Lesdoelen

- ✓ Je begrijpt de taak/rol van de View binnen MVC en kunt een View bouwen voor een blog
- ✓ Je kent de xTemplate template engine en kunt door je View deze laten gebruiken

Model View Controller, deel 2

Inleiding

Vorige week hebben we de basis van het Model View Controller (MVC) ontwerp patroon behandeld. We gaan het MVC patroon inzetten voor het bouwen van een web-log, daar hebben we vorige les het Model voor geschreven welke verantwoordelijk is voor de data afhandeling.

Deze week gaan we verder werken aan het MVC en de onderdelen voor de presentatie (view) en logica (controller) doorlopen. Lees de vorige lesbrief door voor een uitgebreidere omschrijving van het MVC patroon en zorg dat je je HTML templates voor het web-log af hebt.

Installatie

Voor deze les moet je van de vakgroep de bron bestanden van het weblog downloaden in installeren.

Downloaden

1. Download als eerste de zip met bronbestanden, deze vind je hier: http://vakgroep.cmd.hro.nl/fed/y2/q2/w9/blog.zip

Project en bestanden:

1. Maak een nieuw project aan in Easy Eclipse for PHP, genaamt **blog**.
2. Pak het zip bestand uit in de directory van het nieuwe project.

Database:

1. Open phpMyAdmin in je browser, via http://localhost/phpmyadmin/.
2. Maak een nieuwe database aan, genaamd **blog.**
3. Klik op import/importeren.
4. Blader voor het bestand, database.sql, deze staat in je project directory.
5. Kik op start om de SQL code te importeren in de database.

Je kan nu testen op http://localhost/blog/ of de installatie gelukt is, je ziet een overzicht met nieuwsartikelen. Door te klikken op een van de links open je het detail van een bericht.

Bekijk alvast de database structuur in PHPMyAdmin voor elk bericht.

Directory structuur

In de directory van je MVC project zie je een duidelijke directory structuur:

```
./classes
./controller
./model
./view
./index.php
```

./classes

Bevat alle classes die je hebt geschreven die niet direct binnen de code van het MVC patroon vallen. In dit project staat hier de Bericht class, welke wordt gebruikt door het MVC. In deze directory kan je eventueel ook classes neerzetten die je hebt gedownload van het internet, zoals bijvoorbeeld een template engine.

./controller

Bevat controller classes; een controller class handelt alle requests van de gebruiker/browser af. Bij elk request kijkt de controller welke acties er ondernomen moeten worden; zoals het data inladen via een model; de juiste html templates inladen via de views.

./model

Bevat model classes; een model class is verantwoordelijk voor de opslag en ophalen van data uit bijvoorbeeld een datbase. Een model is binnen MVC dan ook de enige plek waar je database queries mag schrijven.

./view

Bevat alle views; dit zijn alle html blokken/templates die kunnen worden ingeladen door een controller. Een enkele HTML pagina bestaat vaak uit meerdere views die gecombineerd worden.

Bestanden

Hieronder worden de bestanden omschreven binnen de voorbeeldcode:

./index.php
Dit bestand is de ingang van het web-log PHP programma en is verantwoordelijk voor zaken als het maken van een database connectie en dat het initialiseren van de juiste controller. In dit voorbeeld hebben we maar een enkele controller, de BerichtController.

./controller/berichtcontroller.class.php
De *BerichtController* class is verantwoordelijk voor afhandelen van de requests die te maken hebben met de berichten binnen het web-log. De controller zoekt aan de hand van de GET parameters uit welke actie precies moet worden uitgevoerd, welk data moet worden ingeladen(model) en welke pagina getoond moet worden (views).

./model/berichtmodel.class.php
De *BerichtModel* class is verantwoordelijk voor het ophalen, opslaan en verwijderen van berichten. Deze class wordt bijvoorbeeld door de controller aangeroepen zodra de details van een bericht moet worden ingeladen uit de database. Het *BerichtModel* geeft dan een nieuwe instantie van de *Bericht* class terug.

./classes/bericht.class.php
De Bericht class is de blauwdruk van een web-log bericht. Alle eigenschappen die terug te vinden zijn in de datbase zijn aanwezig in deze class. De class wordt gebruikt in het *bericht model* en heeft allerlei handige functies waar je binnen een view data kan ophalen.

./view/header.php
Een HTML template welke het opende(head, body) gedeelte van de HTML pagina bevat, wordt altijd als eerste view ingeladen door de controller.

./view/footer.php
Een HTML template welke het afsluitende gedeelte van de HTML pagina bevat, wordt altijd als laatste view ingeladen door de controller.

./view/bericht_detail.php
Een HTML template met de detail pagina voor een bericht, wordt bij het detail ingeladen door de controller.

./view/bericht_overzicht.php
Een HTML template met de overzichtspagina voor een bericht, wordt bij het overzicht ingeladen door de controller.

Doorloop

Hieronder volgt aan de hand van een scenario een doorloop van de code. Volg de doorloop stapsgewijs en bekijk ondertussen de code binnen Easy Eclipse for PHP.

./index.php

1. Een gebruiker open het web-log in de browser: http://localhost/blog/

2. De webserver opent het standaard bestand: *index.php*

3. Binnen *index.php* wordt een verbinding aangemaakt in de database

```php
// de waarden voor de verbinding
$host = "localhost"; // localhost is je eigen computer
$username = "root"; // gebruiker die toegang heeft tot de database
$password = ""; // wachtwoord van de gebruiker
$database = "blog"; // naam van database

// maak een verbinding met mysql
mysql_connect($host, $username, $password);

// kies de juiste database
mysql_select_db($database);
```

4. Hierna wordt binnen *index.php* de juiste bestanden ingeladen

```php
// importeer de bericht class
include_once("classes/bericht.class.php");

// importeer het bericht model
include_once("model/berichtmodel.class.php");

// importeer de bericht controller
include_once("controller/berichtcontroller.class.php");
```

5. Hierna wordt de bericht controller geinitialiseerd en de *start* methode aangeroepen

```php
// initieer de bericht controller
$controller = new BerichtController();

// begin!
$controller->start();
```

./controller/berichtcontroller.class.php

6. De *start* methode van de BerichtController wordt aangeroepen, deze zoekt aan de hand van meegegeven GET variabele welke actie moet worden uitgevoerd.

```php
public function start()
{
	// standaard actie
	$actie = "";

	// is er een andere actie mee gegeven aan de url?
	if (isset($_GET['actie']))
	{
		$actie = $_GET['actie'];
	}

	// functie aanroepen aan de hand van de actie
	switch ($actie)
	{
		case "overzicht" :
			$this->toonOverzicht();
			break;
		case "detail" :
			$this->toonDetail();
			break;
		default :
			// standaard actie
			$this->toonOverzicht();
			break;
	}
}
```

7. Standaard wordt er geen variabele meegegeven via GET en is GET['actie'] leeg, dan wordt de toonOverzicht methode aangeroepen.

8. De toonOverzicht methode laad eerst alle berichten in via het bericht model. Het BerichtModel laad de data uit de database en geeft een lijst (array) met Bericht objecten terug. Zodra het model deze heeft terug gegeven worden de benodigde views ingeladen.

```php
private function toonOverzicht()
{
	// we laden alle berichten in
	$berichten = $this->bericht_model->overzicht();

	// titel variabele
	$pagina_titel = "Overzicht";

	// we tonen het berichten overzicht
	include("view/header.php");
	include("view/bericht_overzicht.php");
	include("view/footer.php");
}
```

./view/bericht_overzicht. php

9. In de *bericht_overzicht.php* view wordt door alle berichten heen gelopen met behulp van een loop, hier wordt een lijst van geprint met links naar de detail pagina. Merk op dat het ID van een nieuwsbericht wordt meegegeven met de link.

```
<h1>Berichten</h1>
<p>
<ul>
<?php
        // we loopen door alle berichten heen die uit het model komen
        foreach ($berichten as $bericht)
        {
                print('<li>');

                // we geven de actie en bericht id mee via de url
                print('<a href="index.php?actie=detail&id=' .
                        $bericht->getId() . '">');

                print($bericht->getHtmlTitel());

                print('</a>');

                print('</li>');
        }
?>
</ul>
</p>
```

Opdrachten

Opdracht 1: Html integreren

Voor deze les heb je HTML geproduceerd, deze moet je nu gebruiken in je blog.

Vervang binnen de views de HTML met die je zelf geschreven hebt, doe dit voor de volgende pagina's:

- ✓ Overzicht pagina met de titels en lees meer links van alle posts
- ✓ Detail pagina met de titel, body en datum

De html met volgens XHTML worden gemaakt (zonder frames) samen met vormgeving in css.