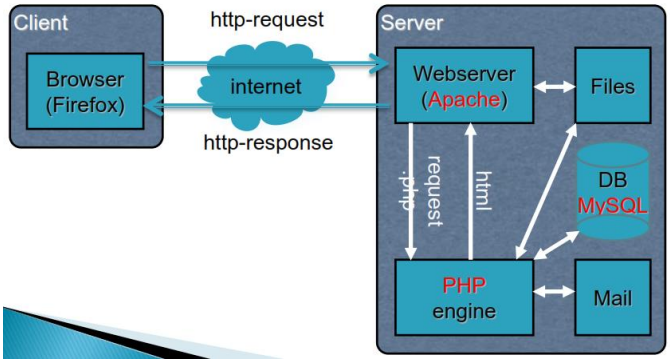
PHP

Inleiding



2 soorten webapplicaties

- **Statische webapplicatie**: na aanvraag van de client, stuurt de server eenvoudige HTML-bestanden
- **Dynamische webapplicatie**: de server heeft meer mogelijkheden, bv. een PHP engige/parser, MySQL server, mail server, etc.

Server-side vs. client-side scripting

- **Server-side scripting**: de code wordt op de server uitgevoerd (bv. PHP)
- **Client-side scripting**: de code wordt op de client uitgevoerd (bv. JavaScript)

PHP

- Oorspronkelijk 'Personal Home Page', nu **Hypertext Preprocessor**
- Server-side scriptingtaal
- Open source, cross-platform, snel, stabiel, eenvoudig te leren en heel populair
- Dynamische webpagina's ontwikkelen in combinatie met HTML
- Bestanden hebben .php-extensie

Apache

- Open source, cross-platform webserver
- Professioneel en robuust, met uitgebreide features
- Ondersteunt PHP

MySQL

- Open source, cross-platform **RDBMS** (Relational Database Management System)
- Gratis, snel, stabiel en populair
- Vaak gebruikt in combinatie met PHP
- Kan beheerd worden via phpMyAdmin

Webserverpakketten

- **WAMP**: Windows, Apache, MySQL, PHP
- **WIMP**: Windows, IIS, MySQL, PHP
- **MAMP**: Mac OSX, Apache, MySQL, PHP
- **XAMP**: Een OS, Apache, MySQL, PHP

Opbouw

- Heeft .php-extensie
- PHP-code kan tussen HTML-code staan, maar enkel PHP-code is ook mogelijk!
- Staat tussen de `<?php` en `?>` tags
- Alternatief tussen `<?` en `?>`, maar kan conflicten geven met tags van andere talen

Uitvoering

- Worden van boven naar onder uitgevoerd
- **thread**: de pointer naar de regel in het PHP bestand die op dat moment wordt geëvalueerd
  - → begint bij het toekomen van een aanvraag voor een webpagina
  - → eindigt na de afhandeling (bij het terugsturen van het antwoord)

Basissyntax

- analoog met Java / C#
- elke coderegel eindigt met een puntkomma
- de hoeveelheid witruimte tussen woorden maakt niet uit
- alles is hoofdlettergevoelig, behalve functienamen

Commentaar

- `// code` voor enkele regels
- `# code` voor enkele regels (best niet gebruiken)
- `/* code */` voor meerdere regels

Variabelen

- voorafgegaan aan `$`
- toewijzing door = operator
- mogelijk om te declareren voor ze gebruikt worden
- hebben geen type, wordt automatisch bepaald a.d.h.v. de waarde
- hebben een standaardwaarde
- eerste letter van de naam moet een letter zijn, daarna mogen letters, cijfers en underscores

Scope

als een variabele `$a` gedefinieerd wordt buiten de scope van een bepaalde functie of klasse, kan deze beschikbaar gesteld worden door `global $a`.

- het keyword `global` definieert geen globale variabele, maar maakt een bestaande variabele beschikbaar in de huidige scope
- de globale variabele blijft beschikbaar
  - tijdens de volledige uitvoering van de thread
  - in andere PHP-bestanden

als een variabele $a gedefinieerd wordt binnen de scope van een bepaalde functie, is deze enkel beschikbaar binnen deze functie

Datatypes

Eenvoudige typen

- integers (gehele getallen)
- doubles (floating-point of decimale getallen)
- booleans (kent twee waarden: true of false)
- NULL (heeft slechts één waarde: null)
- strings (tekenreeksen)
  - aangeduid met enkele of dubbele aanhalingstekens
  - concatenatie van strings gebeurt met een punt, NIET met een plus
  - alles wat tussen dubbele aanhalingstekens staat wordt geëvalueerd door de parser

```
$woord = "vakantie";
echo 'Ik wil $woord'; => Ik wil $woord
echo “Ik wil $woord”; => Ik wil vakantie
```

Samengestelde typen

- arrays (benoemde of geïndexeerde verzameling van andere waarden)
  - geindexeerde array: gebruikt index (zero-based integer) om een item te identificeren

```
$array[0] = "element 1";
$array = array("element 1", "element2");
```

  - associatieve array: gebruikt een case-sensitive string om item te identificeren

```
$array1["key1"] = "Element 1";
$array2 = array("key1" => "Element 1", "key2" => "Element 2");
```

- objecten (instanties van een bepaalde klasse)
- resources (speciale variabelen die een verwijzing bevatten naar externe bronnen (buiten PHP) zoals een databaseverbinding bijvoorbeeld)

Constanten

= variabelen die eenmalig worden geïnitialiseerd

- declareren met commando `define(naam, waarde)` (geen dollarteken)
- wordt geschreven in HOOFDLETTERS

Functies

- moeten éénmaal ergens in het script gedeclareerd worden
- mogen aangeroepen worden voor de thread de definitie is tegengekomen
- functie wordt aangeroepen met te weinig argumenten
  - → ontbrekende argumenten worden behandeld als ongedefinieerde variabelen
  - → standaardwaarden kunnen aangegeven worden in de functiedeclaratie
- functie wordt aangeroepen met te veel argumenten
  - → extra argumenten worden genegeerd

```
function functienaam($argument1 = standaardwaarde, $argument2, ...) {
      // Invulling
}
```

Controlestructuren

zie CPII en CPIII, echt niks nieuws

Basisfuncties

echo en print

= uitvoer van tekst naar het PHP-script (de uiteindelijke HTML-pagina)

- wordt zo veel gebruikt dat hij zonder haakjes geschreven mag worden:
  ```
  <? php echo("Een string afdrukken”); ?>
  <? php echo "Een string afdrukken”; ?>
  ```
- alternatief is `print`, geeft 1 als de output gelukt is en 0 als het niet gelukt is

phpinfo()

geeft nuttige informatie over de status en configuratie van de PHP-installatie

isset()

geeft een boolean terug die aangeeft of er reeds een waarde werd toegekend aan een variabele, en of die waarde niet NULL is

empty()

geeft een boolean terug die aangeeft of een variabele wordt beschouwd als leeg.

- De variabele bestaat niet
- De variabele is gelijk aan:
  - `0 / 0,0 / “0”`
  - `NULL`
  - `false`
  - een lege string
  - een lege array

include

een ander bestand aanroepen

- wanneer het bestand niet gevonden wordt krijg je een warning
- wordt enkel uitgevoerd wanneer de thread er voorbij komt
- met `include_once` zal PHP ervoor zorgen dat een bestand maximum één keer ge-included kan worden, om bijvoorbeeld dubbele functiedefinities te voorkomen

require

een ander bestand aanroepen

- wanneer het bestand niet gevonden wordt krijg je een error
- wordt altijd uitgevoerd
- met `require_once` zal PHP ervoor zorgen dat een bestand maximum één keer ge-required kan worden, om bijvoorbeeld dubbele functiedefinities te voorkomen

die()

de uitvoering van je PHP-script te stoppen voor een bepaalde thread

header()

geeft de mogelijkheid om de http-headers aan te passen

- moet gebeuren voor er iets ge-output wordt (HTML/echo’s/lege lijnen/…), anders werkt het niet!
- redirecten naar een andere pagina gaat door de location-header aan te passen

```
header(“location:http://www.google.be”);
```

Tijd en datum

time()
geeft de tijd *(timestamp)* in UNIX-formaat (aantal seconden sinds 1 januari 1970 om 00:00)

date()
geeft een gegeven tijdstip geformatteerd volgens een format string en optioneel een timestamp

- meer informatie i.v.m. de format string op http://php.net/manual/en/function.date.php

mktime(uur, minuut, seconde, maand, dag, jaar)
geeft de tijd in UNIX-formaat a.d.h.v. bepaalde parameters

Arrays

array_key_exists(*key*, *array*)
gaat na of de key bestaat in de array

array_keys(*array*)
geeft een lijst terug van alle keys

array_merge(*array1*, *array2*, …)
voegt arrays samen

count(*array*)
telt alle elementen in een array

sort(*array*)
sorteert een array, items kunnen nieuwe index toegewezen krijgen

asort(*array*)
sorteert een array, items behouden hun index (belangrijk bij bijvoorbeeld associatieve array)

shuffle(*array*)
schudt de volgorde van de items in een array door elkaar

Stringfuncties

`int` **`strpos`**`(`*`string1, string2`*`)`
geeft de numerische positie van het eerste voorkomen van string 2 in string 1 terug

`string` **`strtolower`**`(`*`string`*`)`
zet de string om naar kleine letters

`string` **`strtoupper`**`(`*`string`*`)`
zet de string om naar hoofdletters

`int` **`strlen`**`(`*`string`*`)`
geeft de lengte van de string terug

`string` **`wordwrap`**`(input, aantal kol, nextlinechar)`
zet string om naar verschillende lijnen

`bool` **`strcmp`**`(`*`string1, string2`*`)`
returnt true indien zelfde string

`bool` **`strncmp`**`(`*`string1, string2, n`*`)`
returnt true indien zelfde eerste n karakters

`string` **`ucfirst`**`(`*`string`*`)`
het eerste karakter van de string wordt een hoofdletter

`string` **`ucwords`**`(`*`string`*`)`
het eerste karakter van elk woord in de string wordt met een hoofdletter geschreven

`string` **`substr`**`(string, startpos, lengte)`
geeft een deel van string terug, afgeleid door een startpositie en lengte

`string` **`trim`**`(string)`
geeft de inputstring terug, met alle whitespace aan het begin en einde ervan verwijderd

→ variaties zijn `ltrim` en `rtrim`

`string` **`addslashes`**`(string)`
zet tekens om naar hun escaped equivalent wanneer nodig

`string` **`stripslashes`**`(string)`
maakt addslashes ongedaan

`string` **`htmlspecialchars`**`(string)`
karakters die in html een speciale betekenis hebben vervangen door hun equivalent (bv. & wordt &)

`$var[5]`
haalt een bepaald karakter van een string op

Formulieren

= informatie sturen van de client naar de server, en deze op de server verwerken d.m.v. PHP

HTML-inputvelden

zie SWD

```
<form action="verwerkpagina.php" method="post" enctype="multipart/form-data">
  …Een aantal formuliervelden…
  <input type="submit" value="Verwerken"/>
</form>
```

Uploaden van bestanden

- d.m.v. invoerveld met type file
- details zijn beschikbaar in de superglobal $_FILES
  - **bestandsnaam**: $_FILES["bestand"]["name"];
  - **bestandstype**: $_FILES["bestand"]["type"];
    = mime-type, bv. image/jpeg
  - **bestandsgrootte** in bytes: $_FILES["bestand"]["size"];
  - **tijdelijke bestandsnaam** op server: $_FILES["bestand"]["tmp_name"];
  - **eventuele fouten** bij het uploaden: $_FILES["bestand"]["error"];
- het bestand op de tijdelijke locatie verplaatsen naar een locatie op de harde schijf met `move_uploaded_file(filename, destination)`

Verzendmethodes

GET-request

- De gegevens worden naar de server gestuurd in een querystring
  - begint met ?
  - bevat een bepaalde hoeveelheid naam/waarde-paren, gescheiden door &
  - wordt toegevoegd aan de URI → zichtbaar in het adresveld van de browser
  - een URL kan een beperkt aantal tekens hebben, afhankelijk van de ontvangende webserver
  - URI = Uniform Resource Identifier (bv. http://www.google.be)
  - URL (Uniform Resource Locator) wordt als synoniem aanvaard, maar is niet volledig hetzelfde.
- In PHP zijn de naam/waardeparen beschikbaar in de superglobale, associatieve `$_GET` array
  - superglobale variabele: variabele die overal en binnen elke scope beschikbaar zijn

Voor- en nadelen

- Alle data is zichtbaar in de url:
  → niet voor vertrouwelijke gegevens
  → zichtbaar in de browsergeschiedenis
  → wordt mee opgeslagen met bookmarks
- Een querystring is beperkt in grootte
- Werkt niet met alle types gegevens

Wanneer gebruiken?

- velden van een zoekformulier
- taalkeuze

Wanneer niet gebruiken?

- pagina's die wijzigingen doorvoeren op de server (kan meerdere keren uitgevoerd worden bij opslagen in de bookmarks)

POST-request

- De gegevens worden naar de server gestuurd in de body van de http request
- In PHP zijn de naam/waardeparen beschikbaar in de superglobale, associatieve `$_POST` array

Voor- en nadelen

- Data wordt verstuurd in de request body:
  - → data is niet onmiddellijk zichtbaar
  - → veiliger, maar niet volledig veilig (encryptie!)
  - → doorgestuurde informatie wordt niet opgeslagen in de geschiedenis/bookmarks
- Mogelijkheid om veel meer gegevens door te sturen
  - → limiet is afhankelijk van de webserver

Wanneer gebruiken?

- registratieformulieren
- uploaden van bestanden
- pagina's die wijzigingen doorvoeren op de server

Naar waar sturen we het formulier?

- een apart PHP-bestand, waar de code die de gegevens verwerkt geisoleerd staat
  - → maakt de code gestructureerder, meer leesbaar en uitbreidbaar
- in het geval van formuliervalidatie sturen we het formulier naar dezelfde pagina
  zo kan de gebruiker blijven een ongeldig formulier indienen, en zijn/haar fouten corrigeren

Client/server-side validatie

	Voordelen	Nadelen
Client-side validatie	- geeft snelle feedback aan de gebruiker - zorg ervoor dat er geen HTTP requests worden gestuurd naar de server indien de ingevulde gegevens toch niet valid waren - geeft minder belasting aan de server	- kan omzeild worden - er bestaat een mogelijkheid dat er niet-valide gegevens worden doorgestuurd naar de server
Server-side validatie	- jij hebt volledige controle over de serveromgeving die gebruikt wordt voor de validatie - veel moeilijker voor een gebruiker om toch nog onjuiste gegevens te laten opslaan/verwerken door de server	- validatie neemt resources van de server in - elke keer de validatie mislukt, moet de pagina met het formulier opnieuw worden teruggestuurd naar de client => meer belasting voor de server
Oplossing:	een combinatie van client-side en server-side validatie	

Formuliervalidatie met regular expressions

= een manier om patronen in strings op te zoeken

- wordt gebruikt in verschillende programmeer- en scriptingtalen
- wordt geimplementeerd via een bibliotheek
- wij gebruiken de PCRE/Perl-stijl (wordt door de meeste programmeertalen ondersteund)

PHP-functies

preg_match`($patroon:String, $string:String):int`
- zoekt `$patroon` in `$string`
- returnwaarde is één van volgende
  - 1 wanneer `$patroon` voorkomt in `$string`
  - 0 wanneer `$patroon` niet voorkomt in `$string`
  - FALSE wanneer zich een fout voordeed

preg_match`($patroon:String, $teOnderzoeken:String, $gevonden:array, $vlaggen:int, $offset:int):int`
- `$gevonden`: zal worden gevuld met een array met de resultaten van de search
- `$vlaggen`:
  - Indien `PREG_OFFSET_CAPTURE` : elk element van `$gevonden` wordt op zich een array waarin de index van de match staat + de match zelf
  - Indien `0`: er verandert niets
- `$offset`: de plaats van de `$string` waar het zoeken begint.

preg_replace`($patroon, $vervanging, $string):uitvoer`
- zoekt `$patroon` in `$string` en vervangt de voorkomens met `$vervanging`
- werkt niet alleen met strings, maar ook met bv. arrays

preg_filter`($patroon, $vervanging, $string):uitvoer`
- zelfde als `preg_replace`, maar returnt enkel indien er een match is

preg_grep`($patroon:String, $array:array):array`
- geeft een array met de elementen van `$array` die aan het patroon voldoen

preg_match_all`($patroon:String, $string:String):int`
- geeft het aantal matches terug of `false` wanneer er zich een fout voordeed
- optioneel kan deze functie ook het resultaat teruggeven in een array die op verschillende manieren gesorteerd kan worden => bekijk de PHP manual voor alle mogelijkheden

preg_split`($patroon:String, $string:String):array`
- splitst `$string` op door middel van een patroon

preg_quote`($string:String):String`
- neemt `$string` en zet een backslash voor elk karakter dat onderdeel is van de regex-syntax
- handig als je een zoekstring hebt die bv. door een gebruiker wordt ingegeven en die speciale karakters kan bevatten

Syntax

- begin en einde wordt aangeduid met een /
- ^ duidt op het begin van de te onderzoeken string
- $ duidt op het einde van de te onderzoeken string
- speciale karakters moeten ge-escaped worden d.m.v. een backslash \

Karakterklasse

- omsloten door vierkante haakjes []
- bevat een reeks karakters waarop gezocht wordt
- bv. `/[0123456789]/` zoekt naar cijfers in een string

- ondersteunt een verkorte notatie voor reeksen van tekens, bv. `[a-f]`, `[0-5]`, `[a-z|0-9]`
- er is de mogelijkheid om te zoeken op alles behalve bepaalde karakters d.m.v. ^, bv. `[^012]`

Subpatronen

- omsloten door ronde haakjes ()
- dient om quantifiers, condities en modifiers toe te passen op een bepaald patroon
- bv. `/([a-z]{1,}\.)/` zoekt naar alle woorden die gevolgd worden door een punt

Quantifiers

- duidt het aantal voorkomens van een subpatroon aan
- bv. `{1,2}` = minimum 1 en maximum 2 voorkomens
  `{1,}` = minimum 1 voorkomen
  `{1}` = exact 1 voorkomen
- ondersteunt enkele verkorte notaties:
  `{0,}` = *
  `{1,}` = +
  `{0,1}` = ?

Modifiers

- een karakter dat na de sluitende backslash wordt toegevoegd en bepaalde parameters bepaald voor het hele patroon

i	Maakt het matchen case-insensitive
m	De onderzochte string mag uit meerdere lijnen bestaan
s	Zorgt ervoor dat . ook met newlines overeenkomt

Handig

http://regex101.com	een online tool om reguliere expressies te schrijven, testen en debuggen
http://regexr.com/	een online tool om reguliere expressies te schrijven, testen en debuggen, met referentie, cheatsheet, voorbeelden en collectie van bestaande reguliere expressies
http://rick.measham.id.au/paste/explain.pl	een online tool om een tekstuele uitleg te krijgen bij een gegeven reguliere expressie
http://regexlib.com/	een online collectie van bestaande reguliere expressies
https://itunes.apple.com/us/app/patterns-the-regex-app/id429449079?mt=12	een offline applicatie voor het schrijven en testen van reguliere expressies (Mac)
http://www.ultrapico.com/Expresso.htm	een offline applicatie voor het schrijven en testen van reguliere expressies (Windows)

Klassen

- aanmaken d.m.v. het sleutelwoord `class`
  ```
  class MijnKlasse { … }
  ```
- constanten in klasses declareren we d.m.v. het sleutelwoord `const`
  ```
  const PI = 3.14;
  ```
- best in aparte PHP-bestanden houden → één bestand per klasse

Constructor/deconstructor

- constructor definiëren d.m.v. de __construct functie
  ```
  function __construct($arg1) {
    $this->mijnVariabele1 = $arg1;
  }
  ```
- deconstructor (bij het verwijderen v.h. object) definiëren d.m.v. de __deconstruct functie
  ```
  function __deconstruct() {
    // implementatie
  }
  ```
- instantie verwijderen door de waarde null toe te wijzen aan de variabele
- instantiëren d.m.v. het sleutelwoord new
  ```
  $object = new MijnKlasse()
  ```

Methodes en klassevariabelen

- methodes en klassevariabelen oproepen op een instantie doen we met `$instance->member`
  ```
  $object->mijnMethode(12,53);
  $object->mijnVariabele2 = 57;
  ```
- methodes en klassevariabelen oproepen binnen dezelfde klasse doen we met `$this->member`
  ```
  $this->mijnMethode(12,53);
  $this->mijnVariabele2 = 57;
  ```
- methodes en klassevariabelen oproepen op een superklasse doen we met `parent::$member`
  ```
  parent::construct(12,53);
  parent::mijnVariabele2 = 57;
  ```
- statische methodes en klassevar. oproepen binnen dezelfde klasse doen we met `self::$member`
  ```
  self::construct(12,53);
  self::mijnVariabele2 = 57;
  ```
- statische methodes en klassevar. oproepen op een klasse doen we met `Class::$member`
  ```
  MijnKlasse::doeIets(12,53);
  MijnKlasse::mijnVariabele2 = 57;
  ```
- de zichtbaarheid van methodes en klassevariabelen tegenover de buitenwereld wordt bepaald met access modifiers (public / private / protected)

Overerving

- overerven doen we met het sleutelwoord `extends` (verwijst naar de superklasse)
  ```
  class MijnKlasse extends ParentKlasse { … }
  ```
- het verhinderen van het overschrijven van methodes in childklasses gebeurt met het keyword `final`
  ```
  final function display() {
        // implementatie
  }
  ```
- het overerven van meerdere klasses tegelijk wordt niet ondersteund in php

Interfaces

Een interface is een overeenkomst tussen ongerelateerde objecten voor het uitvoeren van dezelfde functionaliteit. Een interface stelt je in staat om aan te geven dat een object een bepaalde functionaliteit moet bezitten, maar het bepaalt niet hoe het object dat moet doen. De child class is dus vrij om de hele implementatie te doen, zolang hij maar voldoet aan de functionaliteit die de interface afdwingt.

- We laten de naam beginnen met een hoofdletter i
- Wordt in PHP aangeduid met het keyword `interface`

```
interface IMijnKlasse {
      function mijnMethode();
}
```

Abstracte klasses

Een abstracte klasse is een klasse waarbij niet alle methodes een implementatie moeten hebben, een deel van de functionaliteit kan dus overgelaten worden aan andere klasses die deze abstracte klasse implementeren. Hierdoor kan een abstracte klasse ook niet geïnstantieerd worden.

- Wordt in PHP aangeduid met het keyword `abstract`

Serialisatie

= het converteren van objecten naar een string bytes

- Nodig wanneer we een object willen bewaren in o.a. sessies, (tekst)bestanden, databases
- Serialiseren met serialize($object)
- Deserialiseren met unserialize($object)

HTTP

- **statusloos**: de server handelt commando per commando af
  - → er wordt geen rekening gehouden met voorgaande commando's
  - → aanvragen op de server hebben geen invloed op toekomstige aanvragen
- **connectieloos**: de HTTP-server (de webserver) zal geen verbindingen onthouden of openhouden.
  1. Wanneer een client een pagina wil ophalen, dan verstuurt die een request.
  2. De server haalt het bestand op en verstuurt het met een response.
  3. Nadat de bestanden werden doorgestuurd, sluit de server de verbinding.

Hoe identificeren we de client?

- Bij elke request opnieuw inloggen
  - → niet efficiënt
  - → niet gebruiksvriendelijk
- IP-adres bijhouden
  - → dynamische IP-adressen
  - → meerdere toestellen achter hetzelfde IP-adres

- De oplossing:
  - ✓ cookies: laten toe om gegevens op de client te bewaren, en deze met een volgende request te versturen
  - ✓ sessions: maken het mogelijk om requests te linken aan gebruikers, en op de server gegevens per gebruiker te bewaren.
    - → Steunt op session ID
    - → Wordt opgeslagen in een cookie bij de client of wordt doorgestuurd via POST of GET

Cookies

= laten toe om gegevens op de client te bewaren, en deze met een volgende request te versturen

- zijn gewoon leesbare tekst
- zijn terug te vinden in de webbrowser en eventueel als bestanden op de harde schijf
- kunnen bijdragen tot ongewenste effecten, omdat ze toelaten een gebruiker te volgen

Werking

1. Gebruiker stuurt een HTTP request om een bepaalde webpagina op te vragen
2. De webpagina (hier: PHP-pagina) zal code bevatten die een cookie meestuurt met de response.
3. Nu dat de cookie geplaatst is, geeft de client dit met elke http-request door naar de webserver.
4. De webserver kan de cookie uitlezen en verwerken in zijn response.

Soorten

- **tijdelijke cookies**: cookies die gedurende één webbrowsersessie worden bewaard.
  - → de cookie wordt verwijderd wanneer de gebruiker de webbrowser sluit.
  - → aanmaken met `setcookie(key, value)`
- **semi-permanente cookies**: cookies die gedurende meerdere webbrowsersessies worden bewaard.
  - → de cookie heeft een bepaalde vervaldatum, waarna hij verwijderd wordt
  - → aanmaken met `setcookie(key, value, vervaldatum, [pad, domein])`
    - ✓ vervaldatum: in Unix timestamp-formaat
    - ✓ pad: bv. en.wikipedia.org/**wiki**
    - ✓ domein: bv. en.wikipedia.org, wikipedia.org

Aanmaken

- tijdelijke cookies: aanmaken met `setcookie(key, value)`
- semi-permanente cookies: aanmaken met `setcookie(key, value, time, [pad, domein])`
  - ✓ time: vervaldatum in Unix timestamp-formaat
  - ✓ pad: bv. en.wikipedia.org/**wiki**
  - ✓ domein: bv. en.wikipedia.org, wikipedia.org

Aangezien cookies worden meegestuurd in de http headers, moet setcookie uitgevoerd worden voor er HTML-code wordt ge-output (zowel echo's als witruimte buiten php-tags)

Aangepaste cookiewaarden zijn pas beschikbaar in $_COOKIE bij de volgende http request.

Uitlezen

Cookies die geplaatst zijn en bij de request werden meegestuurd kunnen uitgelezen worden via de superglobale, associatieve array `$_COOKIE` (analoog met `$_GET` en `$_POST`)

Verwijderen

- een lege string meegeven als waarde
```
setcookie("language", "");
```
- false meegeven als waarde
```
setcookie("language", FALSE);
```
- een moment in het verleden meegeven als vervaldatum
```
setcookie("language", "", time() – 1);
```

Sessions

Werking

1. Client die nog geen session heeft vraagt een webpagina op via een HTTP request
2. De PHP-server genereert een nieuw en uniek session ID voor de client en voorziet een bestand op de server waarin de key/waarde paren die bij het session ID horen zullen worden opgeslagen
3. De server genereert een HTTP-response waarin HTML code wordt teruggestuurd naar de client. Eventueel kunnen er reeds bepaalde key/value paren worden bijgehouden voor de session.
4. De client stuurt een nieuwe HTTP request naar de server, die de cookie met de door de server gegenereerde session ID zal bevatten.
5. De server ontvangt de HTTP request, merkt de session ID cookie op, en stelt de key/waarde paren die bij de session ID horen ter beschikking in de $_SESSION superglobal
6. Je kan via jouw code de key/value paren die voor de session ID worden opgeslagen aanpassen. De nieuwe/aangepaste waarden zullen op de file op de server worden opgeslagen.

Aangezien cookies worden meegestuurd in de http headers, moet `session_start` uitgevoerd worden voor er HTML-code wordt ge-output (zowel echo's als witruimte buiten php-tags)

Sessions steunen op 2 pijlers om te werken:

- Sessioncontrole: het toekennen van een session ID uniek aan elke client en session
- Opslag van sessioninformatie, gelinkt aan het session ID

Session starten

`session_start()` maakt een nieuwe session ID aan, of stelt de bestaande session ID ter beschikking

Session verwijderen

- `session_destroy()` verwijdert de session ID met key/valueparen van de server
- De $_SESSION array en cookie van de client moeten manueel verwijderd worden:

```
if (ini_get("session.use_cookies")) {
   $params = session_get_cookie_params();
   setcookie(session_name(), '', time() - 3600,
       $params["path"], $params["domain"],
       $params["secure"], $params["httponly"]
    );
}
```

Databases

database: een georganiseerde en gestructureerde verzameling van gegevens

- → kan op een fysiek aparte server worden gehost t.o.v. de webserver (beter voor schaalbaarheid en hergebruik van de database in andere applicaties)
- → kan aangesproken worden over het netwerk
- → is vaak cross-platform
- → kan afgeschermd worden door het toewijzen van security rechten voor verschillende gebruikers

query: een opdracht gegeven aan een database, om gemakkelijk gegevens op te vragen en te manipuleren

- → d.m.v. een query language zoals MySQL
- → geavanceerde opdrachten zoals het verbinden van gegevenstabellen en het sorteren van gegevens
- → meer mogelijkheden en performanter dan het opslaan van data in tekstbestanden

Database Management System (DBMS): software dat databases beheert

- → Bv. MySQL, MS SQL, Oracle
- → PHP kan met veel DBMS'en samenwerken
- → MySQL is een gratis, performante, betrouwbare en wereldwijd gebruikte DBMS
  - ↳ Gebruikt SQL als query language
  - ↳ Wordt het meest met PHP gecombineerd, d.m.v. een PHP-extensie (wij gebruiken mysqli, biedt een API aan om te verbinden met een MySQL-database)
  - ↳ Automatisch meegeleverd met XAMPP

Content Management System (CMS): software die het mogelijk maakt om een website met bijhorende database op een gebruiksvriendelijke manier te beheren via webformulieren

Werking

```
<?php
// Verbinding maken
$mysqli = new mysqli("localhost", "user", "pass", "dbname");

// Controleren op verbindingsproblemen
if($mysqli->connect_error) {
   // Foutmelding naar gebruiker
}

// Queries uitvoeren
if($result = $mysqli->query( "SELECT * FROM producten")) {
   if($result->num_rows > 0)  {

       // Zet de resultatenrij automatisch om naar een PHP-object
       $object = $result->fetch_object();
       echo "<p>Naam: $object->productnaam<br/>";

       // Zet de resultatenrij automatisch om naar een associatieve array
       $array = $result->fetch_array();
       echo “Prijs: $array[‘prijs’] </p>\n";

       // Zet de resultatenrij automatisch om naar een array met indexen
       $array = $result->fetch_array();
       echo “Prijs: $array[0] </p>\n";

       // Krijg het aantal aangepaste rijen van de laatste query
       echo $mysqli->affected_rows

       // Automatisch loopen over alle beschikbare rijen
       while($product = $result->fetch_object()) {
             echo "<p>Naam: $product->naam<br/>";
             echo “Prijs: $product->prijs</p>\n";
       }

   }
}

// Verbinding sluiten
$mysqli->close();
?>
```

Tekstbestanden

`fopen(filename:String, mode:String):Resource`

- opent een bepaalde resource
  - → kan een lokale file of een URL zijn, PHP vindt het type locatie zelf uit
  - → de webserver moet toegang hebben tot het bestand en voldoende rechten erop hebben om de gewenste acties uit te voeren
- filename: locatie van het bestand
- mode: het gewenste type van toegang

Mode	R/W	Locatie v.d. pointer	Bestand leegmaken
r	read only	Begin	nee
r+	read/write	Begin	nee
w	write only	Begin	ja
w+	read/write	Begin	ja
a	write only	Einde	nee
a+	read/write	Einde	nee

`fget(resource):String`

- leest een lijn van de huidige positie in de file
- returnwaarde van `fopen` wordt meegegeven als argument
- returnt `false` als het einde van het bestand bereikt is → kan geloopt worden met een `while`
- verschillende OS'en hebben verschillen newline-tekens:

Linux/Unix	\n
Windows	\r\n
Mac	\r

`fwrite(resource, string):int`

- schrijft een string naar een bestand
- returnwaarde van `fopen` wordt meegegeven als argument
- returnt het aantal bytes dat werd weggeschreven, of `false` bij een error

`fclose(resource):Bool`

- de connectie met een bestand sluiten, nadat deze geopend werd met `fopen` en de gewenste `fget`- en `fwrite`-taken werden uitgevoerd
- returnwaarde van `fopen` wordt meegegeven als argument
- geeft een boolean terug die aangeeft of het sluiten geslaagd is

MVC (model-view-controller)

Model-view-controller (of MVC) is een ontwerppatroon ("design pattern") dat het ontwerp van complexe toepassingen opdeelt in drie eenheden met verschillende verantwoordelijkheden: datamodel (model), datapresentatie (view) en applicatielogica (controller). Het scheiden van deze verantwoordelijkheden bevordert de leesbaarheid en herbruikbaarheid van code. Het maakt ook dat bijvoorbeeld veranderingen in de gebruikersinterface niet direct invloed hebben op het datamodel en vice versa.

(web) application framework: een softwareframework dat gebruikt kan worden om de standaardstructuur van een (web)applicatie te implementeren

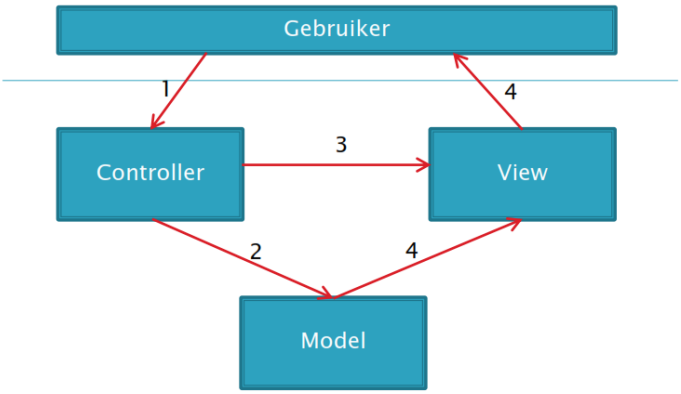
→ bv. Swing, Spring, Zend, Laravel

softwareframework: code die algemene functionaliteit levert en die gecustomized kan worden

Uitleg

Model

- bevat de data
- kan de views en controllers waarschuwen wanneer er aanpassingen zijn voorgekomen
  - → de views kunnen hun output updaten
  - → de controllers kunnen hun commando's aanpassen
  - → bij 'passieve MVC' gebeuren deze waarschuwingen niet



View

- output van de data, onder eender welke vorm
- het is mogelijk om één of verschillende views te hebben

Controller

- controleert de input van de gebruiker
- stuurt het model en/of de views aan

Flow

1. De gebruiker geeft een commando aan de controllerklasse, bijvoorbeeld:
   - ✓ Toon mij een hoeveelheid data
   - ✓ Pas deze data aan
   - ✓ Voer een actie uit
2. De controller past het model aan/voert de actie uit
3. De controller kiest (is gecodeerd door de developer) een geschikte view uit om de data van het model in weer te geven
4. De view wordt gerenderd met de data van het model en weergegeven aan de gebruiker

Geschiedenis

- oorspronkelijk gericht op de client applicatie markt.
- uiteindelijk veel gebruikt op het internet
  - → verschillende (web) application frameworks bouwen hierop

Voordelen

- ✓ elk onderdeel kan onafhankelijk van elkaar worden aangepast, zonder de andere onderdelen (klasses) (te veel) te beïnvloeden.
  - → makkelijker te onderhouden
  - → moeilijker om te bouwen.
- ✓ gemakkelijker verdelen van taken bij het ontwikkelen:
  - → front-end mensen maken views
  - → data mensen verzorgen model
  - → etc.

Toepassing: een website

Model

- in een programmeertaal geschreven.
- intelligente objecten (klasses) die de data bevatten die moet worden weergeven.

View

- een grotendeels in HTML geschreven pagina waarin placeholders voor data voorzien worden
- deze placeholders visualiseren bij een eenvoudige implementatie de data die in het model is weergeven.

Controller

- in een programmeertaal geschreven
- ontvangt de input van de gebruiker
- passen de data in het model aan
- tonen nieuwe data door middel van een view te renderen met aangepaste inhoud