PHP-1

1 Gegevens doorsturen met GET en POST

Leerdoelen

- Je weet wat `GET` en `POST` zijn.
- Je kunt gegevens doorsturen van de ene pagina naar de andere.
- Je kunt de gegevens gebruiken in een tweede PHP-bestand.

Uitleg

Bij een formulier kies je of je `GET` of `POST` gebruikt om de ingevulde gegevens naar de server te sturen:

Methode	Kenmerk	Voorbeeld
`GET`	Toont gegevens in de URL	`pagina.php?naam=Ali`
`POST`	Stuurt gegevens "verborgen" via de browser	Geen zichtbare URL-parameters

Opdracht 1 – formulier.html

Maak een bestand aan met de naam `formulier.html` en zet hier de volgende code in:

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

  <h2>Wat is je naam?</h2>
```

```
  <form action="begroeting.php" method="get">
    <label for="naam">Naam:</label><br>
    <input type="text" id="naam" name="naam"><br><br>
    <input type="submit" value="Verstuur">
  </form>

</body>
</html>
```

> Check, `method="get"` op regel 7

Opdracht 2 – begroeting.php

Maak een nieuw bestand met de naam `begroeting.php` en zet hierin:

```
<?php
$naam = $_GET["naam"];
echo "<h1>Hallo $naam!</h1>";
?>
```

Check of alles werkt.

Als dat zo is verander dan regel 7 in het formulier naar:

```
  <form action="begroeting2.php" method="post">
```

Pas nu de code in begroeting.php aan, zodat het formulier goed werkt.

Reflectie

- Wat is het verschil tussen POST en GET?
- Wat doet `action="begreting2.php"` in het formulier?
- Wanneer zou je liever `POST` gebruiken dan `GET`? Leg uit!

Inleveren

- Beantwoord de drie vragen uit de reflectie en lever die in (.txt of .pdf bestand).

2 Include en Require

Leerdoelen

- Je weet wat `include` en `require` doen in PHP.
- Je kunt een header of footer inladen met `include`.
- Je herkent het verschil tussen `include` en `require`.

Uitleg

Vaak gebruik je op meerdere pagina's dezelfde stukjes HTML, zoals een menu of een footer. Je kunt dat opslaan in een apart bestand en invoegen met `include` of `require`.

- `include "bestand.php"`: probeert het bestand in te laden. Als dat niet lukt, gaat de pagina wel gewoon verder.
- `require "bestand.php"`: probeert het bestand in te laden. Als dat niet lukt, stopt de pagina met een fout.

Voorbeeld:

Stel, je maakt een bestand `header.php` met daarin een simpel menu:

```
<!-- header.php -->
<header>
  <h1>Mijn Website</h1>
  <nav>
    <a href="index.php">Home</a> |
    <a href="info.php">Info</a>
  </nav>
</header>
```

En dan gebruik je `include` in een andere pagina:

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

<h2>Welkom op de homepagina</h2>
<p>Dit is de inhoud van index.php</p>
```

□□ Opdracht – Header en footer maken

1. Maak een bestand `header.php` met een kop en menu zoals hierboven.
2. Maak een bestand `footer.php` met daarin bijvoorbeeld:
   `<footer>© 2025 Mijn Website</footer>`
3. Maak een bestand `index.php` met bovenin een `include("header.php")` en onderin een `include("footer.php")`.

□□ Reflectie

- Waarom is het handig om met `include` te werken?
- Wanneer zou je liever `require` gebruiken?
- Wat zou er gebeuren als je honderd pagina's hebt zonder `include` te gebruiken? In welk geval zou dat onhandig zijn?

□□ Inleveren

- Beantwoord de drie ivragen n eigen woorden uit de reflectie en lever die in (.txt of .pdf bestand).

3 Arrays en Loops

□□eerdoelen

- Je weet wat een array is in PHP.
- Je kunt een array maken met meerdere waarden.
- Je kunt een `foreach`-loop gebruiken om alle items te tonen.

□□ Uitleg

Een **array** is een soort lijstje waarin je meerdere dingen kunt bewaren, zoals hobby’s of namen.

Een array is een variabele die meer dan één item (waarde) kan bevatten.

Je gebruikt een `foreach`-loop om elk item uit de array één voor één te gebruiken.

Voorbeeld:

```
<?php
$hobbies = ["voetbal", "lezen", "gamen"];

foreach ($hobbies as $hobby) {
  echo "<p>Mijn hobby is: $hobby</p>";
}
?>
```

Resultaat:
Mijn hobby is: voetbal
Mijn hobby is: lezen
Mijn hobby is: gamen

□□ Opdracht – Eigen array

Maak een bestand `lijst.php` en vul dit met een array van 5 dingen die jij leuk vindt (hobby’s, favoriete eten, games…).

- Gebruik een `foreach`-loop om ze allemaal op het scherm te tonen.
- Geef elk item weer in een eigen paragraaf (<p>).

□□ Uitleg

Er is ook een ander soort loop; een for-loop.

Gebruik een `for`-loop in plaats van `foreach`:

```
<?php
$games = ["Minecraft", "FIFA", "Fortnite", "Roblox"];
for ($i = 0; $i < count($games); $i++) {
```

```
  echo "<p>" . $i . " Favoriete game: " . $games[$i] . "</p>";
}
?>
```

De waarde van $games[0] is dus "Minecraft", $games[1] = "FIFA", etc, etc.

> Het eerste element in een array heeft een index 0!

□□ Opdracht - deel 2

Plaats in de PHP-code (lijst.php) een eigen voorbeeld waarbij je een `for-loop` gebruikt.

□□ Reflectie

- Wat is het voordeel van een array ten opzichte van losse variabelen?
- Wat is het verschil tussen `for` en `foreach`?
- Wanneer zou je liever een `for`-loop gebruiken?

□□ Inleveren

- Lever een screenshot in van jouw `lijst.php` met minimaal 5 items in een array, getoond in de browser.
- Laat zien dat je zowel `foreach` als `for` gebruikt hebt.

4 Quiz met arrays en loops

□□eerdoelen

- Je kunt een quiz maken met meerdere vragen in een formulier.
- Je kunt de juiste antwoorden opslaan in een array.
- Je kunt een loop gebruiken om automatisch te controleren hoeveel vragen goed zijn.

□□ Uitleg

Met een array kun je makkelijk meerdere juiste antwoorden opslaan. Met een loop kun je daar automatisch doorheen lopen om te controleren hoeveel antwoorden goed zijn ingevuld.

□□ Opdracht 1 – quiz.html

Maak een nieuw bestand `quiz.html` met dit formulier:

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

 <h2>Mini-quiz</h2>

 <form action="uitslag.php" method="get">

  <p>1. Wat is de hoofdstad van Nederland?</p>
  <input type="radio" name="vraag1" value="A"> A. Rotterdam<br>
  <input type="radio" name="vraag1" value="B"> B. Amsterdam<br>
  <input type="radio" name="vraag1" value="C"> C. Utrecht<br>

  <p>2. Wat betekent HTML?</p>
  <input type="radio" name="vraag2" value="A"> A. Hyperlink Text Markup Language<br>
  <input type="radio" name="vraag2" value="B"> B. HyperText Markup Language<br>
  <input type="radio" name="vraag2" value="C"> C. Home Tool Markup Language<br>

  <p>3. Wat doet PHP?</p>
  <input type="radio" name="vraag3" value="A"> A. Styling toevoegen aan websites<br>
  <input type="radio" name="vraag3" value="B"> B. Informatie opslaan en berekenen op de server<br>
  <input type="radio" name="vraag3" value="C"> C. Tekst zichtbaar maken in Word<br>

  <br><input type="submit" value="Verzend antwoorden">

 </form>

</body>
</html>
```

□□ Opdracht 2 – uitslag.php met array en loop

Maak een nieuw bestand `uitslag.php` en gebruik deze code:

```
<?php
$goed = 0;

// juiste antwoorden in een array
$juisteAntwoorden = [
  "vraag1" => "B",
  "vraag2" => "B",
  "vraag3" => "B"
];

$goed = 0;

if (isset($_GET['vraag1']) && $_GET['vraag1'] == 'a') {
  $goed++;
}
if (isset($_GET['vraag2']) && $_GET['vraag2'] == 'c') {
  $goed++;
}
if (isset($_GET['vraag3']) && $_GET['vraag3'] == 'b') {
  $goed++;
}

echo "<h2>Je hebt $goed van de 3 vragen goed!</h2>";
?>
```

Opdracht

Bereid de quiz uit met 2 zelf bedachte vragen.

Zet de code in uitslag.php om, waarbij je de score berekend met behulp van een loop.

Vraag AI om hulp en vraag AI om uit te leggen hoe de loop werkt.

□□ Reflectie

- In het HTML-formulier staan telkens drie <input> elementen met dezelfde naam. Leg uit waarom dat zo is.
- Wat is het voordeel van een array gebruiken voor de juiste antwoorden?
- Wat gebeurt er als je geen antwoord op een vraag invult?
- Wat doet isset() op regel 13, 16 en 19?
- Hoe zou je de quiz uitbreiden naar 5 of 10 vragen met zo min mogelijk extra code?

□□ Inleveren

- `quiz.html` en `uitslag.php`

5 Functies in PHP

□□eerdoelen

- Je weet wat een functie is in PHP.
- Je kunt zelf een functie schrijven en aanroepen.
- Je kunt gegevens meegeven aan een functie (parameters).

□□ Uitleg

Een functie is een blokje code dat je een naam geeft en later opnieuw kunt gebruiken. Zo hoef je niet telkens dezelfde code opnieuw te schrijven.

Je kunt informatie aan een functie meegeven (dat noem je een **parameter**) en je kunt iets teruggeven (dat noem je een **return**).

Voorbeeld:

```
<?php
function begroet($naam) {
```

```
  echo "Hallo, $naam!<br>";
}

begroet("Fatima");
begroet("Ali");
?>
```

□□ Opdracht 1 – Maak je eigen begroetfunctie

1. Maak een nieuw bestand `functie.php`
2. Maak daarin een functie `begroet` die één parameter accepteert: `$naam`
3. Laat de functie een boodschap tonen zoals “Hoi, [naam], welkom terug!”
4. Roep de functie minstens 3 keer aan met verschillende namen.

□□ Opdracht 2 – Maak een rekentool met functie

1. Maak een functie `kortingBerekenen` die twee getallen als parameter gebruikt: `$bedrag` en `$percentage`
2. Laat de functie het bedrag na korting `returnen`
3. Laat het resultaat op het scherm zien met `echo`

Voorbeeldcode:

```
<?php
function kortingBerekenen($bedrag, $korting) {
  $nieuwBedrag = $bedrag - ($bedrag * ($korting / 100));
  return $nieuwBedrag;
}

echo "<p>Je betaalt nu: €" . kortingBerekenen(100, 25) . "</p>";
?>
```

Reflectie

- Waarom is het handig om functies te gebruiken?
- Wat gebeurt er als je geen parameter meegeeft aan een functie die er wel één verwacht?
- Waar zou je functies nog meer voor kunnen gebruiken?

Inleveren

- `functie.php` met ten minste 1 begroetfunctie én 1 rekentool.
  Je moet ten minste 2 functies gebruiken met parameters, waarvan 1 ook een `return` gebruikt.

6 Datum en Tijd in PHP

Leerdoelen

- Je weet hoe je de huidige datum en tijd kunt tonen met PHP.
- Je kunt berekenen hoeveel dagen of jaren geleden iets was.
- Je kunt de leeftijd berekenen op basis van een geboortedatum.

Uitleg

PHP heeft functies om met datum en tijd te werken. De belangrijkste is `date()`, waarmee je de huidige tijd kunt weergeven in verschillende formaten.

- `date("Y")` → jaar (bijv. 2025)
- `date("d-m-Y")` → dag-maand-jaar (bijv. 04-06-2025)
- `date("H:i")` → tijd in uren:minuten (bijv. 14:36)

Voorbeeld:

```php
<?php
echo "Vandaag is het: " . date("d-m-Y") . "<br>";
echo "Het is nu: " . date("H:i") . " uur.";
?>
```

□□ Opdracht 1 – Toon de datum en tijd

1. Maak een bestand `datum.php`
2. Laat daarin zien:
   - Welke dag het vandaag is
   - De datum in formaat `dd-mm-yyyy`
   - De tijd in uren en minuten

□□ Opdracht 2 – Leeftijd berekenen

Bereken iemands leeftijd op basis van geboortedatum:

```php
<?php
$geboortedatum = "2007-03-15";
$geboortedag = new DateTime($geboortedatum);
$vandaag = new DateTime();

$verschil = $vandaag->diff($geboortedag);
echo "Je bent " . $verschil->y . " jaar oud.";
?>
```

Uitleg: PHP kan automatisch het aantal jaren berekenen tussen twee datums met `diff()`.

□□ Reflectie

- Wat is het verschil tussen `date()` en `new DateTime()`?
- Waarom moet je bij het berekenen van leeftijd een `diff()` gebruiken?
- Wat zou er gebeuren als je geboortedatum in een verkeerd formaat schrijft?

□□ Inleveren

- `datum.php` met de juiste datum, tijd en berekende leeftijd.
  Je gebruikt minstens één functie met datum en één met tijd, en `diff()` voor de leeftijd.

7 Sessies en Inloggen met PHP

□□eerdoelen

- Je weet wat een sessie is in PHP.
- Je kunt een eenvoudige inlogpagina maken.
- Je kunt een gebruiker herkennen op een andere pagina.

□□ Uitleg

Een **sessie** in PHP is een manier om informatie te onthouden zolang de gebruiker je website bezoekt. Bijvoorbeeld of iemand is ingelogd of wat zijn/haar gebruikersnaam is.

Je start een sessie met:

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

□□ Opdracht 1 – inloggen.html

Maak een bestand `inloggen.html` met een formulier waarin je gebruikersnaam invoert:

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

  <h2>Inloggen</h2>
```

```
  <form action="login.php" method="post">
    <label for="naam">Naam:</label><br>
    <input type="text" name="naam" id="naam" required><br><br>
    <input type="submit" value="Inloggen">
  </form>

</body>
</html>
```

□□ Opdracht 2 – login.php

Maak een bestand `login.php` dat de naam opslaat in een sessie en doorstuurt:

```
<?php
session_start();
$_SESSION["gebruiker"] = $_POST["naam"];
header("Location: welkom.php");
exit;
?>
```

□□ Opdracht 3 – welkom.php

Maak een bestand `welkom.php` dat de gebruiker begroet:

```
<?php
session_start();
if (isset($_SESSION["gebruiker"])) {
  echo "<h1>Welkom, " . $_SESSION["gebruiker"] . "!</h1>";
  echo '<p><a href="uitloggen.php">Uitloggen</a></p>';
} else {
  echo "<p>Je bent niet ingelogd.</p>";
}
?>
```

□□ Opdracht 4 – uitloggen.php

```php
<?php
session_start();
session_destroy();
header("Location: inloggen.html");
exit;
?>
```

□□ Reflectie

- Waarom moet je altijd `session_start()` gebruiken bovenaan?
- Wat gebeurt er als je probeert `welkom.php` te openen zonder in te loggen?
- Wat zou je kunnen uitbreiden, bijvoorbeeld met wachtwoordcontrole?

□□ Inleveren

- Screenshots van `inloggen.html`, `welkom.php` met jouw naam, en `uitloggen.php` na het uitloggen.
- Je moet gebruik maken van sessies en de naam van de gebruiker correct kunnen tonen op meerdere pagina’s.

8 Inloggen met wachtwoordcontrole

□□eerdoelen

- Je maakt een formulier met gebruikersnaam én wachtwoord.
- Je controleert de invoer in PHP.
- Je begrijpt waarom `$_GET` niet veilig is voor wachtwoorden.
- Je leert werken met een associatieve array.

Uitleg

Een loginformulier stuurt gebruikersnaam en wachtwoord naar PHP. In deze les gebruiken we eerst `$_GET` om te laten zien waarom dat niet veilig is – je ziet het wachtwoord in de URL.

Opdracht 1 – login.html

Maak een bestand `login.html`:

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

  <h2>Loginformulier</h2>

  <form action="controle.php" method="get">
    <label>Gebruikersnaam:</label><br>
    <input type="text" name="gebruiker" required><br><br>

    <label>Wachtwoord:</label><br>
    <input type="password" name="wachtwoord" required><br><br>

    <input type="submit" value="Log in">
  </form>

</body>
</html>
```

Opdracht 2 – controle.php

Maak een bestand `controle.php` waarin je controleert of de gebruiker mag inloggen:

```
<?php
$gebruiker = $_GET["gebruiker"];
$wachtwoord = $_GET["wachtwoord"];

if ($gebruiker == "admin" && $wachtwoord == "geheim123") {
  echo "<h2>Welkom, $gebruiker!</h2>";
```

```
} else {
  echo "<p>Foutieve inloggegevens. Probeer opnieuw.</p>";
}
?>
```

Let op:

Als je dit formulier verstuurt, zie je het wachtwoord in de URL. Dat is niet veilig!

□□ Opdracht 3 – Verbeter met POST

- Pas het formulier aan zodat het `method="post"` gebruikt
- Pas `controle.php` aan zodat het `$_POST` gebruikt
- Test: zie je het wachtwoord nog in de URL?

□□ Uitleg – Associatieve array

Tot nu toe heb je gewerkt met lijsten zoals:

```
$hobby’s = ["voetbal", "gamen", "lezen"];
```

Dit is een **indexed array**: de computer onthoudt zelf de volgorde (index 0, 1, 2).

Een **associatieve array** heeft zelfgekozen namen als index (zogenaamde "keys"):

```
$gebruikers = [
  "admin" => "geheim123",
  "student" => "welkom01"
];
```

Je kunt dan bijvoorbeeld zeggen:

```
echo $gebruikers["admin"]; // toont: geheim123
```

Heel handig voor wachtwoorden of gebruikerslijsten!

□□ Extra opdracht – Meerdere gebruikers

Breid `controle.php` uit met een associatieve array van toegestane gebruikers en wachtwoorden:

```php
<?php
$gebruikers = [
  "admin" => "geheim123",
  "student" => "welkom01",
  "docent" => "phprules"
];

$gebruiker = $_POST["gebruiker"];
$wachtwoord = $_POST["wachtwoord"];

if (isset($gebruikers[$gebruiker]) && $gebruikers[$gebruiker] == $wachtwoord) {
  echo "<h2>Welkom, $gebruiker!</h2>";
} else {
  echo "<p>Inloggen mislukt.</p>";
}
?>
```

Voeg zelf nog twee gebruikers toe: één voor jouw zelf (dus je eigen voornaam) en één voor een klasgenoot.

Reflectie

- Wat is het verschil tussen een indexed array en een associatieve array?
- Waarom is het veiliger om `$_POST` te gebruiken voor wachtwoorden?
- Wat zou je doen om inloggen met een wachtwoord nog veiliger te maken?

Inleveren

- controle.php
- Een .txt. of .pdf bestand met de antwoorden op de drie reflectievragen.

Revision #6
Created 4 June 2025 19:31:20 by Max
Updated 6 June 2025 15:13:04 by Max