Framework PHP

Framework ?

Infrastructure logicielle : composants logiciels organisés suivant des patrons de conception.

Un framework peut utiliser des bibliothèques.

Les bibliothèques sont passives : les fonctions sont en attente d'un appel.

Les framework sont actifs : le déroulement des appels et dirigé par le framework => **inversion de contrôle**

Framework

Logiciel applicatif : Cocoa (Objective C), Microsoft Foundation Class, .net

Persistance : Hibernate, TopLink, …

Application Web :

- Symfony , Laravel, CodeIgniter, …
- Spring, JavaServerFaces, …
- Ruby on Rails
- Django, ….

CodeIgniter 4

- https://codeigniter.com/
- MVC
- https://codeigniter.com/user_guide/intro/requirements.html
  - PHP8.1
  - Intl, mbstring, json

CI 4 installation

- Composer
  - un logiciel gestionnaire de dépendances
    - Installation : https://getcomposer.org/download/
  - Nécessite git
  - Starter : composer create-project codeigniter4/appstarter project-root

- Docker exemple d'image `podman run -e DEBUG=1 -d -v ./container_code:/var/www/html -p 8080:80 shinsenter/php:8.4-fpm-apache`
  (montage avec un répertoire local)

- Manuelle

Mon premier site

Téléchargement du starter

```
composer create-project codeigniter4/appstarter demo-amphi
```

Lancement du serveur de développement sur le port 8000

```
php spark serve --port 8000
```

Analyse de la première page- routage -

http://localhost:8000/

#route.php

```
$routes->get('/', 'Home::index’);
```

La racine est mappée vers la méthode index du contrôleur Home

```
#une nouvelle route
$routes->get('/demo', 'Home::index');
```

Analyse de la première page – contrôleur -

```
class Home extends BaseController
{
    public function index(): string
    {
        return view('welcome_message', );
    }
}
```

Création d’une App – config de base

.env ou config

CI_ENVIRONMENT = development

app.baseURL = 'http://localhost:8000’

⇒La barre de debug est activée

Création d’une App – BD -configuration et création

database.default.database = "demo.db"
database.default.DBDriver = "SQLite3"

```
php spark db:create demo.db
```

(dans writable)

Création d'une App - BD - migrations

```
php spark make:migration create_product
```

```
class CreateProduct extends Migration
{
   public function up()
   {
      $this->forge->addField(array(
         'id' => array(
            'type' => 'INT'
         ),
..
$this->forge->addKey('id', true);
      $this->forge->createTable('product');
   }

   public function down()
   {
      $this->forge->dropTable('ptoduct', true);
   }
}
```

```
php spark migrate
php spark db:table product
```

Création d'une App – BD seed -

```
php spark make:seeder ProductSeeder
```

```
class ProductSeeder extends Seeder
{
    public function run()
    {
        $data = [
            ['id' => 1, 'name'=>'p1', 'price'=>10.0, 'quantity'=>1],
            ['id' => 2, 'name'=>'p2', 'price'=>20.0, 'quantity'=>1]
        ];
        foreach ($data as $row)
            $this->db->table('product')->insert($row);
    }
}
```

```
php spark db:seed ProductSeeder
```

```
php spark db:table product
```

Création d’une App - Entité

```
php spark make:entity Product
```

Création d’une App - Modèle

```
php spark make:model ProductModel
```

```
class ProductModel extends Model
{
    protected $table          = 'products';
    protected $primaryKey     = 'id’;

…
// Dates
// Validation
// Callbacks
}
```

Création d'une App – Création des contrôleurs

- Home réutilisé pour afficher la page d'accueil
- Product pour gérer les produits

```
php spark make:controller Product
```

```
class Product extends BaseController
{
    public function index()
    {
        //
    }
}
```

Création d'une App – affichage de la page d'accueil

- Dans le contrôleur Home
  - On sollicite le modèle et on appelle la vue en lui passant les données

```
class Home extends BaseController
{
    private $productModel;

    public function __construct() {
            $this-> productModel = model('ProductModel');
    }
    public function index(): string
    {

        $products = $this->productModel->findAll();
        return view('home',array("products"=>$products));
    }
}
```

Création d'une App – affichage de la page d'accueil

- Dans la vue
  - On traite les données venant du contrôleur

```
<?php foreach ($products as $product):?>
   <tr>
      <td> <?= $product->id ?> </td>
      <td> <?= $product->name ?> </td>
      <td> <?= $product->price ?> </td>
      <td> <?= $product->qunatity ?> </td>
      <td>  remove </td>
      <td> update </td>
   </tr>
<?php endforeach; ?>
```

Création d'une App – Suppression d'un produit

- On ajoute une route

```
$routes->get('product/remove/(:num)', 'Product::remove/$1');
```

- On ajoute une méthode dans le contrôleur

```
public function remove($id)
{
    $data = [
        'id'   => $id,
    ];

    $rule = [
        'id'   => 'integer',
    ];

    if (! $this->validateData($data, $rule)) {
        return redirect()->back()->with('error', 'Product not found');
    }
    try {
        $products = $this->productModel->delete($id);

    }
    finally {
        return redirect("/");
    }
}
```

Création d'une App – Suppression d'un produit

- On modifier la vue

```
<td> <a href="<?=base_url('product/remove/'.$product->id)?>"> remove
</a></td>
```