

# Webmediaproductie 5

## PHP - Werken met databanken

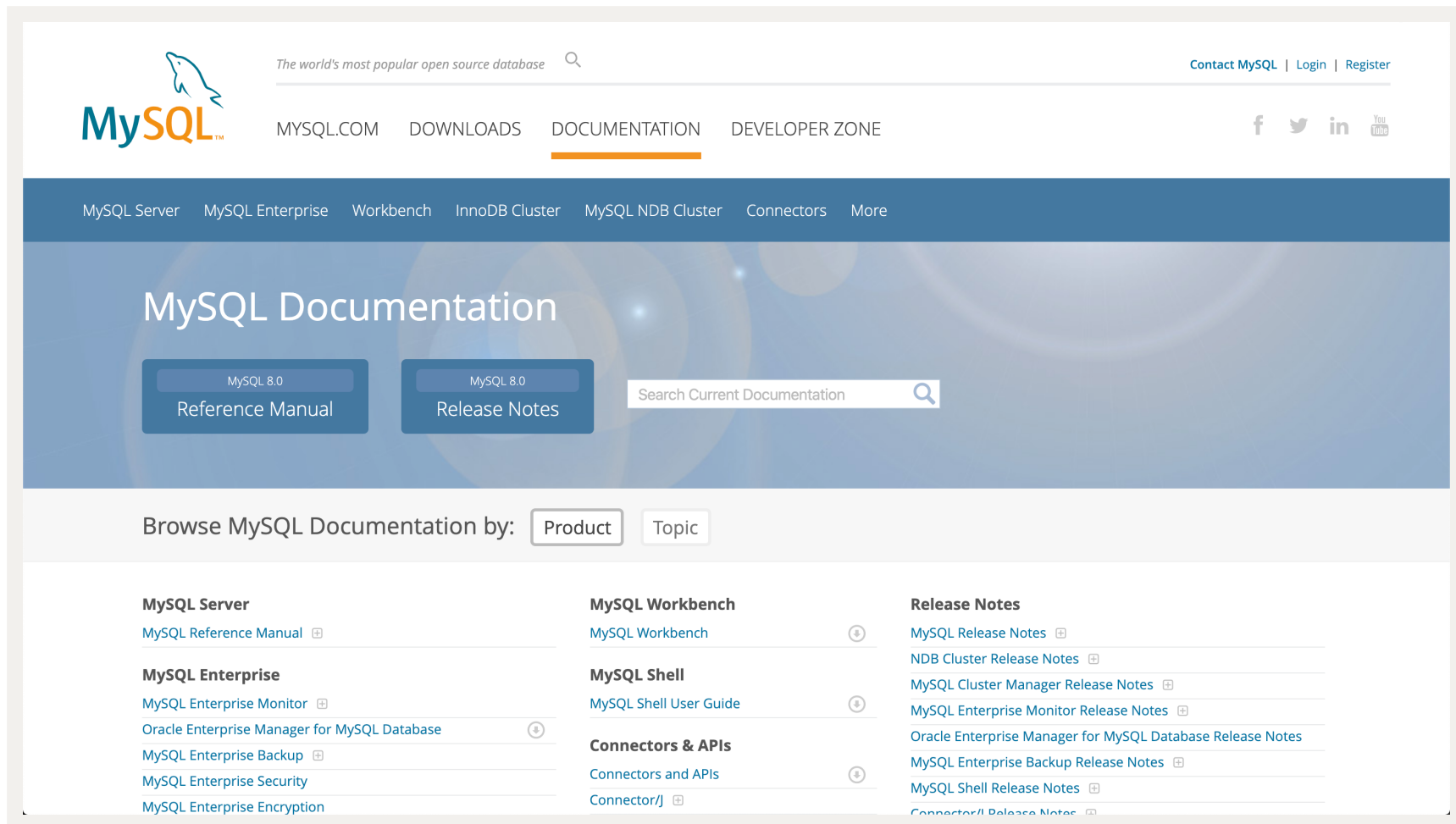
**Kristof Michiels**

MySQL

# Data Opslaan met MySQL

- De basics
- MySQL tabellen bewerken
- Database tabelstructuren
- PHP gebruiken om te werken met MySQL
- Een model-view-controller aanpak voor PHP

# MySQL



The screenshot shows the MySQL Documentation website. At the top, there is a navigation bar with the MySQL logo, the tagline "The world's most popular open source database", and links for "Contact MySQL", "Login", and "Register". Below this, there are links for "MYSQL.COM", "DOWNLOADS", "DOCUMENTATION" (which is highlighted with an orange underline), and "DEVELOPER ZONE". On the right side of the navigation bar, there are social media icons for Facebook, Twitter, LinkedIn, and YouTube.

Below the navigation bar, there is a dark blue header with links for "MySQL Server", "MySQL Enterprise", "Workbench", "InnoDB Cluster", "MySQL NDB Cluster", "Connectors", and "More".

The main content area has a blue background with the text "MySQL Documentation". Below this, there are two buttons: "MySQL 8.0 Reference Manual" and "MySQL 8.0 Release Notes". To the right of these buttons is a search bar labeled "Search Current Documentation".

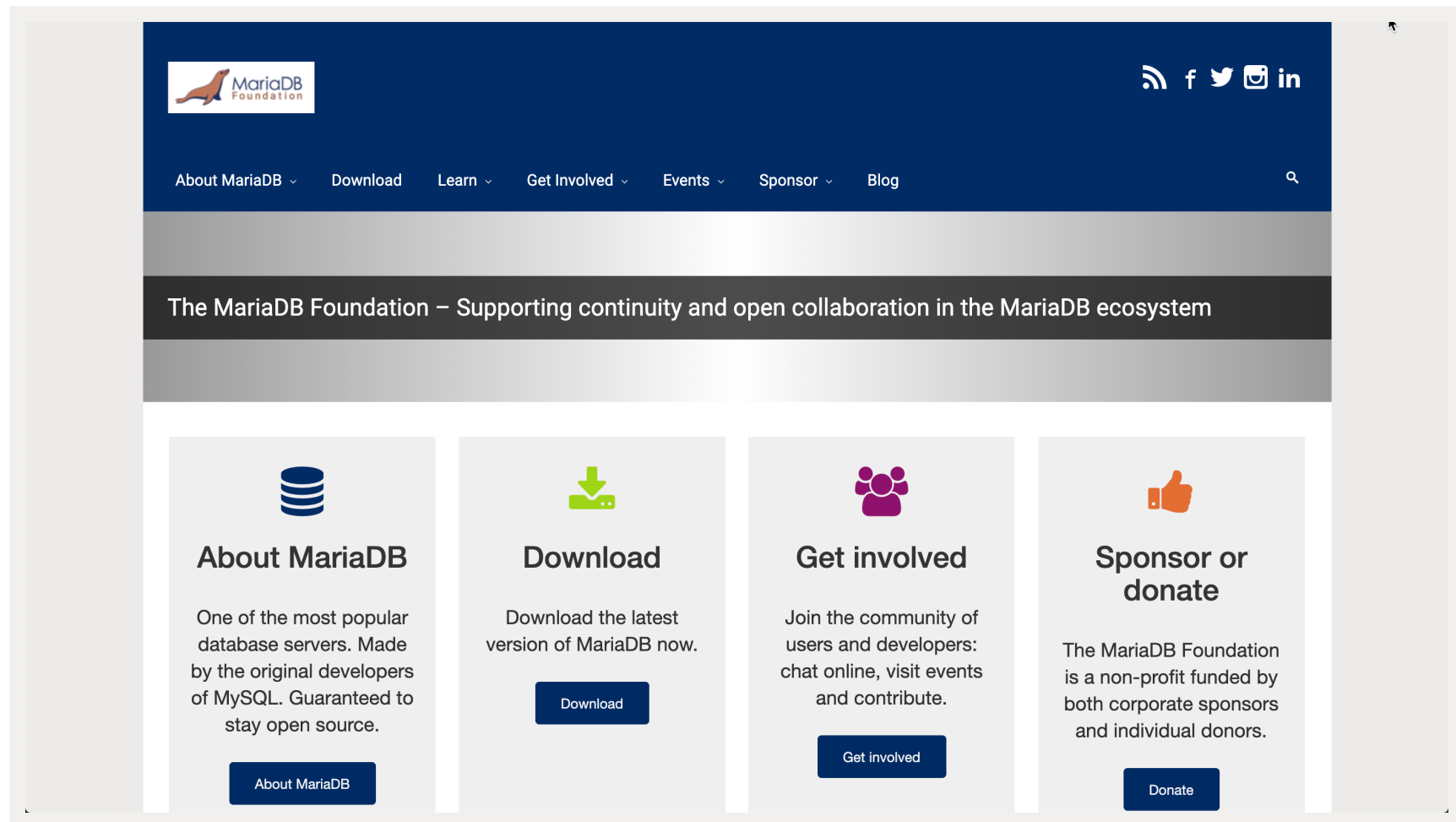
Below the search bar, there is a section titled "Browse MySQL Documentation by:" with two tabs: "Product" and "Topic".

Under the "Product" tab, there are three columns of links:

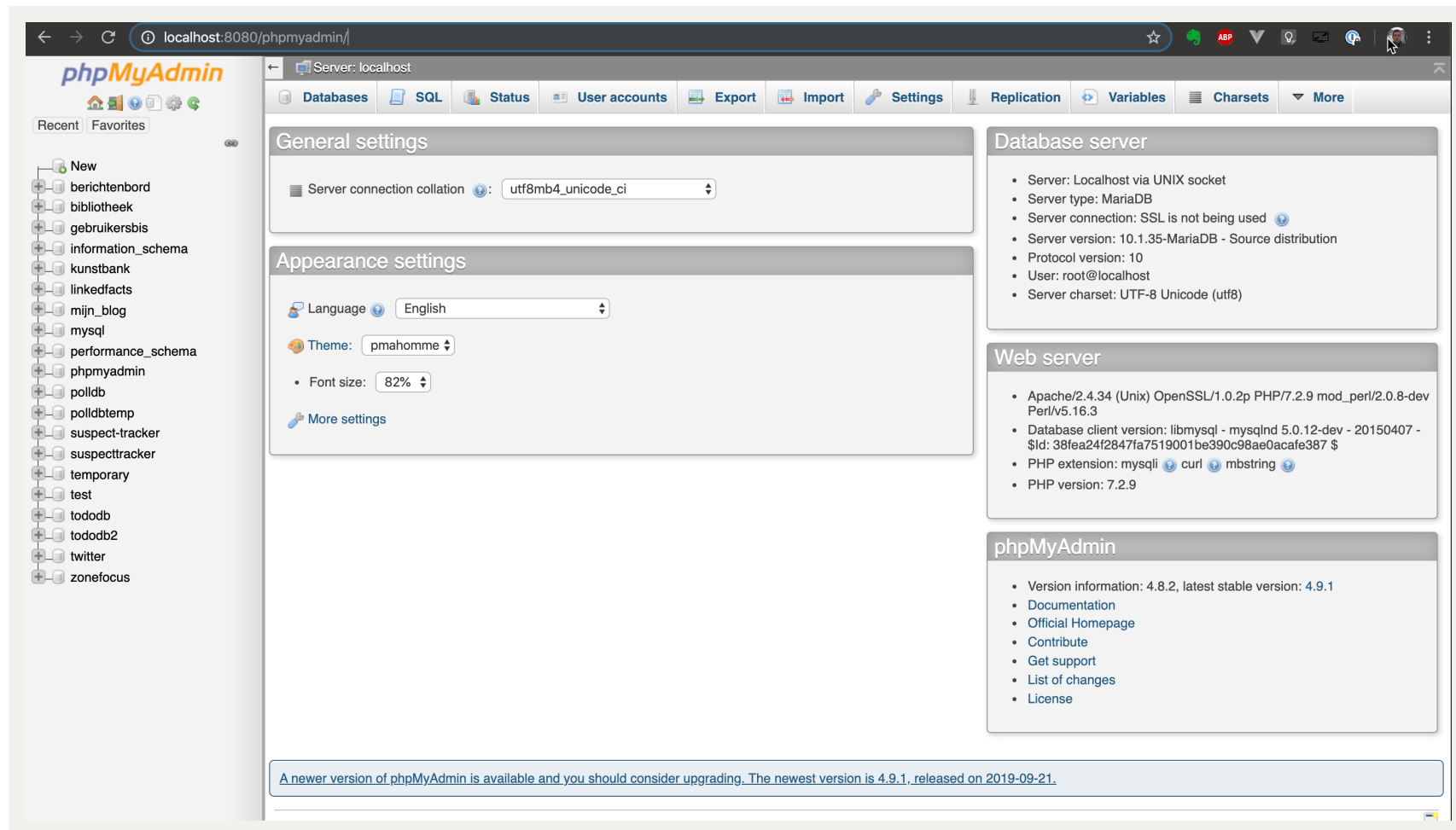
- MySQL Server**
  - [MySQL Reference Manual](#)
- MySQL Enterprise**
  - [MySQL Enterprise Monitor](#)
  - [Oracle Enterprise Manager for MySQL Database](#)
  - [MySQL Enterprise Backup](#)
  - [MySQL Enterprise Security](#)
  - [MySQL Enterprise Encryption](#)
- MySQL Workbench**
  - [MySQL Workbench](#)
- MySQL Shell**
  - [MySQL Shell User Guide](#)
- Connectors & APIs**
  - [Connectors and APIs](#)
  - [Connector/J](#)
- Release Notes**
  - [MySQL Release Notes](#)
  - [NDB Cluster Release Notes](#)
  - [MySQL Cluster Manager Release Notes](#)
  - [MySQL Enterprise Monitor Release Notes](#)
  - [Oracle Enterprise Manager for MySQL Database Release Notes](#)
  - [MySQL Enterprise Backup Release Notes](#)
  - [MySQL Shell Release Notes](#)
  - [Connector/J Release Notes](#)

<https://dev.mysql.com/doc/>

# MariaDB



# phpMyAdmin



# phpMyAdmin

The screenshot shows the phpMyAdmin web interface. On the left is a sidebar with a tree view of databases and tables. The main area displays the 'poll' table from the 'polldb' database. A yellow banner at the top of the main area indicates that 7 rows are shown. Below this, the SQL query 'SELECT \* FROM `poll`' is entered. A table of results is displayed with columns 'poll\_id', 'poll\_vraag', 'ja', and 'nee'. The table contains 7 rows of data. Below the table, there are buttons for 'Check all', 'Edit', 'Copy', 'Delete', and 'Export'. At the bottom, there is a section for 'Query results operations' with buttons for 'Print', 'Copy to clipboard', 'Export', 'Display chart', and 'Create view'. Below this is a 'Bookmark this SQL query' section with a label input field and a checkbox 'Let every user access this bookmark'. A 'Bookmark this SQL query' button is at the bottom right of this section.

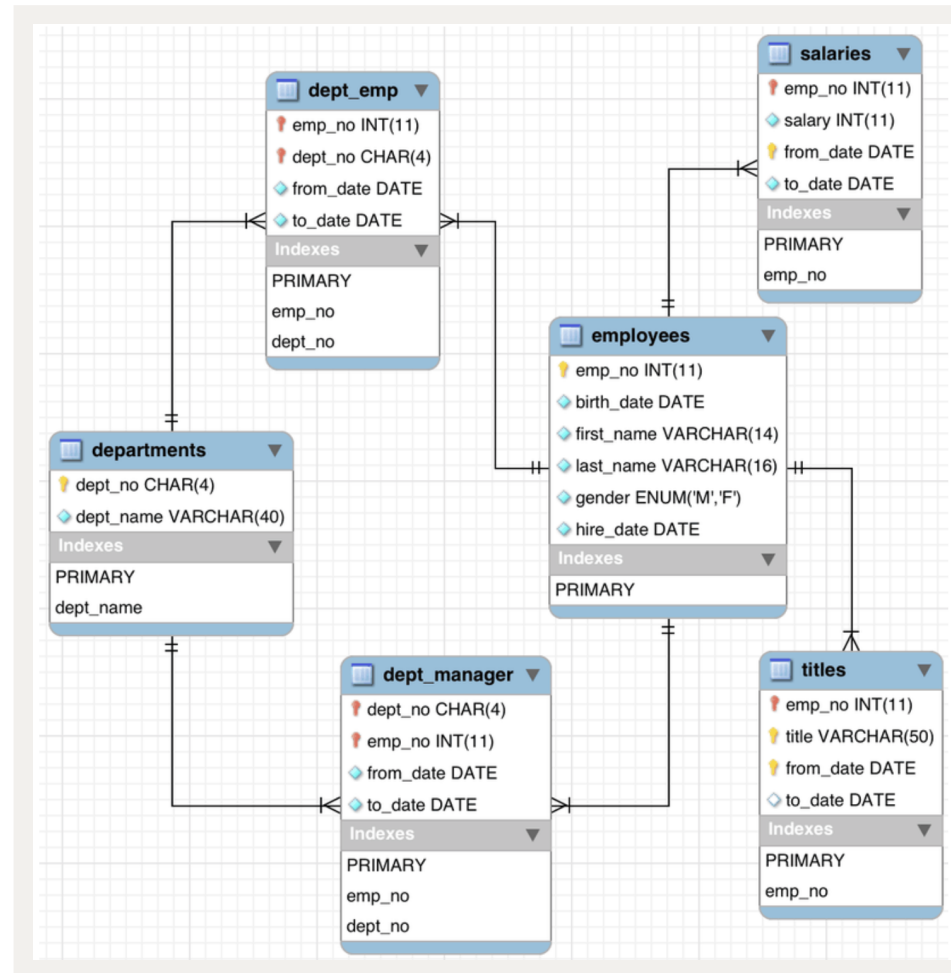
poll_id	poll_vraag	ja	nee
1	Is het moeilijk om PHP te leren?	118	7
2	Is JavaScript leren moeilijk?	23	13
3	Dit is een test	0	0
4	Is Vue.js leren moeilijk?	11	4
5	Is nadenken moeilijk?	2	13
6	Testvraag 23	1230	123445
7	dkeidkiekdeikdeid	0	0

# MySQL Basics

- Data wordt opgeslagen in tabellen (denk Excel sheets)
- Elke tabel bevat een reeks van naam voorziene kolommen, en elke rij bevat data (een record)
- Relationele databank = relaties tussen de (velden van) tabellen
- Tabellen zullen dus ook vaak referenties naar andere tabellen bevatten
- Op die manier kan data opgeslagen in de ene tabel gebruikt worden in andere tabellen
- Door tabellen te ontwikkelen die elk stukje data slechts eenmaal (!) bevatten maak je een robuuste databank met data integriteit



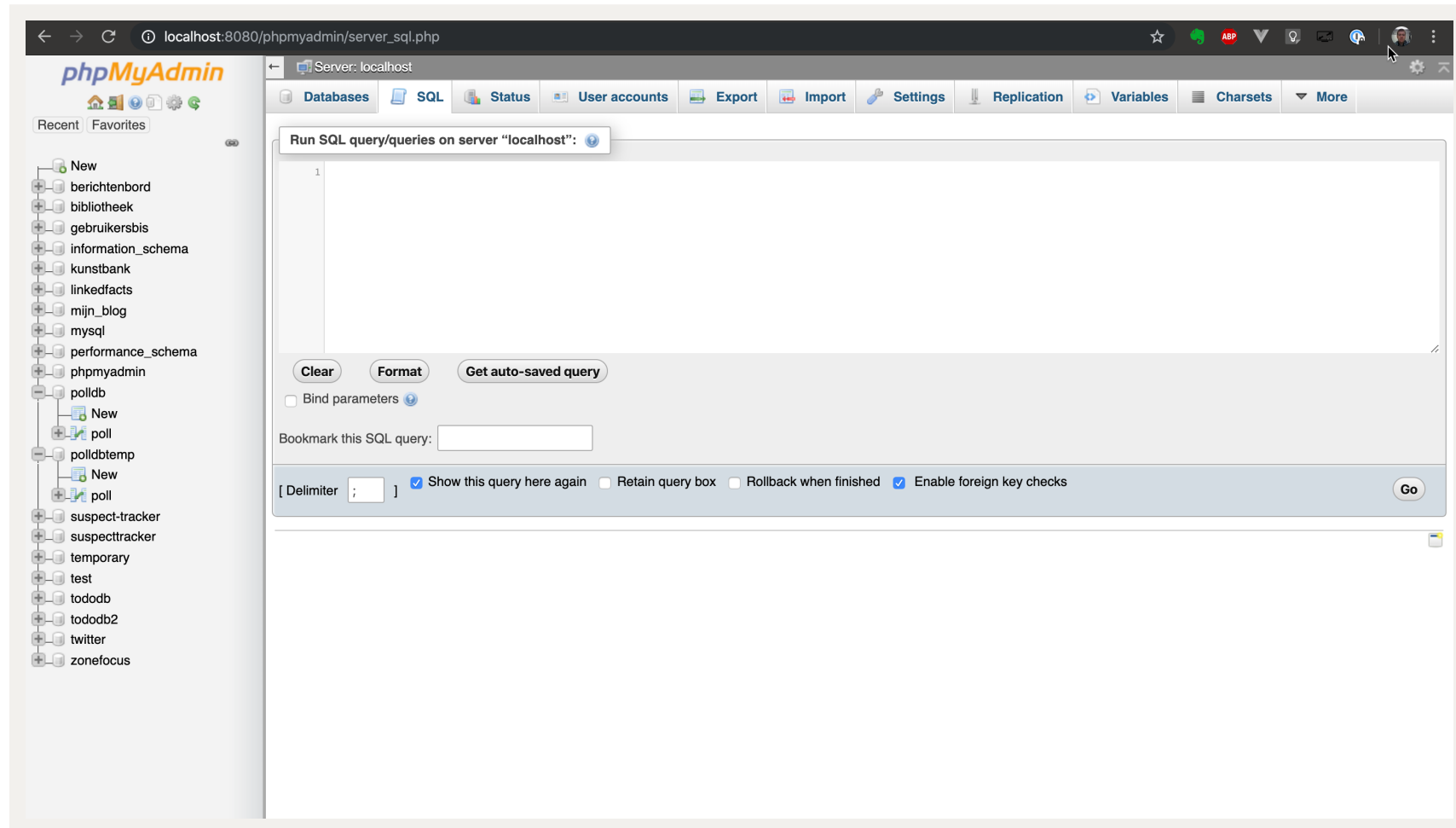
# Relationeel datamodel



# Data manipuleren met SQL

- Je kan data manipuleren in een MySQL tabel door gebruik te maken van de Structured Query Language (SQL)
- SQL is een beperkte taal, in de meeste gevallen niet al te moeilijk om te lezen en te begrijpen
- Je kan SQL ondermeer gebruiken om:
  - Een nieuwe database aan te maken
  - Een tabel in de database aan te maken
  - Data aan de tabel toe te voegen
  - Data uit de tabel op te vragen
  - Bestaande data in de tabel te wijzigen

# SQL toepassen: via phpMyAdmin



Nieuwe app: poll site

# Als voorbeeld: een poll site

- Als voorbeeld zullen we een databankgedreven poll site ontwikkelen
- Het mag dan wel een eenvoudig voorbeeld zijn, maar de belangrijkste principes van databankgedreven ontwikkelen zijn aanwezig
- De meest eenvoudige poll zal één vraag voorzien waarop de bezoeker met ja of nee kan antwoorden
- Je zal ook kunnen zien hoe de andere bezoekers hebben geantwoord
- We zullen gebruik maken van een databank en tabel om de vraag en de antwoorden op te slaan

# De databank aanmaken met CREATE

- SQL gebruikt CREATE om een databank of een tabel aan te maken
- Na CREATE moet je aangeven of je een databank (DATABASE) of een tabel (TABLE) wenst aan te maken
- Je moet dan ook een naam voorzien voor de databank
- We gebruiken steeds de utf-8 karakterset. Geef dit aan met CHARSET utf8

```
CREATE DATABASE polldb CHARSET utf8
```

# De tabel aanmaken met CREATE

- MySQL slaat data op in tabellen
- We moeten dus eerst onze tabel aanmaken
- Hieronder zie je de syntax die we daarvoor moeten gebruiken
- We moeten een naam geven aan onze tabel
- Ook voor de kolommen moeten we een naam, een datatype en eventuele beperkingen (constraints) of standaard (default) waarden voorzien

```
CREATE TABLE tabel_naam (  
    kolom_naam datatype [constraints en default waarden],  
    kolom_naam datatype [constraints en default waarden]  
)
```

# De tabel aanmaken met CREATE

- Onze poll tabel heeft 4 kolommen of attributen: poll\_id, poll\_vraag, ja en nee
- Elk attribuut heeft een data type: poll\_id, ja en nee kunnen enkel integers bevatten (gehele getallen), poll\_vraag kan enkel text bevatten
- ja en nee hebben een standaardwaarde van 0

```
CREATE TABLE poll (  
    poll_id INT NOT NULL AUTO_INCREMENT,  
    poll_vraag TEXT,  
    ja INT DEFAULT 0,  
    nee INT DEFAULT 0,  
    PRIMARY KEY (poll_id)  
)
```



# Mogelijke Data Types

DATE TYPE	SPEC	DATA TYPE	SPEC
CHAR	String (0 - 255)	INT	Integer (-2147483648 to 2147483647)
VARCHAR	String (0 - 255)	BIGINT	Integer (-9223372036854775808 to 9223372036854775807)
TINYTEXT	String (0 - 255)	FLOAT	Decimal (precise to 23 digits)
TEXT	String (0 - 65535)	DOUBLE	Decimal (24 to 53 digits)
BLOB	String (0 - 65535)	DECIMAL	"DOUBLE" stored as string
MEDIUMTEXT	String (0 - 16777215)	DATE	YYYY-MM-DD
MEDIUMBLOB	String (0 - 16777215)	DATETIME	YYYY-MM-DD HH:MM:SS
LONGTEXT	String (0 - 4294967295)	TIMESTAMP	YYYYMMDDHHMMSS
LOBLOB	String (0 - 4294967295)	TIME	HH:MM:SS
TINYINT	Integer (-128 to 127)	ENUM	One of preset options
SMALLINT	Integer (-32768 to 32767)	SET	Selection of preset options
MEDIUMINT	Integer (-8388608 to 8388607)	BOOLEAN	TINYINT(1)

# Wat is PRIMARY KEY?

- Je zal zien dat in phpmyadmin het poll\_id attribuut van een sleutel is voorzien
- Dit betekent dat poll\_id de primary key is van de tabel
- Een primary key moet telkens een unieke waarde bevatten
- Elke primary key vertegenwoordigt dus één datarij in de tabel

# Wat is AUTO\_INCREMENT?

- De primary key van onze tabel is voorzien van de eigenschap "auto-increment"
- Dit is een eenvoudig maar zeer krachtig mechanisme
- De eerste rij zal voor poll\_id de waarde 1 meekrijgen, de tweede rij 2, de derde 3 enz...
- Dit gebeurt volledig automatisch, MySQL houdt dat zelf bij

# Data toevoegen met INSERT

- Nu de tabel is aangemaakt kunnen we ook data gaan opslaan in onze databank
- We gebruiken daarvoor het INSERT statement
- Voor elke nieuwe toevoeging in de poll tabel gaan we een nieuwe datarij aanmaken
- Volgend voorbeeld zal dat voor ons doen. We geven enkel een waarde voor poll\_vraag. De overige waarden worden automatisch voor ons toegevoegd door de databank

```
INSERT INTO poll (poll_vraag) VALUES ("Is het moeilijk om PHP te leren?" )
```

# Data toevoegen met INSERT

Dit is de algemene syntax voor insert:

```
INSERT INTO tabel_naam (  
    kolom_naam, andere_kolom_naam  
) VALUES (  
    [kolom waarde], [andere kolom waarde]  
)
```

# Data opvragen met SELECT

- Je vraagt data op uit je tabel door gebruik te maken van een SQL SELECT statement
- Je ziet hier eerste de algemene syntax, vervolgens een werkend voorbeeld voor onze app
- Een select statement geeft altijd een tijdelijke tabel terug voorzien van alle data die voldoet aan de vraag

```
SELECT kolom_naam, andere_kolom_naam FROM tabel_naam
```

```
SELECT poll_id, poll_vraag, ja, nee FROM poll
```

# Data wijzigen met het UPDATE statement

- In onze voorbeeld app zullen we de ja/nee waardes moeten aanpassen telkens een bezoeker een antwoord doorstuurt
- Dit voorbeeld geeft aan wat we moeten doen als een bezoeker aangeeft dat PHP leren moeilijk is
- WHERE geeft aan voor welke poll\_id waarde we de data moeten wijzigen

```
UPDATE poll SET ja = ja + 1 WHERE poll_id = 1
```

# Data verwijderen met het DELETE statement

- We hebben het in onze app nog niet nodig, maar met het DELETE statement kan je data uit tables definitief verwijderen
- Dit statement neemt volgende algemene gedaante aan:

```
DELETE FROM tabel_naam [WHERE voorwaarde]
```

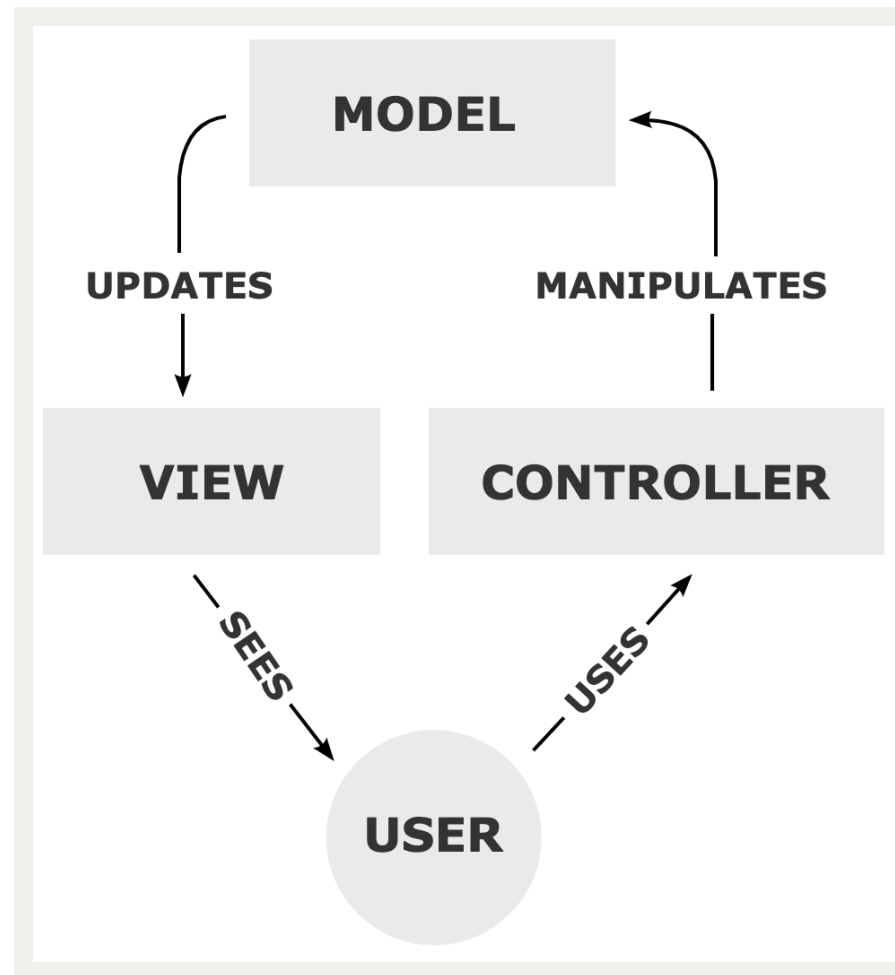


# Terug naar onze PHP code

- We gaan in onze applicatie de databank die we daarnet hebben gemaakt gebruiken als opslagruimte
- We gaan hiervoor een PDO object gebruiken. Dit object zal onze app binden met een MySQL databank
- We zullen ook gebruik maken van een code-design patroon dat ervoor zal zorgen dat onze code gestructureerd kan worden opgeslagen: MVC

# Het MVC-patroon

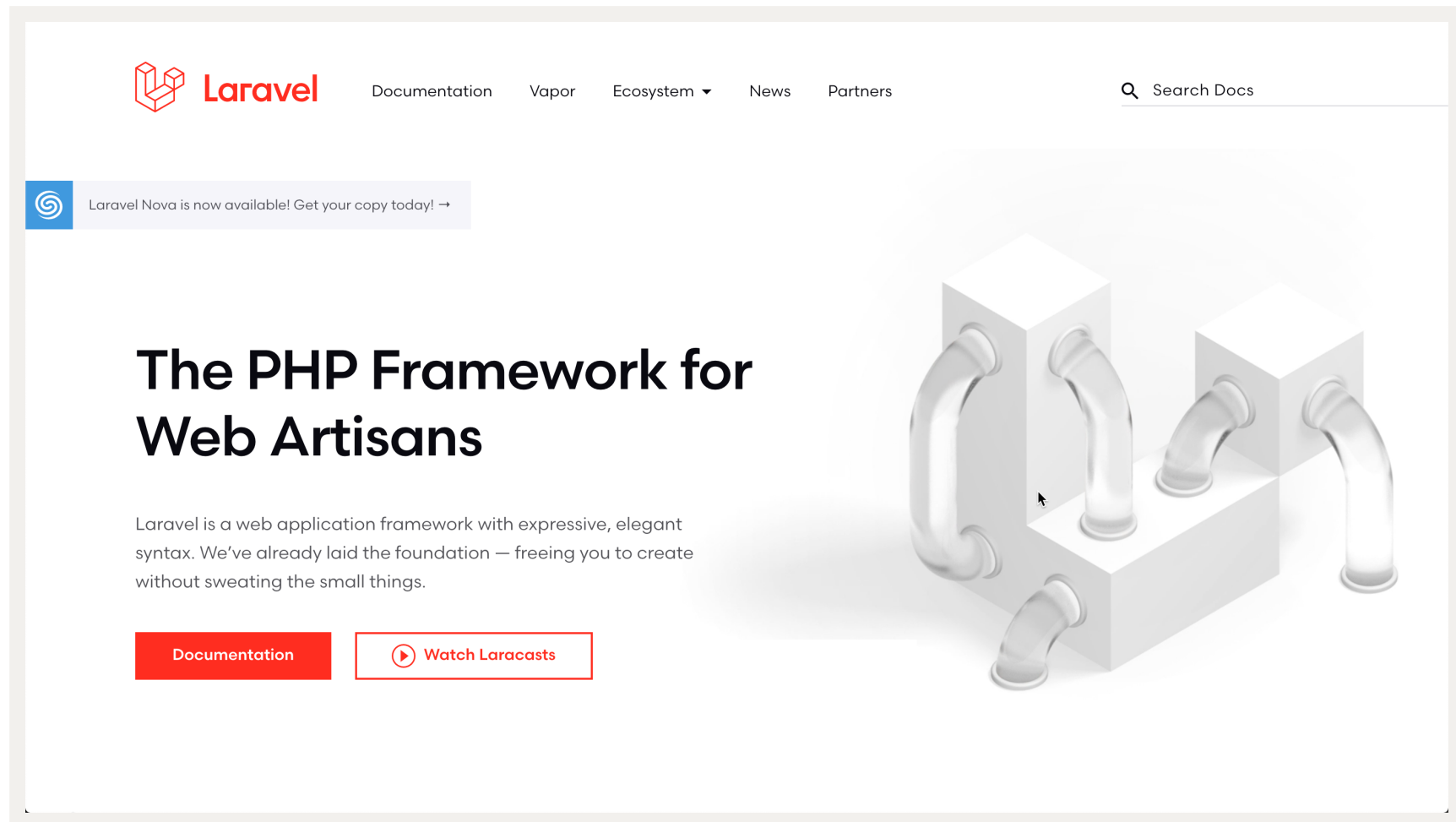
# Model-View-Controller patroon



# Model-View-Controller patroon

- MVC wordt gebruikt om onze scripts op een eenduidige manier op te slaan
- De winst die je hiermee maakt is dat je efficiënter en overzichtelijker kan werken
- Je maakt hierdoor ook minder fouten
- In ons voorbeeld is die winst nog beperkt, maar bij complexere toepassingen is de winst vaak heel groot
- Werken met bepaalde vaste patronen is heel gebruikelijk. Kijk bvb naar Laravel...

# Laravel



# Model-View-Controller patroon

- MVC deelt code in in 3 categorieën: models, views en controllers
- Een model bevat code die data vertegenwoordigt, alsook de meeste systeemlogica
- Een view bevat code die informatie visueel weergeeft. Deze informatie is afkomstig van het model
- Een controller bevat code die input van gebruikers opvangt en bevelen doorstuurt naar de relevante models

# Het opstarten van onze toepassing

- We houden de poll eenvoudig
- We maken een index.php pagina aan die de poll zal tonen. Deze index.php zal een front controller zijn, een "toegangsdeur" tot de web applicatie
- index.php zal een html-pagina weergeven, en zal de poll controller laden
- Elke view zal zijn eigen model en controller hebben: er is een poll model, controller en view
- Ook de pagina heeft een eigen model, view en controller. De front controller is de pagina controller

# Het opstarten van onze toepassing

- In een nieuwe folder voor deze toepassing maak je naast een index.php ook 3 subfolders aan: models, views en controllers
- Gebruik opnieuw pagina.php uit de vorige les en sla ze op in de views folder
- Gebruik opnieuw Pagina\_Data.class.php en sla ze op in de models folder

```
<?php
include_once "models/Pagina_Data.class.php";
$paginaData = new Pagina_Data();
$paginaData->titel = "PHP/MySQL site poll voorbeeld";
$paginaData->content = "<h1>Alles werkt op dit punt!</h1>";
$pagina = include_once "views/pagina.php";
echo $pagina;
```



# Poll controller aanmaken

- We maken een nieuw bestand aan in de controllers folder: poll.php
- We gaan dan de poll controller inladen vanuit index.php

```
<?php
//code voor controllers/poll.php
return "Poll zal hier getoond worden";
```

```
//deze lijn verwijderen in index.php
$paginaData->content = "<h1>Alles werkt op dit punt!</h1>";
//en vervangen door
$paginaData->content = include_once "controllers/poll.php";
```

# Poll model aanmaken

- We gaan een voorlopig poll model aanmaken, in models/Poll.class.php, met slechts één method
- De Poll klasse heeft 1 method: deze zal het hardgecodeerde \$pollData object aanmaken en teruggeven

```
<?php
//code voor models/Poll.class.php
class Poll {
    public function getPollData() {
        $pollData = new stdClass();
        $pollData->poll_vraag = "test 1 2 3...";
        $pollData->ja = 0;
        $pollData->nee = 0;
        return $pollData;
    }
}
```

# Poll view aanmaken

Maak het bestand views/poll-html.php aan:

```
<?php
//code voor views/poll-html.php
return '
    <div id="poll">
        <h1>Poll resultaten</h1>
        <ul>
            <li>$pollData->ja zeiden ja</li>
            <li>$pollData->nee zeiden nee</li>
        </ul>
    </div> ';
```

# Poll view en model verbinden

Verbinden doen we in de poll controller. Pas poll.php aan en we hebben een volledig MVC-patroon gemaakt:

```
<?php
//code voor controllers/poll.php
include_once "models/Poll.class.php";
$poll = new Poll();
$pollData = $poll->getPollData();
$pollView = include_once "views/poll-html.php";
return $pollView;
```

# php data objects (PDO) verbinden php met mysql

- De MVC architectuur maakt het eenvoudig om een verbinding te maken met een databank
- Wij gaan hiervoor PHP data objects (=PDO) gebruiken
- PDO bieden een veilige en efficiënte manier om vanuit PHP met een databank te communiceren
- PDO ondersteunen verschillende databanken en bieden een uniforme set van methods die de meeste databankinteracties ondersteunen
- Meer weten? [www.php.net/manual/en/book.pdo.php](http://www.php.net/manual/en/book.pdo.php)
- We maken op volgende slide enkele wijzigingen in front controller index.php
- We stoppen tijdelijk met het laden van de poll controller
- Probeer en kijk of het verbinden met de databank lukt

# Een databank-connectie openen

```
<?php
include_once "models/Pagina_Data.class.php";
$paginaData = new Pagina_Data();
$paginaData->titel = "PHP/MySQL site poll voorbeeld";
$dbInfo = "mysql:host=localhost;dbname=polldb";
$dbUser = "root";
$dbPassword = "";
try {
    $db = new PDO( $dbInfo, $dbUser, $dbPassword );
    $db->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION );
    $paginaData->content = "<h1>We zijn verbonden!</h1>";
} catch ( Exception $e ) {
    $paginaData->content = "<h1>De connectie lukt niet!</h1><p>$e</p>"; }
$pagina = include_once "views/pagina.php";
echo $pagina;
```

# Een try-catch statement gebruiken

- Er kunnen verschillende zaken mislopen bij het verbinden met een databank
- In dat geval krijg je een exception terug
- Als je exceptions goed opvangt, dan kan de code verder normaal uitgevoerd worden
- Dit kan je met een try-catch statement doen

```
try {  
    //iets uitproberen dat kan mislukken  
} catch ( Exception $e ) {  
    //en het is mislukt  
}
```

# Databank-connectie delen met het Poll model

- We hebben reeds een databankverbinding, een PDO object, gemaakt in index.php
- We gebruiken een constructor-method met argumenten om de db-connectie in het poll model in te brengen
- Pas hiervoor de code in models/Poll.class.php aan

```
<?php
class Poll {
    private $db;
    public function __construct($dbConnection){
        $this->db = $dbConnection; }
    public function getPollData(){
        //...
    }
}
```



# Databank-connectie delen met het Poll model

- De poll class kan nu een PDO object ontvangen als argument ontvangen via de constructor method
- We roepen ons Poll model aan in de poll controller, en dienen ook hier aan te passen
- \$db is gedeclareerd in index.php maar is beschikbaar in de poll controller omdat deze via include\_once is verbonden vanuit index.php

```
<?php //code voor controllers/poll.php
include_once "models/Poll.class.php";
//Enkel hier aanpassen...
$poll = new Poll($db);
$pollData = $poll->getPollData();
$pollView = include_once "views/poll-html.php";
return $pollView;
```

# Data opvragen met een PDO-statement

- We gaan nu terug naar ons poll model en brengen wijzigingen aan in de getPollData
- De method fetchObject() zal automatisch een nieuw StdClass object aanmaken en het teruggeven
- Eigenschappen van dit object zijn identiek aan de kolomnamen van de tabel

```
//code voor models/Poll.class.php
public function getPollData () {
    $sql = "SELECT poll_vraag, ja, nee FROM poll WHERE poll_id = 1";
    $statement = $this->db->prepare($sql);
    $statement->execute();
    $pollData = $statement->fetchObject();
    return $pollData;
}
```

# Data opvragen met een PDO-statement

Hier zie je hoe het \$pollData object kan gebruikt worden om data weer te geven

```
<?php
//code voor views/poll-html.php
return '
    <div id="poll">
    <h1>Poll resultaat</h1>
    <ul>
        <li>$pollData->ja zeiden ja</li>
        <li>$pollData->nee zeiden nee</li>
    </ul>
    </div> ';
```

# Een poll form tonen

- code staat op volgende slide
- We tonen de form in views/poll-html.php
- Het \$pollData object is essentieel voor het functioneren van de form
- Je kan testen of het object beschikbaar is
- Indien niet dan kan je een custom foutboodschap trierden (trigger\_error), zoals we hier gaan doen

# Een poll form tonen

```
<?php
$dataGevonden = isset( $pollData );
if( $dataGevonden === false ){
    trigger_error( 'views/poll-html.php noodzaakt een $pollData object' );
}
return '<div id="poll">
    <form method="post" action="index.php"> <p>$pollData->poll_vraag</p>
    <select name="user-input">
    <option value="ja">Ja, het is moeilijk!</option>
    <option value="nee">Nee, niet echt!</option> </select>
    <input type="submit" value="post">
    </form>
    <h1>Poll resultaten</h1>
    <ul>
    <li>$pollData->ja zeiden ja</li>
    <li>$pollData->nee zeiden nee</li>
    </ul></div> ';
```

# Databank tabel updaten

- Als een gebruiker de poll beantwoordt, dan moeten we de tabel aanpassen
- We maken hiervoor een nieuwe method aan in de Poll class
- De databank updaten hoort te gebeuren in een model, gebruikersinput behandelen: dat is werk voor een controller
- We zullen de updatePoll() method dan ook aanroepen vanuit de controller

# Databank tabel updaten

```
<?php
// nieuwe method in de Poll class...
public function updatePoll ( $input ) {
    if ( $input === "ja" ) {
        $updateSQL = "UPDATE poll SET ja = ja+1 WHERE poll_id = 1";
    } else if ( $input === "nee" ) {
        $updateSQL = "UPDATE poll SET nee = nee+1 WHERE poll_id = 1";
    }
    $updateStatement = $this->db->prepare($updateSQL);
    $updateStatement->execute();
}
```

# Behandelen van de gebruikersinput

- Doen we in de poll controller
- We gebruiken hiervoor de volgende code...
- Hiermee hebben we de laatste steen gelegd aan onze eerste databank-gedreven toepassing
- Probeer deze applicatie zelf te doen functioneren
- In haar eenvoud illustreert ze niet alleen hoe je een databank kan gebruiken om data uit op te vragen en naar weg te schrijven, maar ook hoe je je code scheidt volgens het MVC patroon



# Behandelen van de gebruikersinput

```
<?php
//volledige code voor controllers/poll.php
include_once "models/Poll.class.php";
$poll = new Poll( $db );
$isPollAanwezig = isset( $_POST['user-input'] );
if ( $isPollAanwezig ) {
    $input = $_POST['user-input'];
    $poll->updatePoll( $input );
}
$pollData = $poll->getPollData();
$pollAlsHTML = include_once "views/poll-html.php";
return $pollAlsHTML;
```

## WMP5 - PHP - Werken met databanken