

# Web Backend



# Inhoud

- Inleiding
- Syntax
- Anatomie php.net
- Afdrukken in een document
- Variables
- Operatoren
- Conditional statements
- Arrays

# Inhoud

- Array functions
- Looping statements
- Functions
- \$\_GET
- \$\_POST
- Controle \$\_GET/\$\_POST
- Herhalingsopdracht

# Inhoud

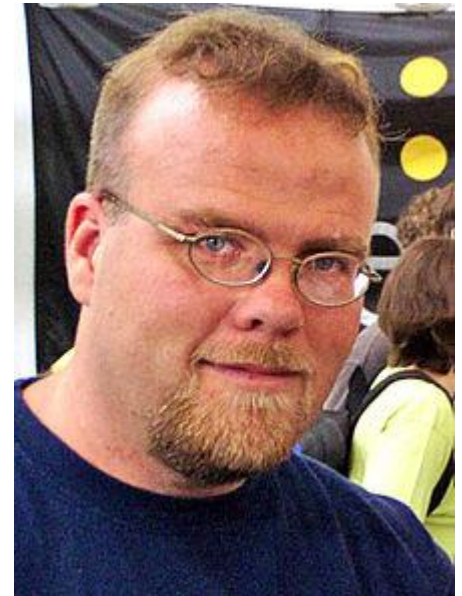
- time functies
  - time()
  - microtime()
  - date()
- sessions
- Cookies
- include/require
- Classes
- namespace
- \_\_autoload() {}
- Error handling
- MySQL
- security
- file upload
- mail

# Inhoud

- constants
- regular expressions
- mod\_rewrite
- MVC-model
  - CodeIgniter
  - Laravel
- Design patterns

# Inleiding

- Wat is PHP
  - Scripttaal (<-> programmeertaal)
  - 1994 IBM ([Rasmus Lerdorf](#))
  - Perl / C / Python
  - **P**ersonal **H**ome **P**age (vroeger)
  - **PHP**: **H**ypertext **P**reprocessor (recursief acroniem)
  - Server side
    - Alternatieven:



# Inleiding

- Doel van php:
  - Dynamisch informatie verwerken (oa.).
    - HTML = statisch
      - Onderhoudsintensief
    - PHP = dynamisch
      - Manier (=script) om het onderhoud van HTML/inhoud te vereenvoudigen.

# Inleiding

- Multiplatform (Windows / Linux / Mac / ...)

- Enkele 'All-in-One'-pakketten

- LAMP ( **L**inux **A**pache **M**ySQL **P**HP )
- MAMP ( **M**acintosh **A**pache **M**ySQL **P**HP )
- WAMP ( **W**indows **A**pache **M**ySQL **P**HP )
- XAMPP ( **X** = Linux, Windows, Mac en Solaris

**A**pache

**M**ySQL

**P**HP

**P**erl )





# Inleiding

- Compatibel met meest gangbare servers (Apache, ISS, ...)
- Meerdere database-types ondersteunen
  - MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, MongoDB, ...
- Gratis te downloaden & gebruiken
- Makkelijk om te leren

# Syntax

- Wat is een PHP-bestand?
  - Extensie: .php (of .php3 of .phtml)
  - Inhoud:
    - PHP-script
    - HTML/JavaScript/CSS
      - geschreven als string binnen `<?php ?>`-tags
      - standaard notatie buiten `<?php ?>`-tags
  - Output kan getoond worden als:
    - HTML, JavaScript, CSS, ...
    - JPEG, PNG, GIF, ...
    - eender welke bestandsextensie

# Syntax

- Script block
  - Start met `<?php` en eindigt met `?>`
  - Elke opdrachtregel wordt afgesloten met een `;` (!!!)
    - Behalve de laatste regel die voor de closing `?>`-tag komt
- 2 manieren om PHP te integreren
  - Inline PHP
    - HTML-document met script-blokken (vb. [voorbeeld-syntax-inline-php](#) )
  - Full PHP
    - PHP-script dat alle output regelt (vb. [voorbeeld-syntax-full-php](#) )

# Syntax

- Beste manier om PHP te integreren
  - scheiden van logica en output
    - variabelen bovenaan tussen scripttags definiëren
    - Onder scripttag -> HTML, CSS, JS
    - via script tags variabelen in HTML invullen

# Syntax

- Commentaar in PHP script-blokken (vb. [voorbeeld-syntax-commentaar](#) )
  - `//` Dit is commentaar
  - `/*` Dit  
is  
comentaar op  
meerdere regels `*/`
  - `#` Dit is ook commentaar
- Alle informatie over de versie van PHP en alle parafenalia:
  - `phpinfo()` (vb. [voorbeeld-syntax-phpinfo](#) )

# Syntax

- Vragen over PHP? Eén adres!

<http://www.php.net>

# Anatomie php.net



[Download 7 documentation](#) | [FAQ](#) | [getting help](#) | [testing help](#) | [news](#) | [issues](#) | [mailing lists](#) | [contact us](#) | [help / system](#) | [your account](#)

### What is PHP?

PHP is a widely-used, general-purpose scripting language that is especially suited for web development and can be embedded into HTML. If you are new to PHP and want to get some idea of how it works, try the [introductory tutorial](#). After that, check out the [online manual](#), and the [example archive](#) sites and some of the other resources available in the [links section](#).

Ever wondered how popular PHP is? See the [Stats](#)!

### Thanks To

asac081

Dr0p5

Earl Redonda

Emilio, Carlos

Husted Solutions

Sony, XPL, Hosting

CGI, Open Source Lab

Taheri, Inc.

HEXCESS.NET

Rockwell

BUTRIST

Unhosted Webhosting

Radial Lingo

Facebook

mxlml.co.uk

SecureScript

Bauer + Smith, GmbH

### Related sites

Apache

MySQL

PostgreSQL

Linux Technology

### Community

...

Upcoming conferences: [DevConf 2012](#) | [Dubai PHP Conference 2012](#)

Calling for papers: [Northwest PHP conference](#)

### PHP 5.4.3 and PHP 5.3.13 Released!

[2012-05-01] The PHP development team would like to announce the immediate availability of PHP 5.4.3 and PHP 5.3.13. All users are encouraged to upgrade to PHP 5.4.3 or PHP 5.3.13.

The releases complete a fix for a [subscriptable](#) in CGI-based setups (CVE-2012-2311). Note: `mod_php` and `php-fpm` are not vulnerable to this attack.

PHP 5.4.3 fixes a buffer overflow vulnerability in the `apache_request_headers()` (CVE-2012-2328). The PHP 5.3 series is not vulnerable to this issue.

For source downloads of PHP 5.4.3 and PHP 5.3.13 please visit our [downloads page](#). Windows binaries can be found on [windows.php.net/downloads](#). The list of changes are recorded in the [ChangeLog](#).

---

### PHP 5.3.12 and 5.4.2 and the CGI flaw (CVE-2012-1823)

[2012-05-01] PHP 5.3.12/5.4.2 do not fix all variations of the CGI issues described in CVE-2012-1823. It has also come to our attention that some sites use an insecure `cgisnapper` script to run PHP. These scripts will use `$*` instead of `"$@"` to pass parameters to `php-cgi` which causes a number of issues. Again, people using `mod_php` or `php-fpm` are not affected.

One way to address these CGI issues is to reject the request if the query string contains a `~` and no `^`. It can be done using Apache's `mod_rewrite` like this:

```
RewriteCond %{QUERY_STRING} "[^~]" [NC]
RewriteRule ^.*$ [F,L]
```

Note that this will block otherwise safe requests like `http-40` so if you have query parameters that look like that, adjust your regex accordingly.

Another set of releases are planned for Tuesday, May, 8th. These releases will fix the CGI flaw and another CGI-related issue in `apache_request_header` (5.4 only).

We apologize for the inconvenience created with these releases and the (lack of) communication around them.

---

### PHP 5.3.12 and PHP 5.4.2 Released!

[2012-05-01] There is a vulnerability in certain CGI-based setups (**Apache+mod\_php and nginx+php-fpm are not affected**) that has gone unnoticed for at least 8 years. [Section 7 of the CGI spec](#) states:

Some systems support a method for supplying a [sic] array of strings to the CGI script. This is only used in the case of an "indexed" query. This is identified by a "GET" or "HEAD" HTTP request with a URL search string not containing any unencoded "+" characters.

If requests that do not have a "+" in the query string are treated differently from those who do in some CGI implementations. For PHP this means that a request containing `T=` may dump the PHP source code for the page, but a request that has `T=&v=1` is fine.

A large number of sites run PHP as either an Apache module through `mod_php` or using `php-fpm` under `nginx`. Neither of these setups are vulnerable to this. Straight shebang-style CGI also does not appear to be vulnerable.

If you are using Apache `mod_cgi` to run PHP you may be vulnerable. To see if you are, just add `T=` to the end of any of your URLs. If you see your source code, you are vulnerable. If your site renders normally, you are not.

To fix this, update to PHP 5.3.12 or PHP 5.4.2.

### Stable Releases

Current PHP 5.4 Stable: 5.4.3

Current PHP 5.3 Stable: 5.3.13

### Upcoming Events (add)

### May

### Other Group Events

09. Wash. DC PHP Developers Group

09. PHP User Group Stuttgart

09. South FL PHP Users

09. PHP South West User Group

09. PHP@LX - LX

09. DC PHP Developers Community

09. Meeting onnigrow Dortmund

09. PHP Usergroup Frankfurt/Main

09. Meet in Java PHP Usergroup

09. PHP User Group Karlsruhe, RUCA

09. CUA Meeting Bonn

09. PHP@LX

09. Nagpur PHP Meetup

09. SecuHackday PHP Karnataka

09. Los Angeles PHP Developers Group

09. Queen City (Charlotte) PHP

09. PHP Bruback, Mexico Group

09. Nashville Enterprise LAMP LG

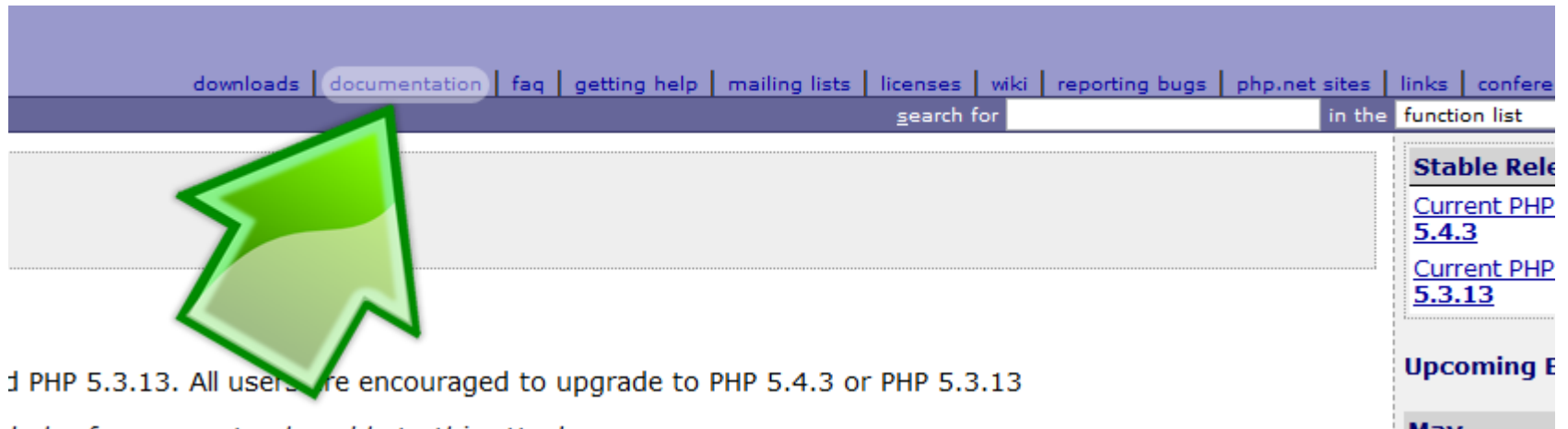
09. Chattanooga PHP Developers

09. PHP North-East User Group

09. WDC PHP User Group Meeting

# Anatomie php.net

- Documentation (= volledig handleiding)





# Anatomie php.net

- Zoeken in de documentation



# Anatomie php.net

- Zoekresultaten

## Related snippet found for 'function'

PHP contains thousands of functions. You might be interested in how a [function is defined](#), or [how to read a function prototype](#).

## PHP Function List

**function** doesn't exist. Closest matches:

[function\\_exists](#)  
[create\\_function](#)  
[runkit\\_function\\_redefine](#)  
[apc\\_dump\\_function\\_table](#)  
[reflectionfunctionabstract](#)  
[register\\_tick\\_function](#)  
[bcompiler\\_write\\_function](#)

[badfunctioncallexception](#)  
[runkit\\_function\\_add](#)  
[runkit\\_function\\_remove](#)  
[override\\_function](#)  
[get\\_defined\\_functions](#)  
[sqlite\\_create\\_function](#)  
[bcompiler\\_write\\_functions\\_from\\_file](#)

## Site Search Results

-  [Functions - Manual](#)  
Note about function names: -- According to the specified regular expression ([a-zA-Z\_\x7F-\xFF]\*), a function name like this one  
[php.net/function](#) - 4 May 2012 - [Cached](#)
-  [User-defined functions - Manual](#)  
User-defined functions. A function may be defined using syntax such as the following:  
[php.net/manual/en/functions.user-defined.php](#) - 8 May 2012 - [Cached](#)
-  [addslashes - Manual](#)



# Anatomie php.net

- Analyse manual (vb. functie "echo")
  - Plaats van zoekterm binnen de manual



The screenshot shows the PHP Manual interface. On the left, a sidebar lists various manual sections: PHP Manual, Function Reference, Text Processing, Strings, and String Functions. Under 'String Functions', a list of functions is shown, including addcslashes, addslashes, bin2hex, chop, chr, chunk\_split, convert\_cyr\_string, convert\_uuencode, convert\_uuencode, count\_chars, crc32, crypt, **echo**, explode, and htmlspecialchars. A green arrow points to the 'echo' function in this list. The main content area displays the details for the 'echo' function, including its description and usage examples.

PHP Manual

Function Reference

Text Processing

Strings

String Functions

- addcslashes
- addslashes
- bin2hex
- chop
- chr
- chunk\_split
- convert\_cyr\_string
- convert\_uuencode
- convert\_uuencode
- count\_chars
- crc32
- crypt
- echo**
- explode
- htmlspecialchars

«crypt

view this page in Brazilian Portuguese

**echo**

(PHP 4, PHP 5)

echo — Output one or more strings

**Description**

void **echo** ( string *\$arg1* [, string *\$...* ] )

puts all parameters.

**echo** is not actually a function (it is a language construct). Additionally, if you want to use it in a function context, you can use the **echo** function.

**echo** also has a shortcut syntax, where you can write:

```
I have <?=$foo> foo
```

# Anatomie php.net

## — Description

[«crypt](#)

view this page in Brazilian Portuguese 

### echo

(PHP 4, PHP 5)

echo — Output one or more strings

Description

```
void echo ( string $arg1 [, string $... ] )
```

Outputs all parameters.

**echo** is not actually a function (it is a language construct), so you are not required to use `pa` context of a function. Additionally, if you want to pass more than one parameter to **echo**, th

**echo** also has a shortcut syntax, where you can immediately follow the opening tag with an

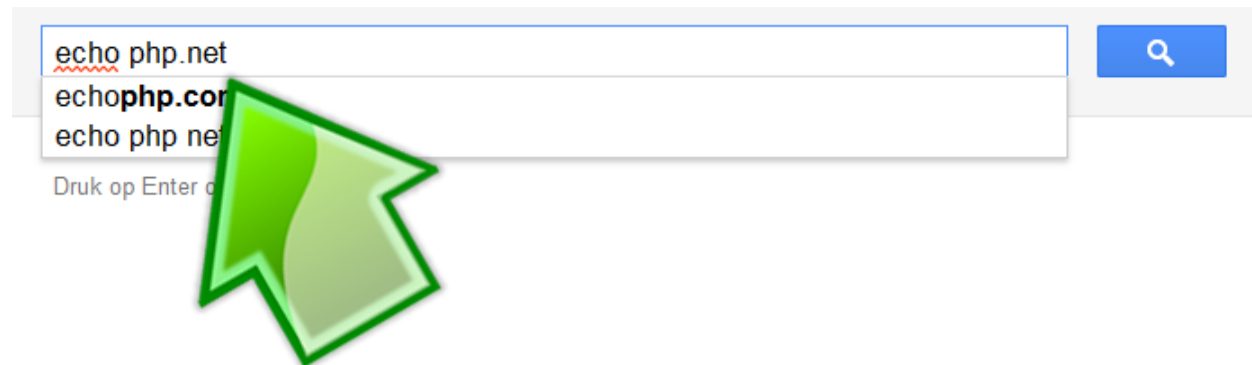
```
I have <?=$foo?> foo.
```

# Anatomie php.net

- Parameters
  - Wat moet meegegeven worden
- Return value
  - Wat de functie teruggeeft
- Examples
- Notes
- See Also
- Comments
  - staan veel kant en klare oplossingen voor veel voorkomende problemen

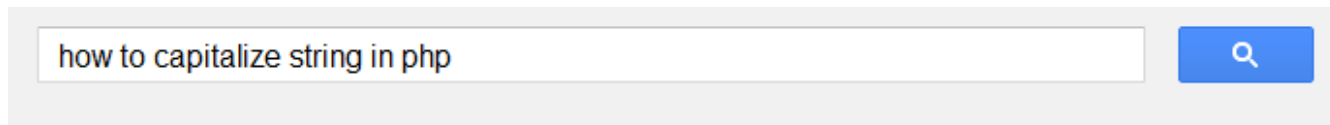
# Anatomie php.net

- **Tip:** handiger zoeken d.m.v. Google
  - Je weet waar je naar op zoek bent
    - Zoekterm = functienaam + php.net



# Anatomie php.net

- Je weet niet naar wat je op zoek bent
  - Zoekterm = Engelse vraag + in php



how to capitalize string in php

Ongeveer 2.060.000 resultaten (0,37 seconden)

---

- Betrouwbaar?

# Anatomie php.net

- Websites **php.net** & **stackoverflow.com** = goed!

## [PHP Tutorial - Capitalization](#)

[www.tizag.com/phpT/php-string-strtoupper-strtol...](http://www.tizag.com/phpT/php-string-strtoupper-strtol...) - Vertaal deze pagina

**PHP - String Capitalization** Functions. If you've ever wanted to manipulate the **capitalization** of your **PHP strings**, then this lesson will be quite helpful to you.

## [ucfirst Make a string's first character uppercase - PHP](#)

[php.net/manual/en/function.ucfirst.php](http://php.net/manual/en/function.ucfirst.php) - Vertaal deze pagina

**string** ucfirst ( **string** \$str ). Returns a **string** with the first character of str **capitalized**, if that character is alphabetic. Note that 'alphabetic' is determined by the ...

## [strtoupper Make a string uppercase - PHP](#)

[php.net/manual/en/function.strtoupper.php](http://php.net/manual/en/function.strtoupper.php) - Vertaal deze pagina

**string** strtoupper ( **string** \$string ). Returns **string** with all alphabetic characters converted to **uppercase**. Note that 'alphabetic' is determined by the current locale.

## [ucwords Uppercase the first character of each word in a string - PHP](#)

[php.net/manual/en/function.ucwords.php](http://php.net/manual/en/function.ucwords.php) - Vertaal deze pagina

Returns a **string** with the first character of each word in str **capitalized**, if that ... (Note: it is advised to use a recent version of **PHP** when implementing this function ...)





# Anatomie php.net

- Wees kritisch/analyseer website
  - » Datum?
  - » Commentaren?
  - » Reclame?
- Het eerste zoekresultaat is niet altijd wat je zoekt
- Neem de tijd om de eerste paar zinnen van elk zoekresultaat te lezen => antwoord zit hier vaak al in vervat

# Afdrukken naar het scherm

## – echo

- Enkel voor strings!
- Gebruik `<?= $var ?>` om dingen in html af te drukken
  - Bv `<p>Hallo, ik heet <?= $naam ?></p>`
  - Gebruik deze notatie énkél om variabelen af te drukken. Laat tekst die niet veranderd moet worden in de HTML staan
  - Dus zeker niet:  
~~`<p><?= 'Hallo, ik heet' . $naam ?></p>`~~
  - Let op: niet elke server heeft deze shorthand echo standaard ingeschakeld (php.ini: `short_open_tag = on`)
  - (vb. [voorbeeld-afdrukken-echo](#) )

# Afdrukken naar het scherm

## – **var\_dump()**

- Dient enkel om te debuggen -> nooit in productieomgeving
- Geeft extra informatie over variabele (string/boolean/array/...) -  
> vergemakkelijkt debuggen
- Standaard onoverzichtelijke opmaak
  - Oplossen door **print\_r()** in samenwerking met <pre>-element
    - » Omslachtig!
  - XDEBUG kan helpen!
- (vb. [voorbeeld-afdrukken-var-dump](#) )

# Afdrukken naar het scherm

- XDEBUG
  - Module die debuggen in PHP aangenamer kan maken
    - » Enorm uitgebreid en krachtig -> best is samenwerking met uitgebreidere IDE (Eclipse, Netbeans, ...)
  - Inschakelen in XAMPP:
    - » C:\xampp\php\php.ini
      - Zoeken naar XDEBUG
      - Alle ; verwijderen onder deze XDEBUG-codeblock
        - Een ; in de php.ini file betekent een lijn in commentaar
    - » XAMPP/Apache heropstarten
    - » **(voor mac-gebruikers:**  
[http://docs.joomla.org/Edit\\_PHP.INI\\_File\\_for\\_XDebug](http://docs.joomla.org/Edit_PHP.INI_File_for_XDebug) )

# Afdrukken naar het scherm

- Opdracht
  - opdracht-comments

- Wat zijn variables?
  - Vergelijkbaar met algebra:

$$x = 5$$

# Variables

**X** = **5**

 **variable**

 **value**

$$\underbrace{x}_{\text{variable}} = \underbrace{5 + y}_{\text{expression}}$$



- Variable in php
  - Start altijd met een **\$**-teken gevolgd door de **naam van de variable**
  - De variable name moet **beginnen met een letter** OF een **underscore** (cijfers zijn NIET toegestaan)
  - De variable name kan **enkel letters, cijfers of een underscore** ( **\_** ) bevatten

- Variable in php

- De variable name mag geen spaties bevatten
- De variable name is case sensitive (A ≠ a)
- Een type declaration is niet noodzakelijk (= loosely typed)
- De variable wordt gecreëerd wanneer er een value aan toegekend wordt

- Schrijfwijze

- Voor een variable die een string (= stuk tekst) bevat

`$myDrink = "Cuba Libre";`

`$myDrink = 'Cuba Libre';`

- Voor een variable die énkél een cijfer bevat

`$myGrade = 20;`

# Variables

- Schrijfwijze

- Voor een variable die niets bevat

```
$containerVariable = null;
```

-> kan zowel een string, integer of ... zijn

```
$containerVariable = "";
```

-> kan enkel een string zijn

- Variables kunnen ook boolean, array, double, object, resource, of "unknown type" zijn (later meer)

# Variables

## – Strings

- Een string is een stuk tekst

```
$albumTitle = "Lullabies to paralyze";
```

(vb. `voorbeeld-variables-string` )

- Verschillende strings kan men samenvoegen =

### **CONCATENATION**

- Gebeurt door middel van een `.` (vb. `voorbeeld-variables-concatenation` )

# Variables

## – Strings

- Probleem met quotes (vb. [voorbeeld-variables-quote-problem](#) )

- `$feeling = 'I'm fine';`

- Oplossing

- `$feeling = 'I\'m fine';`

- `$feeling = "I'm fine";`

# Variables

## – Stringfuncties

- **strlen()** (vb. [voorbeeld-variables-string-strlen-fn](#) )
  - Retourneert de lengte van een string
- **strpos()** (vb. [voorbeeld-variables-string-strpos-fn](#) )
  - Kijkt of een string voorkomt in een andere string
  - Indien deze string teruggevonden wordt => retourneert positie van string
  - Zoniet: retournt *false*

# Variables

## – Data types

- boolean: TRUE of *FALSE* (best altijd in hoofdletters)

```
$a = FALSE; (geeft als waarde 0 of niets)
```

```
$a = TRUE; (geeft als waarde 1)
```

- integer: geheel getal (wordt niet tussen quotes gezet)

```
$a = 1234; // decimaal
```

```
$a = -123; // negatief decimaal
```

```
$a = 0123; // octaal
```

```
$a = 0x1A; // hexadecimaal
```

- float: getal met cijfers na het punt (GEEN komma!)

```
$a = 1.234; // decimaal
```

- string

```
$animal = 'paard'; of $animal = "paard";
```



# Variables

## — Data types

(later)

- array
- object
- resource
- null
- unknown type

# Variables

- Strings (corrigeren)
  - opdracht-verbeter
- Strings (concatenate)
  - opdracht-string-concatenate
- Strings (string functions)
  - opdracht-string-extra-functions

# Operators

Arithmetic Operators	Beschrijving	Voorbeeld	Resultaat
+	add	<code>\$x = 3;</code> <code>\$x = \$x + 4;</code>	7
-	subtract	<code>\$x = 5;</code> <code>\$x = 8 - \$x;</code>	3
*	multiply	<code>\$x = 3;</code> <code>\$x = \$x * 2;</code>	6
/	divide	<code>\$x = 6;</code> <code>\$x = \$x / 2;</code>	3
%	modulo	<code>\$x = 10;</code> <code>\$x = \$x % 4;</code> <code>\$x = \$x % 5;</code> <code>\$x = \$x % 3;</code>	2 (rest) 0 1 (rest)
++	increment	<code>\$x = 3;</code> <code>\$x++;</code>	4
--	decrement	<code>\$x = 3;</code> <code>\$x--;</code>	2

(vb. [voorbeeld-operators-arithmetic](#) )

# Operators

Assignment operators	voorbeeld	Resultaat
=	<code>\$x = 3;</code>	3
+=	<code>\$x = 3;</code> <code>\$x += 5;</code>	$(3 + 5) = 8$
-=	<code>\$x = 3;</code> <code>\$x -= 1;</code>	$(3 - 1) = 2$
*=	<code>\$x = 3;</code> <code>\$x *= 2;</code>	$(3 * 2) = 6$
/=	<code>\$x = 6;</code> <code>\$x /= 2;</code>	$(6 / 2) = 3$
.=	<code>\$x = 3;</code> <code>\$x .= 2;</code>	$(3 . 2) = 32$
%=	<code>\$x = 8;</code> <code>\$x %= 4;</code>	$(8 \% 2) = 0$

**OPM:** waar mogelijk, kies assignment boven de arithmetic operators ( = overzichtelijker!)

# Operators

Comparison operators	beschrijving	voorbeeld	resultaat
==	Is gelijk aan	"test" == "test" "4" == 4	TRUE TRUE
===	Is gelijk aan (incl. data type)	"4" === 4	FALSE
!=	Is niet gelijk aan	4 != 3	TRUE
>	Is groter dan	5 > 9	FALSE
<	Is kleiner dan	3 < 9	TRUE
>=	Is groter dan of gelijk aan	8 >= 9	FALSE
<=	Is kleiner dan of gelijk aan	8 <= 8	TRUE

(vb. [voorbeeld-operators-comparison](#) )

# Operators

Logical operators	beschrijving	voorbeeld	resultaat
&&	and	<code>\$x = 4; \$y = 8; (\$x &lt; 10 &amp;&amp; \$y &gt; 1)</code>	TRUE
	or	<code>\$x = 4; \$y = 8; (\$x == 2    \$y == 3)</code>	FALSE
!	is not	<code>\$x = 4; \$y = 8; !(\$x == \$y)</code>	TRUE

- OPGELET:

~~`if ( $x > 5 && < 10 ) { ... }`~~ => **FOUT**

`if ( $x > 5 && $x < 10 ) { ... }` => **CORRECT**

# Conditional statements

- vier verschillende conditional statements:
  - if statement
  - if ... else statement
  - if ... elseif... else statement
  - switch statement
- Functie: code uitvoeren wanneer er voldaan wordt aan een of meerdere voorwaarden

# Conditional statements

- if statement

- Syntax:

```
if ( condition )
```

```
{
```

```
    uit te voeren code;
```

```
}
```



# Conditional statements

- shorthand if statement

- Syntax:

`( condition ) ? code bij true : code bij false;`

- Wordt vooral gebruikt om op basis van een bepaalde condition een string of int aan een variabele toe te kennen.
    - niet voor het uitvoeren van grote stukken code

( vb. `voorbeeld-conditional-statements-if` )

- Opdracht: `opdracht-conditional-statements-if`

# Conditional statements

- if ... else statement

- Syntax:

```
if (condition)
{
    uit te voeren code;
}
else
{
    uit te voeren code;
}
```

(vb. [voorbeeld-conditional-statements-if-else](#) )

- Opdracht: [opdracht-conditional-statements-if-else](#)

# Conditional statements

- if ... elseif ... else statement

- Syntax:

```
if (condition)
{
    uit te voeren code;
}
elseif (condition)
{
    uit te voeren code;
}
else
{
    uit te voeren code;
}
```

(vb. [voorbeeld-conditional-statements-if-elseif](#) )

- Opdracht: [opdracht-conditional-statements-if-elseif](#)

# Conditional statements

- Switch statement
  - Dient enkel om één variabele te vergelijken met meerdere waarden
  - Syntax:

```
switch ($variablename)
{
    case value1:
        uit te voeren code;
        break;
    case value2:
        uit te voeren code;
        break;
    default:
        uit te voeren code;
}
```

(vb. [voorbeeld-conditional-statements-switch](#) )

# Conditional statements

- Switch statement
  - Operatoren zijn niet toegestaan in de case condition!
  - **break;** is noodzakelijk (anders wordt alles eronder ook uitgevoerd)
  - Default is de actie die uitgevoerd wordt als er aan geen enkele conditie wordt voldaan.
  - Opdracht: [opdracht-conditional-statements-switch](#)


# Arrays

- Een array is een variable die meerdere values kan bevatten.
- Er zijn twee soorten arrays
  - Numeric array (**begint altijd bij 0**)
    - `$frisdrank[0] = 'cola';`
  - Associative array
    - `$frisdrank['type'] = 'zero';`

# Arrays

- Syntax

**\$array[0] = 'fanta';**



arrayname      key      value

# Arrays

- Doel van array: values samenbundelen

– Bv.      `$frisdrank1 = 'Cola';`  
            `$frisdrank2 = 'Fanta';`  
            `$frisdrank3 = 'Spa';`

**Beter:**

`$frisdrank[0] = 'Cola';`  
`$frisdrank[1] = 'Fanta';`  
`$frisdrank[2] = 'Spa';`



# Arrays

- Array samenstellen: numeric array

- `$frisdrank = array('Cola' , 'Fanta' , 'Spa');`

**OF**

- `$frisdrank[] = 'Cola';`  
`$frisdrank[] = 'Fanta';`  
`$frisdrank[] = 'Spa';`

# Arrays

- Array samenstellen: associative Array

- `$frisdrank = array('Cola' => 'Zero', 'Fanta' => 'Regular');`

**OF**

- `$frisdrank['Cola'] = 'Zero';`  
`$frisdrank['Fanta'] = 'Fanta';`

# Arrays

- Inhoud van een array bekijken
  - Specifieke array value:
    - Numeric: `$frisdrank[2];`
    - Associative: `$frisdrank['Cola'];`
  - De volledige array-inhoud:
    - `var_dump($frisdrank);`
    - `print_r($frisdrank);` (gebruik `<pre>...</pre>`)
    - (vb. `voorbeeld-arrays-opbouw` )

# Arrays

- Multidimensional array
  - Array in een array
  - Multidimensional array maken:
    - `$frisdrank = array('Cola' => array('Regular', 'Zero', 'Light'), 'Fanta' => array('Regular', 'Lemon'));`
    - OF
    - `$frisdrank['Cola'] = array('Regular', 'Zero', 'Light');`  
`$frisdrank['Fanta'] = array('Regular', 'Lemon');`
    - (vb. [voorbeeld-arrays-multidimensioneel](#) )

# Arrays

- Opdracht: [opdracht-arrays-basis](#)

# Array functions

- `count()`
  - Telt het aantal waarden van een array
  - Retourneert het aantal waarden (retourneert 0 als er geen waarden zijn)
  - (vb. `voorbeeld-arrays-function-count` )

# Array functions

- `in_array()`
  - Kijkt of een waarde in een array voorkomt
  - Retourneert TRUE indien teruggevonden, FALSE indien niet
  - (vb. [voorbeeld-arrays-function-in-array](#) )

# Array functions

- Verdere array functies?
  - (vb. [voorbeeld-arrays-function-extra](#) )
  - ...
  - [Php.net](#)
- Opdracht
  - [opdracht-arrays-functions](#)



# Looping statements

- Looping statement voert een code block uit gedurende en herhaalt dit code block tot er aan een bepaalde conditie wordt voldaan.
- Vier soorten looping statements:
  - while loop
  - do... while loop
  - for loop
  - foreach loop

# Looping statements

- While loop
  - Voert een code block uit zolang er aan de conditie wordt voldaan
  - Syntax:

```
while (condition)
{
    uit te voeren code;
}
```

(vb. [voorbeeld-looping-statements-while](#) )

- Opdracht: [opdracht-looping-statements-while](#)

# Looping statements

- do... while loop
  - Het verschil met de while loop is dat hier de code block eerst wordt uitgevoerd en daarna pas wordt gekeken of er aan een bepaalde conditie wordt voldaan alvorens de code weer uit te voeren.
  - Syntax:

```
do
{
    uit te voeren code;
}
while ( condition )
```

(vb. [voorbeeld-looping-statements-do-while](#) )

# Looping statements

- for loop
  - Voert een code block een vooropgesteld aantal keer uit.
  - Syntax:

```
for (init; condition; increment) {  
    uit te voeren code;  
}
```

(vb. [voorbeeld-looping-statements-for](#) )
- Opdracht: [opdracht-looping-statements-for](#)

# Looping statements

- foreach loop

- Loopt doorheen een array

- Syntax:

```
foreach ($array as $value) {  
    uit te voeren code die gebruik maakt van $value;  
}
```

(vb. [voorbeeld-looping-statements-foreach-value](#) )

# Looping statements

- foreach loop

```
foreach ($array as $key => $value) {  
    uit te voeren code die gebruik maakt van $key en  
    $value;  
}
```

**OPM:** de naam van de parameters \$key en \$value mag je volledig zelf kiezen.

(vb. [voorbeeld-looping-statements-foreach-key-value](#) )

- Opdracht: [opdracht-looping-statements-foreach](#)

# Looping statements

- Alternatieve syntax voor looping statements
  - if/for/foreach/while/switch hebben alternatieve schrijfwijze

```
<?php if ($a == 5): ?>  
    A is equal to 5  
<?php endif; ?>
```

- Na elke conditie een :
- Om de looping statement af te sluiten:  
endif/endfor/endforeach/endwhile/endswitch

# Looping statements

- Alternatieve syntax voor looping statements
  - **DOEL:** overzichtelijk php mengen in HTML
    - Maak hier gebruik van!
  - Nog altijd niet overzichtelijk -> beste slechtste oplossing
    - Alternatieven
      - Templating languages (Blade, Markdown, ...)
      - [Template animation](#)
  - (vb. **voorbeeld-looping-statements-alternative-syntax**)



# Functions

- Wanneer bepaalde berekeningen vaak terugkomen is het best hiervoor een functie te gebruiken.
- Een functie definiëren

```
function functionName($parameter)
{
    uit te voeren code;
}
```

(vb. [voorbeeld-functions-basis](#) )

# Functions

- Functie met return definiëren (definiëren)
  - functie voert bewerking uit met bepaald resultaat.
  - Dit resultaat kan men teruggeven d.m.v. return
  - Bij een return wordt de functie beëindigd. Alle code erna die binnen dezelfde functie staat, wordt niet meer uitgevoerd.

```
function functionName($parameter)
{
    uit te voeren code;
    return $resultaat;
}
```

(vb. [voorbeeld-functions-return](#) )

# Functions

- Syntax
  - Een function begint altijd met function gevolgd door de function name.
  - Function name begint altijd met een letter of een underscore (**NOOIT** een getal!)
  - Function name moet een duidelijke betekenis hebben (**NOOIT** afkortingen!)
  - Variabelen die met de function name worden 'ingesteld' heten **parameters**

# Functions

- Hoe een functie callen (aanroepen)?

- Zonder arguments:

`functionName()`

- Met één argument

`functionName($argument);`

- Met meerdere arguments

`functionName($argument01, 'argument02value');`

(vb. [voorbeeld-functions-arguments-parameters](#) )

# Functions

- Opdracht: opdracht-functions

# Functions

- Scope van variabelen
  - De reden waarom er in functies met parameters wordt gewerkt: het bereik van variabelen.
  - Variabelen zijn niet overal aanspreekbaar
  - Er is een verschil tussen **global** en **local variables**
    - **Global**: variabele beschikbaar in heel het document
    - **Local**: enkel beschikbaar binnen een functie
    - (vb. **voorbeeld-functions-scope** )

# Functions

- Scope
  - Vanuit een functie een globale variabele aanspreken:
    - define de variable in de function eerst door middel van volgende syntax:

***global*** \$variable;

(vb. **voorbeeld-functions-global** ) (OPM: \$GLOBALS['variablename'])

# Functions

- Scope

- Een functie een local variable laten onthouden

- define de variable in de function door middel van volgende syntax:

***static*** \$variable = value;

- Een static variable kan enkel een value zijn, **nooit een expression** (vb. **voorbeeld-functions-static** )



# Functions

- Callback
  - Een functie kan ook in een variabele opgeslagen worden
  - Hoe:
    - Je definiëert de functie
    - Je benoemt een variabele met de stringnaam van de functie
    - Nu kan je deze variabele overal gebruiken (bv. Als argument in een andere functie)
    - Voer de functie uit door achter de variabelenaam () te plaatsen

# Functions

- Callback

- ```
function test() {  
    echo 'Test func';  
}
```

```
$callback = 'test';
```

```
$callback(); // Resultaat: 'Test func'
```

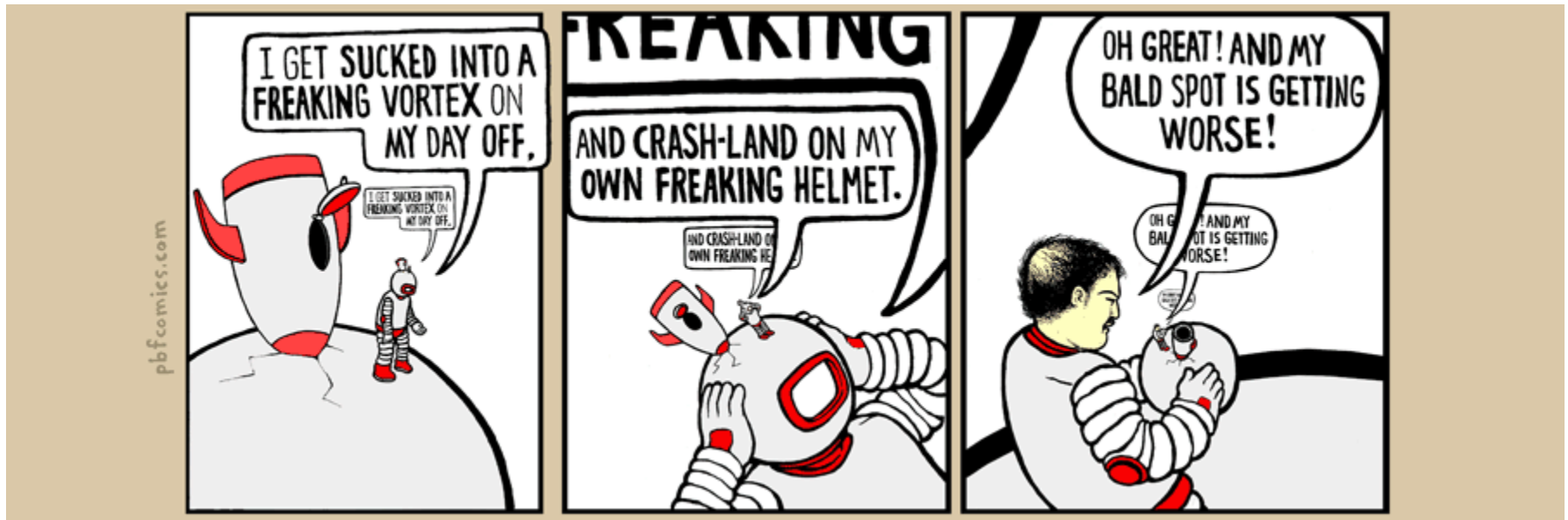
- (vb. [voorbeeld-functions-callback](#) )

# Functions

- Scope
  - Opdracht: opdracht-functions-gevorderd

# Functions

- Recursive functions



# Functions

- Recursive function
  - recursief = zichzelf aanschrijven.
  - Een functie kan zichzelf dus aanschrijven.
  - OPGEPAST: dit kan een infinite loop veroorzaken en dus een crash van het script.
  - (vb. [voorbeeld-functions-recursive-simpel](#) )
  - (vb. [voorbeeld-functions-recursive-return](#) )
  - (vb. [voorbeeld-functions-recursive-uitgebreid](#) )
  - Opdracht: [opdracht-functions-recursive](#)

# \$\_GET

- \$\_GET variable
  - Manier om informatie naar de server te sturen
  - Zichtbaar voor iedereen
  - Zichtbaar in de url
    - Voordeel: kan makkelijk gedeeld worden



- Informatiegrootte is gelimiteerd (max. 2000 karakters)

# \$\_GET

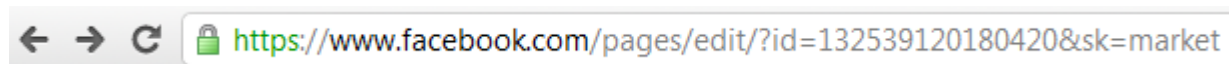
- \$\_GET variable
  - Wordt gebruikt voor het '**tonen van informatie**'
  - Syntax:
    - Eén variable:

`http://www.url.be/contact.html?variablename=value`



- Meerdere variables:

`http://www.url.be/?variablename=value&variablename02=value02`



# \$\_GET

- Hoe gebruiken?

- form.html

```
<form action="validate.php" method="get">  
    <input type="text" name="email">  
</form>
```

- validate.php

```
$_GET['email'];    => is dus een ARRAY!
```

- (vb. [voorbeeld-get-basis](#) )

- Opdracht: [opdracht-get](#)



# \$\_POST

- \$\_POST variable
  - Manier om informatie naar de server te sturen
  - Enkel zichtbaar voor de server
  - Informatiegrootte is niet gelimiteerd
  - Wordt gebruikt voor het **aanpassen** van informatie en voor het doorsturen van **gevoelige informatie** (usernames/passwords)

# \$\_POST

- Hoe gebruiken?

- form.html

```
<form action="validate.php" method="post">  
    <input type="password" name="password">  
</form>
```

- validate.php

```
$_POST['password'];    => is dus een ARRAY!
```

- (vb. **voorbeeld-post-basis** )

# Controle \$\_GET & \$\_POST

- Wanneer \$\_GET of \$\_POST worden aangeroepen zonder dat er iets 'gesubmit' is  
=> error-message!

**Notice:** Undefined index: ... in ... on line ...

# Controle \$\_GET & \$\_POST

- Controleren of een key in een array bestaat:

```
if ( isset( $_POST[ 'key' ] ) )  
{  
    uit te voeren code;  
}
```

(vb. [voorbeeld-get-post-key-controle](#) )

- Opdracht: [opdracht-post](#)

# Controle \$\_GET & \$\_POST

- Controles (kunnen gebruikt worden als condition)
  - `array_key_exists()`
  - `in_array()`
  - `empty()`
  - `isset()`
  - `$variableName`
  - `!$variableName` ( $\Leftrightarrow$  `$variable`)
  - ... ([php.net](http://php.net))

# Herhalingsoefening

opdracht-herhalingsopdracht-01

# time()

- time() geeft het aantal seconden weer dat verstreken is sinds de 'Unix Epoch'.

bv: 1337588931

<http://www.php.net/manual/en/function.time.php>

- UNIX Epoch -> vast referentiepunt nodig

1 januari 1970 00:00:00 GMT

# microtime()

- `microtime()` geeft het aantal microseconden weer dat verstreken is sinds de 'Unix Epoch'.

bv: 0.20361400 1337589462

<http://php.net/manual/en/function.microtime.php>



# mktime()

- mktime() geeft een timestamp terug, gebaseerd op een opgegeven numerieke datum
- mktime(uur,min,sec,maand,dag,jaar) (Amerikaanse volgorde!)

<http://php.net/manual/en/function.mktime.php>

- (vb. **voorbeeld-time**)

# date()

- Zet een timestamp om naar een menselijk leesbaar formaat door middel van parameters

<http://www.php.net/manual/en/function.date.php>

- (vb. **voorbeeld-date** )
- Opdracht: **opdracht-date**

# sessions & cookies

- Zijn globale variabelen binnen eenzelfde domeinnaam die toestaan om data te bewaren over de gebruiker (bv. winkelmandje, logincredentials, ...)
- Verloopt nadat de gebruiker de browser sluit mits er geen expiration date is opgegeven.

# sessions & cookies

A screenshot of the Google login interface. It features a light gray background. At the top left is the word 'Inloggen' and at the top right is the 'Google' logo. Below these are two input fields: 'Gebruikersnaam' (Username) and 'Wachtwoord' (Password). Under the password field is a blue 'Inloggen' button. To the right of the button is a checkbox labeled 'Ingelogd blijven' (Stay logged in). At the bottom is a blue link that says 'Geen toegang tot uw account?' (No access to your account?).

Inloggen Google

Gebruikersnaam

Wachtwoord

Inloggen ☐ Ingelogd blijven

[Geen toegang tot uw account?](#)

- 'Ingelogd blijven' niet aangevinkt -> cookie of session zonder expiration date vervalt na het sluiten vd. browser
- 'Ingelogd blijven' aangevinkt -> cookie of session vervalt na een opgegeven expiration date (bv. 31 dagen)

# sessions & cookies

- Het verschil tussen sessions en cookies
  - Cookies worden opgeslagen op de client (browser)
    - bevatten alle nodige data in de vorm van een array.
    - Informatie leesbaar vanuit de browser.
    - Wordt gebruikt om minder gevoelige data op te slaan.
    - Let op: sla een paswoord nooit op in een cookie!
  - Sessies bevatten enkel een code (=SessionId) op de client (browser) die overeenstemt met een code op de server.
    - Deze code op de server verwijst naar een array met allerlei data over de gebruiker
    - Server staat in voor het opslagen van de data.
    - Wordt gebruikt om gevoelige data op te slaan.

# sessions

- Werkwijze
  - Session moet eerst gestart worden.
    - Session starten: `session_start();`
  - Moet niet expliciet weer gesloten worden, maar kan wel:
    - Session sluiten: `session_destroy();`
  - Session variables aanspreken (gelijkaardig aan `$_GET` & `$_POST`):
    - `$_SESSION['value'];`
  - Session moet geopend of gesloten worden **VOOR** elke echo, `print_r` of `var_dump`, dus: bovenaan het document! (als het niet anders kan: `header('location: url' )` )
- (vb. `voorbeeld-session-start` & `voorbeeld-session` )

# sessions

- <http://www.php.net/manual/en/book.session.php>
- Opdracht: opdracht-sessions

# cookies

- Werkwijze
  - Cookie moet eerst geset worden en vereist een expiration date. Als je geen expiration date meegeeft, vervalst de cookie aan het einde van de sessie (=sluiten van de browser)
    - `setcookie('testCookie', '', time() + 3600);`    (=> cookie vervalst binnen één uur)
  - Om een cookie te verwijderen, moet de expiration date in het verleden liggen (= een negatieve waarde)
    - `setcookie('testCookie', '', time() - 3600);`
    - **MAAR!** Dit is niet veilig als de time van de browser op een andere tijdstip als de browser is ingesteld
      - 100% veilige manier:

```
setcookie('testCookie', '', 1);  
setcookie('testCookie', false);  
unset($_COOKIE[$name]);
```



# cookies

- Werkwijze
  - Session variables aanspreken (gelijkaardig aan `$_GET`, `$_POST` & `$_SESSION`):
    - `$_COOKIE['value'];`
  - Cookie moet geset of geunset worden **VOOR** elke echo, `print_r` of `var_dump`, dus: bovenaan het document! (als het niet anders kan: `header('location: url');` )
- (vb. **voorbeeld-cookie** )

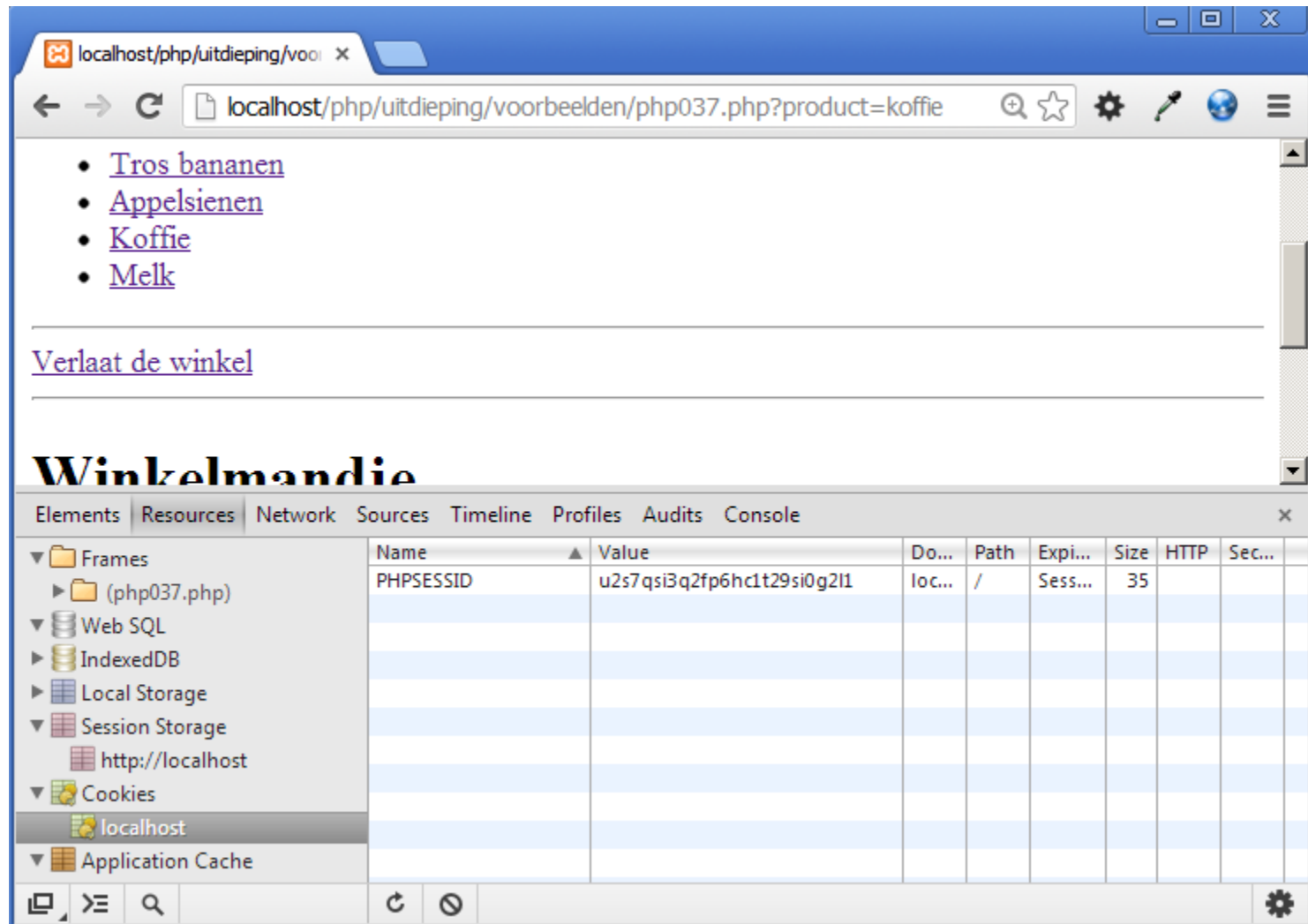
# cookies

- <http://php.net/manual/en/function.setcookie.php>
- Opdracht: opdracht-cookies

# sessions & cookies TIP

- Wat als tijdens het developen je `$_SESSION` en/of `$_COOKIE` array vol komt te staan met corrupte data (bv. na wijzigingen, debugging, ...)
  - `session_destroy()`, `cookie()` met negatieve expiration date, ...
    - omslachtig (je moet dit coderen)
    - Traag
  - Beter: in je webbrowser je sessionid/cookie manueel verwijderen

# sessions & cookies TIP

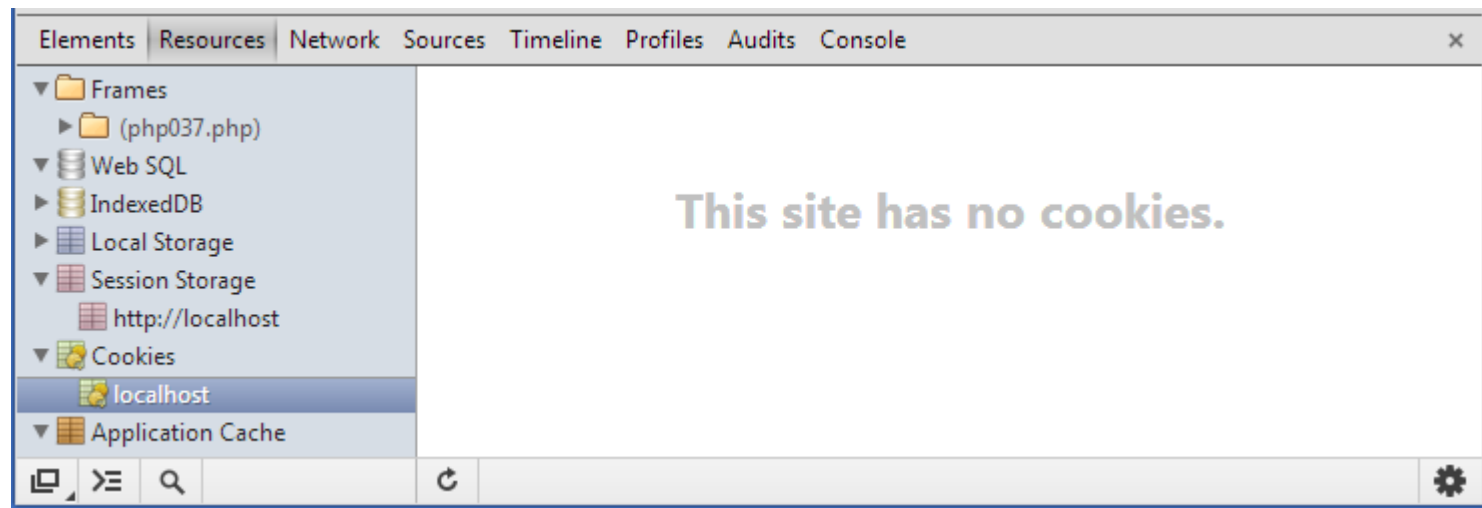


# sessions & cookies TIP

- In de developertools van je browser de RESOURCES nakijken
  - cookies
    - localhost (of domeinnaam waarop je aan het werken bent)
      - » De schuldige cookies/sessionids selecteren en verwijderen

Name ▲	Value	Do...	Path	Expi...	Size	HTTP	Sec...
PHPSESSID	u2s7qsi3q2fp6hc1t29si0g2l1	loc...	/	Sess...	35		

# sessions & cookies TIP



# include/require

- include/require zorgen ervoor dat je de inhoud van een ander bestand kan importeren in het huidige bestand.
  - De inhoud wordt onmiddellijk 'geprint' of gebruikt, naargelang het bestandstype (.txt, .html, .css, .php, ...)
- **require 'bestandsnaam.extensie'**: wanneer het bestand niet geïmporteerd kan worden, zal dit een fatal error (E\_COMPILE\_ERROR) opleveren
  - Dit stopt het script
- **require\_once 'bestandsnaam.extensie'**: Deze functie kijkt na of het bestand al eens is ingeladen. Wanneer het bestand niet geïmporteerd kan worden, zal dit een fatal error (E\_COMPILE\_ERROR) opleveren
  - Dit stopt het script

# include/require

- **Include 'bestandsnaam.extensie'**: wanneer het bestand niet geïmporteerd kan worden, zal dit een warning (E\_WARNING) opleveren
  - Dit stopt het script **niet**
- **include\_once 'bestandsnaam.extensie'**: wanneer het bestand niet geïmporteerd kan worden, zal dit een warning (E\_WARNING) opleveren
  - Dit stopt het script **niet**

(vb. **voorbeeld-include-require** )



# include/require

- Truukje:
  - een .html-bestand includen/requiren dat gebruik maakt van php-variabelen
    - Deze variabelen kan je zetten voor de include/require plaatsvind en zullen zo bekend zijn in alle code die je include/required, mits je gebruik maakt van de juiste php-syntax

(vb. [voorbeeld-include-require-cross-language](#))

# include/require

- Wanneer wat gebruiken? Hangt af van de situatie
- opm: include/require zijn krachtig, maar kunnen global namespace bevuilen & conflicten veroorzaken.
  - Oplossing: include/require in combinatie met classes
- Opdracht: [opdracht-include-require](#)

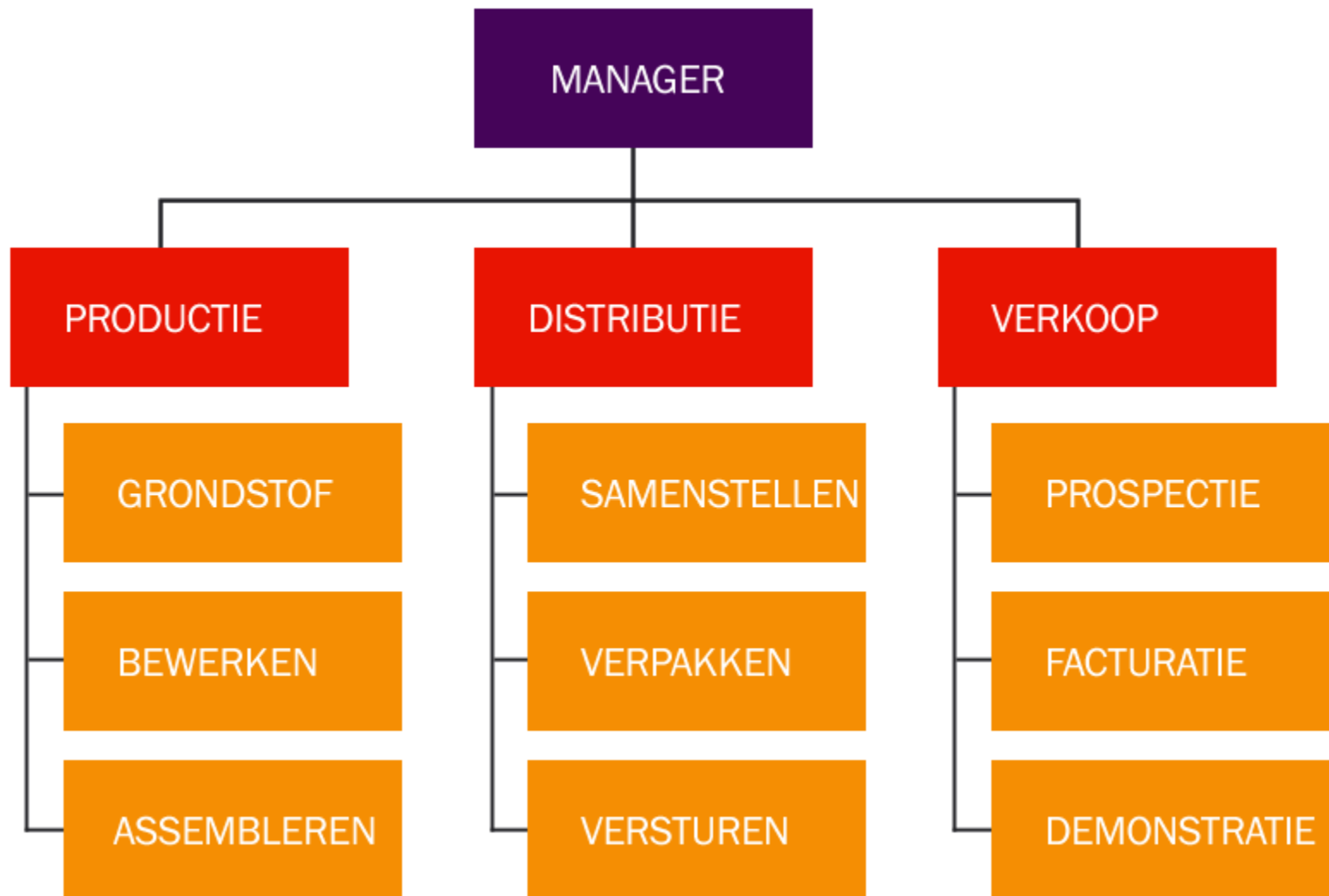
# classes

- Tot nu toe enkel procedurale code gebruikt
  - = functies en variabelen definiëren om daarna te combineren tot één grote applicatie (=de procedure).
  - Nadelen:
    - Bij grote applicaties -> overzicht moeilijk te bewaren
    - Herbruikbaarheid van code is beperkt
    - Nieuwe code implementeren:
      - vergroot kans op conflicten (globale variabelen/functies, ...)
      - vereist grondige kennis van de gehele applicatie
    - Werk moeilijk verdeelbaar onder verschillende developers
    - ...
- Oplossing:
  - Gebruik van classes (= object oriented programmeren)

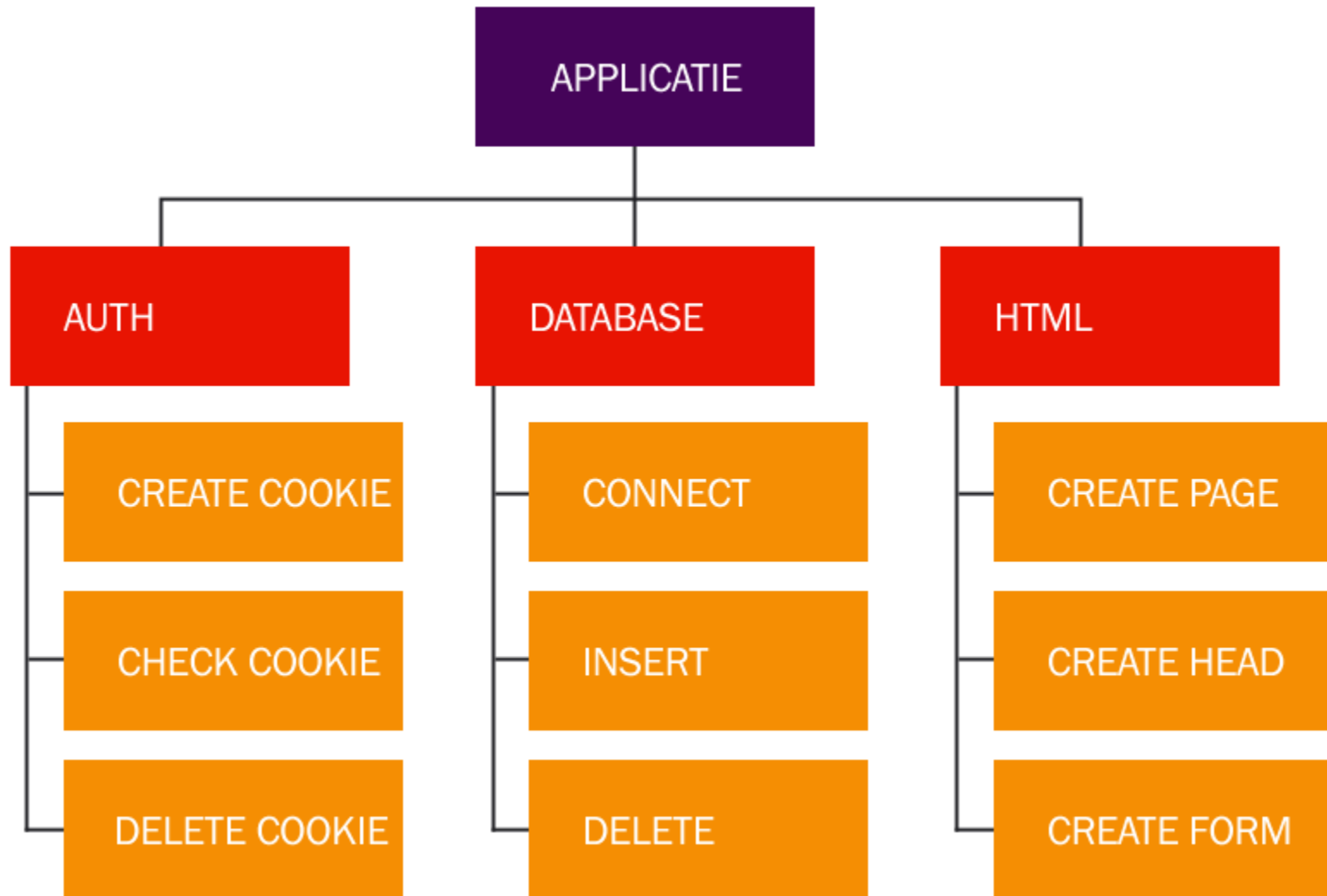
# classes

- Wat is een class?
  - Niet enkel gelijk aan een verzameling van functies en variabelen (-> library)!
  - Vooral: een op zichzelfstaande mini-applicatie (= 'object')
    - Staat in voor een kleine deeltaak van een applicatie
      - Bevat methodes die instaan voor de specifieke werking van de deeltaak
    - Is niet rechtstreeks afhankelijk van gegevens uit de buitenwereld
      - Om informatie uit de buitenwereld op te nemen, worden er setter-methodes gebruikt, ook wel **setters** genaamd
      - Om informatie naar de buitenwereld te sturen, worden er getter-methodes gebruikt, ook wel **getters** genaamd
    - Makkelijk herbruikbaar in verschillende situaties

# classes



# classes



# classes

- Syntax

```
class ClassName {
```

```
    public $variableName;
```

```
    public function functionName() {  
        code block;
```

```
    }
```

```
}
```

- Binnen een klasse zijn **properties** de verzamelnaam van variabelen en functies
  - Binnen een klasse heet een individuele variabele **\$variableName** een **member**
  - Binnen een klasse heet een individuele functie **functionName()** een **method**

# classes

- Syntax

- Toegang krijgen tot de members:
  - Binnen de klasse:

```
$this->memberName;  
$this->methodName();
```

- `$this` slaat op de **instantie** van de klasse
  - nodig om geen conflict te veroorzaken met variabelen eigen aan een bepaalde methode



# classes

## - Syntax

### - Buiten de klasse:

- Eerst een instantie aanmaken van de klasse

```
$classObject = new className();
```

- `new className();` is een **instantie** van de klasse *className* van het datatype 'object'
- De instantie van de klasse bevat alle **publieke properties** van die klasse
- Zoals bij een functie, is het mogelijk om argumenten met een klasse mee te geven -> zie `__construct()`

# classes

## - Syntax

- nadat instance werd aangemaakt:

`$variableInstanceName->memberName`

(bv. `$mysqli->insert_id`)

`$variableInstanceName->methodName()`

(bv. `$mysqli->fetch_assoc()` )

De properties zijn dus nooit rechtstreeks aanspreekbaar buiten de klasse (-> altijd via instance)

- scope van de members is altijd beperkt tot de class zelf

# classes

## - Syntax

- Conventie: klassenamen ALTIJD met een hoofdletter
  - makkelijk te herkennen als klasse
- meestal klasse niet rechtstreeks in de applicatie-code, maar in apart bestand & geïmporteerd dmv include/require
  - klassenaam gelijk stellen aan bestandsnaam
  - Eén klasse per bestand
    - best geen classes in hetzelfde bestand combineren -> zie \_\_autoload)
- (vb. **voorbeeld-classes-instance** )

# classes

- `__construct()`
  - Constructor wordt gebruikt om:
    - bij het aanmaken van de instance de eventueel meegegeven argumenten op te vangen
  - bepaalde handeling automatisch uit te voeren bij het maken van de instance van de klasse
    - bv. class members een waarde toekennen die het resultaat is van de returnwaarde van een functie

# classes

## - \_\_construct

- De variabelen die binnen de klasse gedefinieerd worden, kunnen strings, integers, arrays, ... zijn maar **geen functies**.

```
class className {  
    public $variableName01 = '1 januari 2010';  
    public $variableName02 = 123456789;  
    public $variableName03 = array(1,01,2010);  
    public $variableName04 = time();  
}
```

# classes

- \_\_construct syntax

- class *ClassName* {  
    **public** \$variable;  
  
    **public** function \_\_construct() {  
        \$this->variable = time();  
    }  
}

(vb. voorbeeld-classes-construct )

Opdracht: opdracht-classes-begin

# classes

- extends
  - Een klasse kan properties van een andere klasse overerven of **'extenden'**
    - Dit wil zeggen dat de klasse die een andere klasse extend, toegang heeft tot alle properties van die klasse (afh. v property visibility -> zie later)
  - Werkt bottom up
    - De extending classes (= **children**) hebben toegang tot de properties van de extended classes (= **parents**) .
    - De extended classes (= **parents**) hebben geen toegang tot de properties van de extending classes (= **children**)

# classes

- extends

- Syntax

```
class ParentClassName {
```

```
    codeblock;
```

```
}
```

```
class ChildClassName extends ParentClassName {
```

```
}
```

(vb. voorbeeld-classes-inheritance-extends -aanleiding & voorbeeld-classes-inheritance-extends )



# classes

## - extends

- Een klasse kan maximum één andere klasse extenden

~~—Arbeider extends Werknemer, Burger !!!~~

- Oplossing:
  - Arbeider extends Werknemer
  - Werknemer extends Burger
- Maakt OO op basis van classes archaïsch (<-> prototypal inheritance uit bv. JavaScript)

# classes

- Extends: parent:: & self::
  - Problemen:
    - De constructor van de parent wordt automatisch aangesproken bij aanmaken van een instantie van de child klasse
      - **Enkel** als de child klasse geen constructor heeft
      - Hebben zowel de child als de parent klasse een constructor, dan moet de parent constructor in de child klasse expliciet aangeroepen worden
        - Oplossing: **parent::**

(vb. [voorbeeld-classes-inheritance-parent](#) )

# classes

- Extends: `parent::` & `self::`

- Properties van parents kunnen overschreven worden door properties van de child
  - **\$this** verwijst naar de instantie van van de klasse, dus de referentie naar de originele methode is verloren
    - Oplossing: **self::**

(vb. [voorbeeld-classes-inheritance-self](#) )

# classes

## - Extends: parent:: & self::

- Bij properties van de parent klasse en de child klasse die dezelfde naam dragen, wordt de property van de parent klasse standaard overschreven door die van de child klasse.
  - Enkel als de property visibility op public of protected staat (-> zie property visibility)
  - bij methods enkel op voorwaarde dat het aantal argumenten overeenstemt (anders strict standard error -> in php geen method overloading mogelijk)

# classes

- **Property visibility**

- Probleem:

- Er zijn twee classes, de ene extend de andere en ze hebben elk een member/method met dezelfde naam, maar deze zijn specifiek aan de klasse en mogen **niet overschreven** worden

- Oplossing:

- De property visibility veranderen

# classes

## - Property visibility

- De property visibility beheert de scope van de properties van een bepaalde klasse
- Er zijn drie visibility properties:
  - **Public**
  - **Protected**
  - **Private**
- Elke property heeft **standaard de public property visibility**
  - Hoeft niet altijd geschreven te worden, maar is wel aan te raden (consistentie)

# classes

- Property visibility

- Public

- De property is **overal aanspreekbaar**, de scope reikt zowel tot buiten de klasse als binnen een andere klasse (= een parent klasse)

- Protected

- De scope van de properties reikt tot de **eigen en parent klasse**, maar **niet tot buiten de klasse**.

# classes

- Property visibility

- Private

- De property is **niet aanspreekbaar buiten de klasse** en ook **niet in child classes**. De scope van de property reikt enkel tot de eigen klasse.

- (vb. [voorbeeld-classes-property-visibility](#) )

- Opdracht: [opdracht-classes-extends](#)



# classes

- Static declaration
  - een property static maken, zorgt ervoor dat je deze property buiten de klasse kan aanspreken **zonder eerst een instantie van de klasse te moeten aanmaken.**
    - Gebruik dit niet uit luiheid (= “zo hoef ik niet eerst een instantie aan te maken”)
    - Gebruik dit enkel als je van een klasse een library wil maken die methods bevat die gerelateerd zijn, maar niet van elkaar afhangen.
    - (eventueel voor singletons -> zie design patterns)
    - Fout gebruik/misbruik zorgt voor klassenachtmerries -> gebruik dit dus zuinig en met kennis van zake!

# classes

- Static declaration

- Syntax:

```
public static $variable = 'test';
```

```
public static function functionName() {  
    codeblock;  
}
```

# classes

## - Static declaration

- properties aanspreken binnen de klasse

```
self::$variableName;
```

```
self::functionName();
```

- self is nodig omdat er geen instantie is aangemaakt van de klasse bij statische properties (= verwijst naar de klasse zelf)

- properties aanspreken buiten de klasse

```
ClassName::$variableName;
```

```
ClassName::functionName();
```

(vb. [voorbeeld-classes-static](#) )

# classes

- interface & abstract classes
  - soms is het nodig om in verschillende klassen dezelfde propertynamen én/of dezelfde propertyvalues te hebben.
    - om makkelijk klasseproperties van verschillende klassen met elkaar te vergelijken
    - om consistentie te bewaren
    - om bij extenden van gewone klassen een overvloed van method overwriting te voorkomen

# classes

- interface & abstract classes

- Interfaces

- zegt welke properties een klasse moet implementeren, maar geeft geen invulling aan deze properties
    - Interfaces kunnen elkaar extenden en kunnen meer dan één interface extenden.
    - Alleen methods kunnen aan een interface toegevoegd worden
    - De methods moeten een public visibility hebben
    - een klasse kan meerdere interfaces implementeren

# classes

- interface & abstract classes

- syntax

```
interface InterfaceName
{
    public function memberName;
    public function methodName($para1, $para2);
}
```

...

class ClassName implements InterfaceName

```
{
    public function memberName = 'string';

    public function methodName($para1, $para2)
    {
        // Code block;
    }
}
```

- (vb. **voorbeeld-classes-interface** )

# classes

## - interface & abstract classes

### - abstract class

- Is een combinatie van een gewone klasse en een interface
  - Andere klassen extenden een abstracte klasse en er kan dus maar maximum één abstracte klasse geëxtend worden
  - Definieert properties die een child klasse moet implementeren, maar geeft geen invulling aan deze properties
  - Definieert uitgewerkte properties die op normale wijze aangesproken kunnen worden in de child klasse (worden automatisch in child klasse opgenomen)
  - De visibility van een abstracte property mag maximum even strict zijn in de child klasse, maar kan eventueel een minder strenge visibility krijgen
- Een klasse kan zowel een interface implementeren als maximum één (abstracte) klasse extenden

# classes

- interface & abstract classes

- syntax

```
abstract class AbstractName
{
    abstract public function memberName;
    abstract public function methodName($para1, $para2);

    public function doSomething()
    {
        // Code block
    }
}

...

class ClassName extends AbstractName
{
    public function memberName = 'string';

    public function methodName($para1, $para2)
    {
        $this->doSomething();
        // Code block;
    }
}
```

- (vb. voorbeeld-classes-abstract )



# namespace

- Classes -> afzonderlijke modules (of applicaties) die je kan bundelen zonder dat er conflicten optreden
  - Maar, wat doe je als twee modules een class hebben die hetzelfde heet?
    - Bv. Bij de volgende structuur:  
classes
      - Bing
        - Engine.php
      - Google
        - Engine.php
  - Conflict!
    - Welke Engine-class moet geladen worden?
    - Wat als beide classes geladen moeten worden?

# namespace

- Oplossing: namespaces

**namespace** bing;

- Zorgen ervoor dat alle classes/functies/... die onder de namespace gedefiniëerd worden onder de gedefiniëerde namespace vallen en niet aan de globale namespace worden toegevoegd zoals standaard het geval is

# namespace

- Eigenschappen:
  - Altijd bovenaan .php bestand
  - Namespace niet tussen quotes
  - Namespace best niet beginnen met hoofdletter (= weggelegd voor classnames)
  - afsluiten met ;
  - Namespace kan bevatten:
    - Constants
      - » OPM: mits `const CONSTNAAM = ""` ipv `define( "CONSTNAAM" , "" )`, aangezien `define()` een constant standaard aan de global namespace toevoegt
    - Functions
    - Classes
      - » Meestal enkel voor classes gebruikt
    - Géén variables
  - Buiten de namespace aanspreken dmv `namespacenaam\Classname()`
- vb. `voorbeeld-namespace`

# namespace

- Namespaces kunnen genest worden
  - “sub-namespaces”
  - De namespacenaam volgt best de benaming van de folderstructuur
    - » Makkelijker te gebruiken in samenwerking met autoload
  - Let daarbij goed op hoofdletters (windows ≠ hoofdlettergevoelig, linux = hoofdlettergevoelig -> de meest servers... linux)
- De namespace waar je in zit wordt als root beschouwd
  - Om bij het begin van de namespace te starten, gebruik \
  - voorbeeld-namespace-nesting

# namespace

- Lange namespaces kunnen een alias krijgen
  - Houdt code overzichtelijk

**use** dit\is\veel\te\lang **as** divtl;

- Vb. **voorbeeld-namespace-alias** (google/user/User.php)

# Error handling

- Wanneer een fatale fout zich voordoet -> stopt het script
- Soms komen er fouten voor die niets met PHP-code te maken hebben
  - Non-fatal error
  - ‘Errors’ omdat data niet voldoet aan een bepaalde conditie
- Vroeger: opvangen dmv if-lus
  - betere optie: toetsen via try-catch-block
- Try/catch-block centraliseert foutafhandeling
  - Vermindert te schrijven code
  - Vergemakkelijkt uitbreidingen
  - Alle foutboodschappen kunnen op één plaats gebundeld worden

# Error handling

- Twee delen:
  - Try-block
    - Hierin worden errors **gethrowd**
    - Er moet altijd een datatype Exception gethrowed worden, of een subklasse van het datatype Exception (bv. PDOException)
    - Wanneer er een throw in het script wordt gevonden, gaat het script verder vanaf de catch
  - Catch-block
    - Hierin worden de gethrowde errors opgevangen
    - Heeft een parameter -> deze wordt doorgaans \$e genoemd
      - Is een instantie van Exception (of een subklasse van Exception)
      - Heeft dus specifieke methodes
        - » Bv \$e->getMessage();
    - Kan je bekijken als een functie die instaat voor het afhandelen van eender welke error die gethrowed wordt, op eender welke plaats

# Error handling

- ```
try
{
    if ( FALSE )
    {
        throw new Exception( 'Dit is een fout' );
    }
}
catch ( Exception $e )
{
    echo $e->getMessage();
}
```
- (vb. `voorbeeld-error-handling` &&  
`voorbeeld-error-handling-genest` &&  
`voorbeeld-error-handling-genest-log` )



# Error handling

- In de praktijk:
  - Codes opwerpen ipv tekst
    - Op basis van de code wordt er een stukje tekst uit de db gehaald.
    - Biedt de mogelijkheid om errors naar bv. vertaler te sturen, zonder al teveel grote aanpassingen
    - Opgepast! Best enkel te gebruiken voor fouten die fataal zijn voor de werking van een script (= “kon geen connectie maken met de database”, ~~!= “Het paswoord is niet correct”~~)
  - Zie [opdracht-error-handling](#)

# \_\_autoload() {}

- \_\_autoload()
  - Wanneer meerdere klassen vereist zijn -> verschillende includes/requires uitvoeren.
    - Dit is erg omslachtig
    - Al snel erg grote lijst met veel includes/requires
- **Oplossing:** \_\_autoload() {}

# \_\_autoload() {}

## - \_\_autoload() {}

### - Hoe werkt autoload?

- Van zodra je een instantie van een klasse aanmaakt en deze wordt niet teruggevonden, controleert PHP of de autoload-functie is ingeladen.
- Deze autoload-functie kijkt van welke klassenaam een instantie aangevraagd wordt en zoekt naar een bestand dat deze klassenaam bevat
  - De **naam van de klasse** moet letterlijk **terug te vinden zijn in de bestandsnaam**.

# \_\_autoload() {}

- \_\_autoload() {}

- Syntax

```
function __autoload($classname) {  
    require_once($classname . '.php');  
}
```

- (vb. voorbeeld-autoload )

# \_\_autoload() {}

- \_\_autoload() {}

- OPM: sprake van \_\_autoload() te deprecateren

- Nog geen echte duidelijkheid

- <http://php.net/manual/en/language.oop5.autoload.php>

- Alternatief: spl\_autoload\_register();

- (vb. [voorbeeld-autoload-spl](#) )

# CRUD (database)

- PHP komt pas volledig tot zijn recht in samenwerking met een database.
- Database wordt gebruikt om op een veilige en dynamische manier data opslagen.
  - Van gebruikersnamen, tot paswoorden, artikels en foto's.
- MySQL en PHP hebben elk hun eigen syntax, maar vallen perfect te combineren.
- Gebruik zoveel mogelijk MySQL om data voor te bereiden. Bv. alfabetisch rangschikken op basis van **order by** ipv gebruik te maken van PHP-functies.

# CRUD (database)

- Drie manieren om DB aan te spreken in php:
  - **mysql** (proceduraal -> pre php5)
    - Verouderd, niet meer gebruiken!
  - **Mysqli** (class-based -> php5+)
    - Gebonden aan MySQL
    - Soms omslachtige werkwijze
    - Volstaat, maar betere alternatieven -> PDO
  - **PDO** (class-based -> php5+)
    - Database-agnostisch
    - Meest efficiënte standaard klasse om databasebewerkingen uit te voeren
    - Meest veilige klasse

# CRUD (database)

- MySQLi

- Connectie met MySQL database:

- `$var = new mysqli('serveradres', 'username', 'paswoord');`  
(= instantie aanmaken van de mysqli-klasse)
    - Wordt automatisch afgesloten na het beëindigen van het script
      - Of manueel: `$var->close();`
    - (vb. `voorbeeld-crud-connectie-mysqli` )

- MySQL-database selecteren

- `$db->select_db('databaseName');` (bv. na het creëren van een database)

MAAR meestal is de database op voorhand gekend -> meegeven als parameter in connectie

- `$var = new mysqli('serveradres', 'username', 'paswoord', 'databaseNaam');`



# CRUD (database)

- PDO

- Connectie met MySQL database & database selecteren:

- `$var = new pdo('mysql:host=localhost;dbname=bieren', 'username', 'paswoord');`

- (= instantie aanmaken van de mysqli-klasse)

- Wordt automatisch afgesloten na het beëindigen van het script

- Opgelet: moet tussen try/catch-blokken! (zie error-afhandeling)

- (vb. [voorbeeld-crud-connectie-pdo](#) )

# CRUD (database)

- SQL-queries uitvoeren dmv PHP
  - `$result = $var->query('MYSQL QUERY');`
  - (vb. [voorbeeld-crud-query-execute](#) )
- query omzetten naar php-array (MySQLi):
  - `$result->fetch_row();` //Geeft een array met alle geselecteerde DB-values
  - `$result->fetch_assoc();` //Geeft een associatieve array (tabelnaam => value)
  - `$result->fetch_array();` //Een combinatie van `$result->fetch_row()` en `$result->fetch_array()`
  - (vb. [voorbeeld-crud-query-result-mysqli](#) )

# CRUD (database)

- query omzetten naar php-array (PDO):
  - `$statement->fetch( );` //Geeft een array met alle geselecteerde DB-values
  - `$statement->fetch( PDO::FETCH_ASSOC );` //Geeft een associatieve array (tabelnaam => value)
  - `$statement->fetch( PDO::FETCH_BOTH );` //Een combinatie van numeriek en associatief
  - (vb. `voorbeeld-crud-query-result-pdo`)

# CRUD (database)

- Om alle gevonden resultaten te overlopen:

```
while($row = $result->fetch_array()) {  
    echo $row['kolomnaam1'];  
    echo $row[0];  
}
```

**OPM:** Vergeet `$result->data_seek(0)` (MySQLi) of `$result->execute()` (PDO) niet wanneer je meerdere bewerkingen op hetzelfde resultaat toepast

# CRUD (database)

- Prepared statements
  - Invoerdata (\$\_POST, \$\_GET) mag nooit rechtstreeks in een query-statement worden ingevoerd
    - Groot securityrisico -> SQL-injecties
  - Variabele delen van de query worden in 'placeholders' geplaatst.
  - Placeholder worden ingevuld door middel van een 'bind parameter' functie
    - Escaped SQL-code zodat ze probleemloos in MySQL gebruikt kunnen worden
  - Gebruik deze **áltijd** (MySQLi of PDO)
    - geen MySQLi of PDO ? ->mysql\_real\_escape\_string()

# CRUD (database)

- Prepared statements (MySQLi)
  - `$query = 'SELECT bieren WHERE bieren.alcohol LIKE ?';`  
  
`$db->prepare( $query );`  
  
`$db->bind_param('i', 7);`  
  
`$db->execute( );`
  - MySQLi placeholder ( = ? ) zijn tamelijk omslachtig en verwarrend
    - `bind_param()`
      - 1<sup>ste</sup> argument string
        - » `i=integer`
        - » `s=string`
        - » `d=double`
        - » `b=blob`
      - 2<sup>de</sup> argument: effectieve waarde
    - Meerdere placeholders mogelijk, maar omslachtig -> in dezelfde functie
  - (vb. [voorbeeld-crud-query-result-prepared-statements-mysqli](#) )

# CRUD (database)

- Prepared statements (PDO)

- `$query = 'SELECT bieren WHERE bieren.alcohol LIKE :alcoholPercentage';`

`$db->prepare( $query );`

`$db->bindValue(':alcoholPercentage', 7);`

`$db->execute( );`

- PDO placeholder starten met een :

- Mag je zelf kiezen, maar : wordt niet herkend als MySQLi statement -> meest veilige optie
- Per placeholder een aparte `bindValue()` aanspreking (itt MySQLi)
  - `bindValue()` is doorgegeven op basis van value “pass by value”
  - `bindParam()` is doorgegeven op basis van reference “pass by reference”

- (vb. [voorbeeld-crud-query-result-prepared-statements-pdo](#) )

# CRUD (database)

- Stappenplan om te debuggen

Door de combinatie van PHP/MySQL is het moeilijk te zeggen waar er eventueel een fout zit. Daarom:

1. Maak de query in phpmyadmin-console
    - Pas als deze query het gewenste resultaat geeft, copy-pasten naar PHP
  2. Plaats de query in een variabele en vervang met placeholders
    - Je mag enters gebruik in de query-string, doe dit om het overzicht te bewaren
  3. Echo de query en kijk of alles correct gecopy-paste is
    - geen spaties vergeten?
  4. Pas daarna de query uitvoeren
    - Een fout resultaat? Dan zit de fout vrijwel zeker in de PHP-code
- Opdracht: [opdracht-CRUD-query](#)



# CRUD (database)

- Datarijen **inserten**, **deleten** en **updaten** gebeuren op exact dezelfde manier als een gewone SQL-query
  - Aan de PHP-syntax verandert niets
- Opdracht: opdracht-CRUD-insert
- Opdracht: opdracht-CRUD-delete
- Opdracht: opdracht-CRUD-update
- Opdracht: opdracht-CRUD-query-order-by
































# security

- Algemeen:
  - OWASP
    - [https://www.owasp.org/index.php/PHP\\_Security\\_Cheat\\_Sheet](https://www.owasp.org/index.php/PHP_Security_Cheat_Sheet)

# security

- Tracking details
  - Een extra tabel die wordt gekoppeld aan elke nieuwe rij die in de database wordt toegevoegd.
    - Bv. artikels, gebruikers, ...
  - Wat zit er in deze tabel?
    - **created\_by\_user\_id, created\_on**: wanneer een nieuwe rij wordt aangemaakt
    - **changed\_by\_user\_id, changed\_on**: wanneer een bestaande rij wordt aangepast
    - **is\_archived** (bool), **archived\_by\_user\_id, archived\_on**: wanneer een nieuwe rij wordt 'verwijderd'.
    - **is\_active** (bool), **activated\_by\_user\_id, activated\_on**: wanneer een bestaande rij wordt geactiveerd.

# security

| <div><div> Browse</div><div> Structure</div><div> SQL</div><div> Search</div><div> Insert</div><div> Export</div><div> Import</div><div> Operations</div><div></div></div> |                                |            |           |            |      |         |                |  |  |        |
|---|--------------------------------|------------|-----------|------------|------|---------|----------------|--|--|--------|
| #   | Column                         | Type       | Collation | Attributes | Null | Default | Extra          | Action   |  |        |
| <input type="checkbox"/>  | 1 <b>id</b>                    | int(11)    |           |            | No   | None    | AUTO_INCREMENT |  Change   |  Drop   | More ▼ |
| <input type="checkbox"/>  | 2 <b>created_by_user_id</b>    | int(11)    |           |            | No   | None    |                |  Change   |  Drop   | More ▼ |
| <input type="checkbox"/>  | 3 <b>created_on</b>            | datetime   |           |            | No   | None    |                |  Change   |  Drop   | More ▼ |
| <input type="checkbox"/>  | 4 <b>changed_by_user_id</b>    | int(11)    |           |            | No   | None    |                |  Change   |  Drop   | More ▼ |
| <input type="checkbox"/>  | 5 <b>changed_on</b>            | datetime   |           |            | No   | None    |                |  Change   |  Drop   | More ▼ |
| <input type="checkbox"/>  | 6 <b>is_archived</b>           | tinyint(1) |           |            | No   | None    |                |  Change   |  Drop   | More ▼ |
| <input type="checkbox"/>  | 7 <b>archived_by_user_id</b>   | int(11)    |           |            | No   | None    |                |  Change   |  Drop   | More ▼ |
| <input type="checkbox"/>  | 8 <b>archived_on</b>           | datetime   |           |            | No   | None    |                |  Change  |  Drop  | More ▼ |
| <input type="checkbox"/>  | 9 <b>is_active</b>             | tinyint(1) |           |            | No   | None    |                |  Change |  Drop | More ▼ |
| <input type="checkbox"/>  | 10 <b>activated_by_user_id</b> | int(11)    |           |            | No   | None    |                |  Change |  Drop | More ▼ |
| <input type="checkbox"/>  | 11 <b>activated_on</b>         | datetime   |           |            | No   | None    |                |  Change |  Drop | More ▼ |

# security

- Tracking details
  - created:
    - Wanneer een rij wordt aangemaakt
      - Bv. bij het toevoegen van een gebruiker
    - de created\_on (NOW()) en created\_by\_user\_id (id van de zopas toegevoegde gebruiker) moeten aangepast worden
  - Géén is\_created
    - onlogisch om hier een bool voor te voorzien omdat we weten dat iets is aangemaakt wanneer er een datum/id vermeld is.

# security

- Tracking details
  - changed:
    - Wanneer een rij *inhoudelijk* wordt aangepast
      - Bv. het verbeteren van een taalfout in de titel
    - de `changed_on` (`NOW()`) en `changed_by_user_id` moeten aangepast worden
    - Géén `is_changed`
      - onlogisch om hier een bool voor te voorzien omdat we weten dat iets is aangepast wanneer er een datum/id vermeld is.

# security

- Tracking details

- active:

- Wanneer een rij al is aangemaakt maar nog niet gebruikt mag worden op de hoofdpagina
      - bv. wanneer een artikel nog moet nagekeken worden door de redacteur alvorens te publiceren op de hoofdpagina
      - Bv. wanneer een gebruiker zijn email nog moet bevestigen
    - de is\_active (bool), activated\_on (NOW()) en activated\_by\_user\_id moeten aangepast worden

# security

- Tracking details
  - archived:
    - Wanneer een rij verwijderd moet worden.
    - LET OP: Laat een gebruiker NOOIT data uit de database verwijderen.
    - Werk met `is_archived` en verberg de rijen (dmv SQL) waarbij de `is_archived` op 1 staat.
      - Zo lijkt het alsof de data verwijderd is, terwijl deze nog in de database terug te vinden is.



# security

- Tracking details
    - archived:
      - Voordeel:
        - mensen met kwade bedoelingen kunnen niets verwijderen.
        - Dingen die per ongeluk werden verwijderd, kunnen makkelijk weer opgehaald worden.
      - de is\_archived (bool), archived\_on (NOW()) en archived\_by\_user\_id moeten aangepast worden
    - **Opmerking:** tracking\_details hoeven niet altijd in een aparte tabel, kunnen ook in de te tracken tabel verwerkt worden.
- (vb. [voorbeeld-security-tracking-details](#) )

# security

- Tracking details in samenwerking met user\_type
  - Minst ideaal: geen rekening gehouden met user types (gebruiker, admin, superadmin, ...)
    - Wanneer er iets verwijderd werd of gemonitord moet worden -> opzoeken in de database
  - Betere oplossing: bepaalde pagina's (of opties) enkel toegankelijk maken voor bepaalde gebruikers
    - bv. gearchiveerde artikels wél tonen wanneer het user type gelijk is aan 0 (= superadmin)

# security

- Tracking details in samenwerking met user\_type

Als gebruiker:

U bent ingelogd.  
Ingelogd als test@test.be | [uitloggen](#)

- [Artikels](#)
- [Gegevens wijzigen](#)

Als superadmin:

Ingelogd als superadmin@test.be | [uitloggen](#)

- [Artikels](#)
- [Gegevens wijzigen](#)
- [Overzicht van gebruikers](#)

- (vb. [voorbeeld-security-user-type](#) )

# security

- XSS (Cross-Site Scripting)
  - Het injecteren van JavaScript om zo cookies te stelen of gevoelige informatie te ontfutselen
  - Ontstaat wanneer een gebruiker data kan invoeren die niet gecontroleerd wordt op aanwezigheid van JS/HTML
    - vb. [voorbeeld-security-XSS](#)
  - Beperking: enkel mogelijk waar input-data wordt weergegeven (dus niet de volledige site)
  - Oplossing:
    - Voor insert in database of print: data escaper dmv `html_escape()`

# security

- String hashing (MD5, Sha-512, ...)
  - Probleem: paswoorden zijn zichtbaar in de database.
    - Oplossing: String hash  
bv MD5:  
**paswoord** = 26d01f0b3dcd7cbddb2de08c3a95a740  
**gewooneenhelezin** = e2f70e6cd3e1b3e9bc37c67627eaef02
  - Ongeacht hoeveel karakters, het resultaat is altijd een gescrambelde string met een vast aantal karakters.
  - De hash van een bepaalde string blijft altijd gelijk.

# security

- String hashing (MD5, Sha-512, ...)
  - Voordeel: paswoorden worden onherkenbaar gemaakt
  - Nadeel: niet onfeilbaar
    - door luie gebruikers.
      - azerty = ab4f63f9ac65152575886860dde480a1
      - admin = 21232f297a57a5a743894a0e4a801fc3
      - 123456 = e10adc3949ba59abbe56e057f20f883e
      - ... rainbow tables (=gehashte woordenboeken)
    - collisions (bv. MD5 hash met een andere input, maar met dezelfde MD5 hash)
    - SQL-injecties
  - (Semi-)veilig: Blowfish, SHA-512, ...
  - (vb. [voorbeeld-security-hash](#) )

# security

- String Hash + salt
  - Oplossing voor veel voorkomende paswoorden
  - Salt is een paswoord dat enkel de admin kent en die bij het paswoord/... wordt gevoegd.
  - Dit paswoord geconcateneerd met de salt wordt gehashed, waardoor het minder duidelijk is om welk paswoord het gaat.
    - Niet veilig tegen SQL-injecties (afh. v. mysqli/mysql)
    - Niet veilig tegen rainbow tables
    - Bij statische salts (=dezelfde salt voor alle hashes) is de kraakkans groter dan bij dynamische (= unieke salt per hash)
  - (vb. **voorbeeld-security-hash-salt** )

# security

- Gebruiker validaten
  - Door middel van cookie
    - `$salt = (ophalen uit de database)`  
`$cookieValue = $email . ' ' . hash('SHA512', $email . $salt);`  
`setcookie('login',$cookieValue, time() + 60*60*24);`
  - Hash om te voorkomen dat mensen die e-mail van iemand kennen, een cookie kunnen aanmaken met het e-mailadres van die gebruiker (en dus kunnen inloggen).
  - Salt verplicht gebruiker om via het inlogformulier te passeren, anders kan de salt nooit geweten zijn.



# security

- SQL-injecties (mysql)
  - Voor oude php-code die nog met de mysql-procedure tewerkgaat (mysql ≠ mysql*i*):
    - Deel SQL-query invoeren via een post-formulier
      - `SELECT *`  
`FROM users`  
`WHERE email = " . $_POST['email'] . '"`  
  
mogelijke ingevoerde waarde: " OR "a" = "a"
      - Zal altijd een user inloggen, want
        - » `SELECT *`  
`FROM users`  
`WHERE email = "" OR "a" = "a"`  
  
er is altijd een resultaat  
  
(vb. [voorbeeld-security-sql-injection](#) )

# security

- SQL-injecties (mysql)
  - Oplossing:
    - `mysql_real_escape_string()`
      - `SELECT *`  
`FROM users`  
`WHERE email = '' . mysql_real_escape_string( $_POST['email'] ) . ''`
      - Escaped alle karakters die voor conflicten kunnen zorgen
    - Beste oplossing: werken met prepared statements
  - Opdracht: [opdracht-security-login](#)
  - Synthese-opdracht: [opdracht-CRUD-CMS](#)

# Constants

- Constants zijn variabelen die je één keer kan definiëren en waarvan je de waarde niet meer kan overschrijven
  - `define('CONSTANT_NAME', 'value');`
- Worden altijd in hoofdletters geschreven
  - Conventie, niet verplicht
  - Duidelijk herkenbaar als constante
- Worden in PHP zonder \$-teken geschreven
- Zelfs niet herdefinieerbaar

(vb. [voorbeeld-constants](#) )

# File upload

- File upload
  - De gebruiker kan bestanden uploaden, maar hiervoor moet het formulier aangepast worden:

```
<form action="upload_file.php" method="post"  
  enctype="multipart/form-data">
```

- Ook moet er een apart input-veld worden aangemaakt om een bestand te kunnen uploaden:

```
<input type="file" name="file">
```

# File upload

- File upload
  - Op php-niveau vangen we dit als volgt op
  - Je beschikt over het bestand door middel van de `$_FILES`-variabele (=array)
    - Het bestand krijgt de naam van het input-type-file veld, dus: `$_FILES['input-type-file-name']`
    - Dit is een array, waarin de volgende keys automatisch worden ingevuld
      - **name** (de naam van het bestand)
      - **size** (de grootte van het bestand)
      - **tmp\_name** (de naam die het bestand krijgt in afwachting dat het effectief geupload wordt naar de server)
        - Het bestand staat dus niet onmiddellijk op de server
      - **error** (wanneer er iets misging bij het doorgeven van de file naar de server)

# File upload

- File upload

- Het bestand uploaden naar een definitieve bestemming:

`move_uploaded_file($tempname, $filename);`

- \$tempname is de voorlopige bestandsnaam in de tmp-folder (= \$\_FILES['input-type-file-name']['tmp'])
    - \$filename is de folder geconcateneerd met de bestandsnaam zoals die in de definitieve folder geupload moet worden.

# File upload

- File upload
  - Security
    - file-uploads veroorzaken de grootste securitylekken -> uploaden van bestanden die uitgevoerd kunnen worden (.php)/overschrijven/...
    - Enkele regels:
      - ALTIJD het **type controleren** dmv `$_FILES['inputname']['type'];`
        - NOOIT op basis van extensie in de bestandsnaam
      - Niet blacklisten, maar **whitelisten**
        - (= enkel de extensies/files opsommen die zijn toegestaan ipv de extensies die niet zijn toegestaan)
      - Stel altijd een **maximum bestandsgrootte** in
      - Bij multiple file upload: **limiteer het aantal bestanden** per keer!

# File upload

- File upload
  - Security
    - Sla **bestanden op BUITEN de root folder** en niet in dezelfde map als de code
      - wwwroot/uploads/
      - via .htaccess-bestand toegang van deze bestanden beperken tot die folder
        - Dit .htaccess bestand niet in current directory, maar parent directory plaatsen (zo kan .htaccess niet overschreven worden)
    - De bestandsnamen **ALTIJD hernoemen** naar een unieke naam
      - Op basis van timestamp/hash
      - Om overschrijven te voorkomen
    - Apache **updaten** (nieuwe versies voorkomen veel exploits)
    - ...



# File upload

- File upload
  - (vb. **voorbeeld-file-upload** )
  - Opdracht: **opdracht-file-upload**

# Image manipulation

- Imagemanipulaties via PHP zijn ook mogelijk
  - Om bv. thumbnails te maken
  - Om afbeeldingen te bewerken/
    - bv <http://mustache.me/>
    - ... (wordt nu veel meer opgevangen met HTML5 & <canvas>)
  - Om afbeeldingdata op te slaan
    - Google images > search tools > ...
      - Dimensies
      - Kleur
      - Gezichten
      - ...
- Dmv GD library
  - Zit meestal standaard in php-module van apache
    - Controleren? -> echo phpinfo();
  - <http://www.php.net/manual/en/ref.image.php>

# Image manipulation

- Enkele mogelijkheden:
  - Resizen
  - Stukjes uitsnijden (croppen)
  - Comprimeren (kwaliteit wijzigen -> enkel downgraden)
  - Afbeeldingen combineren
  - Pixelmanipulaties
    - Grijswaarden/highlights/...
    - Kleuren tellen
  - ...
- (vb. [voorbeeld-image-manipulation](#) ) (resizen)
- Opdracht: [opdracht-image-manipulation-thumb](#)
- Opdracht: [opdracht-image-manipulation-gallery](#)

# Mail

- Mail-functie
  - PHP staat toe om vanaf de server mails te versturen
    - Handig voor contactformulieren, mailerapplicaties,...
- Eerst xampp instellen:
  - xampp/php/php.ini
    - Pas aan: smtp = **uit.telenet.be**  
(OPM: Zal enkel werken als je telenet hebt -> belgacom, scarlet, ... hebben dus een andere smtp-servernaam)
    - Herstart xampp

# Mail

## - Mail-functie

- `mail($aan,$titel,$bericht,$headers, $parameters);`
  - \$aan is het e-mailadres van persoon naar wie e-mail moet gestuurd worden
  - \$titel is de titel van het bericht
  - \$bericht is de 'body' van het bericht
  - \$headers is de metadata die bij een mail horen (optioneel afhankelijk van de mailserver)
    - FROM: [afzender@email.com](mailto:afzender@email.com)
    - DATE: ...
    - HTML in bericht toestaan of niet?
    - ... voor meer headers: zoeken naar **header fields**

# Mail

## - Mail-functie

- \$parameters (optioneel, afhankelijk van de mailserver) dient om commandlines mee te geven (bv. om files mee te sturen,...)
- OPM: de standaard mail()-functie in PHP is zeer primitief
  - geen foutafhandeling
  - Moeilijk om berichten aan te maken (cc, bcc, html, geen html, ...)
  - -> beter aparte mail-klasse gebruiken (bv. [PHPMailer](#) of [PEARmail](#))
- (vb. [voorbeeld-mail](#) )
- Opdracht: [opdracht-mail](#)

# AJAX-call opvangen met PHP

- Principe:
  - Een php-pagina aanspreken met JavaScript
    - POST/GET/FILE-waarden doorsturen
  - PHP bepaald antwoord laten genereren op basis van de doorgestuurde data
    - Data meestal omzetten naar JSON-formaat (of XML, ...)
    - Dit antwoord letterlijk echoën
  - JavaScript het antwoord laten parsen en verwerken op de huidige pagina.

# AJAX-call opvangen met PHP

- Voordelen:
  - Vermindert trafiek/laadtijden/... -> enkel dat wat geladen moet worden, wordt doorgestuurd ( <-> ipv de hele pagina te vernieuwen.)
  - Schermt bepaalde gevoelige delen van een applicatie af en biedt toegang op een gecontroleerde manier.
  - Volledig nieuwe stroming van applicaties mogelijk (real-time, ...)
- Nadelen:
  - Complexere logica
  - Moeilijk te debuggen (geen zicht op PHP-pagina)
- (vb. **voorbeeld-ajax-call-receive** & **voorbeeld-ajax-call-process**)
- Opdracht: **opdracht-ajax**



# Regular expressions

- Een krachtige engine die heel efficiënt stukken tekst, woorden, karakters of cijfers uit een string kan selecteren.
- Wordt eveneens ingezet om te controleren of een string aan bepaalde conventies voldoet.
- Geïmplementeerd door verschillende programmeertalen
- Veel makkelijker dan string-bewerkingen, maar leercurve is redelijk stijl door de abstracte operatoren.
- In de meeste gevallen veel sneller dan standaard PHP string-bewerkingen (vooral voor complexe bewerkingen)

# Regular expressions

- **regular expressions methodes in PHP**
  - Returnt GEEN positie van de match, in tegenstelling tot bv. **strpos()**;
  - Kunnen gebruikt worden in php:
    - **preg\_match(\$regEx, \$searchString);**
      - » Returnt:
        - **1** als er een match was,
        - **0** als er geen match was
        - **FALSE** (of error) bij een foutieve regular expression

# Regular expressions

- regular expressions methodes in PHP

- `preg_match_all($regEx, $searchString, $returnArray);`

- » returnt:

- **>0** als er een match was (getal = aantal matches)
        - En bevolkt de `$returnArray` met alle matches
      - **0** als er geen match was
        - En laat de `$returnArray` leeg
      - **FALSE** (of error) bij een foutieve regular expression
        - En laat de `$returnArray` leeg

- » OPM: `$returnArray` is rechtstreeks aanspreekbaar buiten de functie (wordt dus aangemaakt door de functie en hoeft niet elders gedefinieerd te worden)

# Regular expressions

- regular expressions methodes in PHP

- preg\_replace(\$regEx, \$replaceString, \$searchString);

- » Returnt:

- De \$searchstring met de matches vervangen door de \$replaceString
      - De \$searchstring in originele staat
      - **NULL** (of error) bij een foutieve regular expression

- (vb. [voorbeeld-regular-expressions-functions](#))

# Regular expressions

- **regular expressions**

- **OPM:** in PHP moeten alle regular expressions tussen forward slashes.

```
preg_match('/a/', 'banaan');
```

- Dit omdat de preg\_...() functies gebruik maken van PERL en in PERL wordt een string aangeduid door de string tussen / te plaatsen.
- Specifiek voor PHP(PERL) -> andere talen hebben mogelijk andere conventies.

# Regular expressions

- **regular expressions operators**

(vb. [voorbeeld-regular-expressions-operators](#))

- **^karakter** : anchor, selecteert karakter/string aan begin van string  
(OPM: is niet gelijk aan [^] binnen vierkante haakjes!)
- **karakter\$** : anchor, selecteert karakter/string aan einde van string
- **[]** : matcht elk karakter binnen de vierkante haakjes
- **[^ ]** : sluit karakter(s) of range na negatie uit  
(OPM: voor strings moet je een iets complexere handeling uitvoeren)
- **[ - ]** : matcht een range van karakters.
- **|** : OR-operator
- **\** : escaped het volgende karakter en interpreteert het als een literal in plaats van een regular expression operator

# Regular expressions

- **regular expressions operators**

## Literals & Metakaracters

- **Karakters of strings:** selecteert de karakters of strings  
(**OPM:** case sensitive!)
- **.** : matcht elk karakter (cijfers, letters, symbolen, spaties)
- **?** : matcht het voorgaande karakter 0 of 1 keer.
- **\*** : matcht het voorgaande karakter 0 of meer keer
- **+** : matcht het voorgaande karakter 1 of meer keer
- **{n}** : matcht het voorgaande karakter n keer
- **{n,m}** : matcht het voorgaande karakter minstens n keer en maximum m keer

# Regular expressions

- **regular expressions operators**

## Literals & Metakaracters

- **\d** : matcht alle cijfers tussen 0 en 9
- **\D** : matcht alle karakters BEHALVE 0 – 9
- **\s** : matcht een spatie
- **\S** : matcht alles behalve een spatie
- **\w** : matcht alle woord karakters. Equivalent van [a-zA-Z\_0-9]
- **\W** : matcht alle niet-woord karakters. Equivalent van [^a-zA-Z\_0-9]



# Regular expressions

- **Testen?**
  - Sublime Text / Notepad++
  - <http://regexpal.com/>
  - ...
  - Opgelet, deze kunnen verschillende resultaten opleveren, ondanks gelijke regexen. Om zeker te zijn: testen in PHP!

Opdracht: [opdracht-regular-expressions](#)

# mod\_rewrite

- Mod\_rewrite is een module van apache die toelaat URLs te herschrijven.
  - Bv: <http://www.dekrant.be/artikels.php?jaar=2012> omzetten naar:
  - <http://www.dekrant.be/artikels/2012/>
- Wordt gebruikt om gebruiksvriendelijke URLs te maken (+ ook goed voor SEO omdat GET-variabelen soms niet in acht worden genomen)

# mod\_rewrite

- Enkel mogelijk als mod\_rewrite is ingeschakeld op de server (niet altijd het geval!)
  - Hoe controleren?

**phpinfo();**

Kijken naar: **loaded modules.**

Staat mod\_rewrite hier bij, dan is mod\_rewrite ingeschakeld

# mod\_rewrite

## Configuration

### apache2handler

Apache Version	Apache/2.2.21 (Win32) mod_ssl/2.2.21 OpenSSL/1.0.0e PHP/5.3.8 mod_perl/2.0.4 Perl/5.10.1
Apache API Version	20051115
Server Administrator	postmaster@localhost
Hostname:Port	localhost:80
Max Requests	Per Child: 0 - Keep Alive: on - Max Per Connection: 100
Timeouts	Connection: 300 - Keep-Alive: 5
Virtual Server	No
Server Root	C:/xampp/apache
Loaded Modules	core mod_win32 mpm_winnt http_core mod_so mod_actions mod_alias mod_asis mod_auth_basic mod_auth_digest mod_authn_default mod_authn_file mod_authz_default mod_authz_groupfile mod_authz_host mod_authz_user mod_autoindex mod_cgi mod_dav_lock mod_dir mod_env mod_headers mod_include mod_info mod_isapi mod_log_config mod_mime mod_negotiation mod_proxy mod_proxy_ajp mod_rewrite mod_setenvif mod_ssl mod_status mod_php5 mod_perl

# mod\_rewrite

- Hoe gebruik maken van URL-rewriting?
- Eerst en vooral: logging inschakelen:
  - Inschakelen in het bestand **xampp\apache\conf\httpd.conf**

- Zoeken naar LogLevel, dit staat normaalgezien standaard op warn

```
279 #  
280 # LogLevel: Control the number of messages logged to the error_log.  
281 # Possible values include: debug, info, notice, warn, error, crit,  
282 # alert, emerg.  
283 #  
284 LogLevel warn
```

- Wijzigen naar: **LogLevel error rewrite:trace3**
  - Dit zorgt ervoor dat eventuele fouten gelogd zullen worden in de xampp\apache\logs\error.log file
    - Vergemakkelijkt debuggen
- De mod\_rewrite module maakt gebruik van de **regular expressions** engine om URLs te herschrijven

# mod\_rewrite

- .htaccess bestand
  - Dit bestand **staat in voor het 'herschrijven'** van de url.
  - Dit bestand heeft **géén bestandsnaam**, enkel een extensie.
  - Komt **in de map met bestanden** waarop de RewriteRule op toegepast moet worden.
  - Inhoud van bestand:
    - RewriteEngine On
    - RewriteRule ^redirect.php\$ original.php
    - Linkse deel (tussen ^ \$) is de 'alias', rechtse deel is de bron.

(vb. [voorbeeld-mod-rewrite-redirect](#) )

# mod\_rewrite

- Get-variabele herschrijven via RewriteRule (.htaccess):
  - RewriteEngine On  
RewriteRule ^artikels/([^\./]+)/?\$ artikels.php?jaar=\$1
    - **^artikels/** dient om te kijken of de pagina-url begint met de basisurl + de map
    - **([^\./]+)** Dit is een group. Tussen de haakjes alle tekens die niet mogen voorkomen in de naam (=pregmatch)
    - **/?\$** controleert of er enkel een trailing slash op het einde staat. Zoniet wordt de rewriterule genegeerd. (\$ duidt het einde van de alias aan)
    - \$1 is een variabele die automatisch wordt ingevuld in het pregmatch groepgedeelte van de alias.
      - » Er zijn dus meerdere groepen en aliassen mogelijk.
      - » De groepen worden chronologisch genummerd, ()/() => \$1/\$2

(vb. [voorbeeld-mod-rewrite-get](#) )

- Opdracht: [opdracht-mod-rewrite-basis](#)
- Opdracht: [opdracht-mod-rewrite-blog](#)
- Opdracht: [opdracht-mod-rewrite-single-point-of-entry](#)

# cron jobs

- Cron jobs zijn bestanden die je periodiek laat uitvoeren
  - Bv. Het verwijderen van tijdelijke foto's
  - Het scrapen van bepaalde websites
- Instellen in het C-panel van de hosting



# cron jobs

Valid cron time values are the numbers indicated and \*.

You can specify exact times using commas to separate them. eg: 1,2,3 (minutes 1,2 and 3)

You can specify spans using a dash. eg: 5-7 (minutes 5 to 7)

You can specify intervals using a star and a forward slash. eg: \*/2 (every 2nd minute)

You can combine them to create a more precise schedule. eg: 1,5,11-15,30-59/2 (minutes 1, 5, 11 to 15 and every 2nd minute between 30 and 59)

Note that there are no spaces

Current Time: Wed Aug 13 10:36:32 2014

## Create a New Cron Job

Minute	<input type="text" value="*"/>	0-59
Hour	<input type="text" value="*"/>	0-23
Day of Month	<input type="text" value="*"/>	1-31
Month	<input type="text" value="*"/>	1-12
Day of Week	<input type="text" value="*"/>	0-7 (0 or 7 = Sunday)
Command	<input type="text" value="/home/pascaje3/"/>	<input type="button" value="Prevent Email"/>

Add

Advanced Search

Minute	Hour	Day of Month	Month	Day of Week	Command	Select
*/10	*	*	*	*	/usr/local/bin/php /home/pascaje3/domains/pascalculator.be/public_html/btc/btc-monitor.php	<input type="checkbox"/>

Edit

Delete

## Send all Cron output to E-Mail

E-Mail

Save

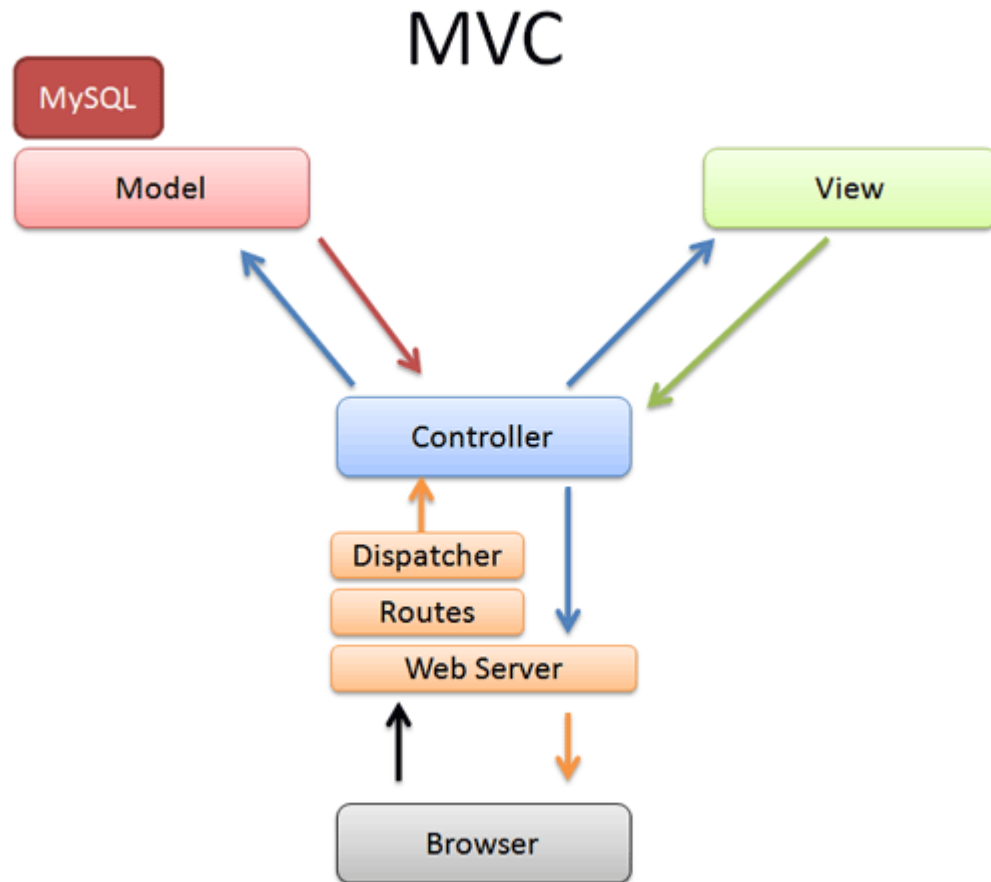
Valid values are an E-Mail address, your DA username: *pascaje3*, or a blank field to prevent any email.

# MVC-model

- MVC-model
  - Zorgt voor een onderverdeling tussen:
    - **Model**: Het uitvoeren van de bewerkingen op databaseniveau
    - **View**: Wat de gebruiker te zien krijgt
    - **Controller**: De command die de gebruiker uitvoert en wat (welke models) er moet aangesproken worden om een resultaat te genereren.

# MVC-model

- MVC-model



# MVC-model

## - MVC-model

- Dient om grote applicaties overzichtelijk te houden
- Makkelijk aanpasbare/onderhoudbare code
- Maakt gebruik van de modulaire opzet van OOP waardoor meerdere mensen makkelijker aan verschillende projecten binnen het MVC-framework kunnen werken zonder dat deze met elkaar in conflict komen.

# MVC-model

- MVC-model
  - Enkele voorbeelden van veelgebruikte 'frameworks' die steunen op het MVC model
    - [Zend](#)
    - [CakePHP](#)
    - [CodeIgniter](#)
    - [Symphony](#)
    - [Laravel](#)
    - ([Drupal](#), [WordPress](#), [Joomla](#), ...)

# MVC-model

- Tiny MVC: stap per stap
  - Stap 1
    - .htaccess in rootmap
      - Leidt alle verkeer om naar public map
    - In public map:
      - .htaccess
        - Zet alle gerequeste url om naar get-variabele (= hook)
      - index.php
        - Staat in voor het inladen van de applicatie
        - Interpreteert de hook
        - Laadt de nodige system/applicatie-classes in
  - (vb. [voorbeeld-mvc-step-by-step-01](#) )

# MVC-model

- Tiny MVC: stap per stap
  - Stap 2
    - System-map
      - Bevat alle internals van de applicatie
      - Application
        - Hier zit alle code specifiek aan de applicatie die gebouwd wordt (= eigen geschreven code!)
        - Code uit deze map extends meestal code uit de overige framework-mappen
      - Alle andere mappen: framework-specifieke code (M, V, C, libraries, helpers, modules, ...)
  - (vb. [voorbeeld-mvc-step-by-step-02](#) )

# MVC-model

- Tiny MVC: stap per stap
  - Stap 3
    - Eigen applicatie in application-map
      - Basis: models, views en controllers
      - Extenden altijd de base-model/view/controller klasse
      - Hier staat alle code die specifiek is aan de applicatie
    - (vb. [voorbeeld-mvc-step-by-step-03](#) )

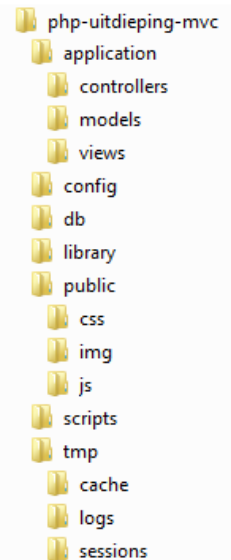


# MVC-model uitdieping

## - MVC-model

### - Mappenstructuur van een MVC-framework

- **application:** hier komen alle onderdelen van de website (bv. Registratie, bieren (overzicht, toevoegen, verwijderen, aanpassen, ...))
  - controllers
  - models
  - views
- **config:** allerlei configuratie (dbname, dbpassword,...)
- **db:** database backup
- **library:** bibliotheek met allerlei functies (bv genereer paswoord-functie, ...)
- **public:** gemeenschappelijke bestanden die door alle pagina's gebruikt worden (css, js, images, ...)
- **scripts:** command-line scripts (bv. batch-processing voor het verkleinen van images, ...)
- **tmp:** alle tijdelijke bestanden (errorlogs, versluisfolder voor het aanpassen van afbeeldingen, ...)



# MVC-model uitdieping

## - MVC-model

### - MVC conventies

- **Sql-tabel-namen** altijd in het meervoud en in **lowercase** (bv. users)
- **Models** zijn **altijd enkelvoudig**, de eerste letter van het woord is een hoofdletter (bv. User)
- Controllers hebben altijd het woord **Controller** achteraan de naam staan (bv. UsersController)
- **Views** staan altijd in een **map** die elke actie in een aparte pagina bevat (bv. users/viewall.php, users/delete.php)

# MVC-model uitdieping

## - MVC-model

### - MVC-verhaal:

- (vb. **voorbeeld-mvc** -> zet deze in xampp/htdocs om .htaccess nachtmerries te vermijden of maak aparte virtual host aan)
- htaccess
  - /.htaccess (in de rootmap):
    - Zorgt voor een automatische redirect naar de public folder
  - public/.htaccess in de public map
    - Zorgt voor een 'single-point-of-entry', alles moet via index.php passeren.
    - Zet mappenstructuur om naar get-variabele die de index.php zal interpreteren

# MVC-model uitdieping

- MVC-model
  - MVC-verhaal:
    - public/index.php
      - Enkele constante bepalen (rootmap, folder-delimiter)
      - De url interpreteren
      - Bootstrap aanspreken

# MVC-model uitdieping

- MVC-model

- MVC-verhaal:

- library/bootstrap.php

- 'bootstrap' = bundelaar.
      - Bundelt alle bestanden die we nodig hebben automatisch samen zodat we deze kunnen aanspreken.
      - Spreekt de config/config.php aan
      - Spreekt library/shared.php aan

# MVC-model uitdieping

- MVC-model
  - MVC-verhaal:
    - library/shared.php
      - Bevat voor bepaalde methodes die automatisch moeten uitgevoerd worden bij elke request
        - **reporting** voor developmentmode/livemode
        - **hook** zorgt voor het omzetten van de url naar een bewerkbare versie die gebruik wordt voor het aanroepen van de juiste klassen
        - **Autoload** functie zorgt voor het automatisch implementeren van de juiste klassen in onze application map

# MVC-model uitdieping

- MVC-model
  - MVC-verhaal:
    - library/shared.php
      - hook: users/view/1/username
        - Controller: users
        - Model: users
        - View: view
        - Action: view
        - Query: array(1, username)

# MVC-model uitdieping

- MVC-model

- MVC-verhaal:

- library/controller.class.php
      - Zorgt voor het aanspreken van het juiste model, view en controller in onze library-map.
    - library/model.class.php
      - Zorgt voor het aanspreken van eventuele uitbreidingen op de algemene model-class (bv, sql query-library)
    - library/sqlquery.class.php
      - Een library die het makkelijk maakt om sql-queries uit te voeren
    - library/template.class.php
      - Importeert de 'views' van de aangesproken applicatie



# MVC-model uitdieping

- MVC-model
  - MVC-verhaal:
    - config/config.php
      - Hier staan alle configuratie-settings (developmentmode, dbserver, dblogin, dbpassword, dbnaam, ...)

# MVC-model uitdieping

- MVC-model
  - MVC-verhaal:
    - application/model/user.php
      - Deze klasse extend de basisklasse Model
      - Hier komen alle extra bewerkingen die uitgevoerd moeten worden op databaseniveau (bv. linken an tracking details, ...)

# MVC-model uitdieping

- MVC-model

- MVC-verhaal:

- application/controller/usersController.php

- Deze klasse extend de basisklasse Controller
      - Deze klasse achterhaalt welke actie er werd aangeroepen en spreekt de methodes aan die uitgevoerd moeten worden bij het aanroepen van deze actie (bv. **public function** add() {})

# MVC-model uitdieping

- MVC-model

- MVC-verhaal:

- application/views/users/
      - De map heeft de naam van de applicatie
      - header.php/footer.php
        - Dit zijn de header en footer die gebruikt moeten worden bij het aanspreken van elke actie. Als er geen header of footer in deze map staat zal de wordt header en/of footer die in de map application/view/ staat, gebruikt
      - view.php, viewall.php bevat de html die gebruikt moet worden wanneer deze acties worden aangesproken.

# MVC-model uitdieping

- MVC-model
  - MVC-verhaal:
    - Afgewerkte applicatie:
    - <http://localhost/voorbeeld-mvc-tiny-mvc/users/viewall/>

Opdracht: [opdracht-MVC-tutorial](#)

# MVC-framework: CodeIgniter

- CodeIgniter MVC-framework.
  - Een van de compactere frameworks (+/- 2Mb)
    - Nadeel: niet alles aanwezig/extra modules nodig.
  - Open source
  - Grote community
  - Goed gedocumenteerd
  - Basis voor ExpressionEngine CMS (ook van EllisLab)
  - URL: <http://ellislab.com/codeigniter>

# MVC-framework: CodeIgniter

- CI-features
  - Full Featured database classes with support for several platforms.
  - Active Record Database Support
  - Form and Data Validation
  - Security and XSS Filtering
  - Session Management
  - Email Sending Class. Supports Attachments, HTML/Text email, multiple protocols (sendmail, SMTP, and Mail) and more.
  - Image Manipulation Library (cropping, resizing, rotating, etc.). Supports GD, ImageMagick, and NetPBM
  - File Uploading Class
  - FTP Class
  - Localization
  - Pagination
  - Data Encryption

# MVC-framework: CodeIgniter

## - CI-features

- Benchmarking
- Full Page Caching
- Error Logging
- Application Profiling
- Calendaring Class
- User Agent Class
- Zip Encoding Class
- Template Engine Class
- Trackback Class
- XML-RPC Library
- Unit Testing Class
- Search-engine Friendly URLs
- Flexible URI Routing
- Support for Hooks and Class Extensions
- Large library of "helper" functions

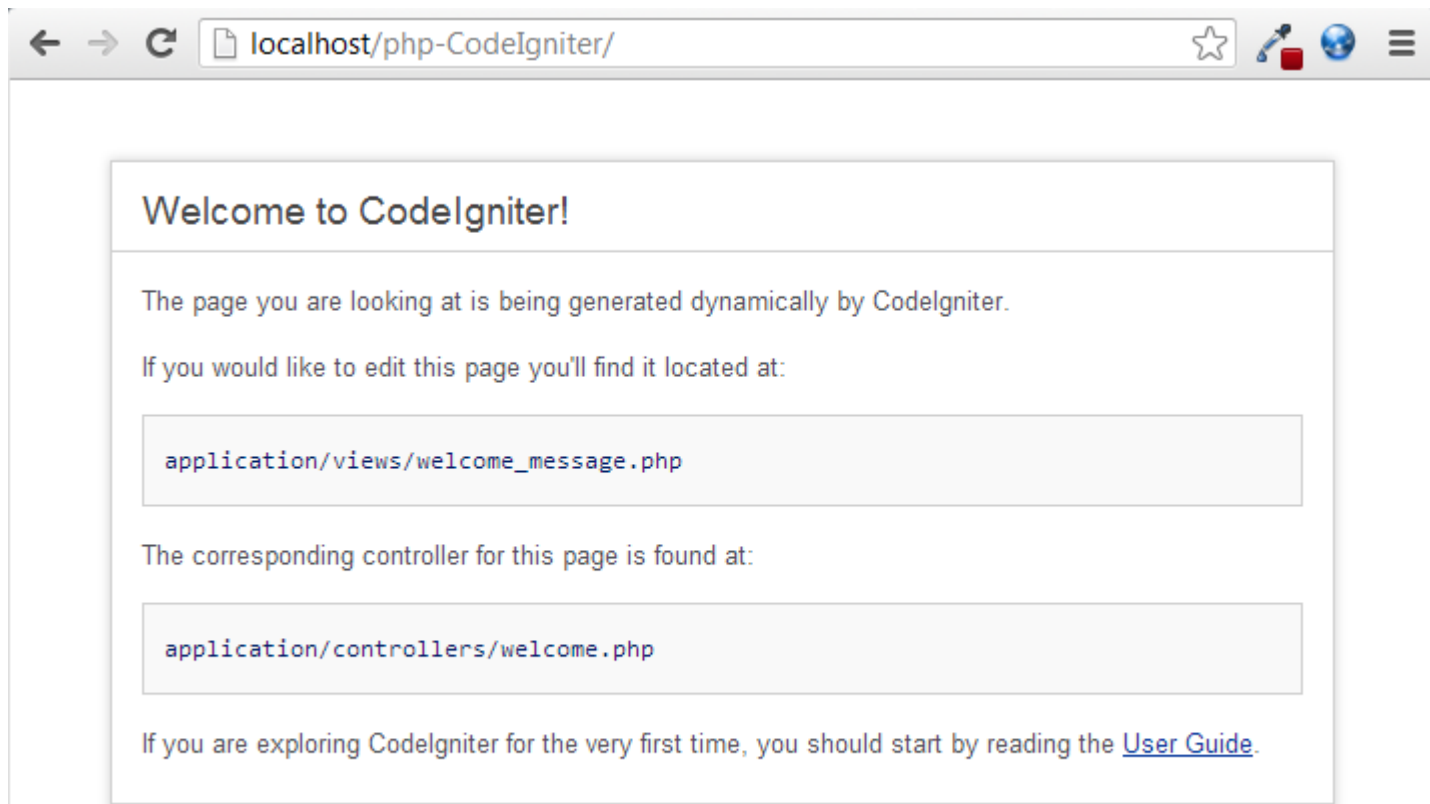


# MVC-framework: CodeIgniter

- Getting started
  - Download CodeIgniter op <http://ellislab.com/codeigniter>
  - Plaats alle files in een map (htdocs, ...) toegankelijk voor je Virtual Server (MAMP, XAMP, ...)

# MVC-framework: CodeIgniter

- Getting started
  - Surf naar deze lokale map



# MVC-framework: CodeIgniter

## - Getting started

- Als de bovenstaande pagina verschijnt is de installatie gelukt.
- Maak een verschil tussen development-omgeving en productie-omgeving
  - Development omgeving:
    - Meestal lokaal
    - Toont specifieke foutboodschappen ("PHP error on line ...")
    - Lokale base paths
    - Lokale database
  - Productie omgeving
    - Op de ftp (live website)
    - Toont algemene foutboodschappen ("Er ging iets mis")
    - Absolute base paths (eigenlijke url van website)
    - Live database (password protected)
- Vooralleer de applicatie te uploaden, moet je deze bepaalde development-eigenschappen wijzigen naar productie-eigenschappen.

# MVC-framework: CodeIgniter

- Getting started
  - Doorneem de User Guide
    - Te vinden in de map user\_guide
      - **Opmerking:** deze map (en license.txt) verwijder je best wanneer je website in productie gaat. Deze dienen enkel als uitleg over het CI-framework en hebben geen applicatie-functie.
  - Of, te vinden op:
    - <http://ellislab.com/codeigniter/user-guide/>

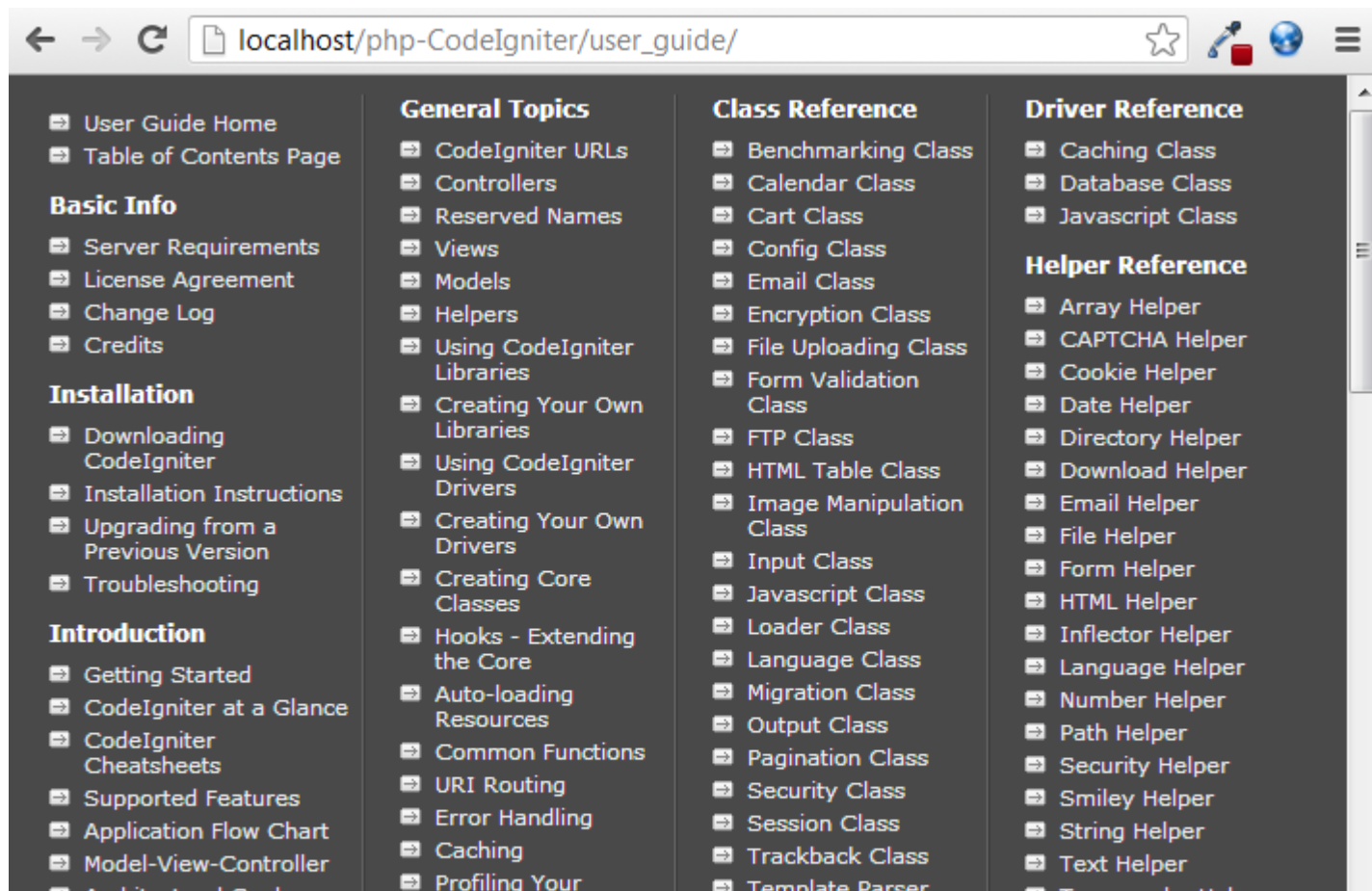
# MVC-framework: CodeIgniter

- Getting started
  - Table of contents: overzicht van alle CI-functies



# MVC-framework: CodeIgniter

## - Getting started



# MVC-framework: CodeIgniter

- Getting started
  - Begin bij Installation

<http://ellislab.com/codeigniter/user-guide/installation/index.html>

- **application/config/config.php**
  - `$config['base_url']` aanpassen

```
16  */  
17  $config['base_url'] = 'http://localhost/php-CodeIgniter/';  
18
```

**OPM:** deze URL hangt af van de map waar jij de CI-bestanden hebt geplaatst.

# MVC-framework: CodeIgniter

- Getting started

- Begin bij Installation

- **application/config/database.php**

- \$db variabelen aanpassen naar databasegegevens van eigen sql-server.

```
51 $db['default']['hostname'] = 'localhost';  
52 $db['default']['username'] = 'root';  
53 $db['default']['password'] = 'root';  
54 $db['default']['database'] = 'php_CodeIgniter';
```

**OPM:** hostname, username, password en database values zijn voor iedereen anders.



# MVC-framework: CodeIgniter

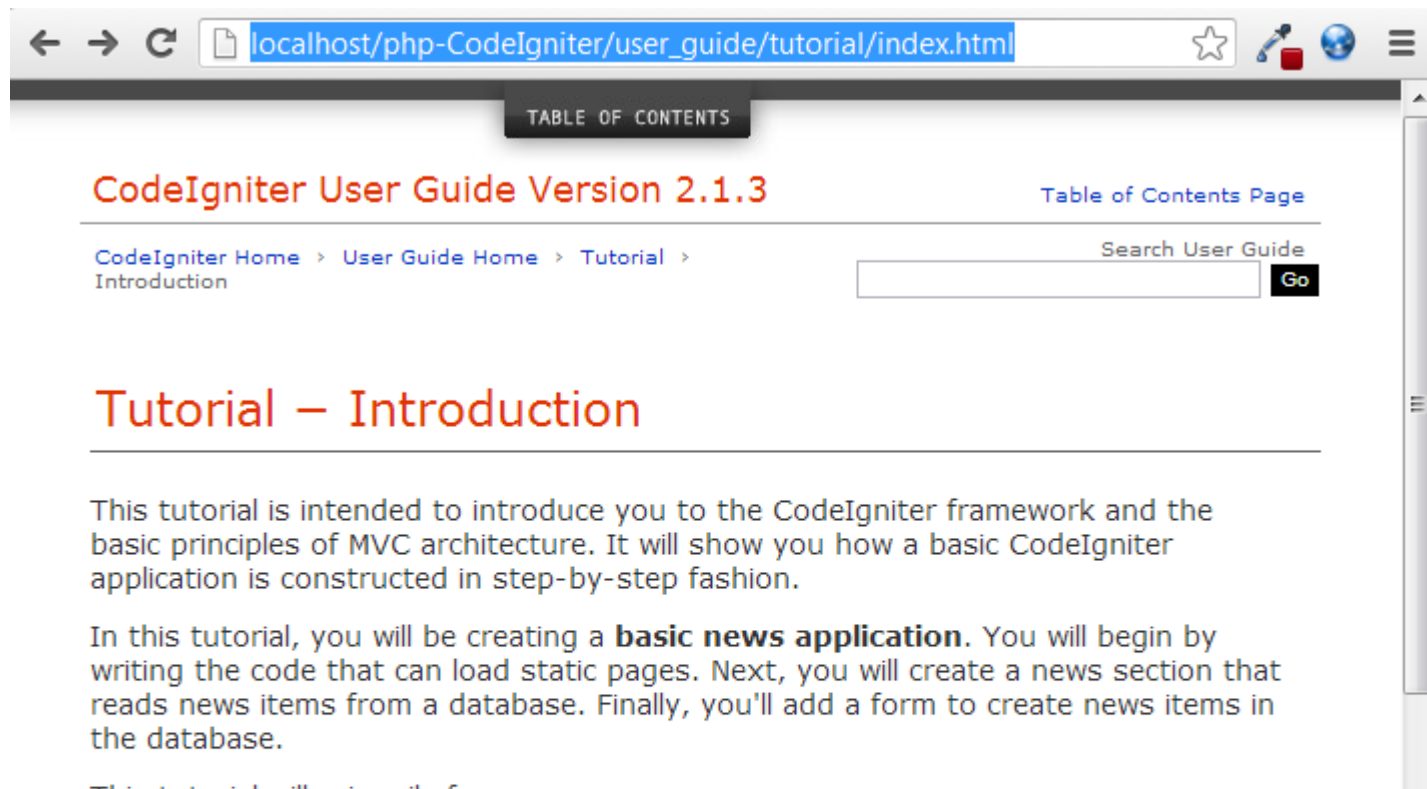
- Getting started
  - Begin bij Installation
    - **index.php**
      - Om te kunnen debuggen moet de constante ENVIRONMENT de value development hebben.

```
20  */
21  define('ENVIRONMENT', 'development');
22  /*
23  *
```

OPM: dit is enkel voor ontwikkelingsomgevingen. Zou een groot security-lek opleveren wanneer dit in productie wordt gebruikt.

# MVC-framework: CodeIgniter

- Getting started
- Tutorial



# MVC-framework: CodeIgniter

- Getting started
  - Opdracht [opdracht-CodeIgniter](#)

# MVC-framework: Laravel

- <http://www.laravel.com>



MENU

THE PHP FRAMEWORK  
FOR WEB ARTISANS.

PHP THAT DOESN'T HURT.  
CODE HAPPY & ENJOY THE FRESH AIR.

# MVC-framework: Laravel

- Voor mensen die problemen ondervinden bij de installatie van een Laravel-project
  - “Mcrypt PHP extension required”
    - <http://andyy.me/mcrypt-php-extension-for-laravel-in-xampp-mac/>

# design patterns

- Design Patterns
  - In het leven geroepen/voor het eerst beschreven door the Gang of Four: Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides in het boek ***Design Patterns: Elements of Reusable Object-Oriented Software***
  - Design patterns zijn bepaalde codeblokken om bepaalde problemen die vaak voorkomen bij OOP op te lossen of om performantie te verhogen.

# design pattern: singleton

- Design Patterns: Singleton
  - Probleem bij databaseconnectie binnen Class  
( vb. [voorbeeld-design-patterns-singleton](#) )

## OPLOSSING: Singleton

- Limiteert het aanmaken van een object.
- Van zodra het object is aangemaakt, wordt de instantie (aangemaakte versie) gebruikt.

# design patterns: singleton

- Design Patterns: Singleton

- *class Database {*  
    **private static** \$connectionInstance;  
  
    **private function** \_\_construct() {  
        connectie met database  
        self::\$connectionInstance = connectie met database  
    }  
  
    **public function** getConnection() {  
        if (!isset(self::\$connectionInstance)) {  
            new Database;  
        }  
        return self::\$connectionInstance;  
    }  
  
    **public function** databaseQuery(\$query) {  
        uitvoeren van de query  
    }  
    *}*



# design patterns: singleton

- Design Patterns: Singleton

- Gebruik:

```
require_once 'Database.class.php';
```

```
Database::getConnection();
```

```
Database::databaseQuery('SELECT * ...');
```

(vb. [voorbeeld-design-patterns-signleton](#))

- Opdracht: [p---](#)

# design patterns: factory

- Design Patterns: Factory
  - Een factory is een klasse die het resultaat uit een andere klasse gaat halen.
    - Dit gebeurt op basis van een argument dat aan de factory klasse wordt opgeworpen.
      - Door middel van een switch wordt aan de hand van het argument bepaald welke klasse de Factory op zijn beurt moet oproepen.

# design patterns: factory

## - Design Patterns: Factory

- De werking heeft drie luiken:
  - De pagina waar het resultaat wordt getoond
  - De factory klassepagina
  - De klassepagina's die door de factory klassepagina worden aangesproken om een bepaald resultaat te reproduceren.
- Wordt gebruikt voor elementen die erg op elkaar lijken en slechts minieme verschillen vertonen.
  - Bv. inputvelden
- (vb. [voorbeeld-design-patterns-factory.php](#))

# design patterns: factory

- Design Patterns: Factory

- Syntax: (bv. TextFactory.class.php)

```
class TextFactory {  
    public static function getResultForType($type, $text) {  
        switch ($type) {  
            case 'Type01':  
                require_once 'Type01.class.php';  
                $result = Type01::getResult($text);  
            case 'Type02':  
                require_once 'Type01.class.php';  
                $result = Type01::getResult($text);  
        }  
        return $result;  
    }  
}
```

# design patterns: factory

- Design Patterns: Factory

- Syntax: (bv. Type01.class.php)

```
class Type01 {  
    public static function getResult($text) {  
        return strtoupper($text);  
    }  
}
```

# design patterns: factory

- Design Patterns: Factory

- Syntax: (bv. index.php)

```
require_once 'TextFactory.class.php';
```

```
echo TextFactory::$getResultForType('Type01', 'dit is  
een string in lowercase' );
```

- Opdracht: ---

# design patterns: varia

- Design Patterns: Observer
- Design Patterns: Chain-of-command
- Design Patterns: Strategy