
•"$field" is the value of the field to be filtered.
•"SANITIZATION TYPE" is the type of sanitization to be performed on the field such as;

- **FILTER_VALIDATE_EMAIL** – it returns true for valid email addresses and false for invalid email addresses.
- **FILTER_SANITIZE_EMAIL** – it removes illegal characters from email addresses. info\@domain.(com) returns info@domain.com.
- **FILTER_SANITIZE_URL** – it removes illegal characters from URLs. http://www.example@.comé returns >http://www.example@.com
- **FILTER_SANITIZE_STRING** - it removes tags from string values. <b>am bold</b> becomes am bold.

PHP Mail

```php
<?php
function sanitize_my_email($field) {
    $field = filter_var($field, FILTER_SANITIZE_EMAIL);
    if (filter_var($field, FILTER_VALIDATE_EMAIL)) {
        return true;
    } else {
        return false;
    }
}
$to_email = 'name @ company . com';
$subject = 'Testing PHP Mail';
$message = 'This mail is sent using the PHP mail ';
$headers = 'From: noreply @ company. com';
//check if the email address is invalid $secure_check
$secure_check = sanitize_my_email($to_email);
if ($secure_check == false) {
    echo "Invalid input";
} else { //send email
    mail($to_email, $subject, $message, $headers);
    echo "This email is sent using PHP Mail";
}
?>
```

PHP Connect to DataBase

- **MySQLi extension (the "i" stands for improved)**
- **PDO (PHP Data Objects)**
- PDO work on different database systems, whereas MySQLi will only work with MySQL databases.

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername,
$username, $password);

// Check connection
if ($conn->connect_error) {
  die("Connection failed: " .
$conn->connect_error);
}
echo "Connected successfully";
?>
```

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

try {
  $conn= new PDO("mysql:host=$servername", $username,
$password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
  echo "Connected successfully";
} catch(PDOException $e) {
  echo "Connection failed: " . $e->getMessage();
}
?>
```

PHP Connect to DataBase

Create Database

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";

try {
  $conn = new PDO("mysql:host=$servername", $username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $sql = "CREATE DATABASE myDB";
  // use exec() because no results are returned
  $conn->exec($sql);
  echo "Database created successfully<br>";
} catch(PDOException $e) {
  echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
?>
```

PHP Connect to DataBase

- The PDO is a class that allows us to manipulate different database engines such as MySQL, PostGres, MS SQL Server etc.

Database access method using the PDO object.

```php
<?php
    try {
    $pdo = new PDO("mysql:host=localhost;dbname=myDB",  'username', 'password');
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $pdo->exec('SET NAMES "utf8"');
    $sql_stmt = "SELECT * FROM `my_contacts`";
    $result = $pdo->query($sql_stmt);
 // set the resulting array to associative
    $result->setFetchMode(PDO::FETCH_ASSOC);
    $data = array();
    foreach ($result as $row) {
    $data[] = $row;
  }
print_r($data);
}
catch (PDOException $e) {
    echo $e->getMessage();
}
?>
```

- “try{…catch…}” is the exception handling block
- “$pdo = new PDO("mysql…” creates an instance of the PDO object and passes the database drivers, server and database names, user id and password.
- “$pdo->setAtt…” sets the PDO error mode and exception mode attributes
- “$pdo->exec('SET NA…” sets the encoding format

PHP Connect to DataBase

Insert Data in DataBase

```
<?php

try{
    $pdo = new PDO("mysql:host=localhost;dbname=myDB", ’username', ’password');

    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $requete= "INSERT INTO `my_contacts` (`id`, `full_names`, `gender`, `contact_no`, `email
`, `city`, `country`) VALUES

    (5, 'Ali', 'Male', '555', 'Ali@gmail.com', 'Marrakech', 'Morocco')";
        // use exec() because no results are returned
    $pdo->exec($requete);
    echo"New record created successfully";
}
catch(PDOException $e){
    echo$requete.'<br>'.$e->getMessage();
    }
?>
```

PHP Security Function: strip_tags, filter_var, Md5

- The kinds of attacks that we need to look out for.
- **SQL Injection** – This type of attack appends harmful code to SQL statements.
  - This is done using either user input forms or URLs that use variables.
  - The appended code comments the condition in the WHERE clause of an SQL statement. The appended code can also;
  - insert a condition that will always be true
  - delete data from a table
  - update data in a table
  - This type of attack is usually used to gain unauthorized access to an application.
- **Cross-site scripting –** this type of attack inserts harmful code usually JavaScript. This is done using user input forms such as contact us and comments forms. This is done to;
  - Retrieve sensitive information such as cookies data
  - Redirect the user to a different URL.
  - Other threats can include – PHP code injection, Shell Injection, Email Injection, Script Source Code Disclosure etc.

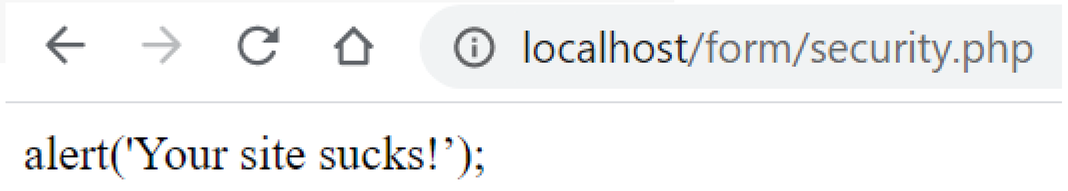
PHP Security Function: strip_tags, filter_var, Md5 and sha1

PHP strip_tags

- The strip_tags functions removes HTML, JavaScript or PHP tags from a string.
- This function is useful when we have to protect our application against attacks such as cross site scripting.

Secure our application from such attacks using strip_tags function.

```
<?php
 $user_input = "<script>alert('Your site sucks!’);</script>";
echo strip_tags($user_input); ?>
```



```
alert('Your site sucks!’);
```

PHP filter_var function

- The filter_var function is used to validate and sanitize data. Validation checks if the data is of the right type. A numeric validation check on a string returns a false result. Sanitization is removing illegal characters from a string. It uses the filter_var function and FILTER_SANITIZE_STRIPPED constant to strip tags.

```
<?php $user_input = "<script>alert('Your site sucks!');</script>";
echo filter_var($user_input, FILTER_SANITIZE_STRIPPED); ?>
```

PHP Security Function: strip_tags, filter_var, Md5 and sha1

- The following SQL statement for validating the user id and password.

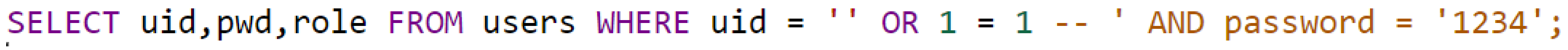
```
<?php SELECT uid,pwd,role FROM users WHERE uid = 'admin' AND password = 'pass'; ?>
```

- A malicious user can enter the following code in the user id text box. ' OR 1 = 1 -- And 1234 in the password text box Let’s code the authentication module

```
<?php $uid = "' OR 1 = 1 -- "; $pwd = "1234";
$sql = "SELECT uid,pwd,role FROM users WHERE uid = '$uid' AND password = '$pwd';";
echo $sql; ?>
```

- “SELECT * FROM users WHERE user_id = ''” tests for an empty user id
- “' OR 1 = 1 “ is a condition that will always be true
- “--" comments that part that tests for the password.

The end result will be



```
SELECT uid,pwd,role FROM users WHERE uid = '' OR 1 = 1 -- ' AND password = '1234';
```

PHP Security Function: strip_tags, filter_var, Md5 and sha1

- mysqli_real_escape_string function is used to protect an application against SQL injection.

To secure our login module:

```
<?php $uid = mysqli_real_escape_string("' OR 1 = 1 -- ");
$pwd = mysqli_real_escape_string("1234");
 $sql = "SELECT uid,pwd,role FROM users WHERE uid = '$uid' AND password = '$pwd’;";
 echo $sql; ?>
```

It will output:

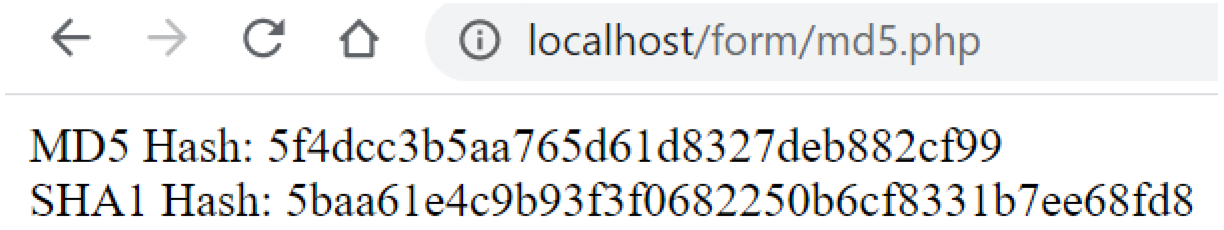
```
SELECT uid,pwd,role FROM users WHERE uid = '\' OR 1 = 1 -- ' AND password = '1234'
```

- *Note the second single quote has been escaped , it will be treated as part of the user id and the password won’t be commented.*

PHP Security Function: strip_tags, filter_var, Md5 and sha1

- **PHP Md5 and PHP sha1**
- Md5 is the acronym for Message Digest 5 and sha1 is the acronym for Secure Hash Algorithm 1.
- They are both used to encrypt strings.
- Once a string has been encrypted, it is tedious to decrypt it.
- Md5 and sha1 are very useful when storing passwords in the database.

```php
<?php
echo "MD5 Hash: " . md5("password");
echo "<br>";
echo "SHA1 Hash: " . sha1("password"
);
?>
```


localhost/form/md5.php

```
MD5 Hash: 5f4dcc3b5aa765d61d8327deb882cf99
SHA1 Hash: 5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8
```

PHP Cookies & Session

- Cookies are small files saved on the user’s computer
- Cookies can only be read from the issuing domain
- Cookies can have an expiry time, if it is not set, then the cookie expires when the browser is closed
- Cookie records your data in the form of an alphanumeric code that is in principle specific to each site, which limits its use.

Exemple

When you browse an e-commerce site, you add certain items to your cart but you do not validate your purchase. The next day, when you return to the same site, without having chosen anything, your cart already contains the items you wanted to buy the day before. The information concerning your purchase intentions was in a cookie retrieved by the site at the time of your connection.

They can also be used to save identifiers, as is the case with Facebook or Twitter cookies. This makes browsing easier for Internet users, who no longer have to go through the authentication page each time they want to consult their news feed.

PHP Cookies & Session

- **Why and when to use Cookies?**

- Cookies allow to track the state of the application using small files stored on the user’s computer.
- The path were the cookies are stored depends on the browser.
- Internet Explorer usually stores them in Temporal Internet Files folder.
- Personalizing the user experience – this is achieved by allowing users to select their preferences.
- The page requested that follow are personalized based on the set preferences in the cookies.
- Tracking the pages visited by a user

PHP Cookies & Session

- **Creating Cookies :** *The php set cookie function must be executed before the HTML opening tag.*

```
<?php
 setcookie(cookie_name, cookie_value, [expiry_time], [cookie_path], [domain], [secure], [httponly]);
?>
```

- Php“**setcookie**” is the PHP function used to create the cookie.
- “cookie_name” is the name of the cookie that the server will use when retrieving its value from the $_COOKIE array variable. It’s mandatory.
- “cookie_value” is the value of the cookie and its mandatory
- “[expiry_time]” is optional; it can be used to set the expiry time for the cookie such as 1 hour. The time is set using the PHP time() functions plus or minus a number of seconds greater than 0 i.e. time() + 3600 for 1 hour.
- “[cookie_path]” is optional; it can be used to set the cookie path on the server. The forward slash “/” means that the cookie will be made available on the entire domain. Sub directories limit the cookie access to the subdomain.
- “[domain]” is optional, it can be used to define the cookie access hierarchy i.e. www.cookiedomain.com
- “[secure]” is optional, the default is false. It is used to determine whether the cookie is sent via https if it is set to true or http if it is set to false.
- “[Httponly]” is optional. If it is set to true, then only client side scripting languages i.e. JavaScript cannot access them.

PHP Cookies & Session

- **Implementation**

cookies.php

```
<?php
setcookie("user_name", "Guru99", time()+
60,'/'); // expires after 60 seconds
 echo 'the cookie has been set for 60 seconds';
?>
```

Output:

```
the cookie has been set for 60 seconds
```

cookies_read.php

```
<?php print_r($_COOKIE); //output the contents
of the cookie array variable ?>
```

Output:

```
Array ( [user_name] => Guru99 )
```

cookie_destroy.php

```
<?php setcookie("user_name", "Guru99", time() - 360,'/'); ?>
```

Note: $_COOKIE is a PHP built in super global variable.
It contains the names and values of all the set cookies.
The number of values that the $_COOKIE array can contain depends on the memory size set in php.ini.
The default value is 1GB.

PHP Cookies & Session

- A session is a global variable stored on the server.
- Each session is assigned a unique id which is used to retrieve stored values.
- Whenever a session is created, a cookie containing the unique session id is stored on the user's computer and returned with every request to the server. If the client browser does not support cookies, the unique php session id is displayed in the URL
- Sessions have the capacity to store relatively large data compared to cookies.
- The session values are automatically deleted when the browser is closed. If we want to store the values permanently, then you should store them in the database.
- Just like the $_COOKIE array variable, session variables are stored in the $_SESSION array variable. Just like cookies, the session must be started before any HTML tags.
- When to use Sessions :
  - Store important information such as the user id more securely on the server
  - Pass values from one page to another
  - Store global variables in an efficient and more secure way compared to passing them in the URL
  - Developing an application such as a shopping cart that has to temporary store information with a capacity larger than 4KB.

PHP Cookies & Session

- Create a session: first call the PHP session_start function and then store values in the $_SESSION array variable.
- Example of the number of times that a page has been loaded

```
<?php
session_start(); //start the PHP_session function
if(isset($_SESSION['page_count'])) {
 $_SESSION['page_count'] += 1;
}
else {
$_SESSION['page_count'] = 1;
 }
echo 'You are visitor number ' . $_SESSION['page_count'];
?>
```

Output:

```
You are visitor number 1
```

PHP Cookies & Session

- **Destroying Session Variables**
- The session_destroy() function is used to destroy the whole Php session variables.
- To destroy only a session single item, then use the unset() function.

```
<?php session_destroy(); //destroy entire session ?>
```

```
<?php unset($_SESSION['product']); //destroy product session item ?>
```

- Session_destroy removes all the session data including cookies associated with the session.
- Unset only frees the individual session variables.

Object-Oriented Programming (OOP)

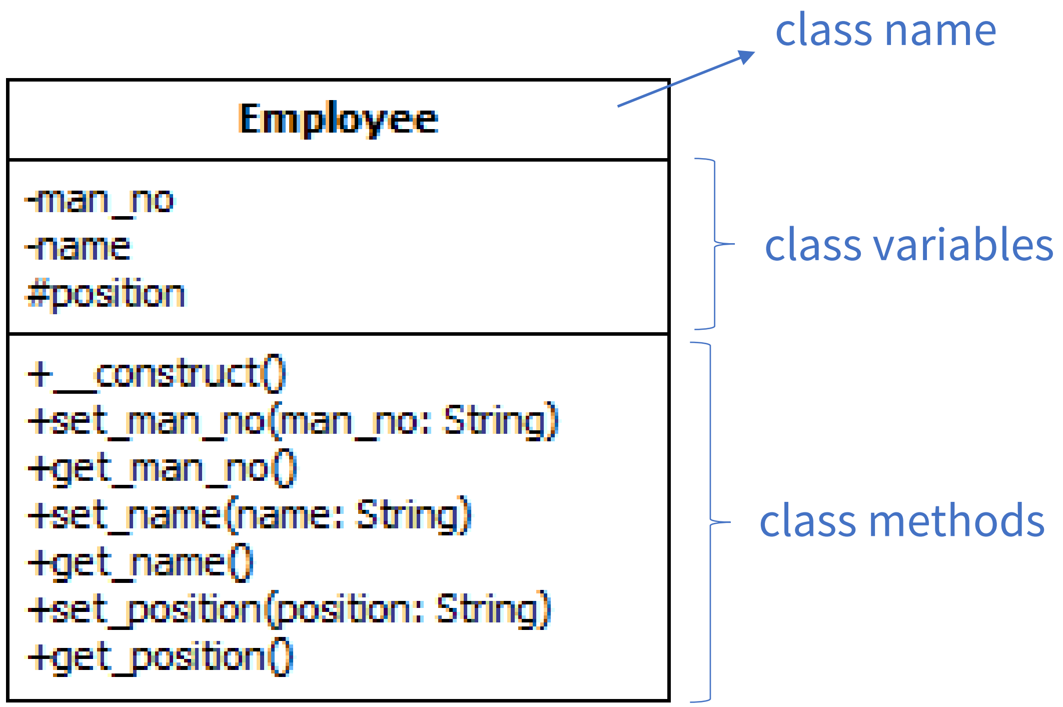
- **Encapsulation** – this is concerned with hiding the implementation details and only exposing the methods. The main purpose of encapsulation is to;
  - Reduce software development complexity – by hiding the implementation details and only exposing the operations, using a class becomes easy.
  - Protect the internal state of an object – access to the class variables is via methods such as get and set, this makes the class flexible and easy to maintain.
  - ➡ via the use of “get” and “set” methods etc.

- **Inheritance** – this is concerned with the relationship between classes. The relationship takes the form of a parent and child. The child uses the methods defined in the parent class. The main purpose of inheritance is;
  - Re-usability– a number of children, can inherit from the same parent. This is very useful when we have to provide common functionality such as adding, updating and deleting data from the database.
  - ➡ via the use of extends keyword

- **Polymorphism** – this is concerned with having a single form but many different implementation ways. The main purpose of polymorphism is;
  - Simplify maintaining applications and making them more extendable.
  - ➡ via the use of implements keyword

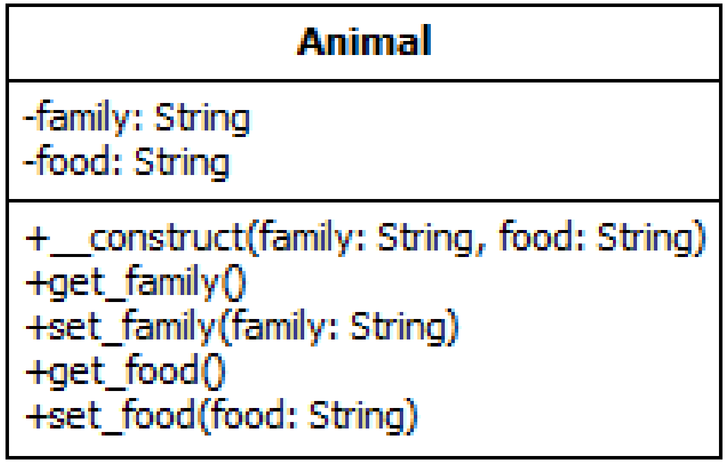
Create a class in PHP

- **Unified Modeling Language UML is a technique used to design and document object oriented systems.**
- UML produces a number of documents, but we will look at the class diagram which is very important to object oriented php programming.
- A class is a representation of real world objects with properties and method
- **Class Diagram Example**



Create a class in PHP: Exemple

```
<?php
 class Animal {
private $family;
 private $food;
 public function __construct($family, $food) {
$this->family = $family;
$this->food = $food; }
public function get_family() {
return $this->family; }
public function set_family($family) {
$this->family = $family; }
public function get_food() {
return $this->food; }
public function set_food($food) {
$this->food = $food; }
} ?>
```



- the variables cannot be accessed directly outside the class (Encapsulation).

•public function __construct($family…)” is called whenever an instance of the class has been created. In this case, we are setting the family and food.

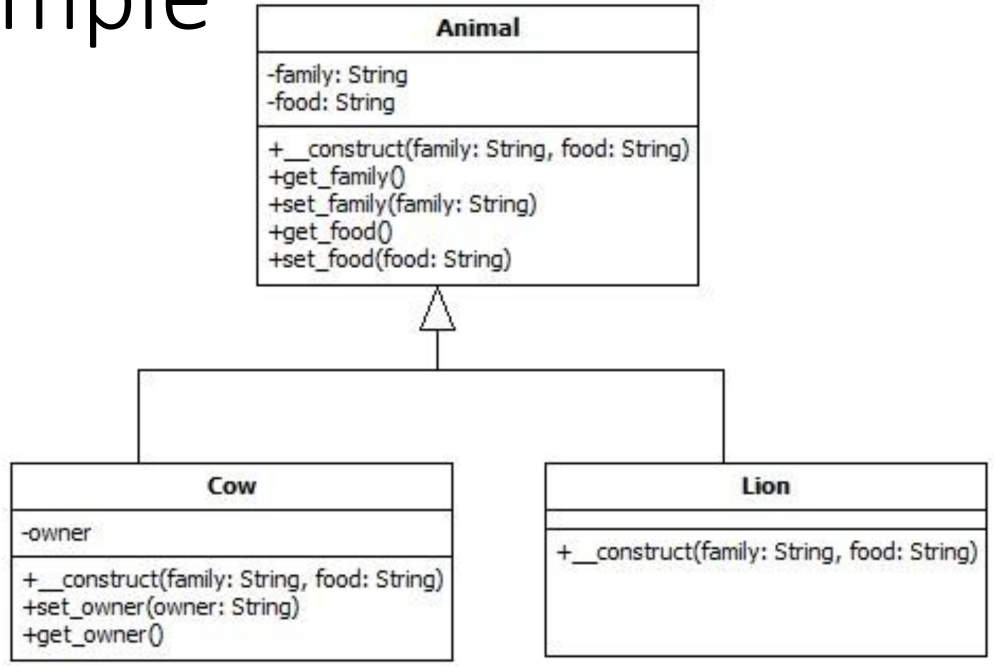
•“public function get…()” is the method used to access the family or food value (Encapsulation)

•“public function set…()” is the method used to set the family or food value (Encapsulation)

Implement Inheritance in PHP: Exemple

```
<?php class Cow extends Animal
 { private $owner;
public function __construct($family, $food) {
parent::__construct($family, $food); }
public function set_owner($owner) {
$this->owner = $owner; }
public function get_owner() {
return $this->owner; }
} ?>
```

```
<?php class Lion extends Animal
{ public function __construct($family, $food) {
parent::__construct($family, $food); }
} ?>
```

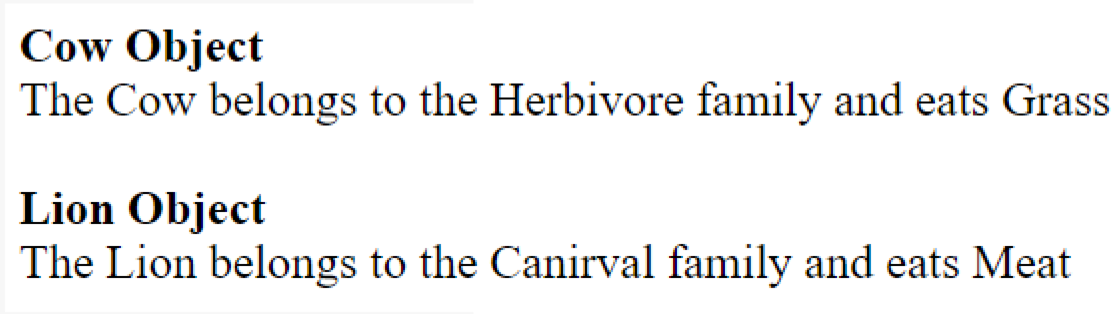


Animal
-family: String
-food: String
+__construct(family: String, food: String)
+get_family()
+set_family(family: String)
+get_food()
+set_food(food: String)
Cow
-owner
+__construct(family: String, food: String)
+set_owner(owner: String)
+get_owner()
Lion
+__construct(family: String, food: String)

- "class … extends Animal" makes the cow and lion use methods from the Animal class (Inheritance).

Create object of the class

```
<?php
require 'Animal.php’;
require 'Cow.php’;
require 'Lion.php’;
$cow = new Cow('Herbivore', 'Grass’);
 $lion = new Lion('Canirval', 'Meat’);
 echo '<b>Cow Object</b> <br>’;
 echo 'The Cow belongs to the ' . $cow->get_family() . ' family
and eats ' . $cow->get_food() . '<br><br>’;
 echo '<b>Lion Object</b> <br>’;
echo 'The Lion belongs to the ' . $lion->get_family() . ' family
and eats ' . $lion->get_food(); ?>
```

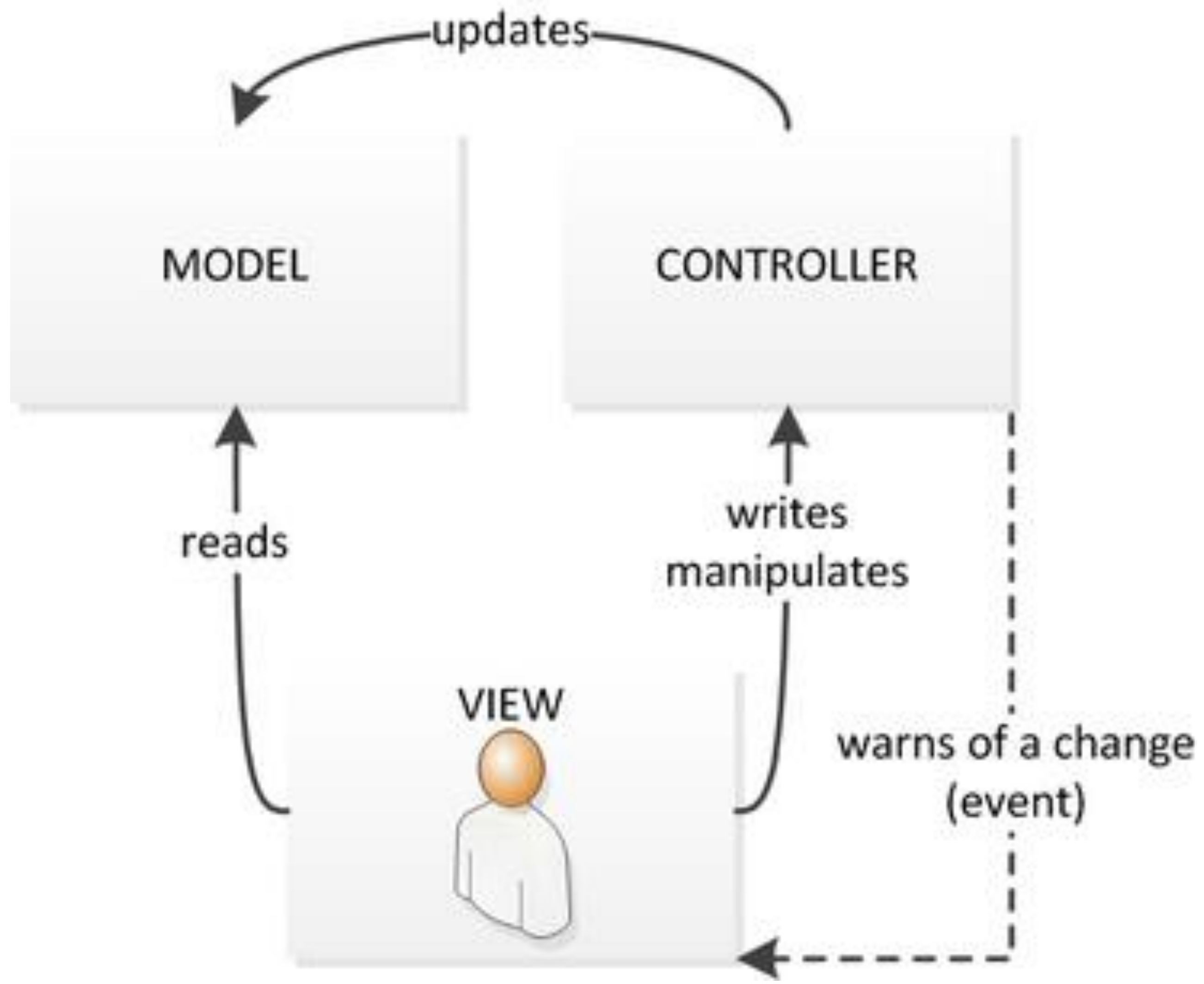


```
Cow Object
The Cow belongs to the Herbivore family and eats Grass

Lion Object
The Lion belongs to the Canirval family and eats Meat
```

Model View Controller: MVC

- The MVC is an architectural pattern that separates an application into 1) Model, 2) View and 3) Controller
- Model: It includes all the data and its related logic
- View: Present data to the user or handles user interaction
- Controller: An interface between Model and View components. It interprets the mouse and keyboard inputs from the user, informing model and the view to change as appropriate.


updates
MODEL
CONTROLLER
reads
writes
manipulates
VIEW
warns of a change
(event)