erasmus

HOGESCHOOL BRUSSEL

Web Development Advanced

erasmus

HOGESCHOOL BRUSSEL

Verzenden van gegevens naar de server

Herhaling: Formulieren

FORMULIEREN

- We wensen de gebruiker de mogelijkheid te geven om informatie door te zenden naar de server.
- Dit kan door middel van een formulier.
- Deze informatie kunnen we dan verwerken op de server door middel van PHP code.
- Bijvoorbeeld:
  - Opslaan in database
  - Files schrijven
  - Emails genereren
  - …

erasmus HOGESCHOOL BRUSSEL

php

FORMULIEREN

Uw voornaam: Frauke

Uw achternaam: Vanderzijpen

Welke besturingssystemen gebruikt u regelematig?

- [x] Windows
- [x] Mac OS X
- [x] Linux

Wat is uw minst favoriete besturingssysteem Andere

Welk wachtwoord wenst u: ••••••••

Uw opmerkingen:

Mijn opmerkingen

Opslaan

erasmus HOGESCHOOL BRUSSEL

FORMULIEREN

- Een formulier bevat invulvelden waar de gebruiker informatie kan invullen. Het formulier wordt weergegeven op de webpagina.
- Nadat het formulier werd “ingediend” (submit) zijn de ingevulde gegevens beschikbaar op de server en kunnen ze daar verwerkt worden
- “Invulvelden” ruim bekijken: vb ook bestanden uploaden.

erasmus
HOGESCHOOL BRUSSEL

php

FORMULIEREN OPBOUWEN

- **action** = locatie/pagina waar de inhoud van het formulier naar wordt gestuurd bij het submitten
- **method** = de wijze waarop de inhoud naar de server wordt verstuurd (GET of POST)

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

erasmus HOGESCHOOL BRUSSEL

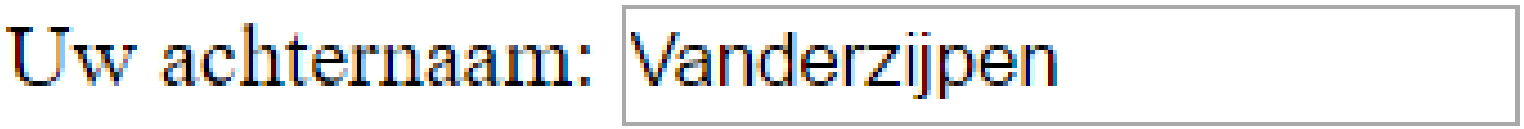
VEEL GEBRUIKTE VELDTYPES

- Input types:
  - Text
  - Password
  - Hidden
  - Checkbox
  - Radio
  - Submit
- Textarea
- Select

erasmus
HOGESCHOOL BRUSSEL

TEXTVELD

- Geeft de mogelijkheid aan de gebruiker om een tekstuele waarde in te voeren



erasmus
HOGESCHOOL BRUSSEL

TEXTVELD

```
<input type="text" name="achternaam" />
```

- type="text"
  - specifieert dat je een textveld wilt maken

- name="achternaam"
  - naam van het input veld
    ➔ dit zal je later in PHP gebruiken om de ingevulde waarde uit te lezen

erasmus
HOGESCHOOL BRUSSEL

PASSWORD-VELD

- Voor het invoeren van wachtwoorden
- De ingevoerde tekst wordt verborgen


```
<input type="password" name="wachtwoord" />
```

erasmus HOGESCHOOL BRUSSEL

HIDDEN-VELD

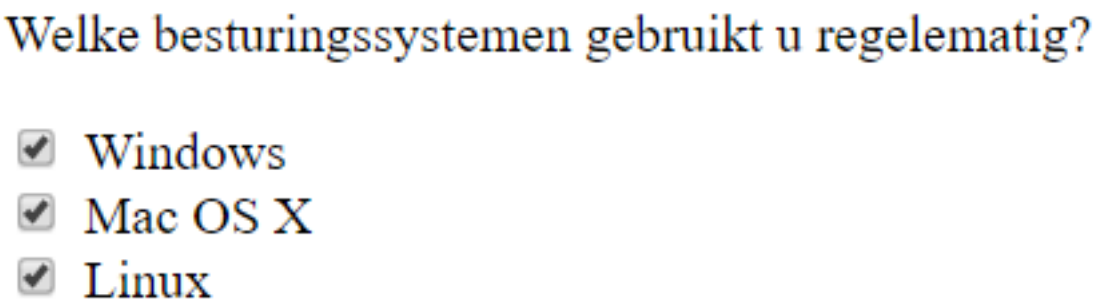
- Is een “invoerveld” dat geen visualisatie heeft naar de gebruiker (de gebruiker ziet niets van een hidden field)
- Het hidden field is wel aanwezig in de HTML code die de webbrowser gebruikt om de pagina weer te geven bij de eindgebruiker.
- We geven het een waarde (value) vanuit de HTML
- Wordt mee opgestuurd naar de server bij het submitten van de form.
- Te vergelijken met een tekstveld dat verborgen is voor de gebruiker en waar de gebruiker de waarde niet rechtstreeks kan van aanpassen

```
<input type="hidden" name="id" value="4" />
```

erasmus HOGESCHOOL BRUSSEL

CHECKBOX

- Een ja/nee vraag naar de gebruiker toe

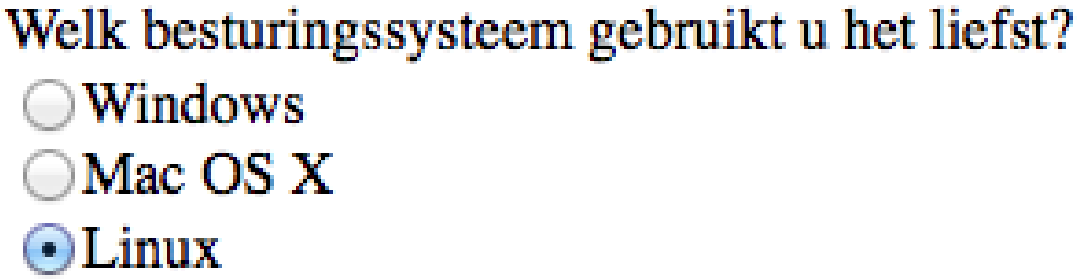



```
<input type="checkbox" name=”useswindows" value="win“ />
<input type="checkbox" name=”usesmac" value="mac“ />
<input type="checkbox" name=”useslinux" value="linux“ />
```

erasmus HOGESCHOOL BRUSSEL

RADIOBUTTON

- Gebruiker een keuze laten maken uit een aantal items




```
<input type="radio" name="ospref" value="win" />
<input type="radio" name="ospref" value="mac" />
<input type="radio" name="ospref" value="linux" />
```

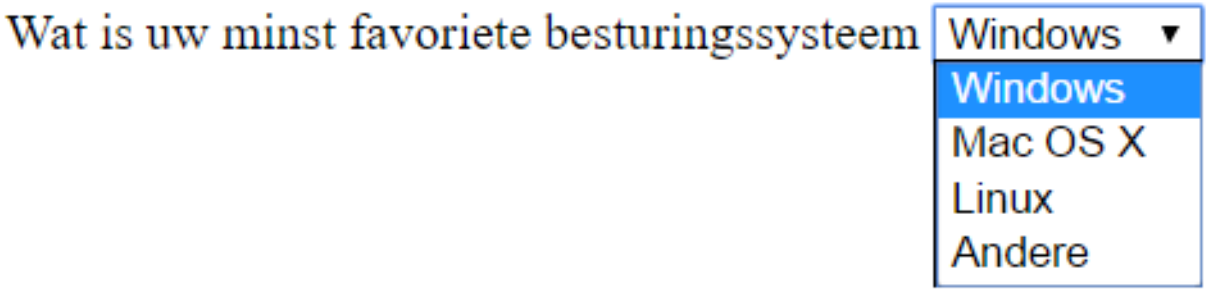
TEXTAREA

- Zoals een textveld, maar kan meer tekst bevatten over meerdere lijnen


```
<textarea name="opmerkingen"></textarea>
```

COMBOBOX/DROPDOWN

- Gebruiker een keuze laten maken uit een aantal items




```
<select name="besturingssysteemafkeur">
   <option value="windows">Windows</option>
   <option value="mac">Mac OS X</option>
   <option value="linux">Linux</option>
   <option value="andere">Andere</option>
</select>
```

erasmus HOGESCHOOL BRUSSEL

Verzenden van gegevens naar de server

GEGEVENS VERZENDEN

- GET
  - http://example.com?q=Banaan
  - Via HTML formulier
- POST
  - Via HTML formulier

GET-METHODE

- Gegevens worden toegevoegd aan de URI
  - URI = Uniform Resource Identifier
    - Vb: http://www.google.be
    - URL (Uniform Resource Locator) wordt als synoniem aanvaard, maar is niet volledig hetzelfde.
- Zichtbaar in het webbrowser adresveld

GET-METHODE

- De gegevens worden naar de server doorgestuurd in de zogenaamde “querystring”

- Querystring zijn een aantal naam/waarde paren die worden verstuurd via de URI
  - http://www.bibliotheek.be/zoekpagina.php**?**zoekterm=bloemen

GET-METHODE

Querystring

<Webpaginanaam>?<naam1>=<waarde1>**&**<naam2>=<waarde2>

Er kunnen extra naam/waarde paren blijven toegevoegd worden, zolang als we ze blijven scheiden met een **&**

GET-METHODE

Querystring

- Let op: maximale lengte adres website is beperkt
  - Afhankelijk van webserver
  - In de praktijk soms niet langer dan 255 tekens
- De webserver leest de extra gegevens in uit de URI/querystring.

GET-METHODE

Querystring opbouwen

- Manuele invoer door de eindgebruiker
- Querystring als developer klaarzetten achter een link
- Gebruik maken van een HTML formulier

GET-METHODE

Querystring opbouwen

- Manuele invoer door de eindgebruiker
  - Rechtstreeks de URI aan passen
  - Niet wenselijk om dit door de eindgebruiker te laten doen
  - Eventueel wel handig bij het testen

GET-METHODE

Querystring opbouwen

- Querystring als developer klaarzetten achter een link

```
<a href="index.php?language=NL">NL</a>
<a href="index.php?language=FR">FR</a>
<a href="index.php?language=EN">EN</a>
```

- Geeft gebruiker nog altijd niet de mogelijkheid om zelf waarden te verzinnen

erasmus HOGESCHOOL BRUSSEL

php

GET-METHODE

Querystring opbouwen

- Gebruik maken van een HTML formulier

```
<form action="index.php" method="GET">
</form>
```

GET-METHODE

Querystring uitlezen

- De querystring komt toe op de webserver
- De PHP parser zal er zelf voor zorgen dat de querystring wordt uitgelezen en wordt omgezet naar PHP variabelen, die wij kunnen gebruiken in ons script
- De querystring waarden zijn toegankelijk via de **$_GET** variabele die beschikbaar is

GET-METHODE

$_GET

- Een globale variabele die PHP ons ter beschikking stelt
- Niet zelf declareren
- Dit is een “Superglobal”
  ➔ Deze variabele is altijd ter beschikking

erasmus HOGESCHOOL BRUSSEL

php

GET-METHODE

$_GET

- Is een associatieve array
- Sleutelwaarde = naam van naam/waarde paar van querystring
- Geassocieerde waarde = waarde van naam/waarde paar van querystring

GET-METHODE

$_GET

http://www.studenten.be/verwerkpagina.php**?**
voornaam=Joske&achternaam=Vermeulen

```
<?php
   $uitgelezenAchternaam = $_GET[“achternaam”];
   $uitgelezenVoornaam = $_GET[“voornaam”];
?>
```

erasmus
HOGESCHOOL BRUSSEL

php

GET-METHODE

Voor- en nadelen

- Querystrings
  - Zijn voor iedereen zichtbaar
    ➔ Niet voor vertrouwelijke gegevens
  - Zijn zichtbaar in de geschiedenis van de door de browser bezochte pagina's (caching)
  - Kunnen mee met een bookmark worden opgeslagen
  - Worden dikwijls op de server gelogd
  - Beperkt in grootte
  - Niet voor alle types gegevens

erasmus HOGESCHOOL BRUSSEL

php

GET-METHODE

Wanneer gebruiken?

- Gebruiken om bijvoorbeeld:
  - Velden van een zoekformulier aan de server te geven
  - Taalkeuze aan server doorgeven

GET-METHODE

Wanneer gebruiken?

- NIET gebruiken bij
  - Bij pagina's die wijzigingen doorvoeren op de server
    - Vb: data bewerken op website
  - Waarom?
    - Querystrings kunnen in browser worden opgeslagen (geschiedenis, bookmarks, …)
    - Bij terug bezoeken van pagina via zo'n opgeslagen link zouden de daaraan gekoppelde acties herhaald worden

POST-METHODE

- Gegevens worden naar de server doorgestuurd in de body van de HTTP request.

```
POST /studenten/registreer.php HTTP/1.1
Host: localhost
Connection: close
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac
OS X 10_6_4; de-de) AppleWebKit/533.16 (KHTML,
like Gecko) Version/5.0 Safari/533.16
Accept:
text/xml,text/html,text/plain,image/png,image/jpe
g,image/gif
Accept-Charset: ISO-8859-1,utf-8
Voornaam=Joske&achternaam=Vermeulen
```

erasmus HOGESCHOOL BRUSSEL

POST-METHODE

Verzenden van POST-variabelen

- Door de gebruiker een formulier te laten invullen

```
<form action="index.php" method="post">
</form>
```

POST-METHODE

Verzenden van POST-variabelen

- Browser zorgt er automatisch voor dat de variabelen in de HTTP post body terecht komen.
- Wordt doorgestuurd naar de server.

POST-METHODE

POST-variabelen opvangen

- De querystring komt toe op de webserver
- De PHP parser zal er zelf voor zorgen dat de POST variabelen worden uitgelezen en worden omgezet naar PHP variabelen, die wij kunnen gebruiken in ons script
- De POST waarden zijn toegankelijk via de $_POST variabele die beschikbaar is

POST-METHODE

$_POST

- Een globale variabele die PHP ons ter beschikking stelt
- Niet zelf declareren
- Is een Superglobal
  ➔ Variabele is altijd ter beschikking

POST-METHODE

$_POST

- Een globale variabele die PHP ons ter beschikking stelt
- Niet zelf declareren
- Is een Superglobal
  ➔ Variabele is altijd ter beschikking

POST-METHODE

$_POST

- Is een associatieve array
- Sleutelwaarde = naam van naam/waarde paar van de POST variabele
- Geassocieerde waarde = naam/waarde paar van de POST variabele

HTML form code:

```
<form action="verwerkpagina.php" method=”post">
   <input type=“text” name=“voornaam”/>
   <input type=“text” name=“achternaam”/>
   <input type=“text” name=“geboortejaar”/>
   <input type=“submit” value=“Opslaan”/>
</form>
```

Verstuurde URI:

http://www.studenten.be/verwerkpagina.php

De formulierinhoud wordt achter de schermen in de HTTP request doorgestuurd naar de server

Uitlezen in PHP (op verwerkpagina.php):

```
<?php
   $uitgelezenAchternaam = $_POST[“achternaam”];
   $uitgelezenVoornaam = $_POST[“voornaam”];
   $uitgelezenGeboortejaar = $_POST[“geboortejaar”];
?>
```

erasmus HOGESCHOOL BRUSSEL

php

POST-METHODE

Voor- en nadelen

- Mogelijk om veel meer gegevens door te sturen.
  - Hoeveelheid gegevens is afhankelijk van de server (php.ini)
- Doorgestuurde informatie is niet onmiddellijk zichtbaar
  - Is veiliger, maar niet volledig veilig => encryptie
- Doorgestuurde informatie komt ook niet terug in bookmarks, history, … van browser

erasmus HOGESCHOOL BRUSSEL

php

POST-METHODE

Wanneer gebruiken

- Bij pagina's die wel wijzigingen doorvoeren
  - Vb: data bewerken op website
- Waarom?
  - POST variabelen worden default niet in browser opgeslagen
  - Bij terug bezoeken van pagina via zo'n opgeslagen link zouden de daaraan gekoppelde acties niet herhaald worden
- Dus gebruiken om bijvoorbeeld:
  - Registratieformulieren te maken
  - Bestanden te uploaden (meer data doorstuurbaar)

erasmus HOGESCHOOL BRUSSEL

php

GEGEVENS VERWERKEN

1. Nagaan of formulier verstuurd werd
   - $_SERVER["REQUEST_METHOD"]

2. Formvalidatie
   - Controleren of variabelen verzonden zijn ➔ isset()
   - Controleren of variabelen een waarde hebben ➔ empty()
   - Andere:
     - Postcode: 4 cijfers
     - E-mail: correct opgebouw
     - …

VALIDATIE VAN FORMULIER

- Wij kunnen op onze webpagina formuliervelden voorzien om gegevens in te vullen.
- We kunnen er niet zeker van zijn dat de gebruiker deze juist gebruikt
- Daarom zullen we een aantal checks moeten doen zodat we er min of meer zeker van zijn dat de formuliervelden juist gebruikt werden

erasmus HOGESCHOOL BRUSSEL

php

CLIENT/SERVER SIDE VALIDATIE

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
- De technologie die je voor de client side validatie gebruikt kan worden uitgeschakeld bv. Javascript
- Er bestaat een mogelijkheid dat er niet-valide gegevens worden doorgestuurd naar de server

CLIENT/SERVER SIDE VALIDATIE

Server side validatie

+ De gebruiker kan de technologie die gebruikt wordt voor de server side validatie niet uitschakelen
+ jij hebt volledige controle over de server omgeving die gebruikt wordt voor de validatie
+ Veel moeilijker voor een gebruiker om toch nog onjuiste gegevens te laten opslaan/verwerken door de server

erasmus HOGESCHOOL BRUSSEL

php

CLIENT/SERVER SIDE VALIDATIE

Server side validatie

- Validatie neemt resources van server in
- Elke keer de validatie mislukt, moet de pagina met het formulier opnieuw worden teruggestuurd naar de client => meer belasting voor de server

CLIENT/SERVER SIDE VALIDATIE

- Probeer zowel server side als client side validatie toe te passen
  - Je krijgt het beste van beide werelden
- Op het einde van de rit blijft SERVER side validatie wel het belangrijkste om te hebben

BEST PRACTICES

- Centraliseer je validatiechecks in aparte functies die je kan aanroepen en hergebruiken.
  - Zet deze functies eventueel in een aparte “validatie.php” die je include in je formulierpagina

File uploads

FILE UPLOADS

- PHP kan een gebruiker toelaten om files zoals een foto te uploaden naar de server
- Dit kan door volgende basiselementen op te nemen in je formulier

```
<form action="upload.php" method="POST"
enctype="multipart/form-data">
   <input type="file" name="bestand">
   <input type="submit" value="Verzend">
</form>
```

erasmus HOGESCHOOL BRUSSEL

FILE UPLOADS - OPMERKINGEN

- De method moet **POST** zijn
- De **enctype** moet ingesteld zijn op "**multipart/form-data**"
  - Je specifieert hiermee het contenttype dat zal worden gebruikt bij het indienen van de form
  - "multipart/form-data" duidt op binaire data. Dit is wat we nodig hebben bij het uploaden van een file
- Je neemt een inputveld van het type "**file**" op in je formulier

FILE UPLOADS - VERWERKING

```php
<?php
    //Bestandsnaam
    echo $_FILES["bestand"]["name"];

    //Bestandstype/mime-type (vb: "image/jpeg")
    echo $_FILES["bestand"]["type"];

    //Bestandsgrootte in bytes
    echo $_FILES["bestand"]["size"];

    //Tijdelijke bestandsnaam op server
    echo $_FILES["bestand"]["tmp_name"];

    //Eventuele fouten die bij uploaden zijn opgetreden
    echo $_FILES["bestand"]["error"];
?>
```

erasmus HOGESCHOOL BRUSSEL

FILE UPLOADS - VERWERKING

- Geuploade files worden op de webserver automatisch tijdelijk opgeslagen
  - `$_FILES["bestand"]["tmp_name"];`
- Gebruik de functie move_uploaded_file om geuploade bestanden naar een locatie op je schijf te verplaatsen.

FILE UPLOADS - VERWERKING

- move_uploaded_file(filename, destination)
  - **Filename** = de naam van de geuploade file
  - **Destination** = de locatie waar je geuploade file naar wilt verplaatsen
  - Als de file reeds bestaat op de bestemmingslocatie, dan wordt die daar overschreven
  - Checkt ook of de file via HTTP POST is geupload

FILE UPLOADS - VERWERKING

- move_uploaded_file(filename, destination)
  - **Filename** = de naam van de geuploade file
  - **Destination** = de locatie waar je geuploade file naar wilt verplaatsen
  - Als de file reeds bestaat op de bestemmingslocatie, dan wordt die daar overschreven
  - Checkt ook of de file via HTTP POST is geupload

```
<?php
   move_uploaded($_FILES["bestand"]["tmp_name"],
   "uploaddir/" . $_FILES["bestand"]["name"]);
?>
```

erasmus HOGESCHOOL BRUSSEL

FILE UPLOADS – OPM.

- De map waarin je de geuploade file wilt opslaan moet writeable zijn voor de webserver gebruiker
- Het toelaten van uploaden van files is een groot security risico.
- De grootte van de bestanden die je kan uploaden is beperkt. Je kan deze instellen in je PHP configuratie

erasmus

HOGESCHOOL BRUSSEL

Sessies en Cookies

HTTP

- Statusloos
- Connectieloos

HTTP

Statusloos

De statusloosheid van de webserver betekent dat de server commando per commando afhandelt, zonder rekening te houden met voorgaande commando's.

Hiermee wordt bedoeld dat aanvragen op de server geen invloed zullen hebben op toekomstige aanvragen.

HTTP

Connectieloos

Dat betekent dat de HTTP-server geen verbindingen zal onthouden of openhouden.

Wanneer een client een pagina wil ophalen, dan verstuurt die een request. De server haalt het bestand op en verstuurt het met een response. Nadat de bestanden werden doorgestuurd, sluit de server de verbinding.

HTTP

Gevolgen

- Welke request komt van welke client?
- Welke gebruiker stuurt welke gegevens naar de server?
- Welke gebruiker mag bepaalde gegevens opvragen?
- Hoe data linken aan de bezoeker van een pagina?
  - Vb: voorkeur van taal

HTTP

Probleem

- Hoe kan de server weten dat twee verschillende HTTP requests van eenzelfde persoon komen of niet?
- We wensen gegevens aan bepaalde gebruikers te kunnen linken over verschillende requests heen.

HTTP

Mogelijke oplossing

- Gebruiker bij elke request opnieuw login credentials laten invullen in een formulier. Deze gegevens worden dan bij de request meegestuurd naar de server

  ➔ Niet praktisch en niet gewenst

SESSIONS EN COOKIES

- 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 client OF
    - Wordt doorgestuurd via POST of GET
- Zijn concepten die niet alleen bij PHP gebruikt worden

Cookies

WAT ZIJN COOKIES?

- Laten toe om gegevens op de **client** te bewaren, en deze met een volgende request te versturen naar de server
  - Vb. Taalkeuze
- Leesbare data
  - Geen scripts, programma's, programeercode, virussen of spyware.
  - Kunnen geen info van harde schijf lezen, wijzigen of verwijderen.

WERKING

1. Gebruiker stuurt een HTTP request om een bepaalde webpagina op te vragen

2. De webpagina (in ons geval de PHP pagina) zal code bevatten waarmee de developer aangeeft dat hij een cookie wilt bewaren op de client. Deze aanvraag wordt meegestuurd met de HTTP response van de webserver

WERKING

HTTP-response

```
HTTP/1.1 200 OK
Date: Thu, 01 Sep 2011 08:30:00 GMT
Server: Apache/2.2.16 (Debian) PHP/5.3.3-7+squeeze3 with
Suhosin-Patch
X-Powered-By: PHP/5.3.3-7+squeeze3
Set-Cookie: language=nl; expires=Fri, 12-Jul-2030 23:59:59 GMT
Cache-Control: must-revalidate
Content-Encoding: gzip
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=utf-8
[body]
```

WERKING

3. Nu dat de cookie geplaatst is, geeft de gebruiker dit bij elke HTTP request door naar de webserver. Dit gebeurt via de HTTP-request

WERKING

HTTP-request

```
POST / HTTP/1.1
Host: iwt.ehb.be
Connection: close
User-Agent: Mozilla/5.0 (Macintosh; U; Intel
Mac OS X 10_6_4; de-de) AppleWebKit/533.16
(KHTML, like Gecko) Version/5.0 Safari/533.16
Accept:
text/xml,text/html,text/plain,image/png,image
/jpeg,image/gif
Accept-Charset: ISO-8859-1,utf-8
Cookie: language=nl
```

WERKING

4. De webserver kan de cookie uitlezen en verwerken in zijn response.

SOORTEN

- Tijdelijke cookies
- Semi-permanente cookies

erasmus
HOGESCHOOL BRUSSEL

php

SOORTEN

Tijdelijke cookies

- Cookies die gedurende één webbrowsersessie worden bewaard.
- De cookie zal dus verwijderd worden wanneer de gebruiker de webbrowser sluit.

SOORTEN

Semi-permanente cookies

- Cookies die gedurende meerdere webbrowsersessies worden bewaard.
- De cookie heeft wel een een bepaalde vervaldatum
- De cookie wordt verwijderd na die vervaldatum

COOKIES AANMAKEN

Tijdelijke cookies

- setcookie(<key>,<value>);

```
<?php
      setcookie("language","nl");
?>
```

COOKIES AANMAKEN

Semi-permanente cookies

- setcookie(<key>,<value>, <vervaldatum>);
- <vervaldatum> is van type integer
  - Het aantal seconden sinds 1 januari 1970
  - Perfect te gebruiken ism functie time() of mktime()

```
<?php
//cookie 7 dagen te bewaren
setcookie("language", "nl", time()+60*60*24*7);
?>
```

erasmus HOGESCHOOL BRUSSEL

COOKIES AANMAKEN

setcookie

- Setcookie aanvaardt nog extra argumenten:
- Hieronder enkele belangrijke, bekijk PHP manual voor alle opties:
- Setcookie(<key>,<value>,
- <vervaltijdstip>,<pad>,<domein>)

COOKIES AANMAKEN

setcookie

- Pad = Op welk pad van het domein zal het cookie beschikbaar zijn?
  - Vb: en.wikipedia.org/wiki
  - Default is de directory waar de cookie in wordt gezet

COOKIES AANMAKEN

setcookie

- Domein = Op welk onderdeel van het domein zal het cookie beschikbaar zijn
  - Vb: en.wikipedia.org
    ➔ enkel beschikbaar in subdomein en van wikipedia.org
  - Vb: wikipedia.org
    ➔ beschikbaar in alle subdomeinen van wikipedia.org, vb: ook in nl.wikipedia.org

erasmus HOGESCHOOL BRUSSEL

php

COOKIES UITLEZEN

- Cookies die geplaatst zijn en bij de request werden meegestuurd kunnen uitgelezen worden via de superglobal $_COOKIE
  - Associatieve array
  - Analoog met $_GET en $_POST

COOKIES UITLEZEN

- De cookie komt toe op de webserver
- De PHP parser zal er zelf voor zorgen dat de cookie wordt uitgelezen en wordt omgezet naar PHP variabelen, die wij kunnen gebruiken in ons script
- De cookies zijn toegankelijk via de $_COOKIE variabele die beschikbaar is

COOKIES UITLEZEN

$_COOKIE

- Een globale variabele die PHP ons ter beschikking stelt
- Niet zelf declareren
- Is een Superglobal
  ➔ Variabele is altijd ter beschikking

COOKIES UITLEZEN

```
<?php
      $uitgelezenTaal = $_COOKIE[“language”];
?>
```

COOKIES VERWIJDEREN

- Opnieuw gebruik maken van setcookie met de key van de te verwijderen cookie, maar
  - Als tweede parameter een lege string meegeven OF
  - Als tweede parameter FALSE meegeven OF
  - Een verlooptijdstip in het verleden meegeven

COOKIES VERWIJDEREN

```
<?php
   setcookie(“language”,“”);
   setcookie(“language”,FALSE);
   setcookie(“language”,””,time()-1);
?>
```

EERST SETCOOKIE, DAN HTML

- De headers van een HTTP response worden geschreven vooraleer de HTML uitvoer wordt gegenereerd. Dit heeft als consequentie dat alle HTTP response headers van de server moeten worden gedefinieerd vooraleer er HTML wordt geoutput.
- In de praktijk wilt dit zeggen dat **setcookie moet aangeroepen worden voordat HTML code wordt gegenereerd in je PHP-script**, want de cookiedefinities moeten in de headers komen.
- **Whitespaces voor/buiten <?php ?> is ook HTML uitvoer !!!**

erasmus HOGESCHOOL BRUSSEL

BESCHIKBAARHEID IN $_COOKIE

- $_COOKIE representeert de cookiewaarden die werden gevonden in de huidige HTTP-request van de client.
- Wanneer een cookie nieuw wordt gedefinieerd in een PHP script, kan je de inhoud van de nieuwe cookie niet onmiddellijk uitlezen uit $_COOKIE binnen dezelfde thread.
- De cookie waarde is pas beschikbaar in $_COOKIE bij de volgende HTTP request

CHECK OP BESTAAN COOKIES

- Je kan opnieuw de functie isset() gebruiken om het bestaan van een cookie te achterhalen.

```
<?php
   if (isset($_COOKIE[“language”])){
      //OK, cookie bestaat => uitlezen
      $var = $_COOKIE[“language”];
   }
?>
```

COOKIES: SECURITY

- Cookies worden als gewone tekst doorgestuurd en meestal ook zo opgeslagen

- Er zijn manieren om cookies te stelen, na te maken, … Een gestolen cookie kan misschien gebruikt worden om een ingelogde browsingsessie over te nemen en zaken uit te voeren onder een andere login.

Sessions

WAT IS EEN SESSION?

- 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 client OF
  - Wordt doorgestuurd via POST of GET

WERKING

- Sessions steunen op 2 pijlers om te werken:
  - Sessioncontrole
  - Opslag van sessioninformatie

WERKING - SESSIONCONTROLE

- Het is nodig om aanvragen van eenzelfde gebruiker te kunnen identificeren over verschillende requests
  ➔ Toekennen van een unieke session-ID aan elke gebruiker die gebruik maakt van sessions

- Session ID is uniek nummer dat aan een unieke gebruiker gedurende een session wordt toegekend.

WERKING - SESSIONCONTROLE

- Toekennen van sessionID gebeurt standaard door bij opstarten van een session een cookie door te sturen naar de gebruiker met daarin het session ID.
- Het session ID is een door de server gegenereerde tekencombinatie.
- Deze cookie met sessionID wordt bij elke HTTP request door de client aan de server doorgestuurd.
- De server kan aan de hand van deze sessionID de gebruiker blijven identificeren.

erasmus
HOGESCHOOL BRUSSEL

php

WERKING - SESSIONCONTROLE

- Standaard gebeurt het bijhouden van de sessionID door middel van tijdelijke cookies.
  - Je kan dit aanpassen naar semi-permanente cookies.

WERKING – OPSLAG VAN INFO

- Omdat de server - na het starten van de session - bij elke request het sessionID van de client krijgt, kan hij de client uniek identificeren over verschillende requests heen
- De server zal dit sessionID gebruiken als key om een hoeveelheid gegevens op de server op te slagen.
- Deze “gegevens” zijn key/value paren.
- De gegevens worden dus bijgehouden op de server per client die een sessionID werd toegewezen.

WERKING – OPSLAG VAN INFO

- De key/value paren die per sessionID worden bijgehouden worden standaard in een tekstbestand op de server opgeslagen.
- De naamgeving van dit tekstbestand kan je lezen in je PHP configuratiefile.
- Het is ook mogelijk om de sessioninformatie op slagen in een database

WERKING

1. Client die nog geen session heeft vraagt een webpagina op via een HTTP request

2. De aangevraagde pagina bevat PHP code om een session te openen => de PHP server genereert een nieuw/uniek sessionID voor de client, en
   - Stuurt deze standaard via een set-cookie in de HTTP-response naar de client
   - Voorziet plaats in een bestand op de server waarin de key/waarde paren die bij het sessionID zullen horen zullen worden opgeslagen

WERKING

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 gebruiker/client stuurt een nieuwe HTTP-request naar de server. Het HTTP-request zal de cookie met de door de server gegenereerde sessionID bevatten.

WERKING

5. De server ontvangt de HTTP-request, merkt de session-ID cookie op, en gaat op zijn schijf op zoek naar de key/waarde paren die bij die sessionID horen.

6. De key/value paren die de server op zijn schijf heeft staan worden beschikbaar gemaakt binnen de PHP omgeving en zijn te bereiken via jou code

WERKING

7. Je kan via jouw code de key/waarde paren die voor de sessionID worden opgeslagen aanpassen. De nieuwe/aangepaste waarden zullen op de file op de server worden opgeslagen. De key/value paren worden dus niet naar de client gestuurd

8. Je kan een HTML pagina genereren rekening houdend met de key/waarde paren die werden opgeslagen in het bestand van de server dat bij de sessionID behoort

WERKING

- Bij een volgende HTTP request van de client kan dit proces zich herhalen

- => sessionID cookie wordt opnieuw opgestuurd, en op basis daarvan worden de bijbehorende key/waardeparen opnieuw ter beschikking gesteld.

SESSION AANMAKEN

- Wanneer men een nieuwe session wenst te starten, moet men dit doorgeven aan PHP.
- Gebruik het commando session_start()

```
<?php
    session_start();
?>
```

SESSION VOORTZETTEN

- Wanneer er aan de client reeds een session is toegewezen, moet men bij een nieuwe HTTP-request opnieuw de opdracht session_start() oproepen.
- De bewaarde data wordt op die manier beschikbaar voor je PHP code.
- PHP engine controleert automatisch of er
  - Een nieuwe sessionID moet worden aangemaakt
  - Een bestaande sessionID met de bijbehorende opgeslagen key/valueparen moet worden gebruikt

KEY/WAARDE PAREN OPSLAAN

- Vooraleer we key/waarde paren kunnen opslaan voor een session, moeten we eerst een session aanmaken of verderzetten.
- Je kan variabelen opslaan via de superglobal $_SESSION.
- Merk op dat we deze keer dus wél waarden toekennen aan de superglobal

KEY/WAARDE PAREN OPSLAAN

```
<?php
   session_start();
   $_SESSION[“language”]=“nl”;
?>
```

KEY/WAARDE PAREN UITLEZEN

- Vooraleer we key/waarde paren kunnen uitlezen voor een session, moeten we eerst een session verderzetten.
- Je kan variabelen uitlezen via de superglobal $_SESSION.

KEY/WAARDE PAREN UITLEZEN

```
<?php
   session_start();
   if(isset($_SESSION[“language”])) {
      $taal = $_SESSION[“language”];
   }
?>
```

SESSIES VERWIJDEREN

- Om een sessie te verwijderen kan men het commando session_destroy() aanroepen.
  - Key/value paren worden van de schijf verwijderd
- Nadeel: $_SESSION blijft beschikbaar, alsook de cookie op de client.
- Als je wilt dat de session volledig vernietigd wordt (bijvoorbeeld om een gebruiker uit te loggen) moet je nog een aantal extra stappen ondernemen

SESSIES VERWIJDEREN

```
<?
   session_start();
   //sessievariabelen resetten
   $_SESSION = array();
   //cookie vernietigen
   if (ini_get("session.use_cookies")) {
      $params = session_get_cookie_params();
      setcookie(session_name(), '', time() - 3600,
      $params["path"], $params["domain"],
      $params["secure"], $params["httponly"]);
   }
   //Sessiedata verwijderen
   session_destroy();
?>
```

erasmus
HOGESCHOOL BRUSSEL

EERST SESSION, DAN HTML

- De headers van een HTTP response worden geschreven vooraleer de HTML uitvoer wordt gegenereerd. Dit heeft als consequentie dat alle HTTP response headers van de server moeten worden gedefinieerd vooraleer er HTML wordt geoutput.
- In de praktijk wilt dit zeggen dat **session_start() moet aangeroepen worden voordat HTML code wordt gegenereerd in je PHP-script**, want de cookiedefinities moeten in de headers komen.
- **Whitespaces voor/buiten <?php ?> is ook HTML uitvoer !!!**

erasmus HOGESCHOOL BRUSSEL

php

CHECK OP BESTAAN SESSION WAARDEN

- Je kan opnieuw de functie isset() gebruiken om het bestaan van een session waarde te achterhalen.

```
<?php
   if (isset($_SESSION[“language”])){
      //OK, key/value paar bestaat => uitlezen
      $var = $_ SESSION[“language”];
   }
?>
```

erasmus
HOGESCHOOL BRUSSEL

php

SECURITY

- SessionID's worden als gewone tekst doorgestuurd en meestal ook zo opgeslagen

- Er zijn manieren om sessionID's te stelen, na te maken, … Een gestolen sessionID kan misschien gebruikt worden om een ingelogde sessie over te nemen en zaken uit te voeren onder een andere login.

erasmus
HOGESCHOOL BRUSSEL

php

erasmus

HOGESCHOOL BRUSSEL

Objectgeoriënteerde PHP

OOP

- Objectgeoriënteerd programmeren
- Ook in PHP
  - Productiever
  - Hergebruik
  - Meer structuur
  - Gemakkelijker om code van anderen te gebruiken
    - Vb: frameworks
  - Minder interessant in zeer kleine applicaties

KLASSEN

Sleutelwoord “class”

```
<?php
   class MijnKlasse {
      var $mijnVariabele1;
      var $mijnVariabele2;
      function mijnMethode($arg1, $arg2) {
       //...
      }
   }
?>
```

KLASSEN

Naam van de klasse
PascalCase

```
<?php
   class MijnKlasse {
      var $mijnVariabele1;
      var $mijnVariabele2;
      function mijnMethode($arg1, $arg2) {
       //...
      }
   }
?>
```

KLASSEN

Keyword "var" voor de variabelen die bij de klasse horen (properties/datamemebers)

```
<?php
   class MijnKlasse {
      var $mijnVariabele1;
      var $mijnVariabele2;
      function mijnMethode($arg1, $arg2) {
       //...
      }
   }
?>
```

erasmus HOGESCHOOL BRUSSEL

KLASSEN

Properties met dollarteken en in camelCase

```
<?php
   class MijnKlasse {
      var $mijnVariabele1;
      var $mijnVariabele2;
      function mijnMethode($arg1, $arg2) {
       //...
      }
   }
?>
```

KLASSEN

Methodenamen in camelCasae

```
<?php
   class MijnKlasse {
      var $mijnVariabele1;
      var $mijnVariabele2;
      function mijnMethode($arg1, $arg2) {
       //...
      }
   }
?>
```

KLASSEN

Constanten in klasses declareert men met sleutelwoord "const" !

```
<?php
   class MijnKlasse {
      var $mijnVariabele1;
      var $mijnVariabele2;
      const PI = 3.14;

      function mijnMethode($arg1, $arg2) {
       //...
      }
   }
?>
```

erasmus HOGESCHOOL BRUSSEL

KLASSEN

Namen van constanten worden in HOOFDLETTERS geschreven, en zonder $

```
<?php
   class MijnKlasse {
      var $mijnVariabele1;
      var $mijnVariabele2;
      const PI = 3.14;

      function mijnMethode($arg1, $arg2) {
       //...
      }
   }
?>
```

erasmus HOGESCHOOL BRUSSEL

KLASSEN

Naamgeving

- Gebruik beschrijvende namen, lengte is van ondergeschikt belang.
- Hoofdletterregels op vorige slides zijn indicatief, maar gebruik ze!

KLASSEN

Bestanden

- Plaats klassen altijd in een apart bestand
- Geef bestand zelfde naam als klasse + .php
  - Ordelijker, beter leesbaar
  - Uitbreidbaarder
  - Betere projectstructuur

KLASSEN

Instantiëren

- Gebruik "new" keyword

```
<?php
      $object = new MijnKlasse();
?>
```

KLASSEN

Methodes/properties aanroepen

- Maak gebruik van een ->

```
<?php
   $object = new MijnKlasse();
   $object->mijnMethode(12,53);
   $object->mijnVariabele2 = 57;
?>
```

KLASSEN

Methodes/properties aanroepen binnen een klasse

```
<?php
   class MijnKlasse {
      var $mijnVariabele1;
      var $mijnVariabele2;

      function mijnMethode($arg1, $arg2) {
         $this->mijnVariabele1 = $arg1;
         $this->mijnVariabele2 = $arg2;
      }
   }
?>
```

KLASSEN

Constructoren

- = een speciale methode die wordt aangeroepen bij het aanmaken van een instantie
- = optioneel
- Implementatie in `__construct`

KLASSEN

Constructoren

```
<?php
class MijnKlasse {
   var $mijnVariabele1;
   var $mijnVariabele2;

   function __construct($arg1, $arg2) {
      $this->mijnVariabele1 = $arg1;
      $this->mijnVariabele2 = $arg2;
   }
}
?>
```

KLASSEN

Destructoren

- Optioneel
- Zo kan men bij het verwijderen van een object automatisch een aantal acties uitvoeren
- Implementatie in `__destruct`

KLASSEN

Destructoren

```
<?php
   class MijnKlasse {
      var $mijnVariabele1;
      var $mijnVariabele2;
      function __destruct() {
       //implementatie
      }
   }
?>
```

KLASSEN

Objecten verwijderen

- Om een object te verwijderen, en dus de implementatie van de destructor te gebruiken, kent men de waarde null toe aan een variabele die verwijst naar een object.

```
<?php
   //constructor wordt aangeroepen
   $obj = new MijnKlasse(12,53);

   //destructor wordt aangeroepen
   $obj = null;
?>
```

KLASSEN

Overerving

```
<?php
   class ParentKlasse {
      $var1;

      function display() {
       echo $this->var1;
      }
   }
?>
```

KLASSEN

Overerving

Gebruik keyword "extends" om naar de superklasse te verwijzen

```
<?php
   class MijnKlasse extends ParentKlasse {
      //...
   }
?>
```

erasmus HOGESCHOOL BRUSSEL

KLASSEN

Overschrijving

- De implementatie van een methode van de subklasse zal voorrang hebben op de implementatie van de superklasse

KLASSEN

Overschrijving

```
<?php
   class ParentKlasse {
      $var1;

      function display() {
            echo $this->var1;
      }
   }
?>
```

KLASSEN

Overschrijving

```
<?php
   class MijnKlasse extends ParentKlasse {
      function display() {
          echo "Waarde van var1: " . $this->var1;
      }
   }
?>
```

KLASSEN

Parent leden aanroepen

- Gebruik het sleutelwoord "parent::" om te verwijzen naar de properties en/of methodes van de parent klasse.
- Hoeft niet als eerste commando.
- Op analoge manier kan je de constructor van de parentklasse aanroepen
  - `parent::__construct()`

KLASSEN

Parent leden aanroepen

```
<?php
   class ParentKlasse {
      $var1;
      function display() {
       echo $this->var1;
      }
   }
?>
```

KLASSEN

Parent leden aanroepen

```
<?php
   class MijnKlasse extends ParentKlasse {
      function display() {
         echo "<p>";
         parent::display();
         echo "</p>";
      }
   }
?>
```

KLASSEN

Overschrijving verhinderen

- Je kan ervoor zorgen dat methodes niet verder overschreven kunnen worden in childklasses
- Gebruik hiervoor het keyword "**final**"

KLASSEN

Overschrijving verhinderen

```
<?php
   class ParentKlasse {
      $var1;
      final function display() {
       echo $this->var1;
      }
   }
?>
```

KLASSEN

Overschrijving verhinderen

```
<?php
   class MijnKlasse extends ParentKlasse {
      function display() {
       echo "Nieuwe implementatie";
      }
   }
?>
```

MAG NIET !!! GEEFT FOUT !!!

KLASSEN

Meervoudige overerving

- Het overerven van meerdere klasses wordt niet ondersteund in PHP
- Je kan wel child/grandchild klasses of parent/grandparent klasses hebben.

ENCAPSULATIE

- Je kan de zichtbaarheid van de leden van een klasse beperken
- Je gebruikt hiervoor access modifiers
- Access modifiers worden gezet vooraan de definities van properties en methodes.
- Klasse
  - Verbergt zijn complexiteit
  - Maakt implementaties/waarden niet rechtstreeks toegankelijk
  - Biedt niet meer functionaliteit aan dan noodzakelijk
  - Laat niet meer over zichzelf weten dan noodzakelijk
  - => verminderd programmeerfouten

ENCAPSULATIE

Access modifiers

- Private:
  - De toegang tot de leden aangeduid met private is beperkt tot de leden binnen de klasse zelf
- Protected:
  - De toegang tot de leden aangeduid met protected is beperkt tot de leden binnen de klasse zelf, en de leden van overervende klasses.

ENCAPSULATIE

Access modifiers

- Public:
  - De leden aangeduid met public zijn vanaf overal toegankelijk.
  - Dit is de standaard. Als je geen access modifier bij een property zet, is ze public.

ENCAPSULATIE

Access modifiers


```
<?php
   class MijnKlasse {
      private $mijnVariabele1;
      private $mijnVariabele2;
      protected function mijnMethode($arg1,
      $arg2) {
         $this->mijnVariabele1 = $arg1;
         $this->mijnVariabele2 = $arg2;
      }
   }
?>
```

erasmus HOGESCHOOL BRUSSEL

ENCAPSULATIE

Access modifiers

Bij specificatie access modifier valt "var" keyword weg bij declaratie datamembers

```
<?php
   class MijnKlasse {
      private $mijnVariabele1;
      private $mijnVariabele2;
      protected function mijnMethode($arg1,
      $arg2) {
         $this->mijnVariabele1 = $arg1;
         $this->mijnVariabele2 = $arg2;
      }
   }
?>
```

erasmus HOGESCHOOL BRUSSEL

ENCAPSULATIE

Access modifiers

Proteced methode

```
<?php
   class MijnKlasse {
      private $mijnVariabele1;
      private $mijnVariabele2;
      protected function mijnMethode($arg1,
      $arg2) {
         $this->mijnVariabele1 = $arg1;
         $this->mijnVariabele2 = $arg2;
      }
   }
?>
```

erasmus HOGESCHOOL BRUSSEL

ABSTRACTE KLASSEN

- Van deze klasses kan geen instantie worden gemaakt. Er kan enkel van worden overgeërfd.
- Niet alle methodes hoeven een implementatie te hebben. Mogelijks sommigen wel.
- Aangeduid door het keyword "abstract"

ABSTRACTE KLASSEN

```
<?php
   abstract class MijnAbstracteKlasse {
      protected function mijnMethode() {
       //implementatie
      }
   }
?>
```

ABSTRACTE KLASSEN

```
<?php
   abstract class MijnAbstracteKlasse {
      protected function mijnMethode() {
       //implementatie
      }
   }
?>
```

STATISCHE PROPERTIES EN METHODES

- Statische leden worden niet aan een object gebonden, maar aan de klassedefinitie zelf
- Statische leden kunnen enkel op de klasse aangeroepen worden, en niet op de leden zelf
- In statische operaties kan men enkel statische en constante leden aanroepen.
- In gewone operaties kan men geen statische leden aanroepen.

STATISCHE PROPERTIES EN METHODES

- Statische leden worden aangeduid met het keyword `static`
- Statische leden worden aangeroepen met "`self::`"

STATISCHE PROPERTIES EN METHODES

```
<?php
class MijnKlasse {
    protected static $var1;
    private $var2;
    const PI = 3.14;

    static function statischeMethode($arg1) {
        self::$var1 = $arg1 * self::PI;
    }
    function gewoneMethode($arg1) {
        $this->var2 = $arg1 * $this->PI;
    }
}

MijnKlasse::statischeMethode(24);

$obj = new MijnKlasse();
$obj->gewoneMethode(16);
?>
```

erasmus HOGESCHOOL BRUSSEL

SERIALISATIE

- Serialisatie = je kan objecten converteren naar een string bytes.
  - functie serialize()
- Deserialisatie = je kan de oorspronkelijke objecten opnieuw reproduceren uit de overeenkomstige strings die je bij het serialiseren hebt geproduceert
  - functie unserialize()

SERIALISATIE

- Nodig wanneer we een object willen bewaren over verschillende scripts
- Geserialiseerde objecten kunnen bewaard worden in
  - Sessies
  - (Tekst)bestanden
  - Database
  - …
- Zonder serialisatie zouden de objecten niet meer bruikbaar zijn wanneer ze worden opgevraagd

SERIALISATIE

```
<?php
   require_once("MijnKlasse.php");

   $obj = new MijnKlasse();
   $geserialObj= serialize($obj);
?>
```

SERIALISATIE

```php
<?php
   require_once("MijnKlasse.php");

   $obj = unserialize($geserialObj);
   $obj->display();
?>
```

SERIALISATIE

- De code die unserialize() aanroept moet ook toegang hebben tot de definitie van de klasse van het object dat je unserializet

- Objecten kunnen enkel uit een geserialiseerde string worden gehaald als de string achteraf niet is aangepast geweest. Het aanpassen van de string kan per ongeluk gebeurt zijn bij het opslaan in een database.
  - eventueel worden slashes toegevoegd of verwijderd