Develop a simple MVC framework in PHP, the use case of SimpleMVC

by Enrico Zimuel
PHP User Group Torino

Feb 23, 2023 - Toolbox CoWorking

PUG Torino

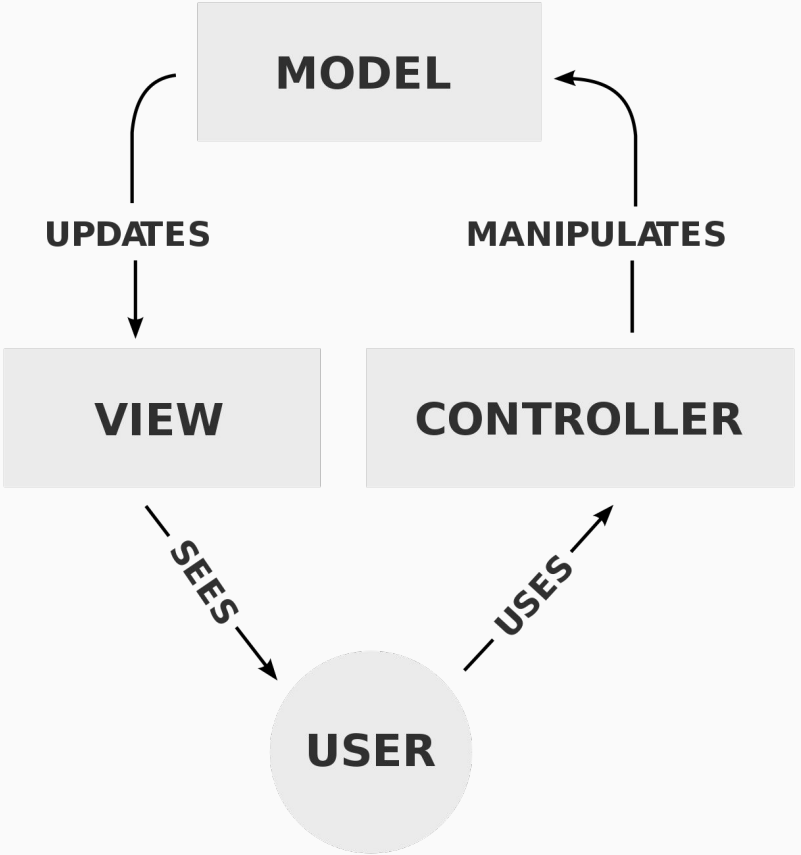
- PHP User Group Torino is a group of web developers interested in the PHP language (and not only)
- We are part of GrUSP association
- On meetup.com we are about 600 members
- We have also a mailing list with more than 100 members
- In the past we organized conferences like PHP.TO.START (2011, 2012, 2013) and Zend Framework Day (2014)
- Toolbox Coworking is sponsoring the group

Model-View-Controller

MVC

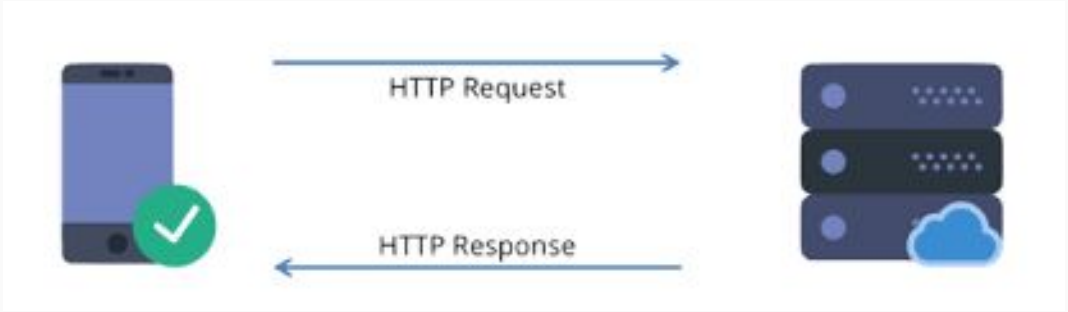
- Il **Model-View-Controller** (**MVC**) is an architectural pattern commonly used for developing user interfaces that divide the related program logic into three interconnected elements
- It separates internal representations of information from the ways information is presented to and accepted from the user
- Es. in a web application a catalog page can use the same **Model** for two different **Views**, HTML and JSON

MVC
diagram

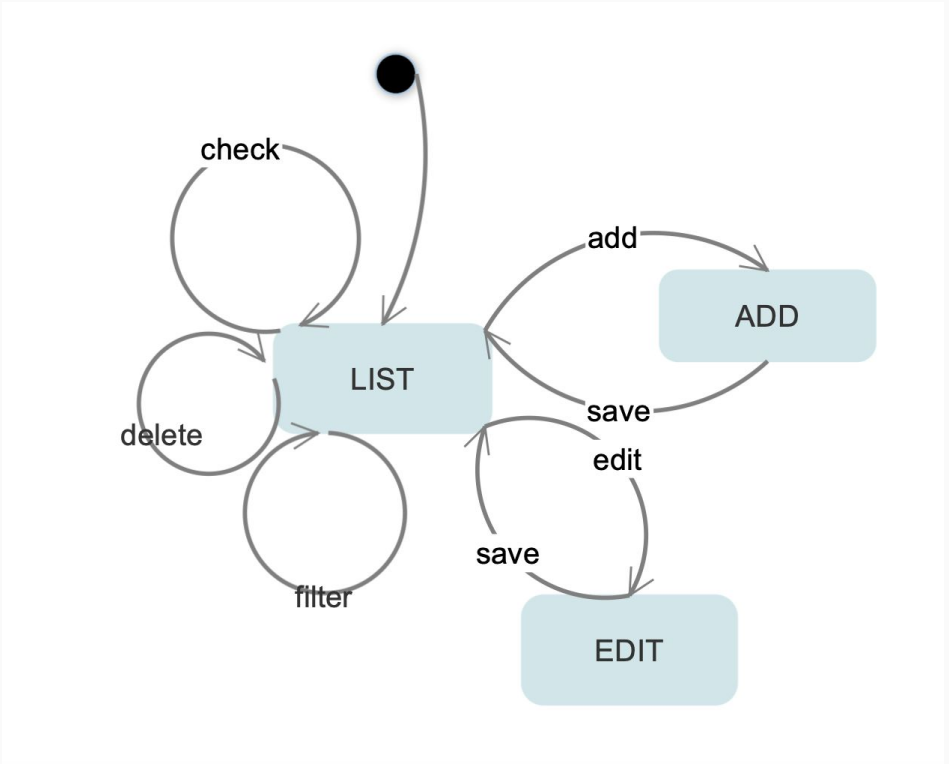


Web application

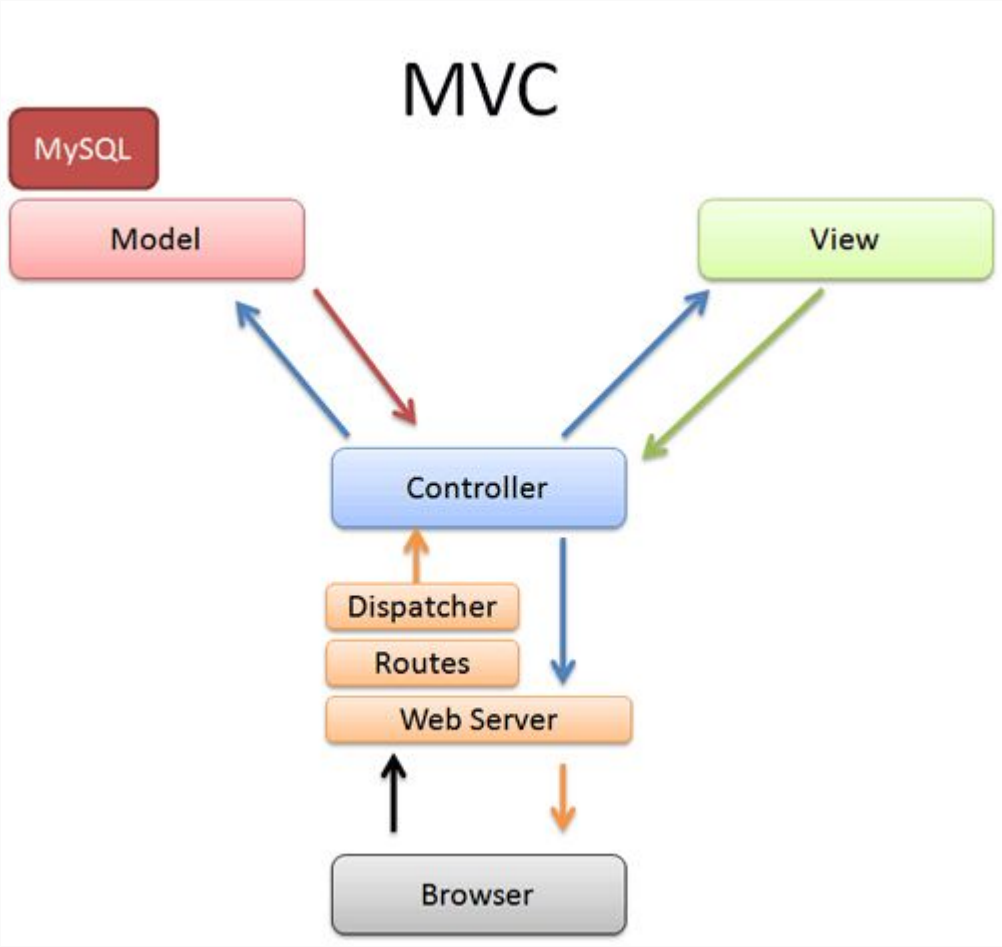
Web application
is a black box that get an HTTP request and produce an HTTP response



State machine



MVC
in a WEB app



Components
of an MVC web application

Components

- Managing HTTP request, response
- Routing
- Dispatch -> choose the Controller to be executed
- Controller
- Model, used by the Controller to extract the data
- View, render the data extracted from the Model

HTTP requests and responses

- PHP manages HTTP requests using global variables $_GET, $_POST, $_REQUEST, $_COOKIE, etc
- Luckily in PHP we have PSR-7 standard
- We can manage HTTP requests and responses using immutable objects!

PSR-7

- **$request->getHeader('foo')**
- **$response->withHeader('foo', 'bar')**
- **$request->getMethod()**, returns GET, POST, etc
- **$request->getBody()**, returns the body as stream
- **$request->getBody()->getContents()**, returns the body string
- **$response->withStatus(404)**

Routing

- Routing is the action of executing a specific portion of the Application (eg. Controller) when an URL request is coming
- GET /login -> execute the LoginController
- A routing systems needs to manage **parameters**,
  eg. /users[/{id}]

Mapping and dispatch

- We need to map URLs with Controllers
- Many frameworks use @annotation
  - Elegant from a code perspective
  - Not so convenient when you need to find a route
- Having a single file that stores this mapping can simplify the management

Complexity

- Many controllers, models and views increase the complexity of the project
- Especially the controllers that consume models and views
- We need to find a way to simplify the management of the class connections
- We have a solution, **Dependency injection**!

Dependency Injection

- **Dependency Injection (DI)** is a design pattern in which an object or function receives other objects or functions that it depends on

```
class Home implements ControllerInterface
{
    protected Engine $plates;

    public function __construct(Engine $plates)
    {
        $this->plates = $plates;
    }
```

Why DI is such important?

- DI **simplify** the complexity of the object dependency
- Makes the dependency between objects **explicitly**
- Makes the code **testable**

PHP-DI

- PHP-DI is a dependency injection container for PHP
- A **container** is a collection of object with all the dependencies resolved
- Implements the **autowiring** feature
  - the ability of the container to automatically create and inject dependencies

Autowiring

No autowiring

```php
class Cart
{
    public function __construct(PDO $pdo)
    {
        $this->pdo = $pdo;
    }
    // ...
}
$pdo = new PDO(/* parameters */);
$cart = new Cart($pdo);
```

Autowiring

```php
class Cart
{
  public function __construct(PDO $pdo)
  {
      $this->pdo = $pdo;
  }
  // ...
}
$container = new DI\Container();
$cart = $container->get('Cart');
```

SimpleMVC

SimpleMVC

- SimpleMVC is a mini framework MVC for PHP
- PSR-7 for managing HTTP requests and responses
- Routing system, using FastRoute
- Dependency injection using PHP-DI
- Focused on the KISS principle
- Developed from the teaching experience at ITS-ICT Piemonte

Demo time!

Thanks!

Contacts:
Enrico Zimuel
enrico@zimuel.it

PHP User Group Torino
torino.grusp.org