

OOP and Ajax in PHP

ITCS 210 Web Programming



Class Objectives

- Students would be able to know **concepts** of object oriented programming (OOP) in PHP, AJAX, the **difference between Synchronous and Asynchronous**, model-view-controller (MVC) architecture paradigm.
- Students should be able to **apply** AJAX in web applications.
- Knowing **frameworks** that use to develop web application based on MVC.

OOP in PHP and Built-in functions

Require and Include

- You can insert the content of one PHP file into another PHP file before the server executes it, with the `include()` or `require()` function.

- **require** – includes and evaluates a specific file; **failure → Fatal Error**

```
<?php
    require 'header.php' ;
?>
```

- **include** - includes and evaluates a specific file; **failure → a Warning**

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

require_once and include_once

- **require_once** – same as `require` except if the file has already been included, it will not be included again

```
<?php
    require_once 'header.php' ;
?>
```

- **include_once** - same as `include` except if the file has already been included, it will not be included again

```
<?php
    include_once 'header.php' ;
?>
```

- Use when the same file might be included and evaluated more than once during a particular execution of a script, and you want to be sure that it is included exactly once to avoid problems with function redefinitions, variable value reassignments, etc.

Example of include

Main file: index.php

```
<html>
<body>
<div class="leftmenu">
<?php include("Menu.php"); ?>
</div>
<h1>Welcome to my home
page.</h1></body>
</html>
```

Menu.php

```
<a href="/default.php">Home</a>
<a href="/about.php">About Us</a>
<a href="/contact.php">Contact Us</a>
```

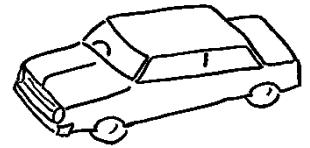
Result in browser after PHP codes are executed

```
<html>
<body>
<div class="leftmenu">
<a href="/default.php">Home</a>
<a href="/about.php">About Us</a>
<a href="/contact.php">Contact Us</a>
</div>
<h1>Welcome to my home page!</h1>
</body>
</html>
```

Object

- An object is a bunch of variables and functions all lumped into a single entity.
- The object can then be called rather than calling the variables or functions themselves. Within an object there are **methods** and **properties**.
- The methods are functions that manipulate data within the object.
- The properties are variables that hold information about the object.

Class



Class: vehicle

- Class is the blueprint for user defined object
- In OOP, the class holds the definition, and the object holds the value.
- Class can contain
 - Object Property
 - Method
- For clearer idea about class and object

All vehicles share similar characteristics

number of doors
color
price

Properties

All vehicles do similar things

Drive
turn left
turn right
stop

Methods

2 doors
Red
\$34,000



Object: mini cooper

Drive
turn left
turn right
stop

5 doors
Blue
\$29,285



Object: Ford

Drive
turn left
turn right
stop

Basic Syntax

- Creating a new object

```
$someVariable = new SomeClassDefiningAnObject;
```

- Executing a method in the object

```
$someVariable->someMethod($someArgumentJustLikeARegularFunction);
```

- Assigning the return value to a variable

```
$returnValue = $someVariable->someMethodThatReturnsSomething();
```

- Setting and retrieving the current property value

```
$someVariable->someProperty = 'SomeValue';  
$currentValue = $someVariable->someProperty;
```

Defining Class

```
1.class Dog
2.{
3.    public $hungry = 'yeah.';
4.
5.    function eat($food)
6.    {
7.        $this->hungry = 'not so much.';
8.    }
9.}
```

Property

Method

```
1.$dog = new Dog;
2.echo $dog->hungry;
3.
4.$dog->eat('cookie');
5.echo $dog->hungry;
```

Output

```
yeah.
not so much.
```



Class

▪ Syntax

```
class classname
```

```
{
```

```
var $property_name;
```

← Object Property

```
function __construct(parameters)
```

← Constructor

or

```
function classname(parameters)
```

```
{ code }
```

```
function __destruct()
```

← Destructor

```
{ code }
```

```
function func_name()
```

← Method

```
{ code }
```

```
}
```

- Constructor is a method automatically called when an object is created. It is usually used for object property initialization.
- Destructor is a method automatically called when all references to a particular object are removed or when the object is explicitly destroyed or in any order in shutdown sequence.

Example

```
class student
{
    var $student_id;
    function student($param) //constructor for PHP Version < 5.3.2
    {
        $this->student_id = $param;
    } //end constructor

    function show_id(){
        echo "student id = " . $this->student_id;
    } //end function
}

$stu = new student(46330001);           // create instance
$stu->show_id();                         // show student id
```

How to use

- Use class by creating object

Syntax: `$objName = new className(parameter);`

`$Kitty = new student(5288999);`

- Refer to an property in the class using the symbol

Syntax: `$objName -> propertyName = "value";`

`$Kitty -> student_id = "5288000";` *//change value of property*

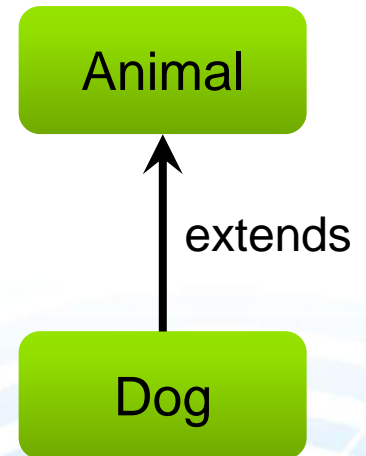
- Use method in the class

Syntax: `$objName -> methodName(parameter);`

`$Kitty -> show_id();` *//call a method*

Inheritance

```
1.class Animal {
2.    public $hungry = 'yeah.';
3.
4.    function eat($food) {
5.        $this->hungry = 'not so much.';
6.    }
7.}
8.
9.class Dog extends Animal {
10.    function eat($food) {
11.        if($food == 'cookie') {
12.            $this->hungry = 'not so much.';
13.        }
14.        else {
15.            echo 'barf, I only like cookies!';
16.        }
17.    }
18.}
```

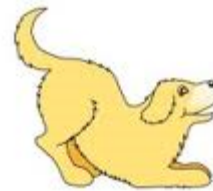


Constructor

```
1.class Dog extends Animal
2.{
3.    public $breed;
4.
5.    function __construct($breed){
6.        $this->breed = $breed;
7.    }
8.
9.    function eat($food){
10.        if($food == 'cookie'){
11.            $this->hungry = 'not so much.';
12.        }
13.        else {
14.            echo 'barf, I only like cookies!';
15.        }
16.    }
17.}
```

Two Underscores

```
$dog = new Dog('Golden Retriever');
$dog->breed = 'Golden Retriever';
```



Scope Resolution Operator

```
12. class Dog extends Animal {  
13.     public $breed;  
14.  
15.     function __construct($breed){  
16.         $this->breed = $breed;  
17.  
18.         Animal::__construct();  
19.  
20.     }  
21.
```

```
22.     function eat($food){  
23.         if($food == 'cookie'){  
24.             Animal::eat($food);  
25.         }  
26.         else {  
27.             echo 'barf, I only like cookies!';  
28.         }  
29.     }  
30. }  
31.  
32. $dog = new Dog('Rotweiler');  
33. $dog->eat('cookie');  
34. echo $dog->hungry;
```

```
1. class Animal {  
2.     public $hungry = 'yeah.';  
3.  
4.     function __construct(){  
5.         echo 'I am an animal.';  
6.     }  
7.  
8.     function eat($food){  
9.         $this->hungry = 'not so much.';  
10.    }  
11. }
```

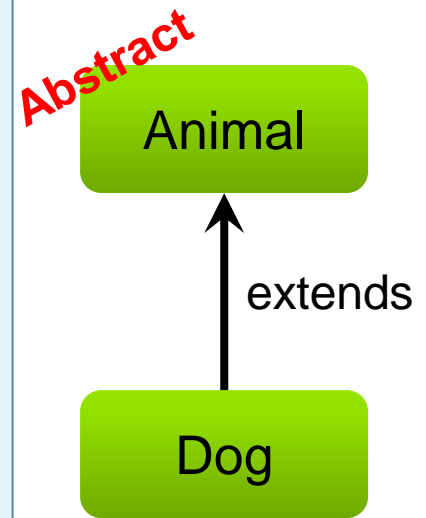
Perform static calls to methods & class members

Output

```
I am an animal.  
not so much.
```

Abstract Class

```
1.abstract class Animal {
2.    public $hungry = 'yeah.';
3.
4.    abstract public function eat($food);
5.}
6.class Dog extends Animal{
7.    function eat($food){
8.        if($food == 'cookie'){
9.            $this->hungry = 'not so much.';
10.        }
11.        else {
12.            echo 'barf, I only like cookies!';
13.        }
14.    }
15.}
16.$dog = new Dog();
17.echo $dog->hungry;
18.$dog->eat('peanut');
```

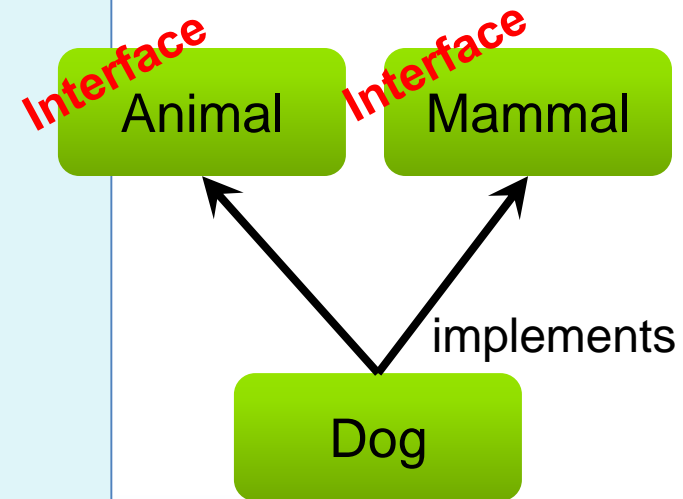


Output

```
yeah.
barf, I only like cookies!.
```

Interfaces

```
1.interface Animal {
2.   public function eat($food);
3.}
4.interface Mammal {
5.   public function giveBirth();
6.}
7.class Dog implements Animal, Mammal{
8.   public $gender = 'male';
9.
10.  function eat($food){
11.      if($food == 'cookie'){
12.          $this->hungry = 'not so much.';
13.      }
14.      else {
15.          echo 'barf, I only like cookies!';
16.      }
17.  }
18.  function giveBirth(){
19.      if($this->gender == 'male'){
20.          echo 'I can\'t, I am a boy :P';
21.      }
22.      else {
23.          echo 'I\'m not even pregnant yet.';
24.      }
25.  }
26.}
```

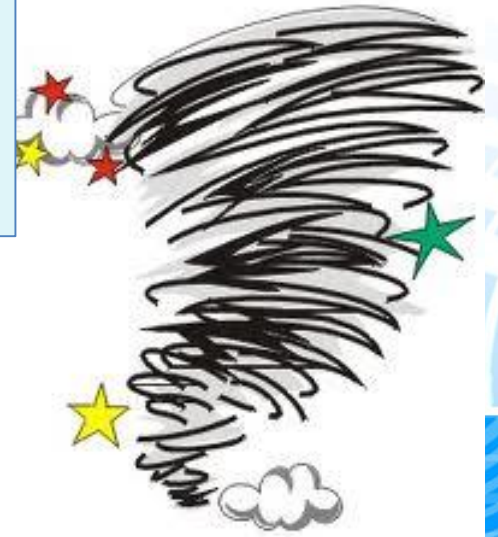


Destructor

```
1.class Example {  
2.     private $_name;  
3.  
4.     public function __construct($name){  
5.         $this->_name = $name;  
6.     }  
7.     function __destruct() {  
8.         echo "Destructing object '$this->_name'.";  
9.     }  
10.}  
11.  
12.$objectOne = new Example('Object one');  
13.$objectTwo = new Example('Object two');  
14.unset($objectOne);  
15.echo 'Script still running.';  
16.unset($objectTwo);
```

Output

```
Destructing object 'Object one'.  
Script still running.  
Destructing object 'Object two'.
```



Visibility

```
1.class Teeth {
2.     protected $_colour = 'white';
3.
4.     public function stain(){
5.         $this->_colour = 'yellow';
6.     }
7.}
8.class Dog {
9.     public $teeth;
10.
11.     public function __construct(){
12.         $this->teeth = new Teeth();
13.     }
14.     public function eat($food){
15.         if($food == 'cookie'){
16.             $this->hungry = 'not so much.';
17.             //Attempt to turn teeth green:
18.             $this->teeth->_colour = 'green';
19.         }
20.         else {
21.             echo 'barf, I only like cookies!';
22.         }
23.     }
24.}
25.$dog = new Dog();
```

Output

```
Fatal error: Cannot access protected property
Teeth::$_colour
```

- **Public:**
available to every classes
- **Protected:**
available to parent and child classes
- **Private:**
Available to its own class

Constants

```
1.class Dog{
2.    const NUMBER_OF_LEGS = '4';
3.
4.    public function __construct(){
5.        echo 'I have '.self::NUMBER_OF_LEGS.' legs,
6.        and you can\'t take that away from me!';
7.    }
8.}
9.$dog = new Dog();
```

`$this -> NUMBER_OF_LEGS` ❌

`$dog -> NUMBER_OF_LEGS` ❌

PHP is looking for a non-existent object

Output

I have 4 legs, and you can't take that away from me!

I have legs, and you can't take that away from me!



Exception

```
1.class LiarException extends Exception {}
2.
3.try {
4.    if($doggy->talk() == 'Doggie likes broccoli.'){
5.        throw new LiarException(
6.            'Doggie is a big fat liar. He only likes cookies.'
7.        );
8.    }
9.    else {
10.        throw new Exception('Just because we can.');
```

```
1.class Dog {
2.    public function talk(){
3.        return 'Doggie likes something.';
4.    }...
```

Output

Somebody threw an exception: Just because we can.

ToString

- `__toString` method can be used to show a string from an object

```
1.class Dog
2.{
3.    function __toString()
4.    {
5.        return 'I am Dog.';
6.    }
7.}
8.$dog = new Dog();
9.echo $dog;
```

Output

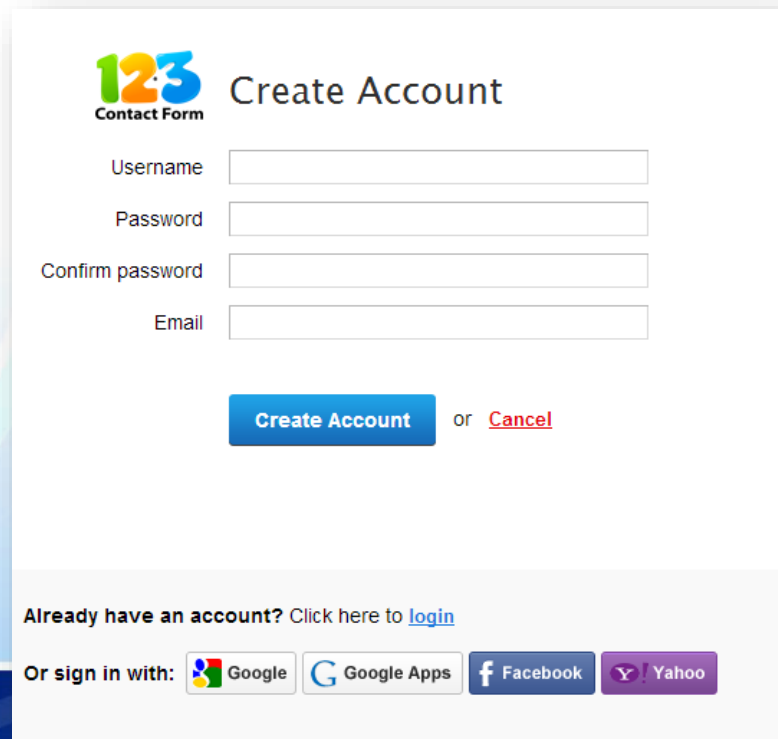
I am Dog.

fatal error: Object of class Dog could not be converted to string

AJAX in PHP

AJAX

- Asynchronous JavaScript and XML
- To build asynchronous web application



123
Contact Form

Create Account

Username





Password

Confirm password

Email

[Create Account](#) or [Cancel](#)

Already have an account? Click here to [login](#)

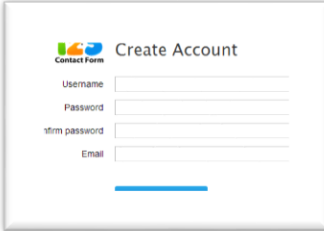
Or sign in with:  Google  Google Apps  Facebook  Yahoo



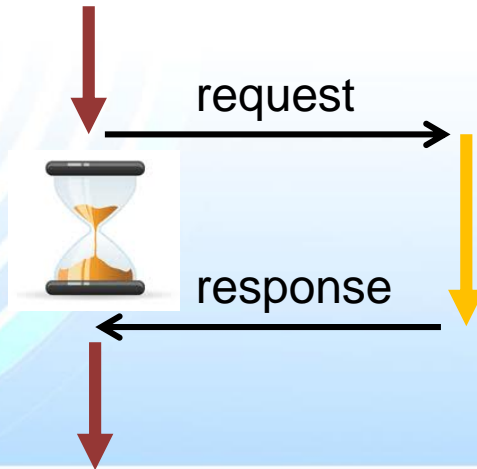
AJAX: Asynchronous

- Synchronous
Client

Server

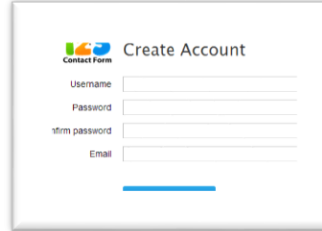


A 'Create Account' form with fields for Username, Password, Confirm password, and Email. It includes a 'Contact Form' logo and a blue submit button.

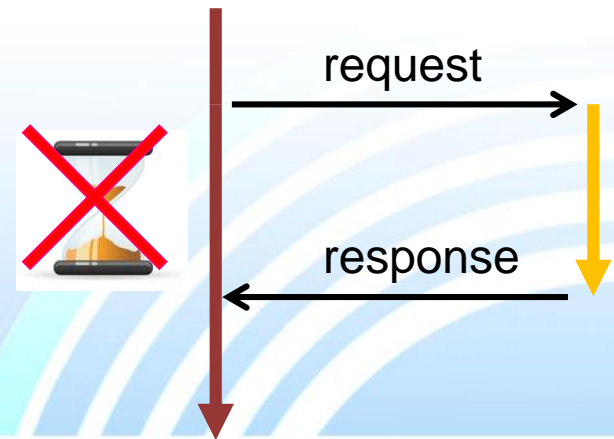


- Asynchronous
Client

Server

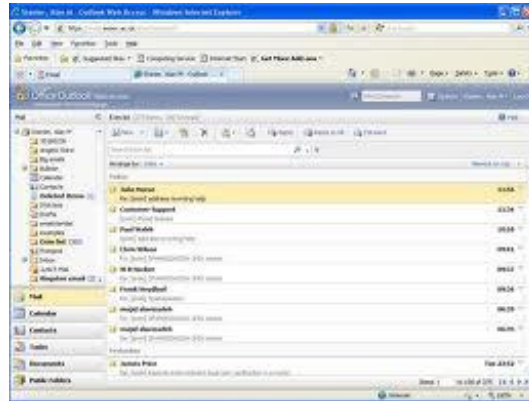


A 'Create Account' form with fields for Username, Password, Confirm password, and Email. It includes a 'Contact Form' logo and a blue submit button.

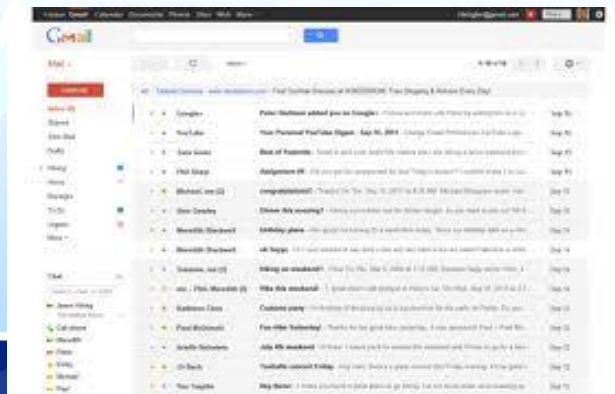


Ajax: Example

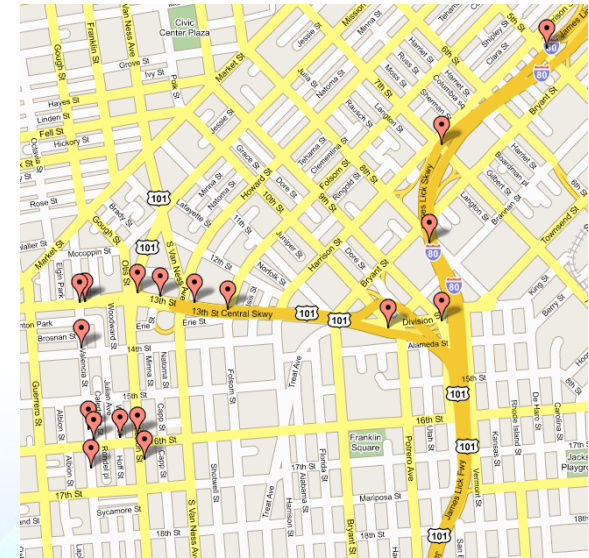
- Microsoft Outlook Web Access



- Gmail



- Google Maps



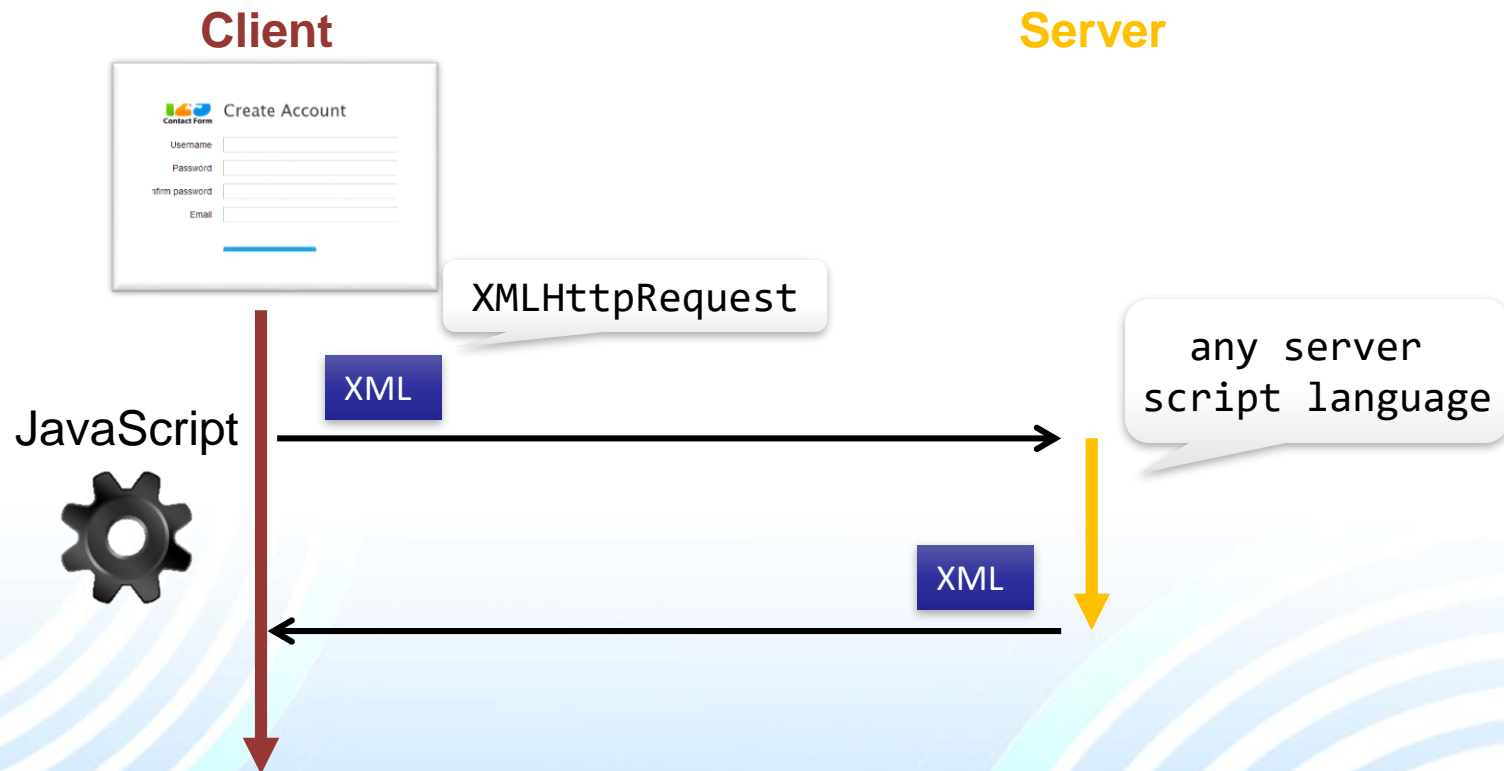
- Google Suggest



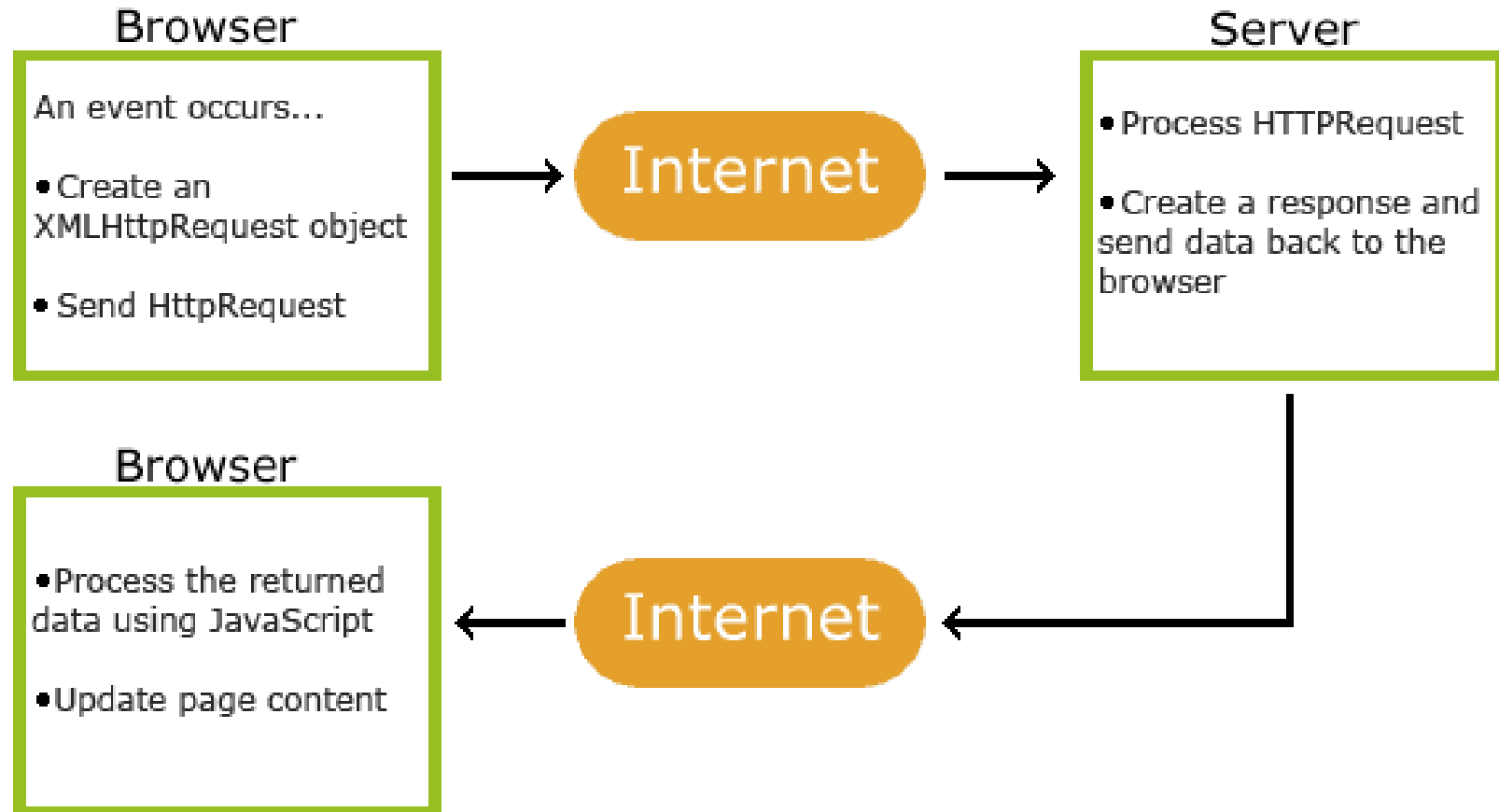
Ajax

- The Ajax engine works within the Web browser
- Through JavaScript and the **DOM** (Document Object Model)
- **XML** is commonly used as the format for receiving server data (In this lecture, we will show response in **text**)
- AJAX is a web browser technology independent of web server software
- A user can **continue** to use the application while the client program requests information from the server in the background

Ajax: JavaScript and XML



How Ajax works



How Ajax works

1. An event occurs in a web page (the page is loaded, a button is clicked)
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read by JavaScript
7. Proper action (like page update) is performed by JavaScript

Ajax: Create XMLHttpRequest Object

- The **XMLHttpRequest** Object can be used to exchange data with a server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.
- Create an XMLHttpRequest Object

variable = new XMLHttpRequest();

Example

```
var xhttp = new XMLHttpRequest();
```

XMLHttpRequest Object Methods

Method	Description
<code>new XMLHttpRequest()</code>	Creates a new XMLHttpRequest object
<code>abort()</code>	Cancels the current request
<code>getAllResponseHeaders()</code>	Returns header information
<code>getResponseHeader()</code>	Returns specific header information
<code>open(method, url, async, user, psw)</code>	Specifies the request <i>method</i> : the request type GET or POST <i>url</i> : the file location <i>async</i> : true (asynchronous) or false (synchronous) <i>user</i> : optional user name <i>psw</i> : optional password
<code>send()</code>	Sends the request to the server Used for GET requests
<code>send(string)</code>	Sends the request to the server. Used for POST requests
<code>setRequestHeader()</code>	Adds a label/value pair to the header to be sent

XMLHttpRequest Object Properties

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

Ajax: Send a Request to a Server

```
xmlhttp.open("GET","getcd.php?q="+str,true);  
xmlhttp.send();
```

Syntax

```
open(method, url, async)
```

- *method*: the type of request: GET or POST
url: the location of the file on the server
async: true (asynchronous) or false (synchronous)
- GET : simpler & faster
- POST: sending large amount of data, secure

GET or POST?

- **GET** is simpler and faster than **POST**, and can be used in most cases.
- However, **always** use POST requests when:
 - A cached file is not an option (update a file or database on the server).
 - Sending a large amount of data to the server (POST has no size limitations).
 - Sending user input (which can contain unknown characters), POST is more robust and secure than GET.

Ajax: Send a Request to a Server

Get Request

- A simple GET request:

```
xhttp.open("GET", "demo_get.php", true);  
xhttp.send();
```

- If you want to send information with the GET method, add the information to the URL:

```
xhttp.open("GET", "demo_get2.php?fname=Henry&lname=Ford", true);  
xhttp.send();
```

Ajax: Send a Request to a Server

Post Request

- A simple POST request:

```
xhttp.open("POST", "demo_post.php", true);  
xhttp.send();
```

- To POST data like an HTML form, add an HTTP header with `setRequestHeader()`. Specify the data you want to send in the `send()` method:

```
xhttp.open("POST", "ajax_test.php", true);  
xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");  
xhttp.send("fname=Henry&lname=Ford");
```

Ajax: Send a Request to a Server

```
xhttp.open("GET", "ajax_test.php", true);
```

- The **url** parameter of the open() method, is an address to a file on a server
- Server requests should be sent **asynchronously**. The async parameter of the open() method should be set to true
 - execute other scripts while waiting for server response
 - deal with the response after the response is ready

Ajax: Server Response

- Server Response Properties

Property	Description
responseText	get the response data as a string
responseXML	get the response data as XML data

- Server Response Methods

Method	Description
getResponseHeader()	Returns specific header information from the server resource
getAllResponseHeaders()	Returns all the header information from the server resource

Ajax: Server Response

- The onreadystatechange function is called every time the readyState changes.
- When readyState is 4 and status is 200, the response is ready.

```
function loadDoc() {  
    var xhttp = new XMLHttpRequest();  
    xhttp.onreadystatechange = function() {  
        if (this.readyState == 4 && this.status == 200) {  
            document.getElementById("demo").innerHTML =  
                this.responseText;  
        }  
    };  
    xhttp.open("GET", "ajax_info.txt", true);  
    xhttp.send();  
}
```

The.responseText Property

```
<!DOCTYPE html>
<html>
<body>

<div id="demo">
<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Change Content</button>
</div>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
</script>

</body>
</html>
```

The XMLHttpRequest Object

Change Content

ajax_info.txt



AJAX

AJAX is not a programming language.

AJAX is a technique for accessing web servers from a web page.

AJAX stands for Asynchronous JavaScript And XML.

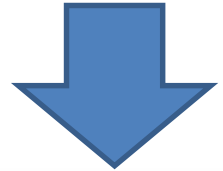
The responseXML Property

```
<!DOCTYPE html>
<html>
<body>
<div id="demo">
<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadXML()">Change Content</button>
</div>
<script>
function loadXML() {
  var xmlhttp, xmlDoc, txt, x, i;
  xmlhttp = new XMLHttpRequest();
  xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      xmlDoc = this.responseXML;
      txt = "";
      x = xmlDoc.getElementsByTagName("ARTIST");
      for (i = 0; i < x.length; i++) {
        txt = txt + x[i].childNodes[0].nodeValue + "<br>";
      }
      document.getElementById("demo").innerHTML = txt;
    }
  };
  xmlhttp.open("GET", "cd_catalog.xml", true);
  xmlhttp.send();
}
</script>
</body>
</html>
```

The XMLHttpRequest Object

Change Content

Cd_catalog.xml



Bob Dylan
Bonnie Tyler
Dolly Parton
Gary Moore
Eros Ramazzotti
Bee Gees
Dr.Hook
Rod Stewart
Andrea Bocelli
Percy Sledge
Savage Rose
Many
Kenny Rogers
Will Smith

Ajax: Form Suggestion: Form

Start typing a name in the input field below:

First name:

Suggestions:



Start typing a name in the input field below:

First name:

Suggestions: Eva , Eve , Evita , Elizabeth , Ellen

```
1.<p><b>Start typing a name in the input field below:</b></p>
2.<form>
3.First name: <input type="text" onkeyup="showHint(this.value)" size="20" />
4.</form>
5.<p>Suggestions: <span id="txtHint"></span></p>
```

1

Ajax: Form Suggestion: JavaScript

2

```
1. <script>
2. function showHint(str) {
3.   if (str.length==0) {
4.     document.getElementById("txtHint").innerHTML="";
5.     return;
6.   }
7.
8.   if (window.XMLHttpRequest) { // code for IE7+,Firefox,Chrome,Opera,Safari
9.     xmlhttp = new XMLHttpRequest();
10.  }
11.  else { // code for IE6, IE5
12.    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
13.  }
14.
15.  xmlhttp.onreadystatechange=function() {
16.    if (xmlhttp.readyState==4 && xmlhttp.status==200) {
17.      document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
18.    }
19.  }
20.
21.  xmlhttp.open("GET","gethint.php?q="+str,true);
22.  xmlhttp.send();
23.}
24.</script>
```

Ajax: Form Suggestion: PHP

3

```
1.<?php
2.// Fill up array with names
3.$a[]="Anna";
4.$a[]="Brittany";
5.$a[]="Cinderella";
6....
7.$a[]="Elizabeth";
8.$a[]="Ellen";
9.$a[]="Wenche";
10.$a[]="Vicky";
11.
12.$q=$_GET["q"];
13.
14.if (strlen($q) > 0) {
15.    $hint="";
16.    for($i=0; $i<count($a); $i++) {
17.        if (strtolower($q)==strtolower(substr($a[$i],0,strlen($q)))) {
18.            if ($hint=="")
19.                $hint=$a[$i];
20.            else
21.                $hint=$hint." , ".$a[$i];
22.        }
23.    }
24. }
25. // Set output to "no suggestion"
26. // if no hint were found
27. // or to the correct values
28.if ($hint == "")
29.    $response="no suggestion";
30.else
31.    $response=$hint;
32.
33.//output the response
34.echo $response;
35.?>
```


Ajax: Form Suggestion

2

```
<html>
<head>
<script>
function showHint(str)
{
if (str.length==0)
{
document.getElementById("txtHint").innerHTML="";
return;
}
if (window.XMLHttpRequest)
{// code for IE7+, Firefox, Chrome, Opera, Safari
xmlhttp=new XMLHttpRequest();
}
else
{// code for IE6, IE5
xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
xmlhttp.onreadystatechange=function()
{
if (xmlhttp.readyState==4 && xmlhttp.status==200)
{
document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
}
}
xmlhttp.open("GET","gethint.php?q="+str,true);
xmlhttp.send();
}
</script>
</head>
<body>
```

1

```
<p><b>Start typing a name in the input field below:</b></p>
<form>
First name: <input type="text" onkeyup="showHint(this.value)" size="20" />
</form>
<p>Suggestions: <span id="txtHint"></span></p>
```

```
</body>
</html>
```

3

```
<?php
// Fill up array with names
$a[]="Anna";
$a[]="Brittany";
...

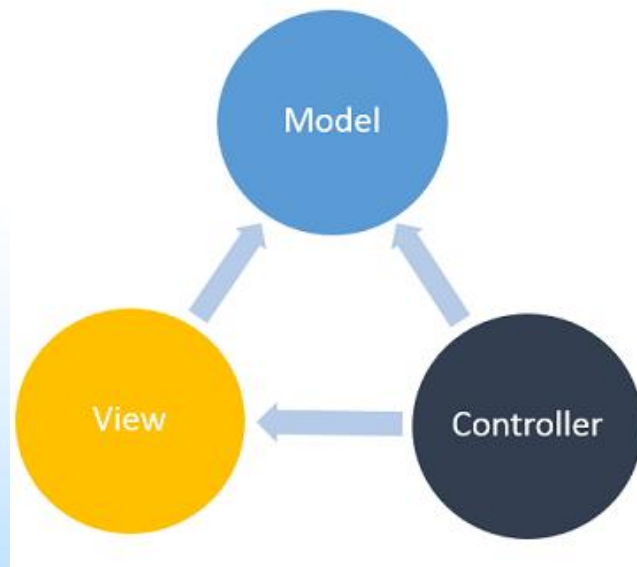
//get the q parameter from URL
$q=$_GET["q"];

//lookup all hints from array if length of q>0
if (strlen($q) > 0) {
$hint="";
for($i=0; $i<count($a); $i++){
if (strtolower($q)==strtolower(substr($a[$i],0,strlen($q)))){
if ($hint==""){
$hint=$a[$i];
}
else{
$hint=$hint." , ".$a[$i];
}
}
}
}

if ($hint == "")
$response="no suggestion";
else
$response=$hint;

//output the response
echo $response;
?>
```

MVC in Web Programming



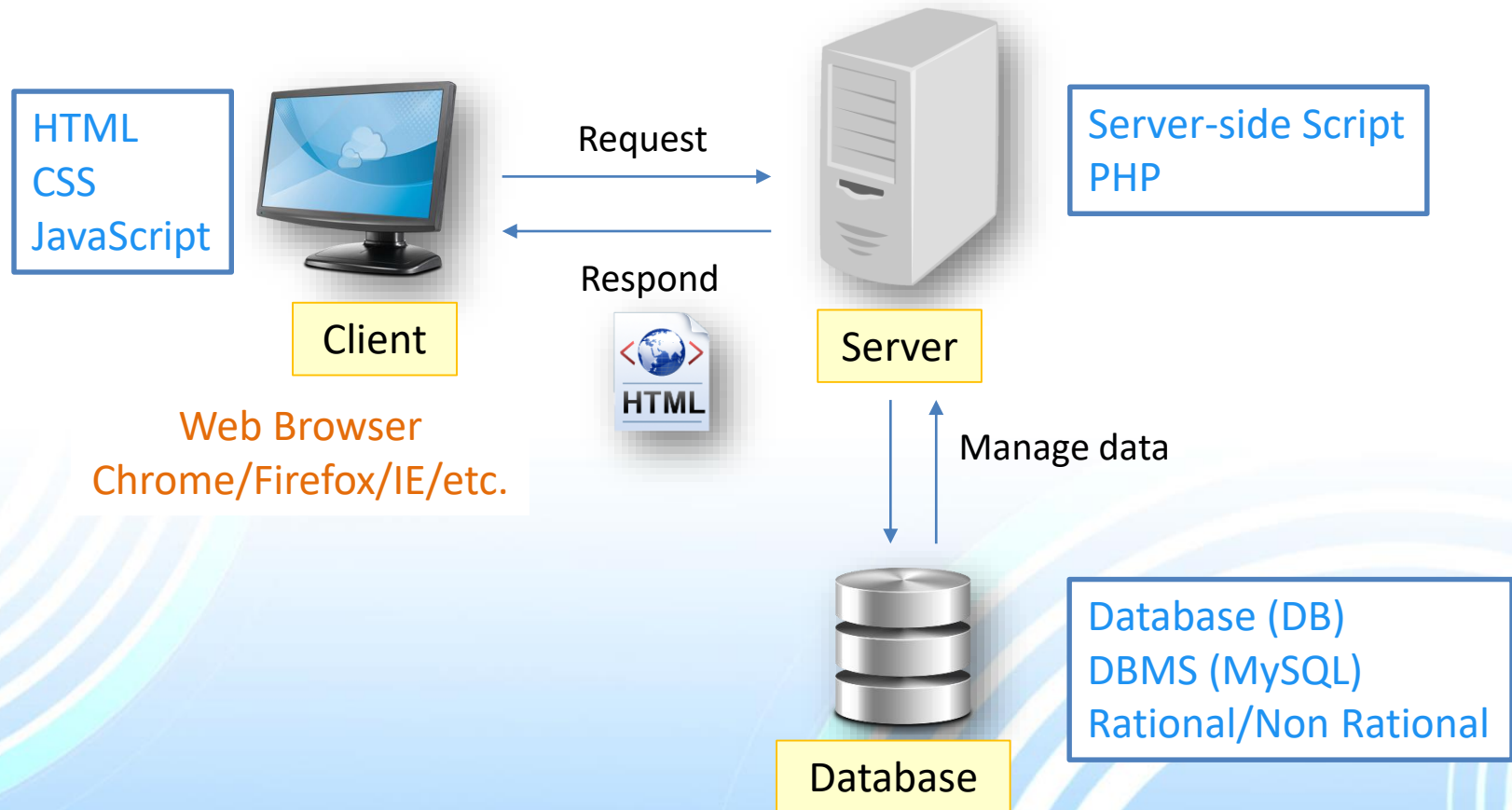
MVC (Model-View-Controller)

- A software **architectural**, or **design pattern**, for implementing user interfaces.
- Although originally developed for desktop computing, model–view–controller has been widely adopted as an architecture for World Wide Web applications in major programming languages such as PHP, ASP.NET, etc.
- adapted MVC to different contexts.
 - hierarchical model–view–controller (HMVC),
 - model–view–adapter (MVA),
 - model–view–presenter (MVP),
 - model–view–viewmodel (MVVM)

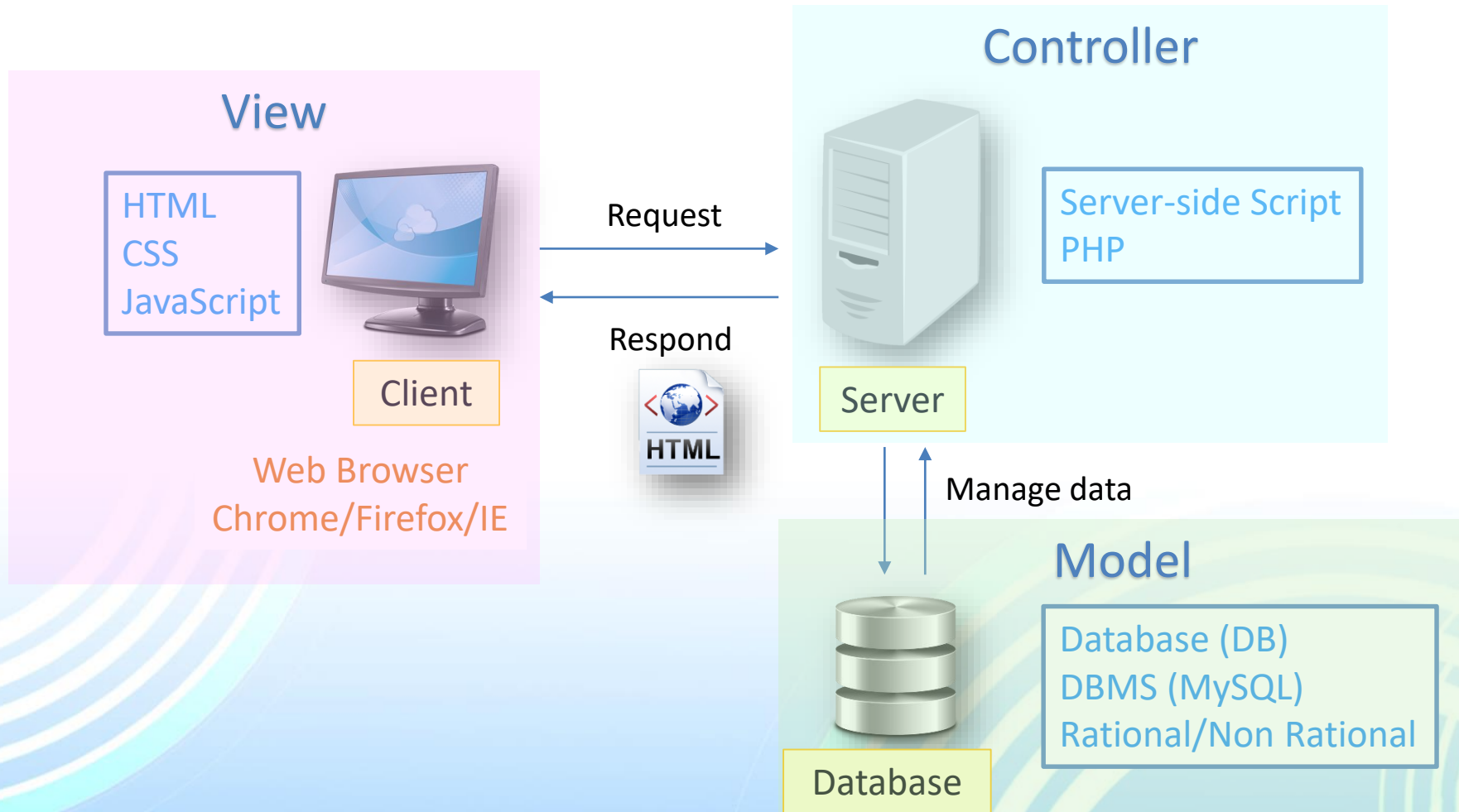
MVC (Model-View-Controller)

- It divides a web application into three interconnected parts, Model-View-Controller, which make it powerful, scalable, clean, and robust.
- Dividing the component based on responsibility.
- Efficient code reuse and parallel development

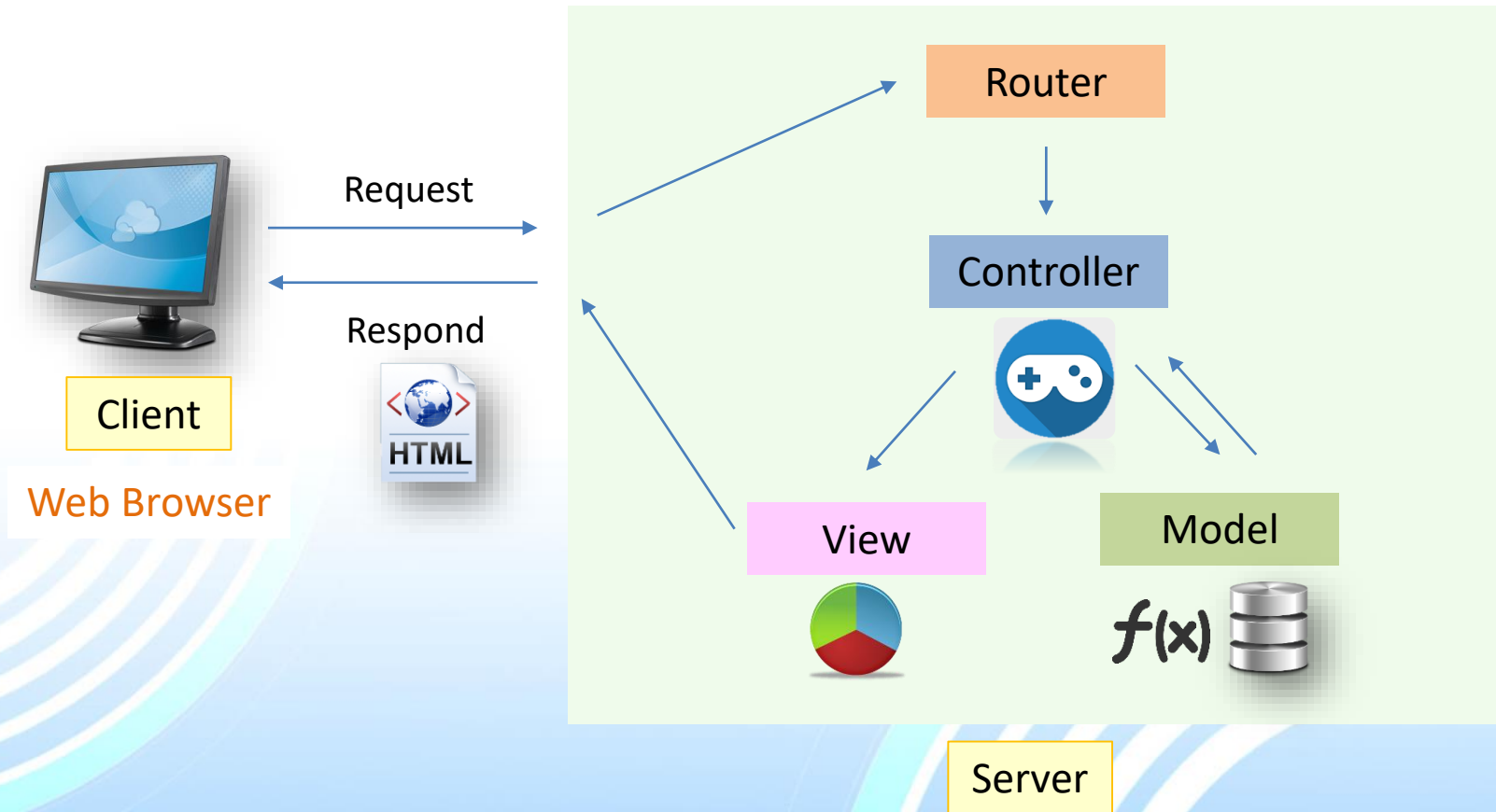
The flow based on responsibility



The flow based on responsibility



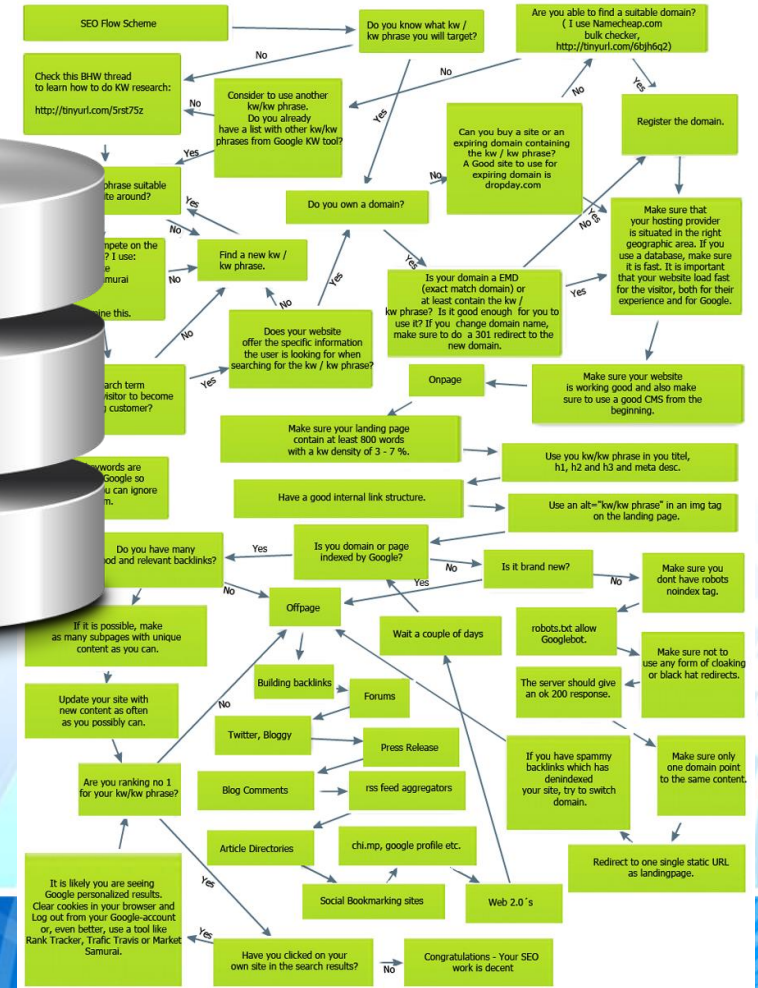
MVC Architecture



MVC: Model

- Application data
- Business rules
- Logic
- Functions

$f(x)$



MVC: Model

- Directly manages the **data, logic and rules** of the application
- Adding and retrieving data from database
- Processing data from or to database
- Speaking only with the Controller (not to the view directly)
- Independent of user interface

MVC: View

- Representation of data
 - Chart or diagram
 - Table
 - Article



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam eget leo in urna tincidunt eleifend. Ut aliquet eleifend neque, a commodo leo viverra vel. Curabitur justo erat, rutrum in ornare laoreet, venenatis nec ipsum. Sed nisi nisl, scelerisque vitae dapibus a, sagittis sit amet nunc. Integer vitae gravida tellus. Aenean dapibus non velit id elementum. Donec venenatis sed orci ac tristique.

MVC: View

- Output **presentation**, e.g., Charts/ Diagrams/ Content
- Provide multiple view of the same information
- Using HTML and CSS
- The only thing that user ever sees
- Only listen to the Controller

MVC: Controller

- Receives input
- Connects between models and views (Middle man)



MVC: Controller

- Responsible for **handling HTTP request**
- Apply server side logic and process GET/POST requests
- Take information from user
- Talk to model to get data
- Tell view how to display

“We need SMART Models,
THIN Controllers,
and DUMB Views”

Unknown

Example of the flow in MVC

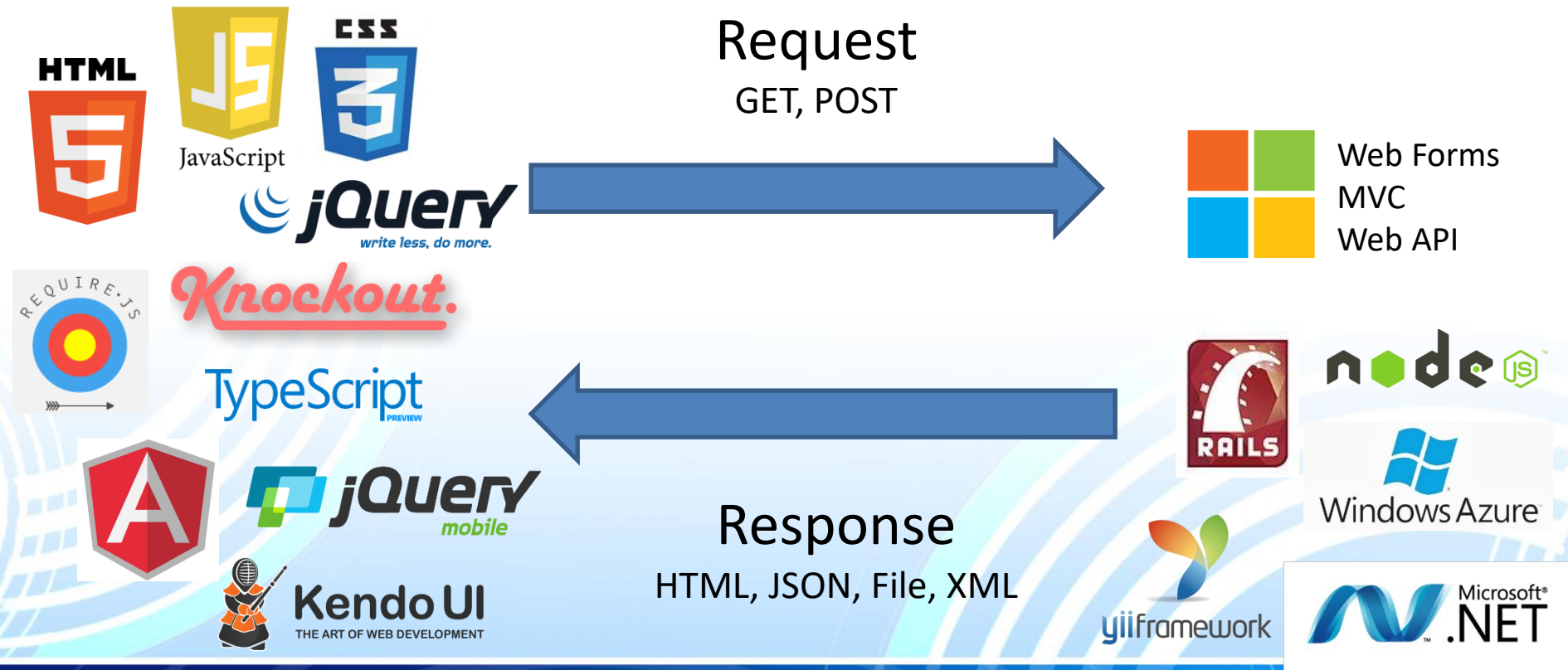
- A user interacts with the **view** - by clicking on a link or submitting a form.
- The **Controller** handles the user input, and transfers the information to the **model**
- The **Model** receives the information and updates it's state (adds data to a database, for example, or calculates today's date)
- The **View** checks the state of the Model and responds accordingly (listing the newly entered data, maybe)
- The **View** waits for another interaction from the user.

Framework: Implementing MVC

- A very good implementation of MVC is the DRY (**Don't Repeat Yourself**) philosophy.
- Universal & Reusable software platform to develop applications
- Include support programs, compilers, code libraries, tools, or APIs



More abstractions, libraries, frameworks on top of HTTP



Examples of implementing frameworks

- Yii



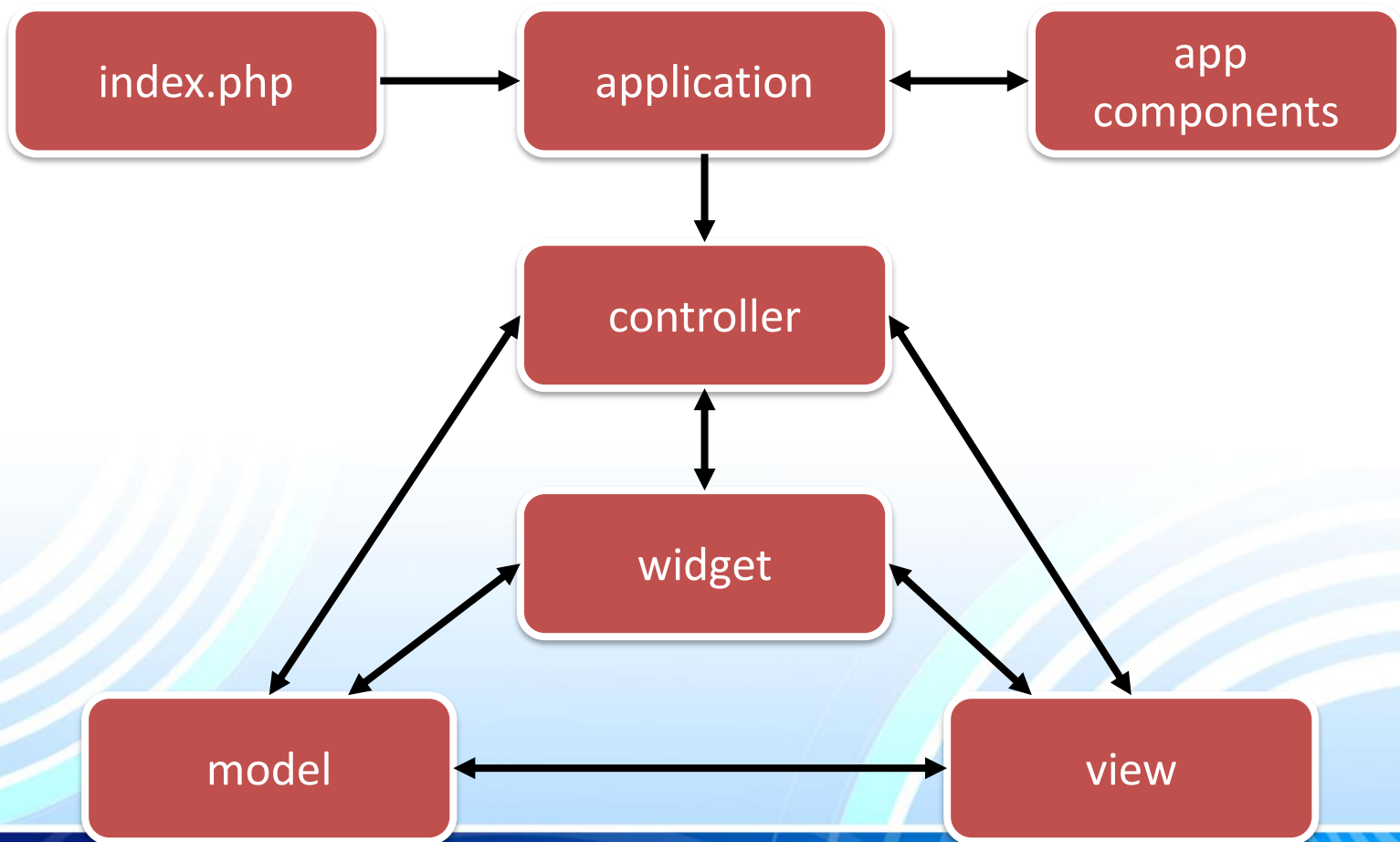
- ASP.NET



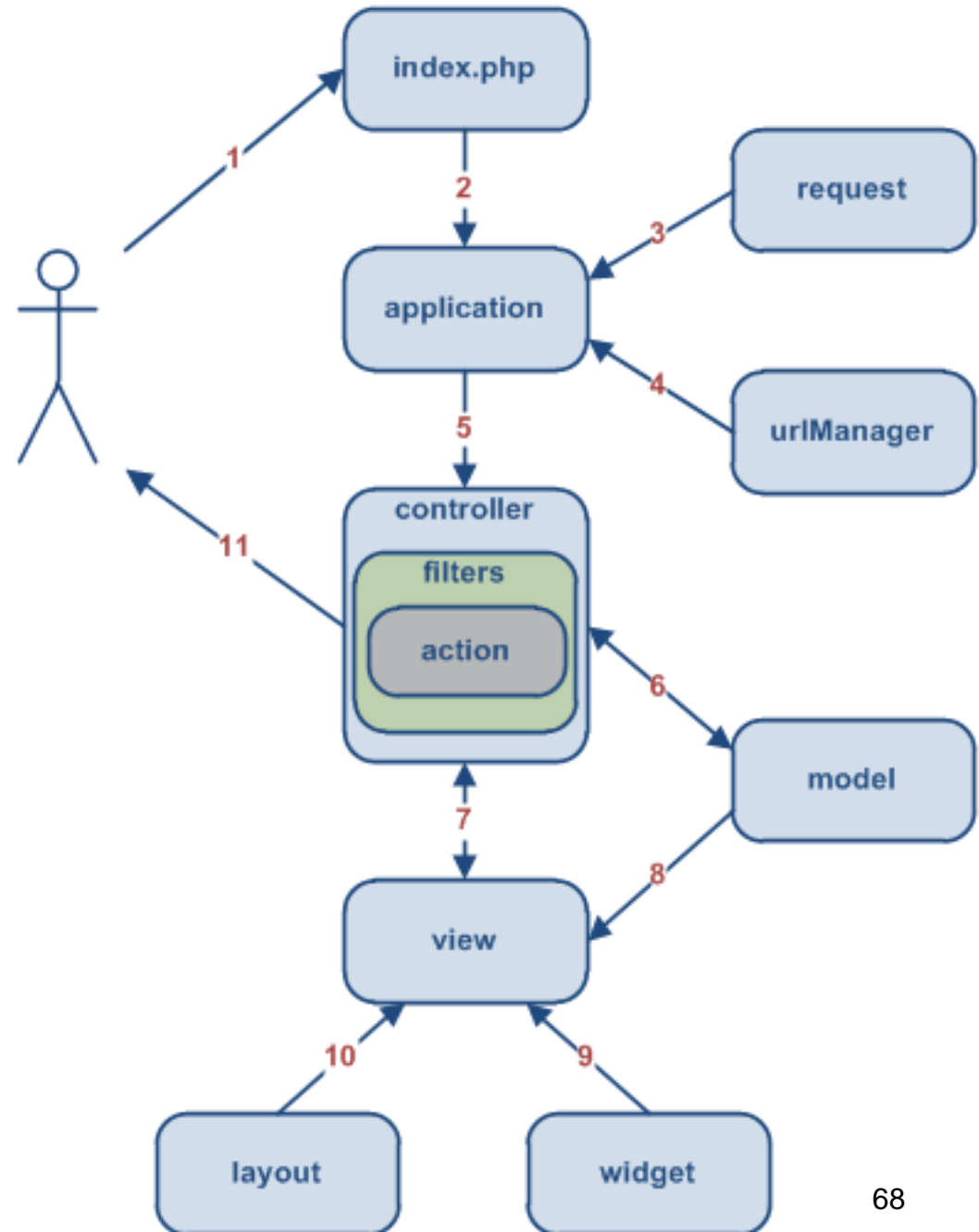
Yii Framework

- Yii is a high-performance, component-based PHP framework for developing large-scale Web applications rapidly.
- It enables maximum reusability in Web programming and can significantly accelerate your Web application development process.
- The name **Yii** (pronounced Yee or [ji:]) is an acronym for "Yes It Is!".

Yii Structure



Typical Workflow



Yii Structure: protected

controllers/
 SiteController.php

containing controller class files
the default controller class

data/
 schema.mysql.sql
 schema.sqlite.sql
 testdrive.db

containing the sample database
the DB schema for the sample MySQL database
the DB schema for the sample SQLite database
the sample SQLite database file

extensions/
messages/

containing third-party extensions
containing translated messages

models/
 LoginForm.php
 ContactForm.php

containing model class files
the form model for 'login' action
the form model for 'contact' action

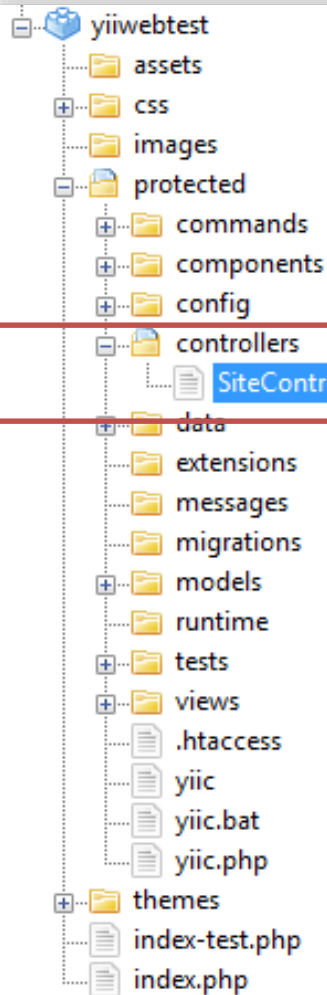
runtime/
tests/

containing temporarily generated files
containing test scripts

views/
 layouts/
 main.php
 column1.php
 column2.php
 site/
 pages/
 about.php
 contact.php
 error.php
 index.php
 login.php

containing controller view and layout files
containing layout view files
the base layout shared by all pages
the layout for pages using a single column
the layout for pages using two columns
containing view files for the 'site' controller
containing "static" pages
the view for the "about" page
the view for 'contact' action
the view for 'error' action (displaying external errors)
the view for 'index' action
the view for 'login' action

Controller



Controller under
“controllers” folder

Controller ID

```
class SiteController extends CController
{
    public function actionIndex()
    {
        // ...
    }

    public function actionContact()
    {
        // ...
    }
}
```

Action ID

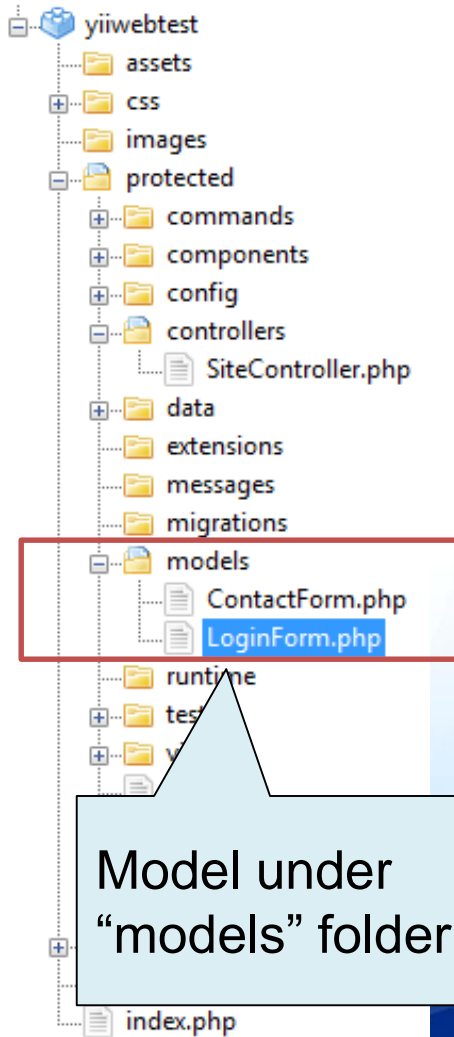
Controller

```
class SiteController extends CController
{
    public function actionIndex()
    {
        // ...
    }

    public function actionContact()
    {
        // ...
    }
}
```

- When a controller runs, it performs the **requested action**, which usually brings in the **needed models** and renders an **appropriate view**.
- When the user request does not specify which action to execute, the **default action, index**, will be executed.

Model



```
class LoginForm extends CFormModel
```

```
{
```

```
public $username;  
public $password;  
public $rememberMe=false;  
  
private $_identity;
```

Model attributes

```
public function rules()
```

```
{
```

```
return array(  
    array('username, password', 'required'),  
    array('rememberMe', 'boolean'),  
    array('password', 'authenticate'),  
);
```

Validation Rules

```
    array('username, password', 'required'),  
    array('rememberMe', 'boolean'),  
    array('password', 'authenticate'),  
);
```

```
}
```

```
public function authenticate($attribute,$params)
```

```
{
```

```
    $this->_identity=new UserIdentity($this->username,$this->password);  
    if(!$this->_identity->authenticate()  
        $this->addError('password','Incorrect username or password.');
```

```
}
```

```
}
```

Method in model

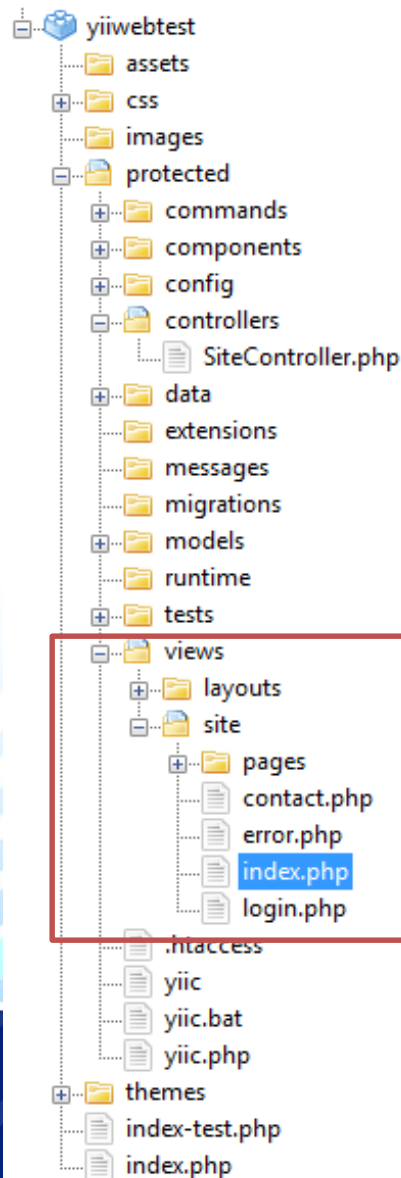
Model rules validation

```
public function rules()
{
    return array(
        array('username, password', 'required'),
        array('password_repeat', 'required', 'on'=>'register'),
        array('password', 'compare', 'compareAttribute'=>'password_repeat', 'on'=>'register'),
    );
}
```

```
array(
    // mandatory arguments
    'attribute list',
    'validator name',
    // optional parameters
    'on'=>'scenario name',
    'message'=>'The attribute didn\'t validate!',
    ...validation parameters...
);
```

- *attribute list*: specifies the attributes (separated by commas) to be validated;
- *validator name*: specifies the validator to be used. See the [next section](#) for details.
- *on*: this specifies the scenarios when the validation rule should be performed. Separate different scenarios with commas. If this option is not set, the rule will be applied in any scenario. See the section [Scenarios] [#Scenarios] for details.
- *message*: replaces the default error message if validation fails.
- *...validation parameters...*: any number of extra parameters to be used by the specified validator.

View



```
1 <?php
2 /* @var $this SiteController */
3
4 $this->pageTitle=Yii::app()->name;
5 ?>
6
7 <h1>Welcome to <i><?php echo CHtml::encode(Yii::app()->name); ?></i></h1>
8
9 <p>Congratulations! You have successfully created your Yii application.</p>
10
11 <p>You may change the content of this page by modifying the following two files:</p>
12 <ul>
13     <li>View file: <code><?php echo __FILE__; ?></code></li>
14     <li>Layout file: <code><?php echo $this->getLayoutFile('main'); ?></code></li>
15 </ul>
16
17 <p>For more details on how to further develop this application, please read
18 the <a href="http://www.yiiframework.com/doc/">documentation</a>.
19 Feel free to ask in the <a href="http://www.yiiframework.com/forum/">forum</a>,
20 should you have any questions.</p>
21
```

My Web Application

[Home](#) [About](#) [Contact](#) [Login](#)

Welcome to My Web Application

Congratulations! You have successfully created your Yii application.

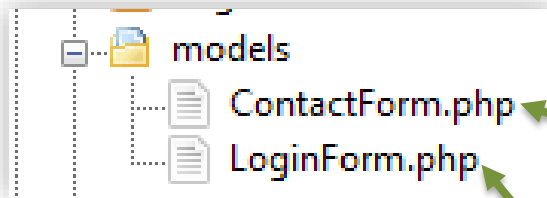
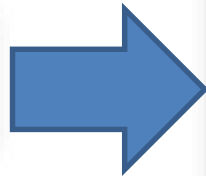
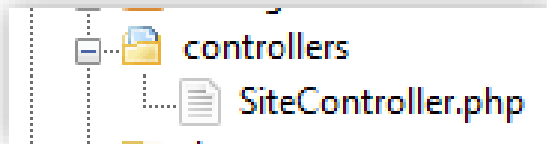
You may change the content of this page by modifying the following two files:

- View file: C:\Uniserver\www\yiiwebtest\protected\views\site\index.php
- Layout file: C:\Uniserver\www\yiiwebtest\protected\views\layouts\main.php

For more details on how to further develop this application, please read the [documentation](#). Feel free to ask in the [forum](#), should you have any questions.

Copyright © 2013 by My Company.
All Rights Reserved.
Powered by [Yii Framework](#).

MVC Relation in Yii



```
<?php

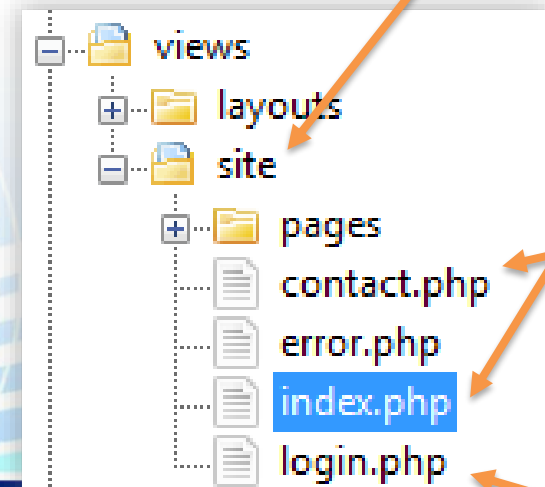
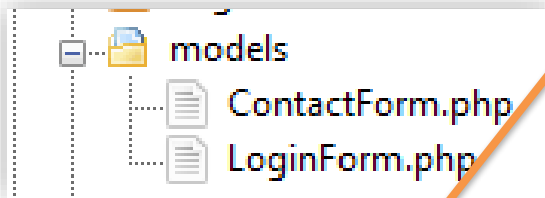
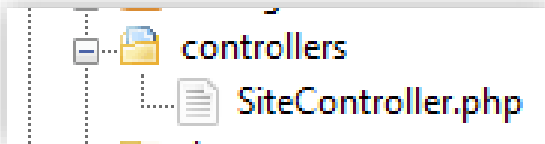
class SiteController extends Controller
{
    /** Declares class-based actions. ... */
    public function actions() {...}

    /** This is the default 'index' action that is invoked ... */
    public function actionIndex()
    {
        // renders the view file 'protected/views/site/index.php'
        // using the default layout 'protected/views/layouts/main.php'
        $this->render('index');
    }

    /** Displays the contact page ... */
    public function actionContact()
    {
        $model=new ContactForm;
        if(isset($_POST['ContactForm']))
        {
            $model->attributes=$_POST['ContactForm'];
            if($model->validate()) {...}
        }
        $this->render('contact',array('model'=>$model));
    }

    /** Displays the login page ... */
    public function actionLogin()
    {
        $model=new LoginForm;
        // if it is ajax validation request
        if(isset($_POST['ajax']) && $_POST['ajax']==='login-form'){...}
        // collect user input data
        if(isset($_POST['LoginForm'])){...}
        // display the login form
        $this->render('login',array('model'=>$model));
    }
}
```


MVC Relation in Yii



```
<?php
class SiteController extends Controller
{
    /** Declares class-based actions. ...*/
    public function actions(){...}

    /** This is the default 'index' action that is invoked ...*/
    public function actionIndex()
    {
        // renders the view file 'protected/views/site/index.php'
        // using the default layout 'protected/views/layouts/main.php'
        $this->render('index');
    }

    /** Displays the contact page ...*/
    public function actionContact()
    {
        $model=new ContactForm;
        if(isset($_POST['ContactForm']))
        {
            $model->attributes=$_POST['ContactForm'];
            if($model->validate()){...}
        }
        $this->render('contact',array('model'=>$model));
    }

    /** Displays the login page ...*/
    public function actionLogin()
    {
        $model=new LoginForm;
        // if it is ajax validation request
        if(isset($_POST['ajax']) && $_POST['ajax']==='login-form'){...}
        // collect user input data
        if(isset($_POST['LoginForm'])){...}
        // display the login form
        $this->render('login',array('model'=>$model));
    }
}
```


Contact Page

My Web Application

[Home](#) [About](#) [Contact](#) [Login](#)

[Home](#) » [Contact](#)

Contact Us

If you have business inquiries or other questions, please fill out the following form to contact us. Thank you.


*Fields with * are required.*

Name *

Email *

Subject *

Body *

Verification Code
 [Get a new code](#)

Please enter the letters as they are shown in the image above.
Letters are not case-sensitive.

URL

`http://localhost/yiiwebtest/index.php?r=site/contact`

Controller ID

Action ID

Redirect

```
$this->redirect(array('controllerID/actionID',  
    'var1' => $value1,  
    'var2' => $value1));
```

For Example

```
$this->redirect(array('changeEmailResult','scb'=>$successBool,  
    'email'=>$studentModel->m_email));
```

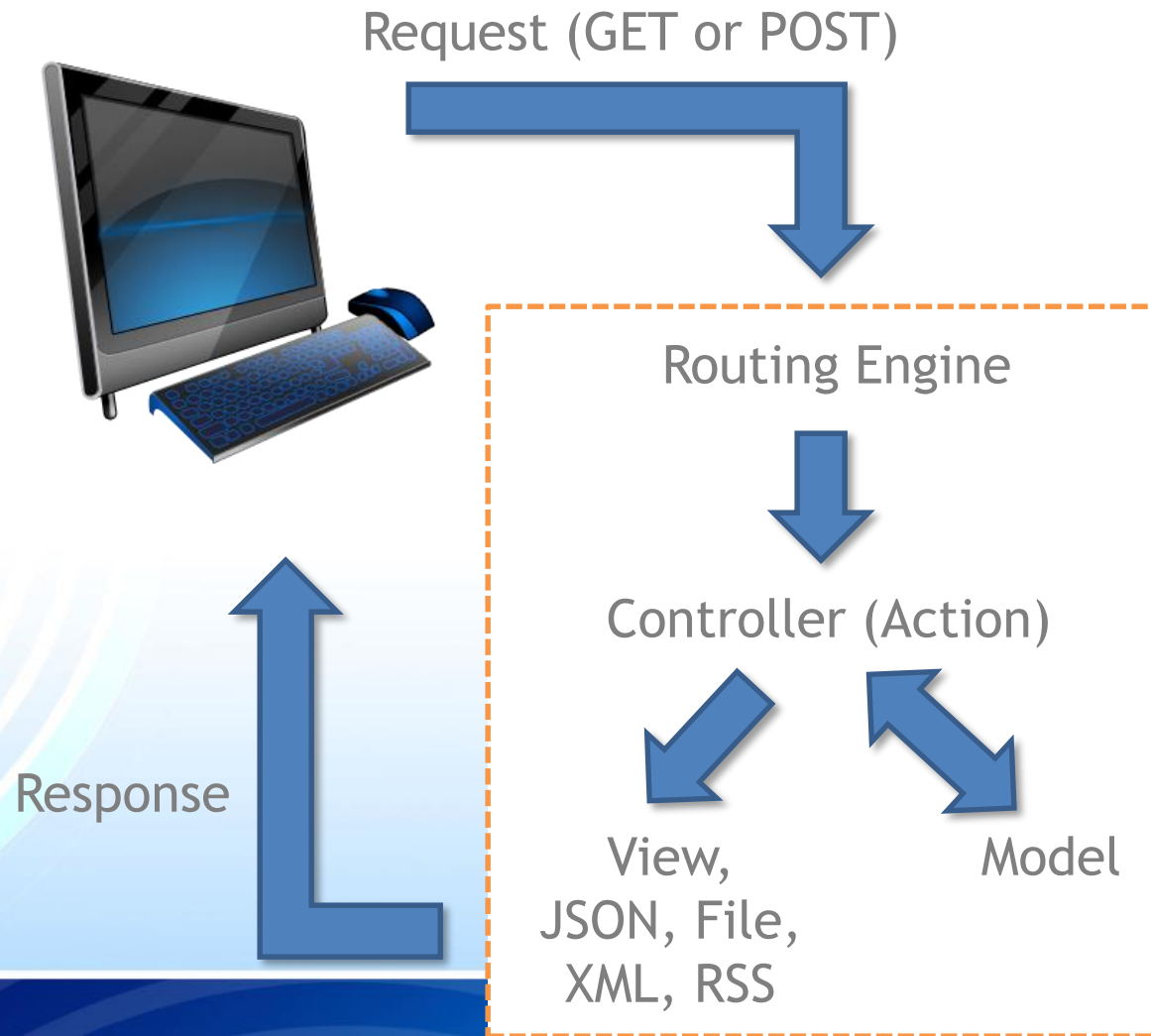
ASP.NET

- ASP.NET is an open-source **server-side web application framework** designed for web development to produce dynamic web pages. It was developed by Microsoft to allow programmers to build dynamic web sites, web applications and web services.
- ASP.NET offers three frameworks for creating web applications: ASP.NET **Web Forms**, ASP.NET **MVC**, and ASP.NET **Web Pages**.

ASP.NET MVC

- ASP.NET MVC can make it easier to manage complexity in larger applications.
- Multiple teams can work on a web site because the code for the business logic is **separate** from the code and markup for the presentation layer.
- Developers can work on the business logic while designers work on the markup and JavaScript that is sent to the browser.

ASP.NET MVC



ASP.NET MVC

Style Sheets, Images

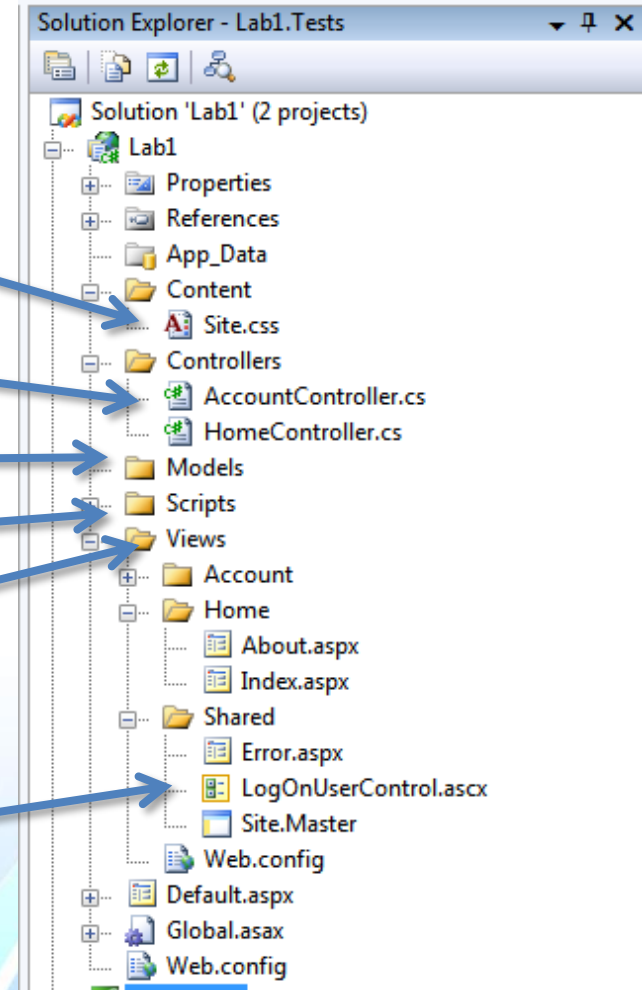
Controllers

Models

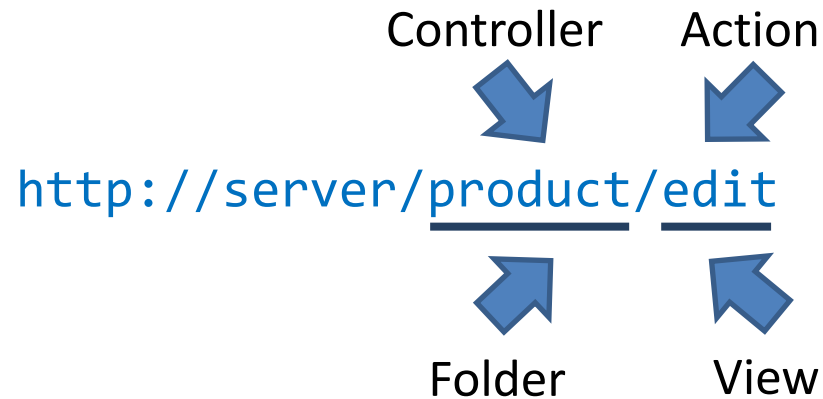
JavaScript

Views

Master Pages



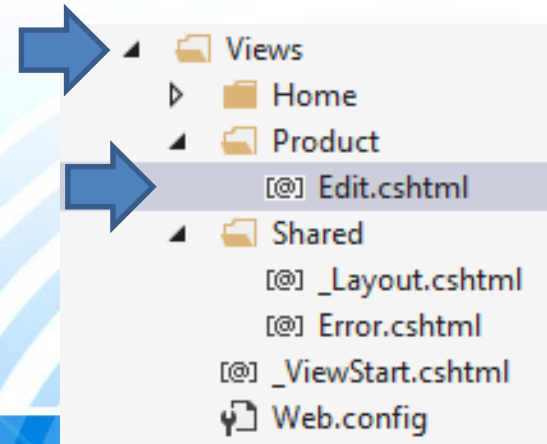
ASP.NET MVC - Routing



↓

```
public class ProductController : Controller
{
    public ActionResult Edit()
    {
        return View();
    }
}
```

↑



MVC Advantages

- *Simultaneous development* – Multiple developers can work simultaneously on the model, controller and views.
- **High cohesion** – MVC enables logical grouping of related actions on a controller together. The views for a specific model are also grouped together.
- **Low coupling** – The very nature of the MVC framework is such that there is low coupling among models, views or controllers
- *Ease of modification* – Because of the separation of responsibilities, future development or modification is easier
- *Multiple views for a model* – Models can have multiple views

MVC Disadvantages

- *Code navigability* – The framework navigation can be complex because it introduces new layers of abstraction and requires users to adapt to the decomposition criteria of MVC.
- *Multi-artifact consistency* – Decomposing a feature into three artifacts causes scattering. Thus, requiring developers to maintain the consistency of multiple representations at once.
- *Pronounced learning curve* – Knowledge on multiple technologies becomes the norm. Developers using MVC need to be skilled in multiple technologies.

References

- W3Schools: <http://www.w3schools.com>
- What is Ajax?:
<http://webdesign.about.com/od/ajax/a/aa101705.htm>
- What is Ajax?:
http://www.tutorialspoint.com/ajax/what_is_ajax.htm
- PHP Freaks: <http://www.phpfreaks.com>
- Why Use Model View Controller
<http://www.htmlgoodies.com/beyond/php/article.php/3912211>
- <https://r.je/mvc-tutorial-real-application-example.html>
- Yii Framework:
<http://www.yiiframework.com/doc/guide/1.0/en/quickstart.what-is-yii>
- Microsoft site for MVC: <https://msdn.microsoft.com/en-us/library/4w3ex9c2.aspx>

Activity 1

1. Develop a PHP program by using the provided classes (input.php and output.php).
Create HTML as the example below:



Parents:	John, 35; Jane, 30
Children:	Tim, 12; Tom, 10; Tam, 13
<input type="button" value="Process"/>	



Extra: Apply AJAX in order to display the result on the browser

Parent

Total member is 5

Name: John	Name: Jane
Age: 35	Age: 30
	

Children

Name: Tim	Name: Tom	Name: Tam
Age: 12	Age: 10	Age: 13
