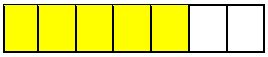
Symfony 4

Version 1.1.0

Niveau requis : 5/7



Mise en œuvre de Symfony 4

Sommaire

I. Préambule

I.I. *Objet*

L'objet de ce document est de décrire comment installer la version projet de Symfony 4 et de le personnaliser pour en réaliser un site web de gestion des cours.

I.II. *Pré-requis*

Avoir suivi le cours « Mise en oeuvre de Doctrine ».

Avoir installé Visual Studio Code comme IDE (IDE Gratuit) avec la prise en compte des extensions PHP, PHP Debug et Composer.

Nous continuerons par la suite d'utiliser VSCode qui permet aussi d'avoir le mode Debug et d'éditer du PHP sans payer de licence.

I.III. *Versions du document*

Version	Date	Auteur	Description
1.0.0	08/05/2018	Péquignat.eu	Version initiale du document
1.1.0	08/04/2022	Péquignat.eu	Retrait de l'Auto-Entreprise

I.IV. *Documents de référence*

#	Document	Version	Auteur(s)
[R1]	Mise en œuvre de Doctrine	1.2.0	Péquignat.eu

II. Installation de la version de Demo de Symfony 4

II.I. *Installation Demo Symfony*

Aller dans votre répertoire en ligne de commande

```
C:\wamp64\www
```

Lancer la commande **composer** suivante :

```
composer create-project symfony/symfony-demo
```

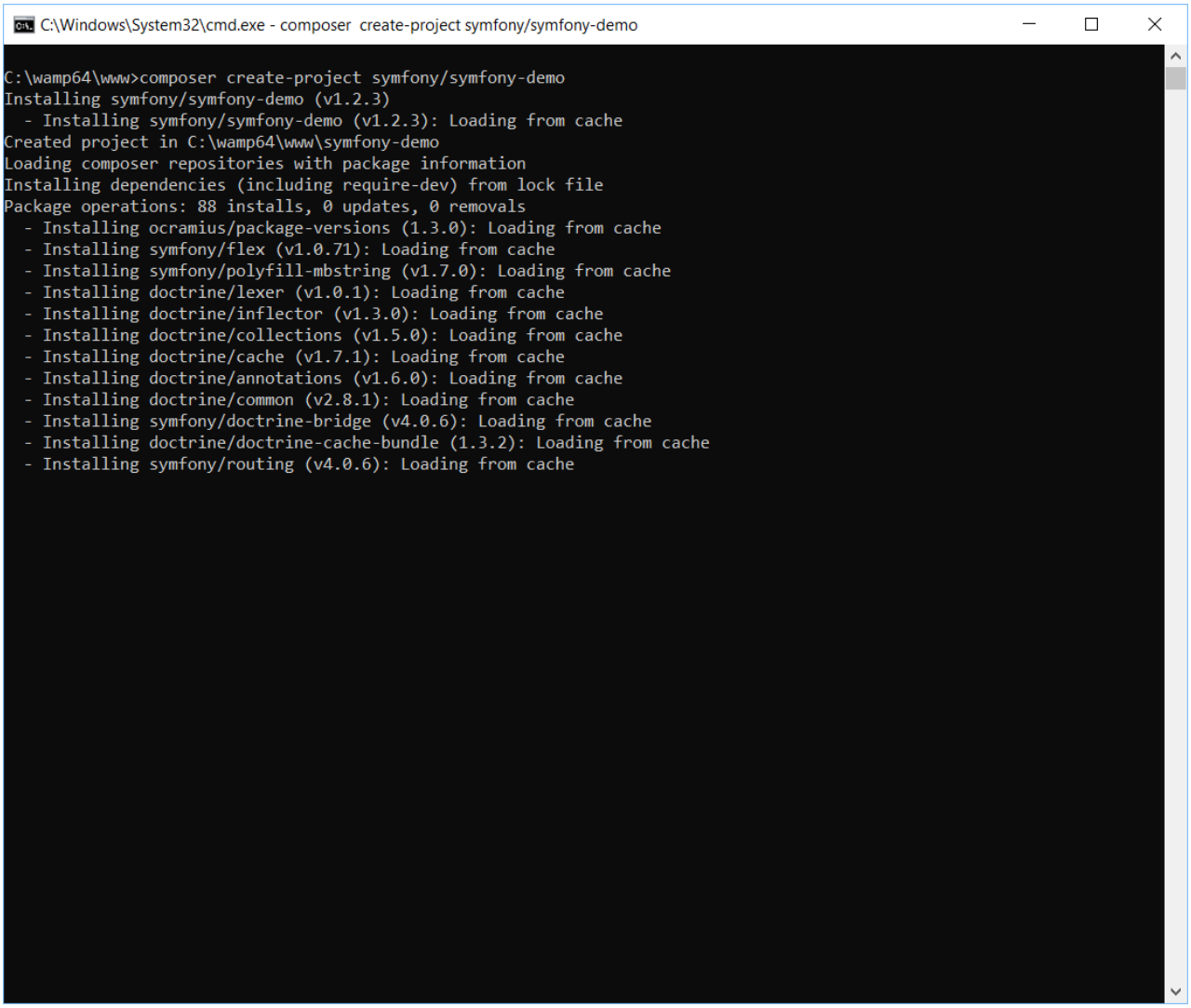


Figure 1 - Installation demo symfony

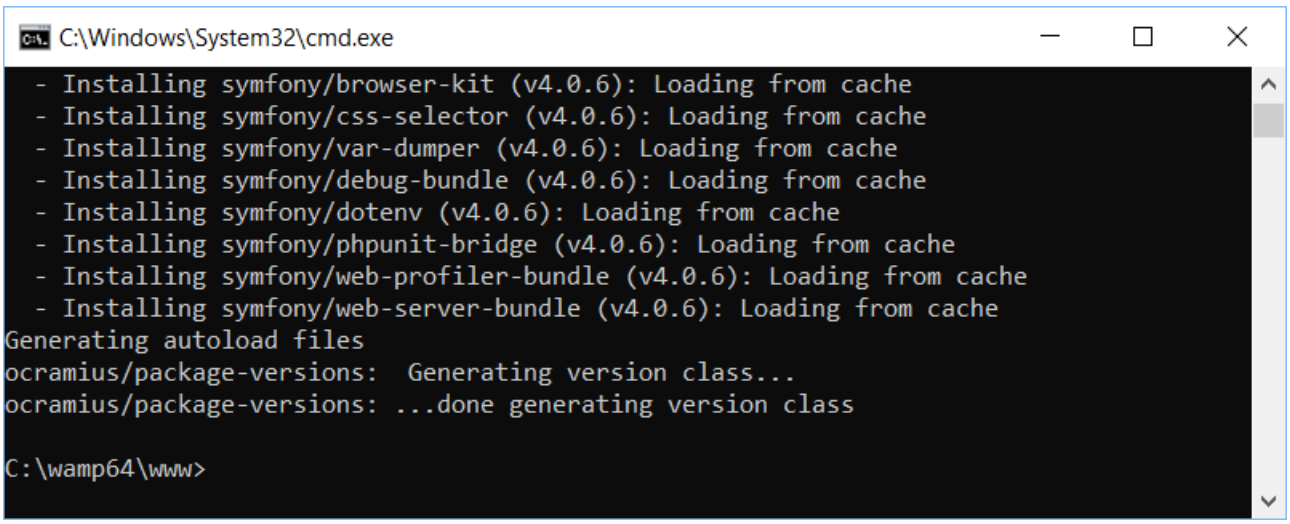


Figure 2 - Fin installation Symfony de base

Maintenant que Symfony Demo vous pouvez le tester de deux manières :

- Soit en lancer la commande PHP serveur

- Soit en installant un Virtual Host dans Apache

II.II. *Lancement serveur PHP*

Dans la ligne de commande faîtes, pour vous situer dans le répertoire créé :

```
cd symfony-demo/
```

Puis lancer le serveur directement :

```
php bin/console server:run
```

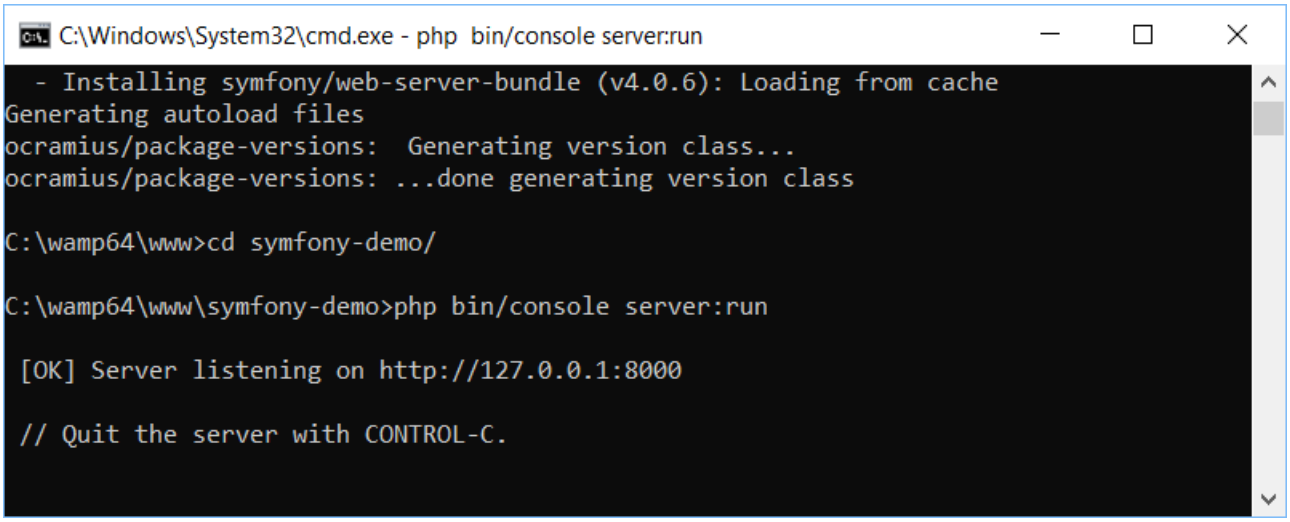


Figure 3 - Lancement en mode serveur

II.III. *Visualiser le site*

Pour visualiser le site : cliquer sur le lien : http://localhost:8000

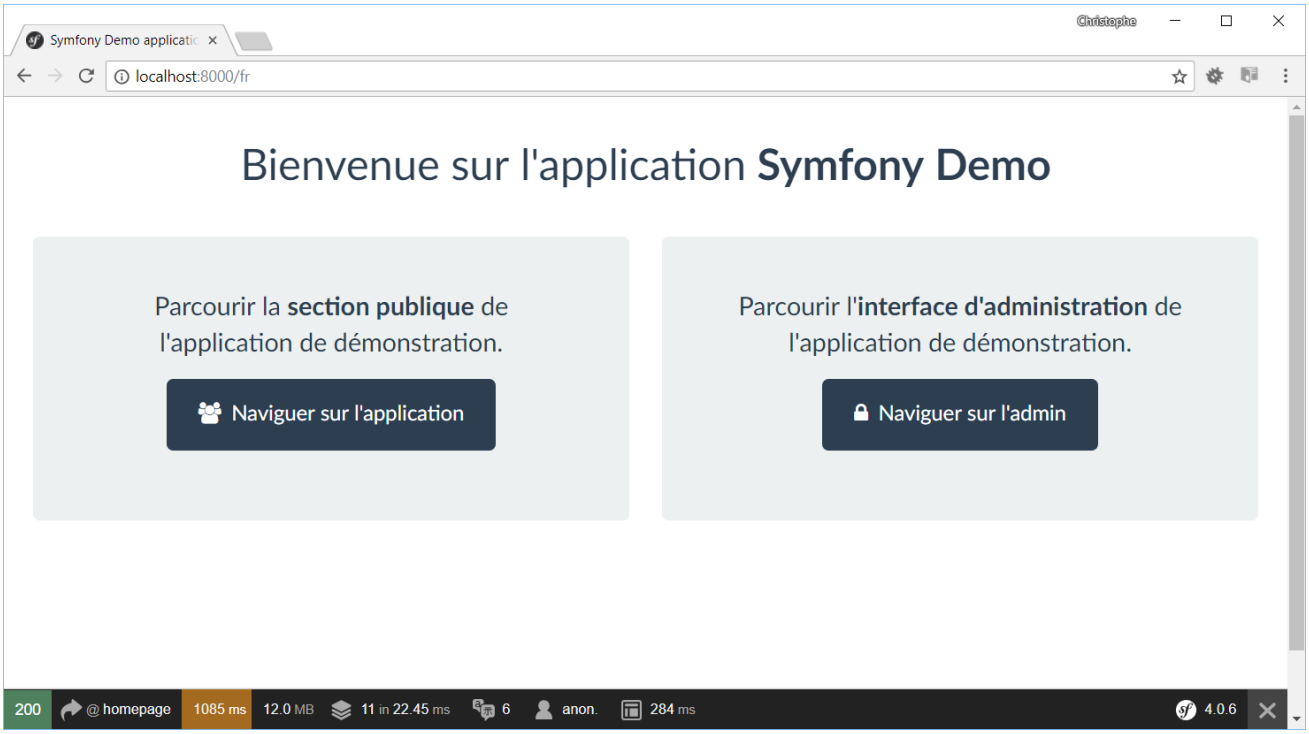


Figure 4 - Site demo en mode serveur PHP

Vous noterez que le site à détecté automatiquement la langue du navigateur et est gérer en mode multi-langue.
En bas, vous avez la barre de **Profiling** qui n'est affiché que pour le site en mode développement.

II.IV. *Lancement par serveur Apache*

II.IV.1 Installation du pack apache

Afin de pouvoir lancer convenablement avec un support apache, il convient de lancer en plus une instruction **composer** suivante : (A la racine du projet)

```
composer require symfony/apache-pack
```

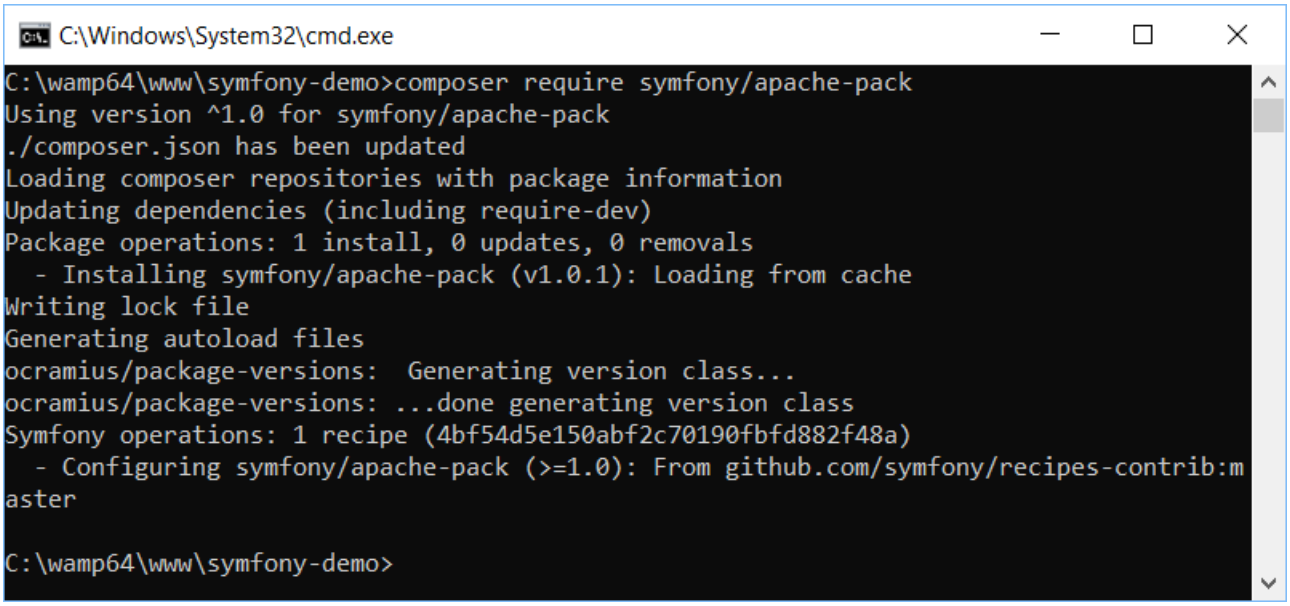


Figure 5 - Lancement du pack Apache

II.IV.2 Mise en place du Virtual Host

Cliquer sur le lien : http://localhost/add_vhost.php?lang=french

Renseigner avec les informations suivantes :

Le nom :

```
DemoSymfony
```

Le Chemin :

```
C:/wamp64/www/symfony-demo/public/
```

Et cliquer sur Démarrer la création …

Figure 6 - Ajout réalisé avec succès

Redémarrer les services Wamp

II.IV.3 Visualiser dans Apache

Cliquer sur le lien : http://demosymfony

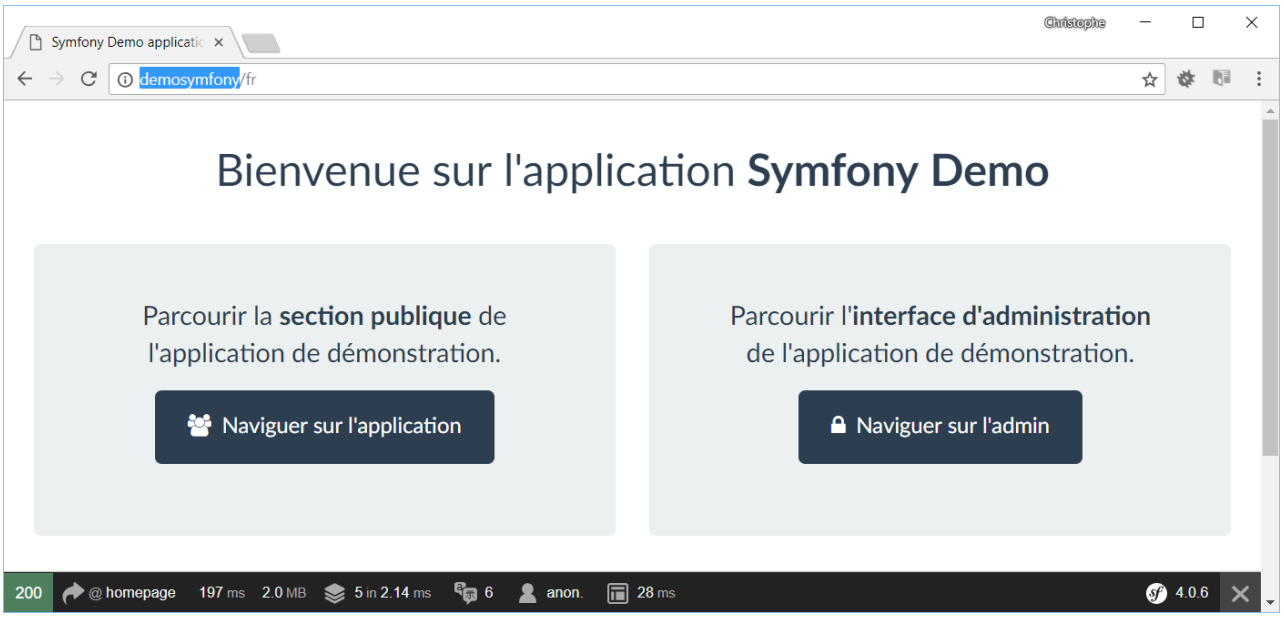


Figure 7 - Visualisation par serveur apache

De nouveau le site est détecté comme étant ouvert depuis un navigateur en Français.

Cette version de démo est là pour vous montrer une application réalisée en Symfony et du coup d'avoir un exemple pour apprendre par comparaison.

III. Installation de la version projet de Symfony 4

Relancer une commande MSDOS (cmd) dans votre répertoire C:/wamp64/www
Et entrer la ligne suivante :

```
composer create-project symfony/skeleton CoursSymfony
```

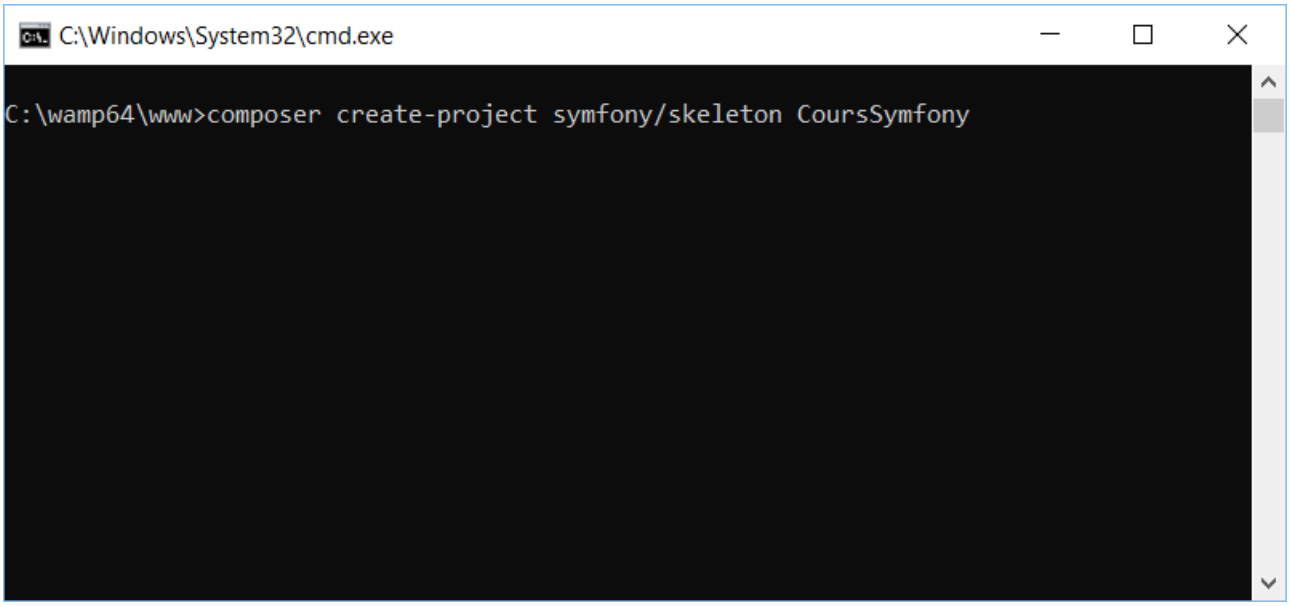


Figure 8 - Création du projet Symfony CoursSymfony

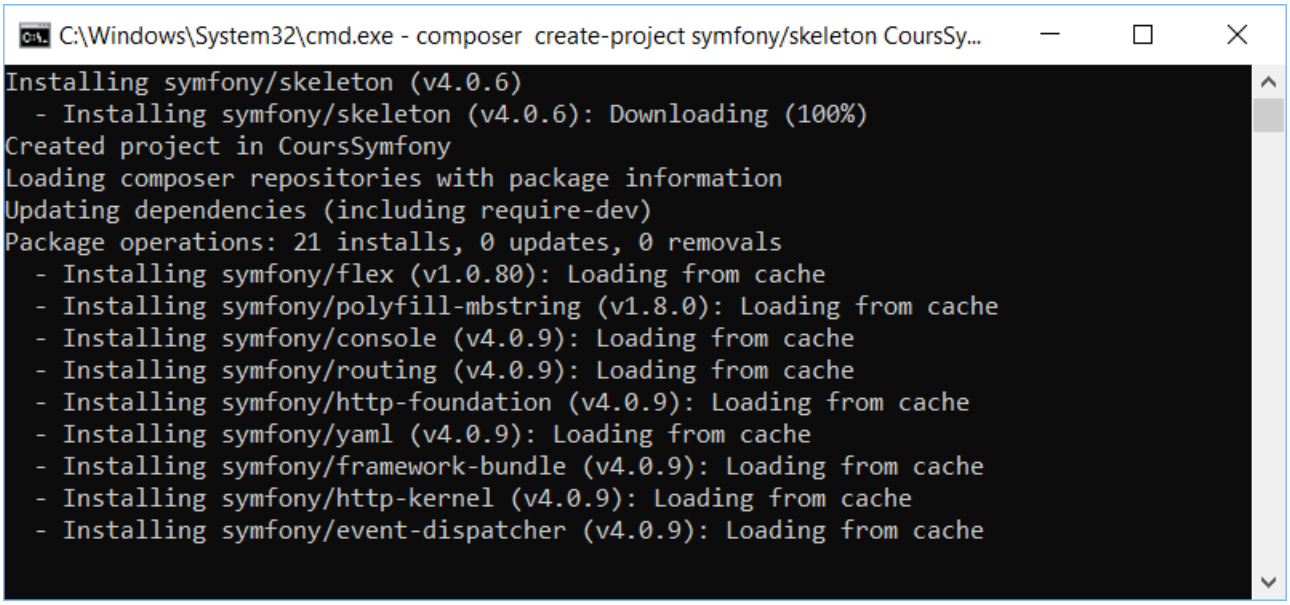


Figure 9 - Installation en cours

Installation du pack apache

```
cd CoursSymfony
composer require symfony/apache-pack
```

Répondre « y » à la question.

Installer un VirtualHost CoursSymfony

Cliquer sur le lien : http://localhost/add_vhost.php?lang=french

Renseigner avec les informations suivantes :
Le nom :

```
CoursSymfony
```

Le Chemin :

```
C:/wamp64/www/CoursSymfony/public/
```

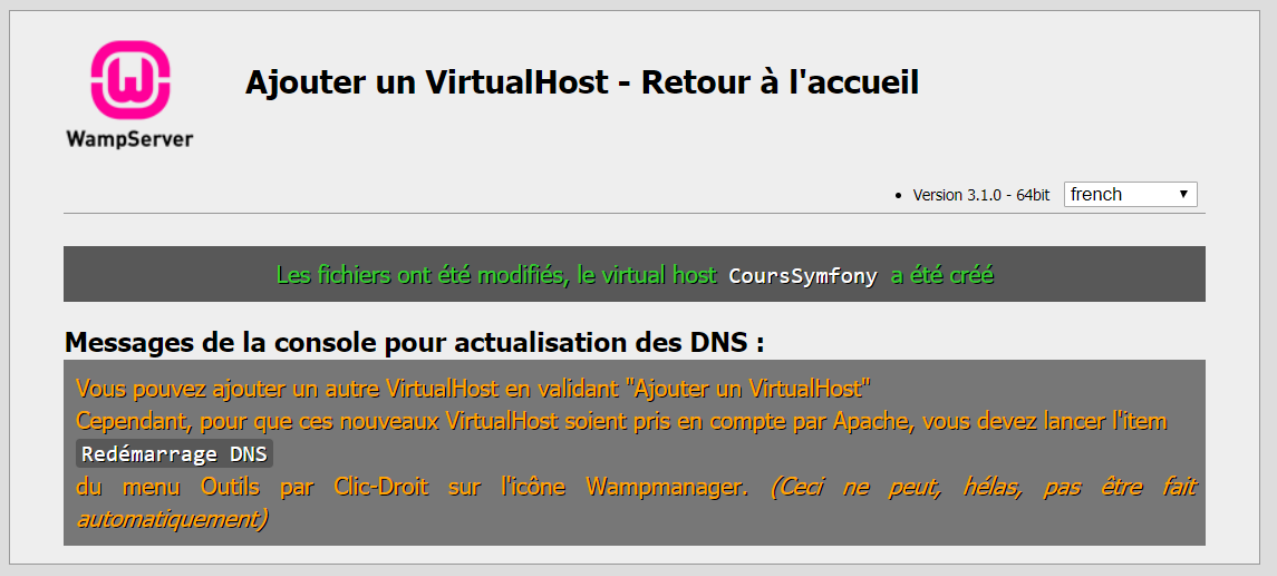


Figure 10 - Installation VirtualHost CoursSymfony

IV.Mise en œuvre du Projet

IV.I. *Ouvrir le projet sous de VSCode :*

A partir de maintenant nous allons utiliser l'IDE VSCode.

Ouvrir le répertoire :

```
C:\wamp64\www\CoursSymfony
```

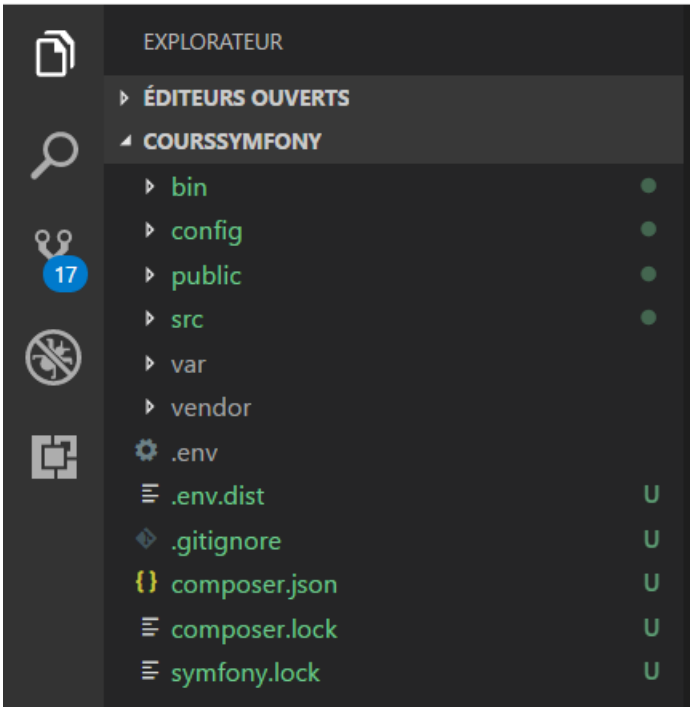


Figure 11 - VSCode

IV.II. *Installer les plugins dans l'IDE VSCode*

Vous devez avoir installé les extensions :

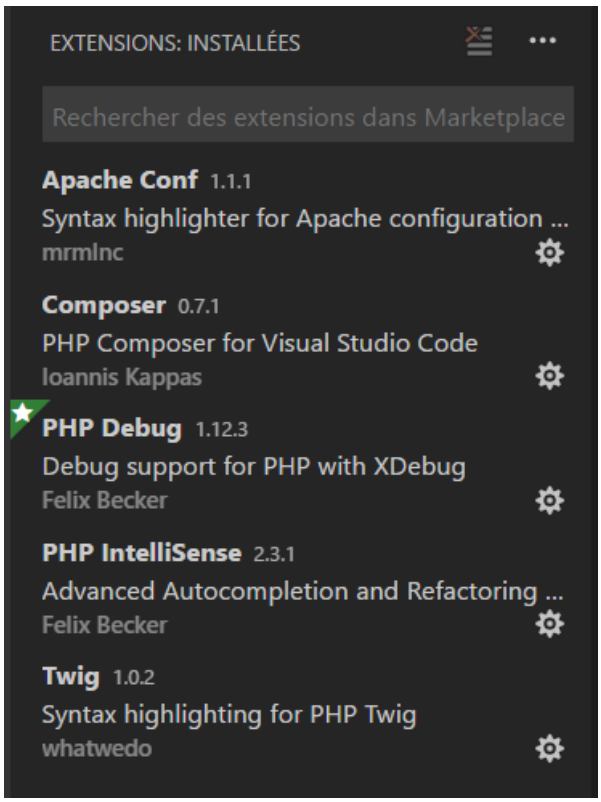


Figure 12 - Extensions VSCode Installées

IV.III. *Installer le « package » annotations*

Dans le terminal faîtes :

```
composer require annotations
```

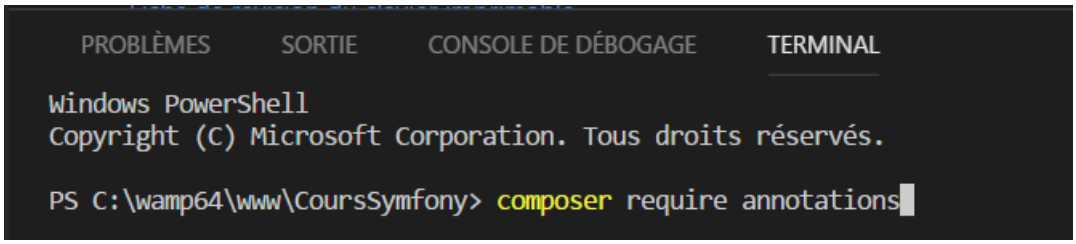


Figure 13 - Installation Annotations

```
PS C:\wamp64\www\CoursSymfony> composer require annotations
Using version ^5.1 for sensio/framework-extra-bundle
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 7 installs, 0 updates, 0 removals
  - Installing doctrine/lexer (v1.0.1): Loading from cache
  - Installing doctrine/inflector (v1.3.0): Loading from cache
  - Installing doctrine/collections (v1.5.0): Loading from cache
  - Installing doctrine/cache (v1.7.1): Loading from cache
  - Installing doctrine/annotations (v1.6.0): Loading from cache
  - Installing doctrine/common (v2.8.1): Loading from cache
  - Installing sensio/framework-extra-bundle (v5.1.6): Loading from cache
Writing lock file
Generating autoload files
Symfony operations: 2 recipes (3e7dfaed729c33eb4dd8b43c0e1bcd5d)
  - Configuring doctrine/annotations (>=1.0): From github.com/symfony/recipes:master
  - Configuring sensio/framework-extra-bundle (>=4.0): From github.com/symfony/recipes:master
Executing script cache:clear [OK]
Executing script assets:install --symlink --relative public [OK]

Some files may have been created or updated to configure your new packages.
Please review, edit and commit them: these files are yours.

PS C:\wamp64\www\CoursSymfony>
```

Figure 14 - Installation annotations OK

IV.IV. *Installer la barre de debug « profiling »*

Entrer dans le terminal :

```
composer require --dev profiler
```

```
PS C:\wamp64\www\CoursSymfony> composer require --dev profiler
Using version ^1.0 for symfony/profiler-pack
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 8 installs, 0 updates, 0 removals
  - Installing twig/twig (v2.4.8): Loading from cache
  - Installing symfony/polyfill-php72 (v1.8.0): Loading from cache
  - Installing symfony/var-dumper (v4.0.9): Loading from cache
  - Installing symfony/twig-bridge (v4.0.9): Loading from cache
  - Installing symfony/web-profiler-bundle (v4.0.9): Loading from cache
  - Installing symfony/twig-bundle (v4.0.9): Loading from cache
  - Installing symfony/stopwatch (v4.0.9): Loading from cache
  - Installing symfony/profiler-pack (v1.0.3): Loading from cache
Writing lock file
Generating autoload files
Symfony operations: 2 recipes (4e8014456210ba9e904fd1ff372b1d47)
  - Configuring symfony/web-profiler-bundle (>=3.3): From github.com/symfony/recipes:master
  - Configuring symfony/twig-bundle (>=3.3): From github.com/symfony/recipes:master
Executing script cache:clear [OK]
Executing script assets:install --symlink --relative public [OK]

Some files may have been created or updated to configure your new packages.
Please review, edit and commit them: these files are yours.

PS C:\wamp64\www\CoursSymfony>
```

Figure 15 - Installation de la barre de debug

Vous devriez voir en base de votre site la barre de debug apparaître :

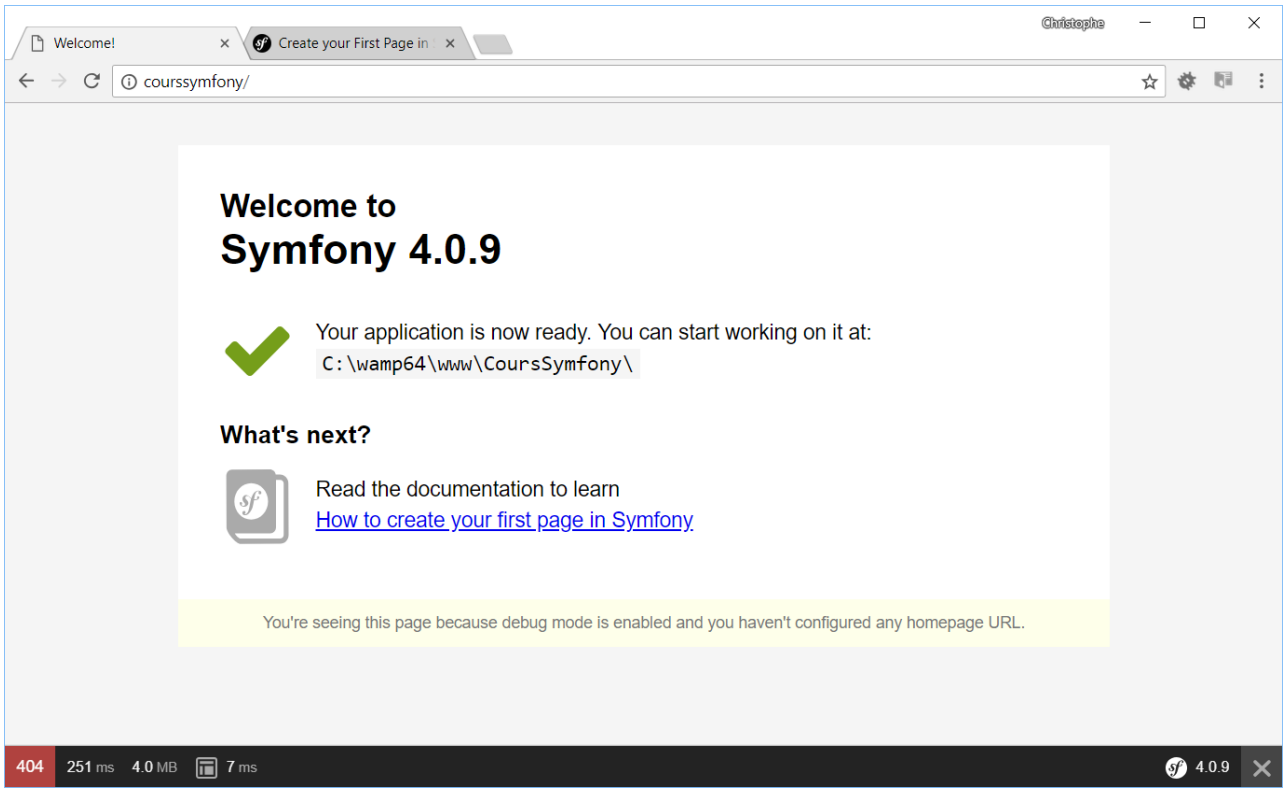


Figure 16 - barre de debug

IV.V. *Ajout du moteur de templating twig*

Dans le terminal lancer la commande :

```
composer require twig
```

```
PS C:\wamp64\www\CoursSymfony> composer require twig
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Nothing to install or update
Writing lock file
Generating autoload files
Executing script cache:clear [OK]
Executing script assets:install --symlink --relative public [OK]

PS C:\wamp64\www\CoursSymfony>
```

Figure 17 - Initialisation de TWIG

IV.VI. *Mise en œuvre d'un premier Controller*

Créons un fichier DefaultController.php dans le répertoire

```
src/Controller/DefaultController.php
```

Dont le contenu est le suivant :

```php
<?php
// src/Controller/DefaultController.php
namespace App\Controller;

use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class DefaultController
{
    /**
     * @Route("/")
     */
    public function index()
    {
        $number = mt_rand(0, 100);

        return new Response(
            '<html><body>Lucky number: '.$number.'</body></html>'
        );
    }
}
```

Cette page renvoie directement une page HTML dans l'objet Response avec un nombre entier compris entre 0 et 100.

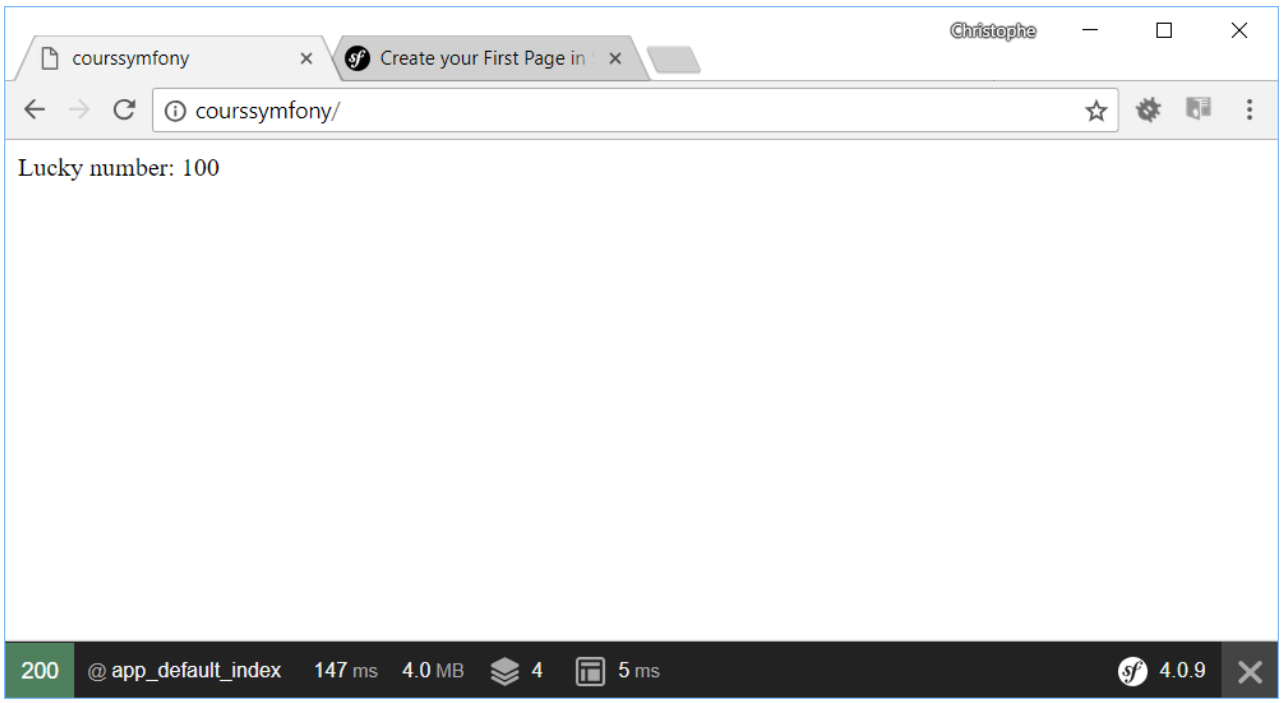


Figure 18 - Default Controller sur la route « / »

V. Création du contenu du projet de CoursSymfony

V.I. *Première page en utilisant TWIG*

V.I.1 Qu'est ce que TWIG

TWIG est un moteur de templating qui permet de s'abstraire de la partie codage de PHP en se positionner sur la notion de rendu graphique. Cela permet de faciliter le travail des web designer et ainsi de mieux focaliser le codage dans les couches PHP applicative comme le Controller, Service, Repository, Modèle Entity (Entité mappé en base de données).

V.I.1 Le Controller DefaultController

Le Controller se transforme ainsi :

```php
<?php
// src/Controller/DefaultController.php
namespace App\Controller;

use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;

class DefaultController extends Controller
{
    /**
     * @Route("/")
     */
```

```
    public function index()
    {
        $number = mt_rand(0, 100);

        return $this->render('index.html.twig', array(
            'number' => $number,
        ));
    }
}
```

On fait une dérivation de la classe Controller de :

```
use Symfony\Bundle\FrameworkBundle\Controller\Controller;

class DefaultController extends Controller
```

On renvoie le Rendering vers la page « index.html.twig » en lui passant en paramètre le nombre dans une variable de la vue « number ».

```
return $this->render('index.html.twig', array(
            'number' => $number,
        ));
```

Le code TWIG permet d'avoir l'affichage suivant :

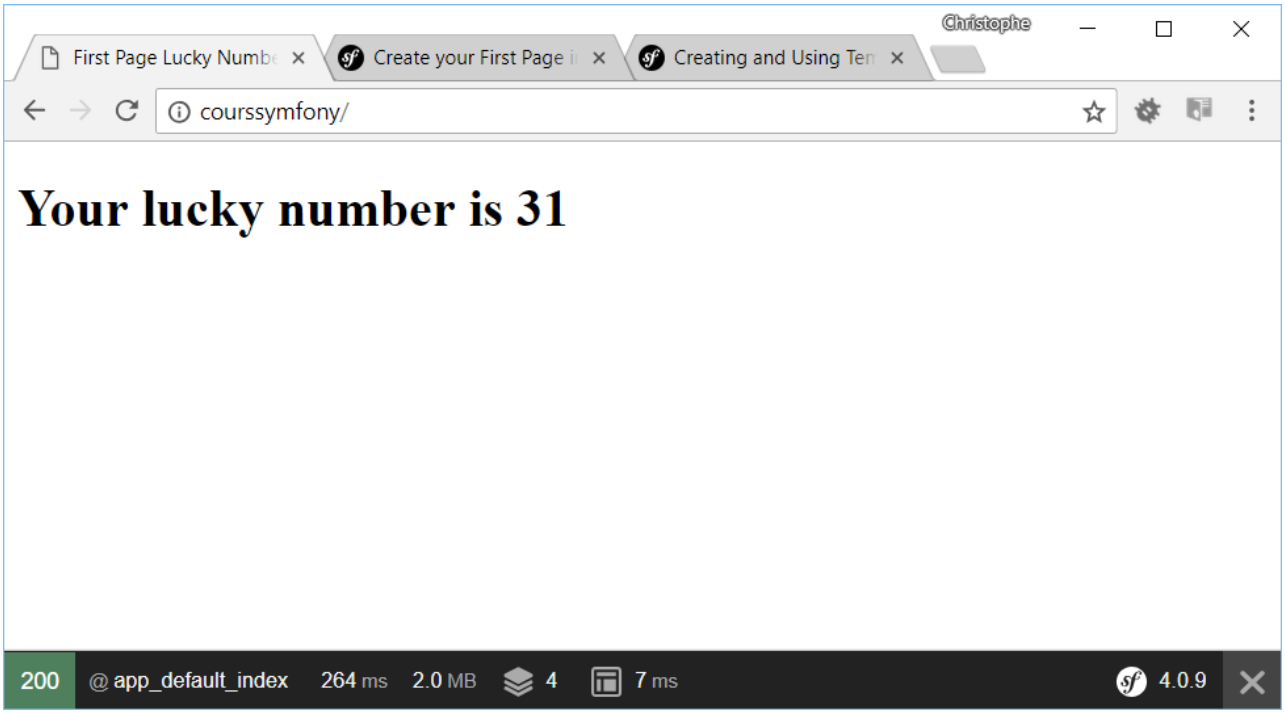


Figure 19 - page index.html.twig

Le code associé :

```
{# templates/index.html.twig #}
{% extends 'base.html.twig' %}

{% block title %}First Page Lucky Numbering{% endblock %}
```

```
{% block body %}
    <h1>Your lucky number is {{ number }}</h1>
{% endblock %}
```

Il est réalisé plusieurs opérations :

```
{% extends 'base.html.twig' %}
```

Permet d'étendre en surchargeant le modèle présent dans le fichier « base.html.twig »

```
{% block title %}First Page Lucky Numbering{% endblock %}
```

Remplace le block « title » par le contenu « First Page Lucky Numbering »

```
{% block body %}
    <h1>Your lucky number is {{ number }}</h1>
{% endblock %}
```

Remplace le block body de la base par :

```
    <h1>Your lucky number is {{ number }}</h1>
```

Où « number » est récupérer de la vue fournie par le Controller vue plus tôt.

Le fichier « base.html.twig » :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>{% block title %}Welcome!{% endblock %}</title>
        {% block stylesheets %}{% endblock %}
    </head>
    <body>
        {% block body %}{% endblock %}
        {% block javascripts %}{% endblock %}
    </body>
</html>
```

V.II. *Création du modèle de base pour les cours*

V.II.1 Création du schéma de la base de données Cours

Connectez-vous sur le lien : http://localhost/phpmyadmin/

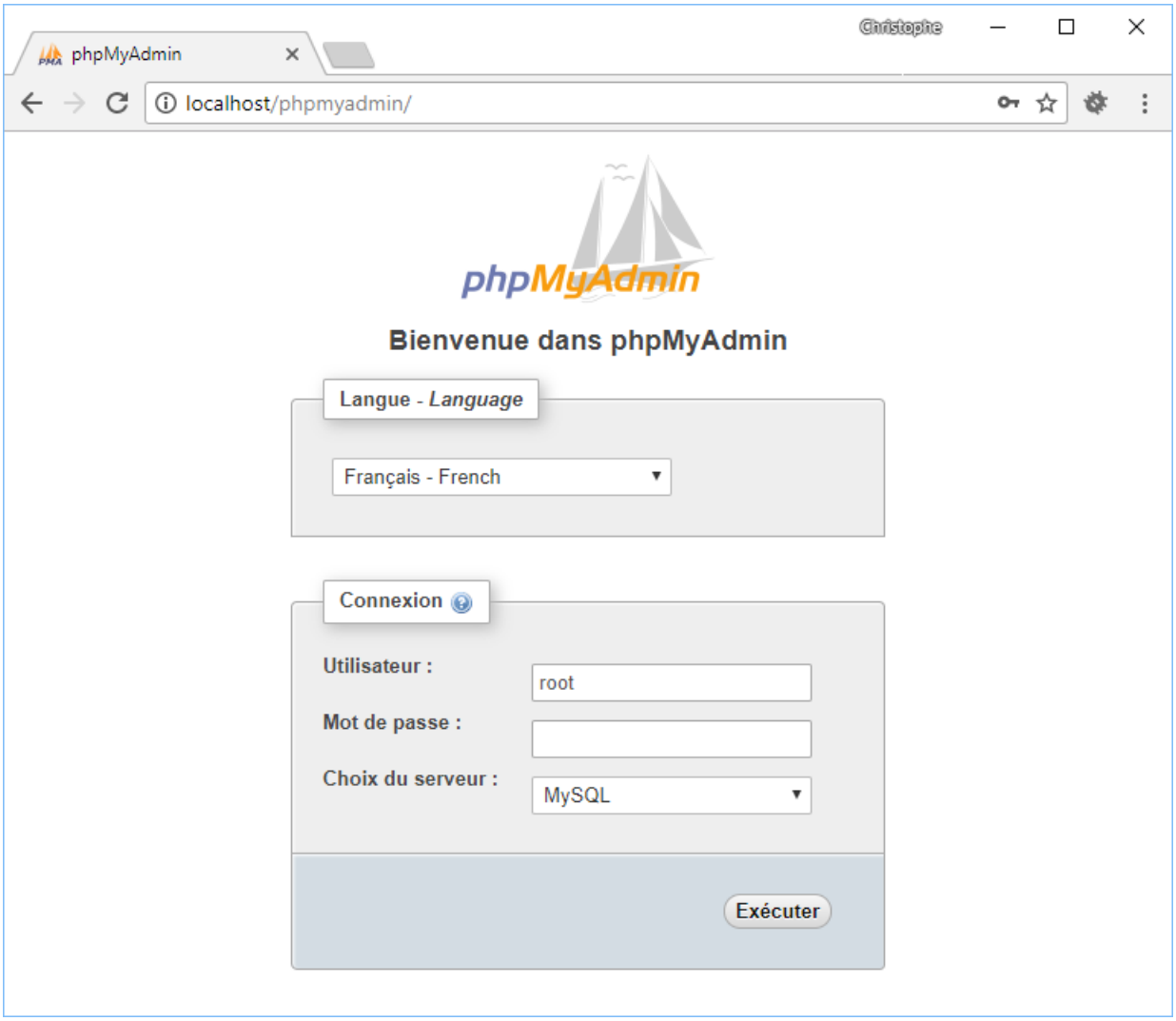


Figure 20 - Connexion PHPMyAdmin

Cliquez sur Exécuter

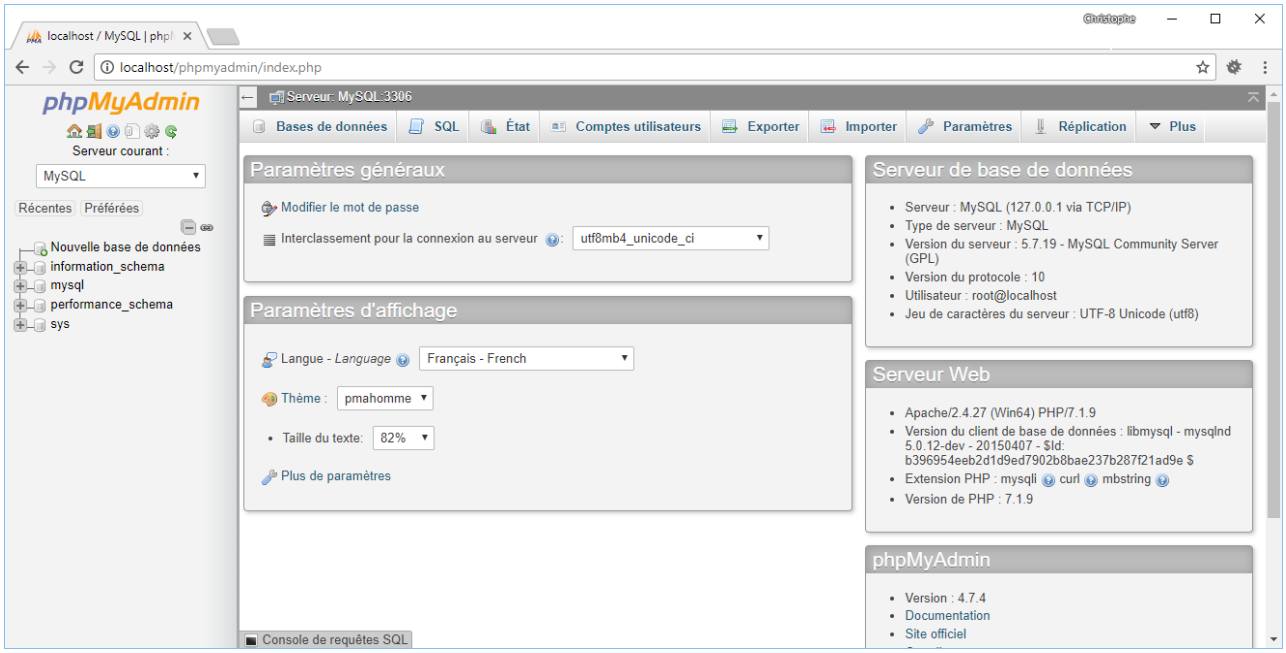


Figure 21 - Accueil PHPMyAdmin

Cliquez sur Bases de données

Compléter le nom de la base de données : cours
Encodage : utf8_general_ci

Figure 22 - Création du schéma de données

Cliquez sur « Créer »
On est redirigé vers :

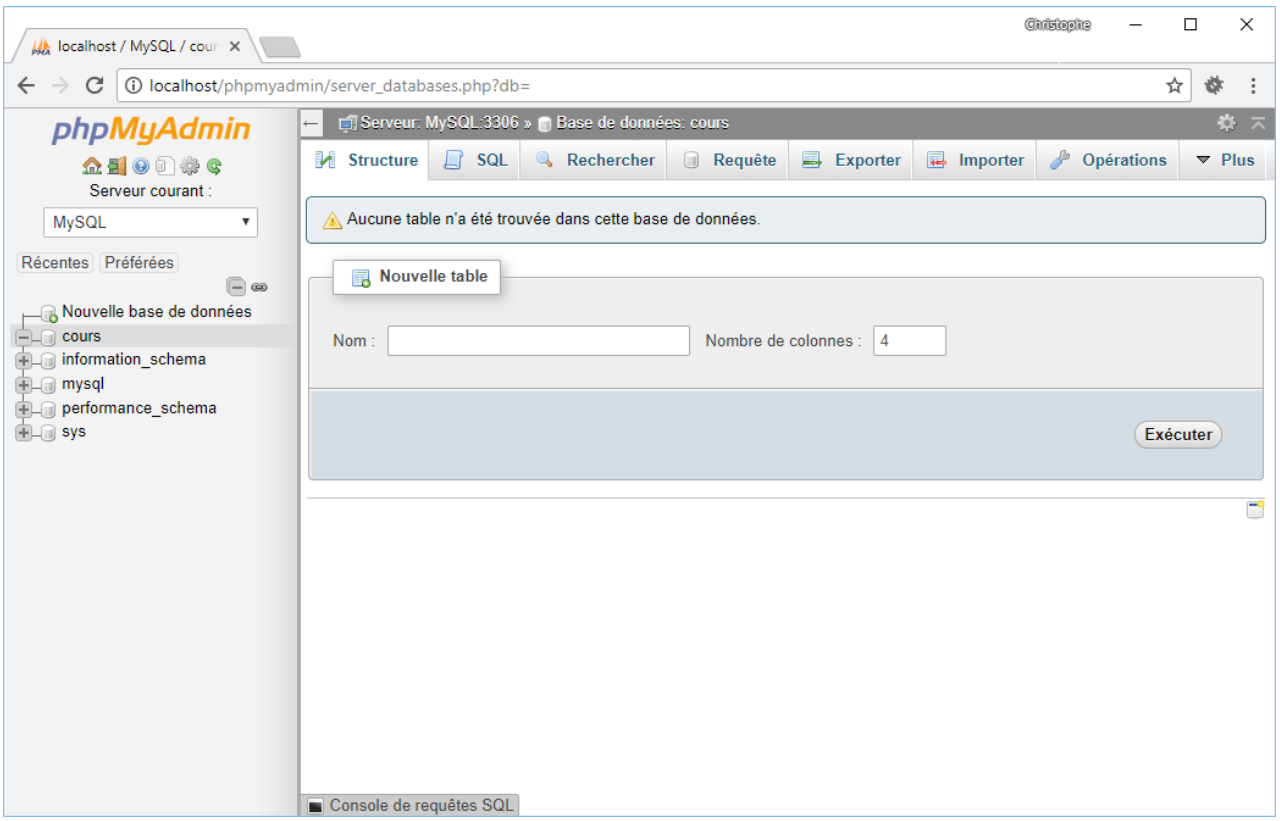


Figure 23 - Schéma cours créé

V.II.2 Création des tables

Aller dans l'éditeur de requête et insérer les lignes suivantes :

```
-- --------------------------------------------------------
-- Hôte :                        127.0.0.1
-- Version du serveur:           5.7.19 - MySQL Community Server (GPL)
-- SE du serveur:                Win64
-- HeidiSQL Version:             9.4.0.5125
-- --------------------------------------------------------


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET NAMES utf8 */;
/*!50503 SET NAMES utf8mb4 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0
*/;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;


-- Export de la structure de la table cours. course
```

```
CREATE TABLE IF NOT EXISTS `course` (
  `COURSE_ID` int(11) NOT NULL AUTO_INCREMENT,
  `TITLE` varchar(1024) COLLATE utf8_unicode_ci NOT NULL,
  `AUTHOR_FK` int(11) NOT NULL,
  PRIMARY KEY (`COURSE_ID`),
  KEY `IDX_E666A83F9DDDAA4A` (`AUTHOR_FK`),
  CONSTRAINT `FK_E666A83F9DDDAA4A` FOREIGN KEY (`AUTHOR_FK`) REFERENCES `person` (`PERSON_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- Les données exportées n'étaient pas sélectionnées.
-- Export de la structure de la table cours. person
CREATE TABLE IF NOT EXISTS `person` (
  `PERSON_ID` int(11) NOT NULL AUTO_INCREMENT,
  `NAME` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `FIRST_NAME` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `JOB` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`PERSON_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- Les données exportées n'étaient pas sélectionnées.
-- Export de la structure de la table cours. person_course
CREATE TABLE IF NOT EXISTS `person_course` (
  `PERSON_ID` int(11) NOT NULL,
  `COURSE_ID` int(11) NOT NULL,
  PRIMARY KEY (`PERSON_ID`,`COURSE_ID`),
  KEY `IDX_63CA64765DF4E348` (`PERSON_ID`),
  KEY `IDX_63CA64762593919D` (`COURSE_ID`),
  CONSTRAINT `FK_63CA64762593919D` FOREIGN KEY (`COURSE_ID`) REFERENCES `course` (`COURSE_ID`),
  CONSTRAINT `FK_63CA64765DF4E348` FOREIGN KEY (`PERSON_ID`) REFERENCES `person` (`PERSON_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- Les données exportées n'étaient pas sélectionnées.
/*!40101 SET SQL_MODE=IFNULL(@OLD_SQL_MODE, '') */;
```

```
/*!40014 SET FOREIGN_KEY_CHECKS=IF(@OLD_FOREIGN_KEY_CHECKS IS NULL, 1,
@OLD_FOREIGN_KEY_CHECKS) */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
```

V.II.3 Création de l'utilisateur course/pass

Faîtes de même pour y insérer l'utilisateur

```
CREATE USER 'course'@'localhost' IDENTIFIED BY 'pass';
GRANT USAGE ON *.* TO 'course'@'localhost';
GRANT SELECT, EXECUTE, SHOW VIEW, ALTER, ALTER ROUTINE, CREATE, CREATE ROUTINE,
CREATE TEMPORARY TABLES, CREATE VIEW, DELETE, DROP, EVENT, INDEX, INSERT,
REFERENCES, TRIGGER, UPDATE, LOCK TABLES  ON `cours`.* TO 'course'@'localhost'
WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

V.III. *Configuration et modèle Entity*

Le modèle Entity est un objet métier qui va être représenter en base de données relationnelles.

V.III.1 Installer Doctrine

Dans le terminal lancer :

```
composer require doctrine
composer require maker --dev
```

V.III.2 Configurer le fichier .env

Dans VSCode, configurer le fichier « .env » se trouvant à la racine du projet.

```
DATABASE_URL=mysql://course:pass@127.0.0.1:3306/cours
```

V.III.3 Création du répertoire Entity dans src/

Veuillez créer le répertoire src/Entity afin d'y créer par la suite les entity.

V.III.4 Création de l'Entity Course

```php
<?php

namespace App\Entity;

use App\Entity\Person;
use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity(repositoryClass="App\Repository\CourseRepository")
 * @ORM\Table(name="COURSE")
 *
 * @author Christophe PEQUIGNAT
 */
class Course
{

    /**
     * @ORM\Id
     * @ORM\Column(type="integer", name="COURSE_ID")
     * @ORM\GeneratedValue
     *
     * @var int
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=1024, name="TITLE")
     *
     * @var string
     */
    private $title;

    /**
     * @ORM\ManyToOne(targetEntity="App\Entity\Person", inversedBy="myCourses")
     * @ORM\JoinColumn(name="AUTHOR_FK", referencedColumnName="PERSON_ID", nullable=false)
     *
     * @var Person
     */
    private $author;

    /**
     * Plusieurs cours ont plusieurs étudiants.
     * @ORM\ManyToMany(targetEntity="App\Entity\Person", mappedBy="myTrainingCourses")
     *
     * @var \Doctrine\Common\Collections\ArrayCollection
     */
    private $students;

    public function __construct()
    {
        $this->students = new \Doctrine\Common\Collections\ArrayCollection();
    }

    /**
     *
     * @return the $id
     */
    public function getId()
    {
        return $this->id;
    }
```

```php
    /**
     *
     * @return the $title
     */
    public function getTitle()
    {
        return $this->title;
    }

    /**
     *
     * @return Person $author
     */
    public function getAuthor()
    {
        return $this->author;
    }

    /**
     *
     * @param number $id
     */
    public function setId($id)
    {
        $this->id = $id;
    }

    /**
     *
     * @param string $title
     */
    public function setTitle($title)
    {
        $this->title = $title;
    }

    /**
     *
     * @param Person $author
     */
    public function setAuthor(Person $author)
    {
        $this->author = $author;
    }

    /**
     *
     * @param Person $student
     */
    public function addStudent(Person $student): void
    {
        $this->students->add($student);
    }

    /**
     *
     * @return \Doctrine\Common\Collections\ArrayCollection
     */
    public function getStudents(): \Doctrine\Common\Collections\ArrayCollection
    {
        return $this->students;
    }

    public function __toString(): string
    {
        $str='Course{id:'.$this->id.',title:'.$this->title.',author:'.$this->author.'students:[';
        $isFirst = true;
        foreach($this->students as $student){
            if (! $isFirst){
```

```
                $str.=',';
            }
            $str.=$student;
            $isFirst=false;
        }
        $str.=']}';
        return $str;
    }
}
```

V.III.1 **Création de l'Entity Person**

```php
<?php

namespace App\Entity;

use App\Entity\Course;
use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity(repositoryClass="App\Repository\PersonRepository")
 * @ORM\Table(name="PERSON")
 *
 * @author Christophe PEQUIGNAT
 */
class Person
{

    /**
     * @ORM\Id
     * @ORM\Column(type="integer", name="PERSON_ID")
     * @ORM\GeneratedValue
     *
     * @var int
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=255, name="NAME")
     *
     * @var string
     */
    private $name;

    /**
     * @ORM\Column(type="string", length=255, name="FIRST_NAME")
     *
     * @var string
     */
    private $firstname;

    /**
     * @ORM\Column(type="string", length=255, name="JOB", nullable=true)
     *
     * @var string
     */
    private $job;

    /**
     * Une Person rédige plusieurs Course.
     * @ORM\OneToMany(targetEntity="App\Entity\Course", mappedBy="author")
     */
    private $myCourses;
```

```php
/**
 * Plusieurs Persons suivent plusieurs cours.
 * @ORM\ManyToMany(targetEntity="App\Entity\Course", inversedBy="students")
 * @ORM\JoinTable(name="PERSON_COURSE",
 * joinColumns={@ORM\JoinColumn(name="PERSON_ID", referencedColumnName="PERSON_ID")},
 * inverseJoinColumns={@ORM\JoinColumn(name="COURSE_ID", referencedColumnName="COURSE_ID")}
 * )
 */
private $myTrainingCourses;

public function __construct()
{
    $this->myCourses = new \Doctrine\Common\Collections\ArrayCollection();
    $this->myTrainingCourses = new \Doctrine\Common\Collections\ArrayCollection();
}

/**
 *
 * @return the $id
 */
public function getId()
{
    return $this->id;
}

/**
 *
 * @return the $name
 */
public function getName()
{
    return $this->name;
}

/**
 *
 * @return the $firstname
 */
public function getFirstname()
{
    return $this->firstname;
}

/**
 *
 * @return the $job
 */
public function getJob()
{
    return $this->job;
}

/**
 *
 * @param number $id
 */
public function setId($id)
{
    $this->id = $id;
}

/**
 *
 * @param string $name
 */
public function setName($name)
{
    $this->name = $name;
```

```
    }

    /**
     *
     * @param string $firstname
     */
    public function setFirstname($firstname)
    {
        $this->firstname = $firstname;
    }

    /**
     *
     * @param string $job
     */
    public function setJob($job)
    {
        $this->job = $job;
    }

    /**
     *
     * @param Course $course
     */
    public function addMyCourse(Course $course): void
    {
        $this->myCourses->add($course);
    }

    /**
     *
     * @return \Doctrine\Common\Collections\ArrayCollection
     */
    public function getMyCourses(): \Doctrine\Common\Collections\ArrayCollection
    {
        return $this->myCourses;
    }

    public function addMyTrainingCourse(Course $myTrainingCourse): void
    {
        $this->myTrainingCourses->add($myTrainingCourse);
    }

    public function getMyTrainingCourses(): \Doctrine\Common\Collections\ArrayCollection
    {
        return $this->myTrainingCourses;
    }


    public function __toString(): string
    {
        return 'Person{id:' . $this->id . ',firstname:' . $this->firstname . ',name:' . $this->name . ',job:' .
$this->job . '}';
    }
}
```

V.IV. *Création des Repository*

Le Repository, c'est ce qui permet d'interroger la base de données afin de récupérer le Model vue au dessus.

V.IV.1 Créer le fichier CourseRepository.php dans src/Repository

Ce repository utilise l'extension pour simplifier l'écriture avec :

```
class CourseRepository extends EntityRepository
```

On utilise ici le Langage de requête DQL :

```
return $this->getEntityManager()->createQuery('SELECT c FROM App\Entity\Course c')
        ->getResult();
```

```
<?php
namespace App\Repository;

use App\Entity\Course;
use Doctrine\ORM\EntityRepository;
/**
 *
 * @author Christophe PEQUIGNAT
 *
 */
class CourseRepository extends EntityRepository
{

    public function getAllCourses()
    {
        return $this->getEntityManager()->createQuery('SELECT c FROM App\Entity\Course c')
        ->getResult();
    }

    public function addCourse(Course $course){
        $this->getEntityManager()->persist($course);
    }

    public function updateCourse(Course $course){
        $this->getEntityManager()->persist($course);
    }

    public function removeAll() : void{
        foreach($this->getAllCourses() as $course){
            $this->getEntityManager()->remove($course);
        }
    }

}
```

V.IV.1 Créer le fichier PersonRepository.php dans src/Repository

```
<?php
namespace App\Repository;

use App\Entity\Person;
use Doctrine\ORM\EntityRepository;

class PersonRepository extends EntityRepository
{

    public function getAllPersons()
    {
        return $this->getEntityManager()->createQuery('SELECT p FROM App\Entity\Person p')
        ->getResult();
    }
```

```
    public function addPerson(Person $person) : void{
        $this->getEntityManager()->persist($person);
    }

    public function removeAll() : void{
        foreach($this->getAllPersons() as $person){
            $this->getEntityManager()->remove($person);
        }
    }

}
```

V.V. *Insérer une donnée en base de données MySQL :*

```
INSERT INTO `cours`.`person` (`NAME`, `FIRST_NAME`, `JOB`) VALUES ('Christophe',
'PEQUIGNAT', 'Architect');


INSERT INTO `cours`.`course` (`TITLE`, `AUTHOR_FK`) VALUES ('Cours 1', '1');
```

V.VI. *Ajouter le rendu de la liste des cours*

V.VI.1 Ajouter le Controller CourseController

```
<?php
// src/Controller/CourseController.php
namespace App\Controller;

use App\Entity\Course;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\Routing\Annotation\Route;

class CourseController extends Controller
{
    /**
     * Matches /course exactly
     *
     * @Route("/course", name="course_list")
     */
    public function list()
    {
        $repository = $this->getDoctrine()->getRepository(Course::class);

        $courses = $repository->getAllCourses();
        return $this->render('course/list.html.twig', array(
            'courses' => $courses,
        ));
    }

}
```

V.VI.2 Réaliser le rendu du twig : course/list.html.twig

```
{% extends 'base.html.twig' %}

{% block title %}List of Courses{% endblock %}

{% block body %}
    <h1>List of Courses</h1>
    <ul>
    {% for course in courses %}
        <li>{{course.title}} [Author :  {{course.author.name}} {{course.author.firstName}}]</li>

    {% endfor %}
    </ul>
{% endblock %}
```

Visualiser le résultat sur la page : http://courssymfony/course

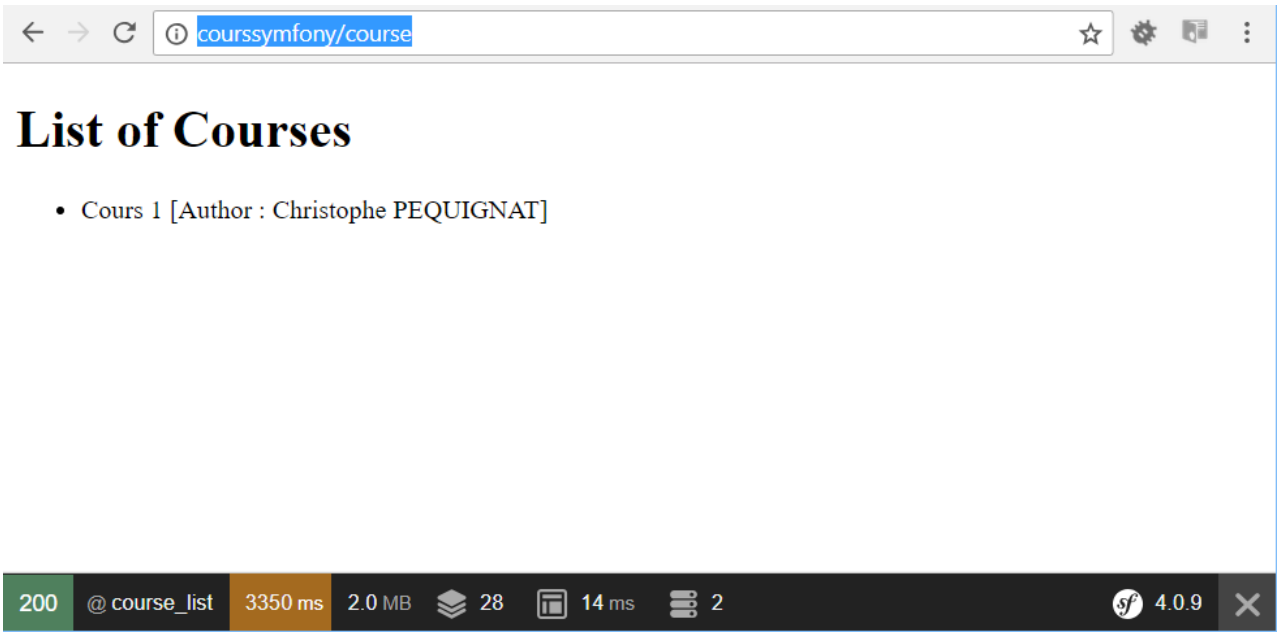


Figure 24 - Liste des cours

V.VII. *Modifier la vue principale afin de rajouter le lien vers la page des Cours*

```
{# templates/index.html.twig #}
{% extends 'base.html.twig' %}

{% block title %}First Page Lucky Numbering{% endblock %}

{% block body %}
    <h1>Your lucky number is {{ number }}</h1>

    <a href="{{ path('course_list') }}">
      Go to the courses list!
    </a>
{% endblock %}
```

VI.Sources d'informations

#	Source	Lien
[S1]	Page creation	https://symfony.com/doc/current/page_creation.html
[S2]	http Fundamentals	https://symfony.com/doc/current/introduction/http_fundamentals.html
[S3]	Templating	https://symfony.com/doc/current/templating.html
[S4]	Routing	http://symfony.com/doc/current/routing.html
[S5]	Doctrine	http://symfony.com/doc/current/doctrine.html

VII. Fin du document