

## WORKSHOP – Introductie tot Kenmerk-gebaseerd Filteren in PHP

In dit project leer je hoe je een complexe dataset van personages kunt uitlezen en filteren. We werken met Arrays, Loops en Logische Voorwaarden.

### Wat zijn Arrays? (Lijstjes met data)

Arrays zijn variabelen waarin je meerdere waardes kunt opslaan.

#### Array

Een standaard array is een genummerde lijst. De computer begint altijd met tellen bij 0.

```
$names = ["Alex", "Bernard", "Susan"];  
echo $names[0]; // Output: Alex
```

#### Associatieve Array (Labels)

Vaak is het handiger om zelf labels (sleutels of keys) te gebruiken in plaats van nummers.

```
$person = [  
    "name" => "Tom",  
    "is_man" => true,  
    "glasses" => true  
];  
echo $person["name"]; // Output: Tom
```

## De Project Structuur & var\_dump (Onze Data)

In dit project is de data complexer: we hebben een grote lijst waarin de naam van het personage de sleutel is. De waarde is vervolgens weer een array met daarin de features.

```
$dataset = [  
  "Bernard" => [  
    "features" => ["man" => 1, "bald" => 0, "glasses" => 1]  
  ],  
  "Susan" => [  
    "features" => ["man" => 0, "bald" => 0, "glasses" => 1]  
  ]  
];
```

Hoe kun je zien wat hierin zit? Deze structuur is lastig te onthouden. Gebruik `var_dump()` om de inhoud van een variabele te inspecteren. Dit is je belangrijkste hulpmiddel! De `<pre>`-tags maken de output leesbaar.

```
echo '<pre>';  
var_dump($dataset);  
echo '</pre>';
```

## Loopen met foreach (Door de data heenlopen)

Om elke persoon in de dataset te kunnen bekijken, gebruiken we een foreach-loop. Omdat de naam van het personage de sleutel is, gebruiken we de => syntax.

```
// $name wordt de sleutel (bijv. "Bernard")
// $data wordt de waarde (de array met features)
foreach ($dataset as $name => $data) {
    echo "Dit is personage: " . $name . "<br>";
}
```

## Filteren met if (Voorwaarden stellen)

Nu we door de data kunnen lopen, kunnen we voorwaarden stellen om selecties te maken.

### Stap 1: Toegang krijgen tot een genest kenmerk

De eigenschappen zitten diep in de structuur. Om te zien of iemand een vrouw is, volg je het pad: \$data → features → woman.

### Stap 2: Een voorwaarde schrijven

```
foreach ($dataset as $name => $data) {
    // We kijken in $data, dan in 'features', dan naar 'woman'
    if ($data['features']['woman'] === 1) {
        echo $name . " is een vrouw.<br>";
    }
}
```

### Stap 3: Meerdere voorwaarden combineren met && (EN)

```
foreach ($dataset as $name => $data) {
    $f = $data['features']; // Korte variabele voor leesbaarheid

    if ($f['man'] === 1 && $f['glasses'] === 1) {
        echo $name . " is een man met een bril.<br>";
    }
}
```

## De juiste vergelijking kiezen: == VERSUS ===

Je hebt hierboven === gezien. Waarom is dat, en kun je ook == gebruiken? Onze JSON-data gebruikt 1 en 0. De meest exacte manier is strikt vergelijken met het getal 1.

```
// BESTE AANPAK (Strikt en Duidelijk):
if ($data['features']['woman'] === 1) {
    // ...
}

// WERKT OOK (Soepel, maar minder precies):
if ($data['features']['woman'] == true) {
    // ...
}
```

## Speciale functies

### Functie 1: Een willekeurige waarde kiezen (array\_rand)

Voor het spel moet de computer een willekeurig personage kiezen. array\_rand geeft je een willekeurige sleutel uit een array.

```
// Dit geeft je een willekeurige NAAM (bijv. "Susan")
$randomName = array_rand($dataset);
echo "Het gekozen personage is: " . $randomName;
```

### Functie 2: Sessies (Het geheugen van de server)

HTTP vergeet alles. Als de pagina herlaadt, weet de server niet meer wie het geheime personage was. PHP lost dit op met Sessies: een soort "rugzak" die op de server blijft voor de gebruiker.

```
// Stap 1: Rugzak openen
session_start();

// Stap 2: Iets in de rugzak stoppen
$_SESSION['secret_character'] = "Yunus";

// Stap 3: Iets uit de rugzak halen
if (isset($_SESSION['secret_character'])) {
    echo "Ik heb onthouden dat het " . $_SESSION['secret_character'] . " is.";
}
```

```
}  
// Stap 4: Iets specifiek uit je rugzak verwijderen  
Unset ($_SESSION['secret_character']);  
  
// Stap 5: De rugzak leegmaken (alles eruit)  
session_destroy();
```

## Formulieren als Associatieve Arrays (Vorbereiding op het Spel)

Eerder hebben we gewerkt met associatieve arrays, zoals:

```
$person = [  
    "name"    => "Tom",  
    "is_man"  => true,  
    "glasses" => true  
];
```

Hierbij is "name" de sleutel en "Tom" de waarde.

```
echo $person["name"];
```

Wanneer een gebruiker een formulier verstuurt, maakt PHP automatisch ook een associatieve array aan.

**Bij een formulier zoals:**

```
<form method="post">  
    <input type="text" name="name">  
    <input type="text" name="question">  
    <button type="submit">Verstuur</button>  
</form>
```

**maakt PHP intern deze array:**

```
$_POST = [  
    "name" => "waarde die de gebruiker intypt",  
    "question" => "waarde die de gebruiker intypt"  
];
```

De naam van het inputveld (name="") bepaalt de sleutel in de associatieve array.

```
if (isset($_POST["name"])) {
```

```
    echo "Gebruiker: " . $_POST["username"];  
}
```

Net zoals bij de dataset kun je altijd inspecteren wat erin zit:

```
echo "<pre>";  
  
var_dump($_POST);  
  
echo "</pre>";
```

In het spel worden vragen dus via een formulier gesteld en als associatieve array verwerkt.