Dynamische Websites

Week 7

AGENDA

- **Lesschema**
- Herhaling
- Front Controller

LESSCHEMA

- Lesweek 7: Framework - Front Controller
- Lesweek 8: Framework - 2 Step Design
- Lesweek 9: Framework - Forms en ...
- Lesweek 10: Framework - Vragen
- Lesweek 11: Extra topics
- Lesweek 12: Herhaling belangrijkste concepten
- Lesweek 13: Vragen en uitleg examen

AGENDA

- Lesschema
- **Herhaling**
- Front Controller

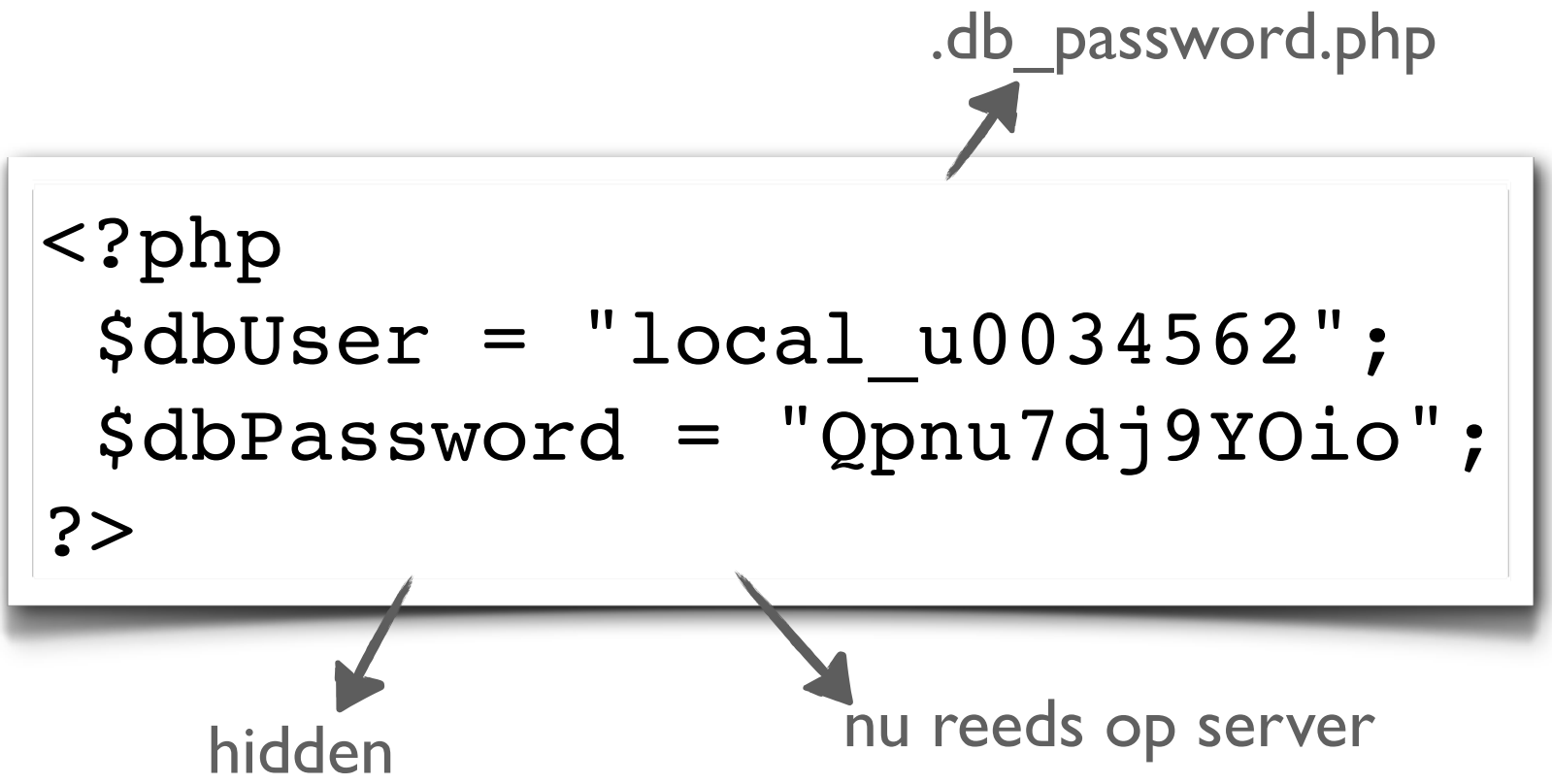
DB Gegevens

```php
<?php
date_default_timezone_set('Europe/Brussels');

// local_u0034562 via webontwerp => 'host' => 'localhost'
// u0034562 via mamp, wamp, xampp => 'host' => 'gegevensbanken.khleuven.be'
require_once('../../../../.db_password.php');

$db_config = array(
    'driver' => 'pgsql',
    'username' => $username,
    'password' => $password,
    'schema' => 'public',                          // dit moet je veranderen naar je eigen schema
    'dsn' => array(
        'host' => 'gegevensbanken.khleuven.be',
        'dbname' => 'webontwerp',                  // dit moet je veranderen naar db van je reeks
        'port' => '51314',
    )
);
```

.DB_PASSWORD.PHP



Agenda

- Lesschema
- Herhaling
- **Front Controller**

Front Controller

- = is een dispatching controller naar wie alle requests worden gedaan
- alles gaat via index.php
  - controllers, actions en parameters worden hieraan meegegeven

- https://en.wikipedia.org/wiki/Front_Controller_pattern

Front Controller via $_GET en IF

- Voorbeelden
  - http://example.com/index.php?action=new
  - http://example.com/index.php?action=home
- Zie
  - dynweb2013examples/theorie07/frontcontroller_stap01

VOORBEELD HOME

- http://example.com/index.php?action=home

INDEX.PHP

```
<?php
require_once('Controller.php');

$controller = new Controller();
$controller->run();
```

Code Onveranderd

CONTROLLER.PHP

```
<?php
class Controller{
    private $_vehicles;
    private $_vehicleMapper;
...
    public function run(){
        $nextPage = 'index.php';
        if ($this->_action == 'register') {
            $nextPage = $this->register();
        } elseif ($this->_action == 'new') {
            $nextPage = $this->newRegistration();
        } elseif ($this->_action == 'home') {
            $nextPage = $this->home();
        }
        require_once($nextPage);
    }
...
```

Code Onveranderd

CONTROLLER.PHP

```
    ...

      private function home()
      {
       return 'home.php';
    }
}

?>
```

Code Onveranderd

HOME.PHP

```
<body>
<h1>Welkom!</h1>
<a href="index.php?action=new">Registreer voertuig</a>
</body>
```

Code Onveranderd

Front Controller via $_GET en IF

- Uitbreiding
  - als we een detail pagina willen toevoegen waarmee we alle informatie van dat specifiek vehicle willen tonen

Front Controller via $_GET en IF

- Problemen?
  - run methode wordt veel te lang
  - voldoet niet aan het Open-Closed principe
    - nieuwe functionaliteit kunnen toevoegen door zo weinig mogelijk bestaande code aan te passen

Front Controller OO via $_GET

- Controller
  - if-then-else structuur eruit halen
  - delegeren naar een controller per onderwerp (bijvoorbeeld vehicle)
  - maakt code **herbruikbaarder** en **uitbreidbaarder**

Front Controller OO via $_GET

- Voorbeelden
  - http://example.com/index.php?controller=vehicle&action=new
  - http://example.com/index.php?controller=vehicle&action=detail&id=1

- Zie
  - dynweb2013examples/theorie07/frontcontroller stap02

VOORBEELD VEHICLE OVERVIEW

- http://example.com/index.php?controller=vehicle&action=index

- $object = 'VehicleController';
- $method = 'index';
- $object = new $object();
- $object->$method();

INDEX.PHP

```
<?php
require_once('Controller.php');

$controller = new Controller();
$controller->run();
```

Code Onveranderd

CONTROLLER.PHP

```php
<?php
class Controller{

    public function run()
    {
        define('DEFAULT_CONTROLLER', 'home');
        define('DEFAULT_ACTION', 'index');

        // getting the requested controller from $_GET['controller']
        $controller = (isset($_GET['controller'])) ? $_GET['controller'] : DEFAULT_CONTROLLER;

        // setting the object and including the controller objectfile
        $object = ucfirst(strtolower($controller));
        $file = $object . 'Controller' . '.php';
        require_once($file);

        // getting the requested action frm $_GET['action']
        $method = (isset($_GET['action'])) ? $_GET['action'] : DEFAULT_ACTION;
        $object = $object . 'Controller';

        // creating the controller as an instance of the controller object
        $object = new $object();

        // executing the method
        $object->$method();
    }
```

```
$object = 'Vehicle';

$file = 'VehicleController.php'

$method = 'index';

$object = 'VehicleController'

$object = new $object();

$object->$method();
```

VehicleController.php

```php
<?php

class VehicleController
{
    private $_vehicleMapper;

    public function __construct() {
        $this->_vehicleMapper = new VehicleMapper();
    }

    public function index() {
        $vehicles = $this->_vehicleMapper->getAll();
        require_once('vehicle_overview.php');
    }

    public function detail() {...}

    public function add() {...}

    public function register() {...}
}
```

VehicleMapper.php

```php
<?php
require_once('Db.php');
require_once('Vehicle.php');

class VehicleMapper
{
    private $_db;

    public function __construct() {
            $this->_db = Db::getInstance();
    }

    public function add($object) {...}

    public function getAll() {
            $sql = "SELECT * FROM vehicles";
            $data = $this->_db->query($sql);
            $objects = array();
            foreach ($data as $row) {
                $object = new Vehicle($row['color'], $row['brand']);
                $objects[] = $object;
            }
            return $objects;

    }
}
```

Code Onveranderd

VEHICLE.PHP

```php
<?php
class Vehicle {
    private $color;
    private $brand;

    public function __construct($color = 'blauw', $brand = 'audi') {
        $this->setColor($color);
        $this->setBrand($brand);
    }
    public function __toString() {
        return 'een voertuig met de kleur ' . $this->getColor() . ' en het merk ' . $this->getBrand();
    }
    public function setColor($color){
        $this->color = $color;
    }
    public function getColor(){
        return $this->color;
    }
    public function setBrand($brand){
        $this->brand = $brand;
    }
    public function getBrand(){
        return $this->brand;
    }
}

?>
```

Code Onveranderd

VEHICLE_OVERVIEW.PHP

```
<!DOCTYPE html>
<html>
<body>
<a href="index.php?controller=home&action=home">Home</a>
<a href="index.php?controller=vehicle&action=add">Nieuwe registratie</a>
<br/>
<br/>
<table>
    <th>Kleur</th>
    <th>Merk</th>
    <?php foreach ($vehicles as $vehicle) { ?>
        <tr>
            <td><?php echo $vehicle->getColor(); ?></td>
            <td><?php echo $vehicle->getBrand(); ?></td>
        </tr>
    <?php } ?>
</table>
</body>
</html>
```

Front Controller OO via $_GET

- Uitbreiding
  - als we een detail pagina willen toevoegen waarmee we alle informatie van dat specifiek vehicle willen tonen

Front Controller OO via $_GET

- Problemen?
  - URLs worden te lang en minder leesbaar
  - URLs niet conform de standaard

Front Controller zonder $_GET

- Leesbaarheid verhogen van URLs
  - http://example.com/controller/action/parameters

Front Controller zonder $_GET

- Voorbeelden
  - http://example.com/vehicle/new
  - http://example.com/vehicle/detail/1
- Zie
  - dynweb2013examples/theorie07/frontcontroller_stap03

Front Controller zonder $_GET

- 2 stappen
  - Stap 1: de webserver zeggen dat alles via index.php moet gaan altijd
  - Stap 2: alle informatie uit de URL halen en mappen op de juiste controller, action en parameters

STAP 1

- via mod-rewrite van apache (webserver)
  - .htaccess file maken met daar code in

- http://www.addedbytes.com/articles/for-beginners/url-rewriting-for-beginners/

.HTACCESS

```
RewriteEngine On                                  // activate the rewrite engine
RewriteCond %{REQUEST_FILENAME} -s [OR]           // if (requested filename is a file with a size > 0
RewriteCond %{REQUEST_FILENAME} -l [OR]           // or requested filename is a symbolic link
RewriteCond %{REQUEST_FILENAME} -d                // or requested filename is a directory)
RewriteRule ^.*$ - [NC,L]                         // then show the requested filename
RewriteRule ^.*$ index.php [NC,L]                 // else redirect to index.php page
```

STAP 2

- Mappen structuur volgens MVC patroon
  - application
    - controller
    - model
    - view
  - system
    - controller
    - model
    - view

STAP 2

- Controller klasse wordt gerefactord

Voorbeeld Vehicle Detail

- http://example.com/vehicle/detail/1

INDEX.PHP

```php
<?php
session_start();

// constanten ivm de paths
define('APPLICATION_PATH', 'application/');
define('SYSTEM_PATH', 'system/');

require_once(APPLICATION_PATH . 'config.php');

require_once(SYSTEM_PATH . 'model/Db.php');
require_once(SYSTEM_PATH . 'controller/Controller.php');

$controller = new Controller();
$controller->run();
```

CONTROLLER.PHP

```
<?php

class Controller
{
    const DEFAULT_CONTROLLER = "Home";
    const DEFAULT_ACTION = "index";
    const CONTROLLER_PATH = 'application/controller/';
    const CONTROLLER_FILE = 'index.php';

    private $controller = self::DEFAULT_CONTROLLER;
    private $action = self::DEFAULT_ACTION;
    private $params = array();

    public function __construct() {
        $this->parseUri();
    }
```

CONTROLLER.PHP

```
private function parseUri() {
    // strip the controllerfile out of the scriptname
    $scriptprefix = str_replace(self::CONTROLLER_FILE, '', $_SERVER['SCRIPT_NAME']);
    $uri = str_replace(self::CONTROLLER_FILE, '', $_SERVER['REQUEST_URI']);

    // get the part of the uri, starting from the position after the scriptprefix
    $path = substr($uri, strlen($scriptprefix));

    // strip non-alphanumeric characters out of the path
    $path = preg_replace('/[^a-zA-Z0-9]\//', "", $path);

    // trim the path for /
    $path = trim($path, '/');
```

$_SERVER[‘SCRIPT_NAME’] = /2013/lessons/theorie07/frontcontroller_stap03/index.php
$_SERVER[‘REQUEST_URI’] = /2013/lessons/theorie07/frontcontroller_stap03/vehicle/detail/1
$scriptprefix = /2013/lessons/theorie07/frontcontroller_stap03/
$uri = /2013/lessons/theorie07/frontcontroller_stap03/vehicle/detail/1
$path = vehicle/detail/1

CONTROLLER.PHP

```
    // explode the $path into three parts to get the controller, action and parameters
    // the @-sign is used to supress errors when the function after it fails
    @list($controller, $action, $params) = explode("/", $path, 3);

    if (isset($controller)) {
        $this->setController($controller);
    }
    if (isset($action)) {
        $this->setAction($action);
    }
    if (isset($params)) {
        $this->setParams(explode("/", $params));
    }
}
```

$controller = vehicle
$action = detail
$params = 1

CONTROLLER.PHP

```
private function setController($controller) {
    $controller = ($controller) ? $controller : self::DEFAULT_CONTROLLER;

    $controllerfile = self::CONTROLLER_PATH . ucfirst(strtolower($controller)) . 'Controller' . '.php';

    // check if controller file exists
    if (!file_exists($controllerfile)) {
        die("Controller '$controller' could not be found.");
    } else {
        require_once($controllerfile);
        $this->controller = $controller. 'Controller';
    }

    return $this;
}
private function setAction($action) {...}
private function setParams(array $params) {...}
```

$this->controller = vehicleController

$this->action = detail

$this->params = Array

array(1) { [0]=> string(1) "1"}

CONTROLLER.PHP

```
    public function run()
    {
        // checking the parameter count, using Reflection (http://www.php.net/reflection)
        $reflector = new ReflectionClass($this->controller);
        $method = $reflector->getMethod($this->action);
        $parameters = $method->getNumberOfRequiredParameters();

        if (($parameters) > count($this->params)) {
            die("Action '$this->action' in class '$this->controller' expects $parameters mandatory
parameter(s), you only provided " . count($this->params) . ".");
        }

        // create an instance of the controller as an object
        $controller = new $this->controller();

        // call the method based on $this->action and the params
        call_user_func_array(array($controller, $this->action), $this->params);
    }

}
```

$reflector = Class [class VehicleController]
$method = Method [public method detail]
$parameters = 1

VehicleController.php

```
<?php
class VehicleController
{

    private $_vehicleMapper;
    private $_vehicle;

    public function __construct ()
    {
        require_once(APPLICATION_PATH . 'model/VehicleMapper.php');
        $this->_vehicleMapper = new VehicleMapper();
    }

    public function index() {...}

    public function detail($id)
    {
        $this->_vehicle = $this->_vehicleMapper->get($id);
        require_once(APPLICATION_PATH .'view/vehicle_detail.php');
    }

    public function add() {...}

}
```

Code Onveranderd

VehicleMapper.php

```
<?php
require_once(SYSTEM_PATH . 'model/Db.php');
require_once(APPLICATION_PATH . 'model/Vehicle.php');

class VehicleMapper
{
    private $_db;

    public function __construct()
    {
        $this->_db = Db::getInstance();
    }

    public function add($object) {...}

    public function getAll() {...}

    public function get($id)
    {
        $id = (int)$id;
        $query = "
            SELECT *
            FROM vehicles
            WHERE id = ?;
        ";
        return $this->_db->queryOne($query, 'Vehicle', array($id));
    }

}
```

DB.PHP

```
<?php
class Db
{
    private static $instance = null;
    private $_db;

    ...

    public function execute($sql, $arguments = array()) {
        if (!is_array($arguments)) {
            $arguments = array($arguments);
        }

        try {
            $stmt = $this->_db->prepare($sql);
            $stmt->execute($arguments);
            $stmt->setFetchMode(PDO::FETCH_ASSOC);
        } catch(PDOException $e) {
            error_log($e->getMessage());
        }

        return $stmt;
    }
```

DB.PHP

```
    public function queryAll($sql, $type, $arguments = array()) {
        $stmt = $this->execute($sql, $arguments);

        return $stmt->fetchAll(PDO::FETCH_CLASS, $type);
    }

    public function queryOne($sql, $type, $arguments = array()) {
        $stmt = $this->execute($sql, $arguments);

        return $stmt->fetchObject($type);
    }
}
```

VEHICLE.PHP

```php
<?php
class Vehicle
{
    private $color;
    private $brand;

    public function __construct() {
    }

    public function __toString() {
        return 'een voertuig met de kleur ' . $this->getColor() . ' en het merk ' . $this->getBrand();
    }

    public function setColor($color) {
        $this->color = $color;
    }

    public function getColor() {
        return $this->color;
    }

    public function setBrand($brand) {
        $this->brand = $brand;
    }

    public function getBrand() {
        return $this->brand;
    }
}
```

VEHICLE_DETAIL.PHP

```
<!DOCTYPE html>
<html>
<body>
<table>
    <th>Kleur</th>
    <th>Merk</th>
        <tr>
            <td><?php echo $this->_vehicle->getColor(); ?></td>
            <td><?php echo $this->_vehicle->getBrand(); ?></td>
        </tr>
</table>
</body>
</html>
```

Code Onveranderd

Front Controller zonder $_GET

- Uitbreiding
  - Categorie toevoegen

Front Controller zonder $_GET

- Problemen?
  - veel dubbele code nu

FRONT CONTROLLER ZONDER $_GET

- Refactoren dubbele code
  - Mapper klasse
  - Identifiable klasse

- Zie
  - dynweb2013examples/theorie07/frontcontroller_stap04

MAPPER.PHP

```
<?php
require_once(SYSTEM_PATH . 'model/Db.php');

class Mapper
{

    protected $_db;
    protected $_table;
    protected $_type;

    public function __construct($table, $type)
    {
        $this->_db = Db::getInstance();
        $this->_table = $table;
        $this->_type = $type;
    }

...
```

MAPPER.PHP

```
  public function get($id)
   {
      $query = "
         SELECT *
         FROM $this->_table
        WHERE id = ?
      ";

      return $this->_db->queryOne($query, $this->_type, array($id));
   }

   public function getAll()
   {
      $query = "
         SELECT *
         FROM $this->_table
      ";

      return $this->_db->queryAll($query, $this->_type);
   }

    ...

}
```

VehicleMapper.php

```
<?php
require_once(APPLICATION_PATH . 'model/Vehicle.php');
require_once(SYSTEM_PATH . 'model/Mapper.php');

class VehicleMapper extends Mapper
{
    public function __construct()
    {
        parent::__construct('vehicles', 'Vehicle');
    }
}
```

IDENTIFIABLE.PHP

```php
<?php

class Identifiable {
    private $_id;

    public function getId() {
        return $this->_id;
    }

    public function setId($id) {
        $this->_id = $id;
    }

    function __get($property) {
        $method = "get{$property}";
        if (method_exists($this, $method)) {
            return $this->$method();
        }
    }

    public function __set($property, $value){
        $method = "set{$property}";
        if (method_exists($this, $method)) {
            $this->$method($value);
        }
    }

}
```

VEHICLE.PHP

```
<?php
require_once(SYSTEM_PATH . 'model/Identifiable.php');

class Vehicle extends Identifiable
{
    private $_color;
    private $_brand;

    public function __construct() {
    }

    public function setColor($color) {
        $this->_color = $color;
    }

    public function getColor() {
        return $this->_color;
    }

    public function setBrand($brand) {
        $this->_brand = $brand;
    }

    public function getBrand() {
        return $this->_brand;
    }
```

VEHICLE.PHP

```
    public function __toString() {
        return 'een voertuig met de kleur ' . $this->getColor() . ' en het merk ' . $this->getBrand();
    }

    public function toArray() {
        $fields['color'] = $this->_color;
        $fields['brand'] = $this->_brand;
        return $fields;
    }
}
```

Agenda

- Lesschema
- Herhaling
- Front Controller

VOORBEREIDING LABO

- Bekijk de code in svn
  - dynweb2013examples/theorie07/frontcontroller

Voorbereiding Labo

- URLs
  - home
  - home/display/elke
  - vehicle
  - vehicle/detail/1
  - category
  - category/detail/2