Web Integration – 2TI

HC2 – Formulieren

erasmus
HOGESCHOOL BRUSSEL

Formulieren

Opbouw

Verzenden

Data uitlezen en gebruiken

Validatie

Formulieren en PHP

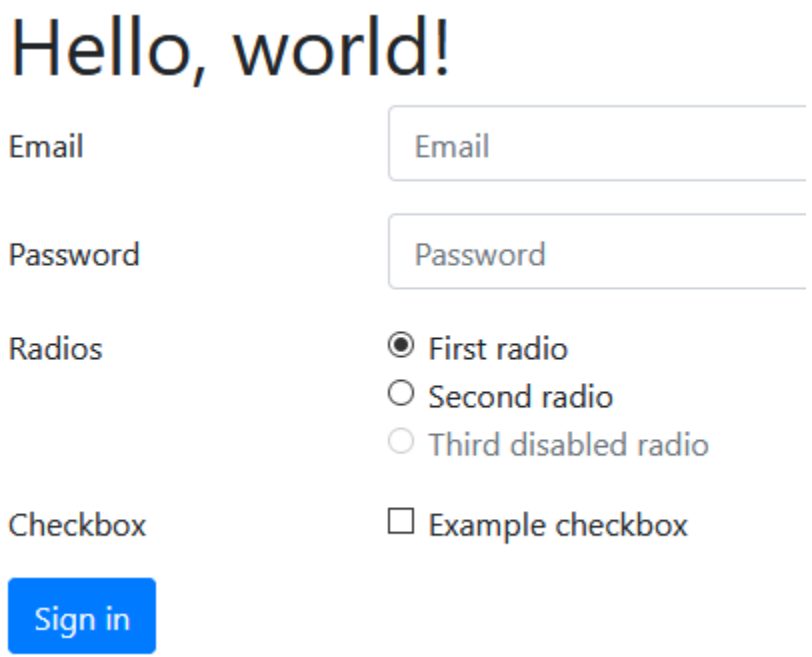
Formulieren laten toe om data naar de server te verzenden en te verwerken.

Na de nodige client side validatie (JavaScript), kunnen we deze verwerken in onze backend (PHP server script).

Data wordt gebruikt voor: files aanmaken, emails sturen, opslaan in database, … Alles om een website dynamisch te maken.

Opbouw van een formulier

- Reeks input velden
- Knop om de data in te dienen: Submit
- Invulvelden variatie
  - Email
  - Paswoord
  - Radio buttons
  - Files
  - …



Opbouw van een formulier

- Twee belangrijke parameters in HTML
  - **Action**: de locatie naar waar de data gestuurd wordt bij een submit
  - **Method**: hoe de data verstuurd wordt
    - GET vs POST

```
<form action="process.php" method="POST">
        <!-- Some fields -->
        <button type="submit" class="btn btn-primary">Sign in</button>
</form>
```

Velden in HTML

Om correcte velden op te bouwen, dien je op een aantal zaken te letten

Basis invoerveld met attributen

```
<input type="text" name="username" value="testuser"/>
```

- **type**: geeft het type invoerveld aan
- **name**: Geeft het invoerveld een naam. Wordt gebruikt om in PHP om de waarde van dit veld te koppelen.
- **value**: bevat op elk moment de waarde van het inputveld

Velden in HTML

Radiobuttons – type: “radio”

```
<input type="radio" name="courses" value="Web Essentials"/> Web Essentials
<input type="radio" name="courses" value="Web Advanced"/> Web Advanced
<input type="radio" name="courses" value="Web Integration"/> Web Integration
```

radiobuttons die bij elkaar horen dienen dezelfde ‘name’ te hebben.

Dankzij het attribuut value, kunnen de waarde opvragen in PHP.

Velden in HTML

Dropdown

```
<p>Je niveau in PHP is:</p>
<select name="niveau">
    <option value=“1">Beginner</option>
    <option value=“2">Matig</option>
    <option value=“3">Goed</option>
    <option value=“7">Godlike!</option>
</select>
```

Dankzij het attribuut value, kunnen de waarde opvragen in PHP. Niet zichtbaar voor de eindgebruiker.

Velden in HTML

Listbox

```
<p>Je niveau in PHP is:</p>
<select name="niveau“ size=“4”>
    <option value=“1">Beginner</option>
    <option value=“2">Matig</option>
    <option value=“3">Goed</option>
    <option value=“7">Godlike!</option>
</select>
```

Door toevoeging van attribuut ‘size’ krijg je een listbox: je ziet meerdere opties tegelijk.

Formulier verzenden

- Gegevens worden verzonden vanaf dat het submit event aangeroepen wordt
- Client side validatie kan dit tegengaan
  - e.preventDefault();
- Twee manieren
  - GET request
  - POST request

GET data

- Data opsomming wordt een ‘querystring’ genoemd
- Paren van data
  - http://www.ehb.be/opleidingen.php?opleiding=ti

- Meerdere paren of datasets mogelijk: &
  - http://www.ehb.be/opleidingen.php?opleiding=ti&jaar=2

- Lengte url beperkt: webserver afhankelijk. Meestal 255 tekens.

GET querystring

- Manueel door de eindgebruiker
  - Geen praktische use case
  - Makkelijk voor testen
- Developer voorbereiding achter link

```
<a href="opleidingen.php?opleiding=ti">Toegepaste Informatica</a>
```

- Html formulieren
  - Automatisch door de methode aan te passen

```
<form action="process.php" method="GET">
```

GET

- Voordelen
  - Voor iedereen zichtbaar: delen van link
  - Tracked in de geschiedenis van de browser
- Nadelen
  - Voor iedereen zichtbaar: vertrouwelijk
  - Beperkt in grootte
  - Niet voor alle types gegevens

GET data aanspreken

- Querystring wordt ontvangen op de server
- PHP parser gooit die automatisch om naar een bruikbare variabele:

$_GET

$_GET

- Globale variabele
- Niet declareren
- Ook wel een ‘Superglobal’ genoemd
  - Altijd beschikbaar
- Associatieve array: key/value pairs

http://www.ehb.be/opleidingen.php?opleiding=ti&jaar=2

```
<?php
    $opleiding = $_GET['opleiding']; // ti
    $jaar = $_GET['jaar']; // 2
?>
```

Post

- Gegevens worden in de body van de HTTP request bewaart.

```
POST /opleidingen/toegepasteinformatica.php HTTP/1.1
Host: localhost
Connection: close
User-Agent: Mozilla/5.0 (Macintosh; U; Intel MacOS 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
jaar=2&id=123456
```

Post

- Post request wordt verzonden met een formulier

```
<form action="process.php" method=“POST">
```

- Analoog met een GET request worden deze ook verzameld in een superglobal:

  $_POST

$_POST

- Globale variabele
- Niet declareren
- Waardes bevinden zich in de body van de request en worden automatisch opgezet door de php parser
- Ook wel een ‘Superglobal’ genoemd
  - Altijd beschikbaar
- Associatieve array: key/value pairs

http://www.ehb.be/opleidingen.php

```
<?php
    $opleiding = $_POST['opleiding']; // ti
    $jaar = $_POST['jaar']; // 2
?>
```

Post

- Voordelen
  - Laat veel meer gegevens toe
    - Door server bepaalt in config
  - Info niet (volledig) zichtbaar
  - Geschiedenis bewaart geen POST data
- Nadelen
  - Encryptie nodig om alle data volledig te verbergen

Requests afhandelen

We sturen de data naar een aparte PHP pagina waar de formuliergegevens verwerkt worden: Data opslag, emails,…

Dit brengt meer structuur en leesbaarheid. Maakt het makkelijk voor uit te breiden.

Enkel bij validatie is het een best practice om naar de eigen pagina te submitten

Requests afhandelen

Wanneer alle validaties slagen, kan je een apart script oproepen voor de effectieve verwerking.

Indien de validaties incorrect zijn, tonen we het formulier weer met een foutboodschap.

Requests afhandelen

```php
<?php
    if(formSubmitted()){
        //Het formulier is correct ingevuld
        //Gepost naar deze pagina
        //Validatie kan starten
        if(validatieFormulierCorrect()){
            // Correct ingevuld
            // verwerken gegevens
            // toon resultaten of redirect
        }else {
            //Niet correct ingevuld
            //Maak de foutboodschappen
            //Toon het formulier weer.
        }
    }
?>
```

Formulier submitted

We doen een controle op te kijken of het formulier gesubmit is naar deze pagina:
formSubmitted();

Enkel dan voeren we de validatie uit.

Controle op correcte submit: hidden fields in form!

Hidden input

Je kan aan een formulier een verborgen invoerveld meegeven.

```
<input type="hidden" name="loginform" value="10">
```

- Onzichtbaar voor eindgebruiker
- Value en name nodig voor PHP
- Wordt automatisch opgestuurd bij een submit

Formulier submitted

Nagaan of hidden field aanwezig is

- Aanwezig: submit van formulier. Data valideren
- Niet aanwezig: Geen data van formulier. Eerste bezoek op de pagina. Toon formulier

Gegevens valideren

Nagaan of alles correct ingevuld is.

Gebeurt meestal in combinatie met client-side controle (JavaScript en HTML).

Combinatie van controles is ideaal en gewenst.

Centraliseer je validaties van PHP in een aparte file. Zo breid je makkelijk uit.

Client-side validation

- Voordelen
  - Snelle feedback aan gebruiker
  - Geen HTTP requests
  - Minder belasting voor de server
- Nadelen
  - Cross site scripting
  - JavaScript kan uitgeschakeld worden
  - Resultaten kunnen door eindgebruiker gemanipuleerd worden

Server-side validation

- Voordelen
  - Technologie kan niet uitgeschakeld worden
  - Volledige controle
- Nadelen
  - Server resources worden gebruikt
  - Validatie fail => terug sturen naar eindgebruiker

File uploads

- Doordat de PHP parser op de server alles doorloopt, heb je de mogelijkheid om files op te slaan.
- Hiervoor dien je iets extra in je formulier toe te voegen om data door te sturen

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

File uploads - requirements

- De method van de form moet POST zijn
- Enctype moet ingesteld zijn op “multipart/form-data”
  - Specifieert het contenttype
  - Duid op binaire data. Nodig voor file upload
- Inputveld dient het type ‘file’ te hebben

File uploads - data

```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"];
?>
```

File uploads - Verwerken

- Files worden steeds op een tijdelijke locatie geplaatst
  - $_FILES[‘file’][‘tmp_name’]

- Gebruik de functie move_uploaded_file om de bestanden te verplaatsen naar een locatie naar keuze
  - https://www.php.net/manual/en/function.move-uploaded-file.php

File uploads - Verplaatsen

- De folder die je wilt gebruiken dient schrijfrechten op de server te hebben
- File uploads zijn een risico. Bouw genoeg controles in
- Grootte van de files kan je beperken via server settings of zelf controles uit te voeren