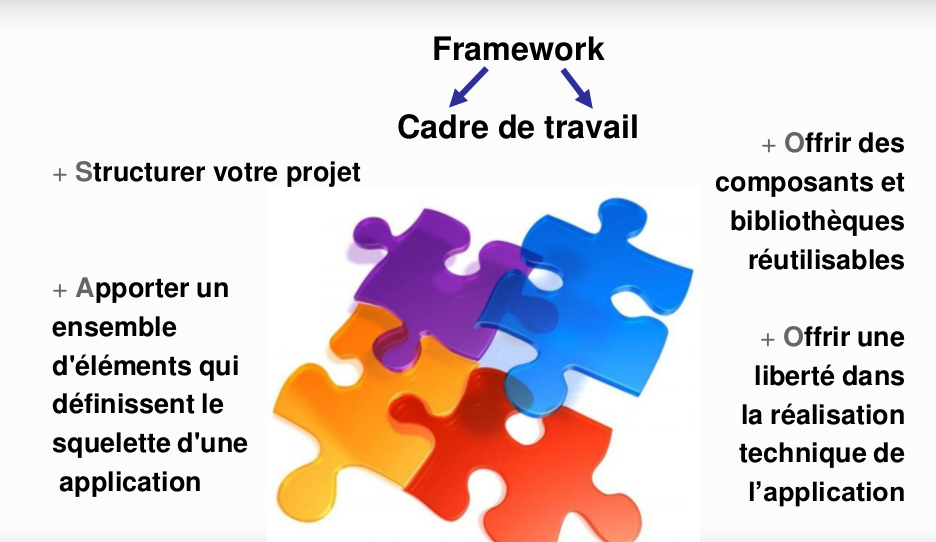
# INTIATION AU DÉVELOPPEMENT AVEC LE FRAMEWORK SYMFONY2

# Rappel :

C’est quoi un Framework ?



# MVC

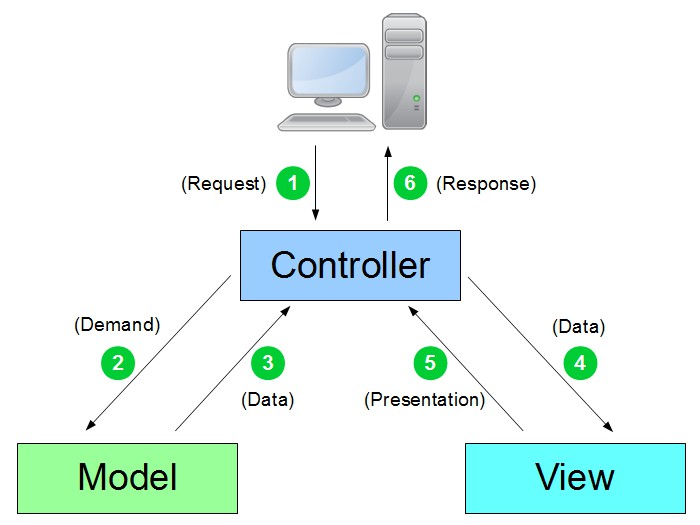
Pour réduire les risques de maintenance, il a fallu exploiter un niveau supplémentaire dans la conception objet : les modèles de conception (design patterns).

Le design pattern MVC permet d’organiser une application en trois couches distinctes à savoir:

**Le modèle :** C’est la couche des "données" persistantes.

**La vue :** C’est l’interface d’interaction entre l’utilisateur et l’application. Elle se charge de présenter les résultats renvoyés par le modèle qui lui sont passés grâce à un contrôleur. La vue reçoit aussi toutes les actions de l’utilisateur telles que la soumission d’un formulaire, la demande d’une interface, etc.

**Le contrôleur :** Son rôle est de traiter les événements en provenance de l’interface utilisateur et de les transmettre au modèle pour le faire évoluer ou à la vue pour modifier son aspect visuel.



Programmation orientée objet

La programmation orientée objet se base sur une approche de conception et de développement de logiciels

Représenter les éléments du monde réel par des entités informatiques appelés "objets" en adoptant un haut niveau d'abstraction.

Approche procédurale :

"Que doit faire mon programme ?  "

Approche orientée-objet :

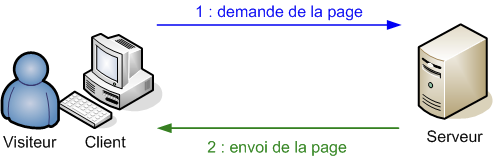
"De quoi doit être composé mon programme ?"

# Design Pattern :

Un design pattern ou modèle de conception consiste en un ensemble de règles et de bonnes pratiques visant à améliorer l’organisation de l’application en privilégiant la ré-utilisabilité du code. Une telle organisation permet de catalyser les temps de développements, et améliore considérablement la maintenance de l’application.

[LIEN DESIGN PATTERN](https://openclassrooms.com/courses/programmez-en-oriente-objet-en-php/les-design-patterns)

# Model http :

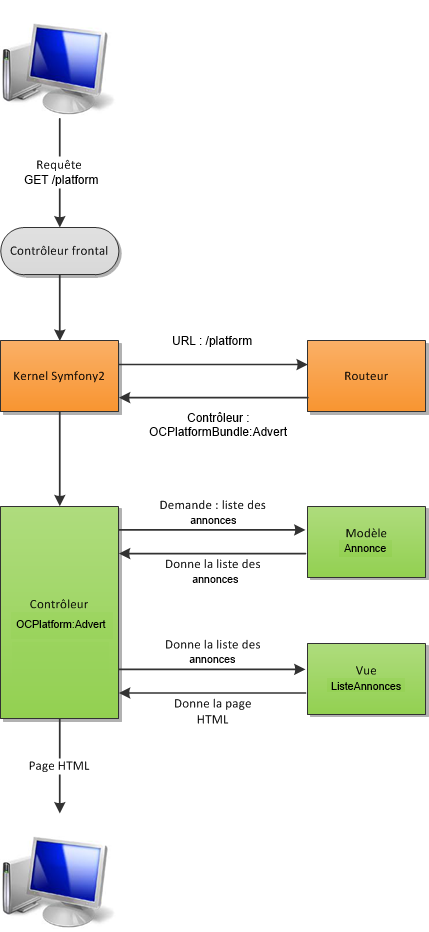


# Symfony2 et MVC

Un projet symfony 2, contient 5 principaux répertoires :

1. **app/** : Contient les fichiers de l’applications sauf le code source, notamment; la configuration, le cache, les fichiers logs etc...
2. **vendor/** : contient toutes les librairies (ou bundles) externes
3. **Web/** : Contient tous les fichiers destinés aux visiteurs : images, fichiers CSS et JavaScript. C’est le seul répertoire qui doit être accessible par les internautes)
4. **src/** : Contient le code source de l’applications organisé sous forme de bundles (par défaut il contient un bundle d’exemple fourni par Symfony).

[LIEN ARCHITECTURE PROJET SYMFONY](http://symfony.com/doc/2.8/quick_tour/the_architecture.html)



# Architecture du framework

Front-end Controller

Le contrôleur frontal (*front controller*, en anglais) est le point d'entrée de votre application. C'est *le* fichier par lequel passent toutes vos pages. Vous devez surement connaître le principe d'index.php et des pseudo-frames (avec des URL du type index.php?page=blog) et bien, cet index.php est un contrôleur frontal. Dans Symfony2, le contrôleur frontal se situe dans le répertoire /web, il s'agit de app.php ou app\_dev.php.

MVC dans Symfony2

# Composer

# Autoloading + namespace

Autoloading :

"autoload": {

       "psr-4": {"MicroCMS\\": "src"}

   }

La nouvelle entrée autoload dans ce fichier permet d'ajouter notre propre code source, défini dans le répertoire src, au mécanisme de chargement automatique (autoloading) géré par Composer.

Finies les instructions require partout dans le code ! Pour que cela fonctionne, il faut que notre code source respecte le standard [PSR-4](http://www.php-fig.org/psr/psr-4/).

namespace:

Vous aurez peut-être remarqué que cette classe est définie dans un espace de nommage ou espace de noms (instruction PHP namespace au début du fichier).

En PHP comme dans les autres langages qui les supportent, un espace de nommage permet d'éviter les conflits de nommage des éléments (exemple : deux classes portant le même nom) en regroupant ces éléments dans des espaces dédiés. Ainsi, le nom complet de notre classe Article est MicroCMS\Domain\Article. Pour plus d'informations sur les espaces de noms en PHP, consultez la [documentation du langage](https://secure.php.net//manual/fr/language.namespaces.php).

# Routing

• Les routes en Symfony2 déterminent quel contrôleur/action est à exécuter en fonction de l’URL appelée par l’internaute.

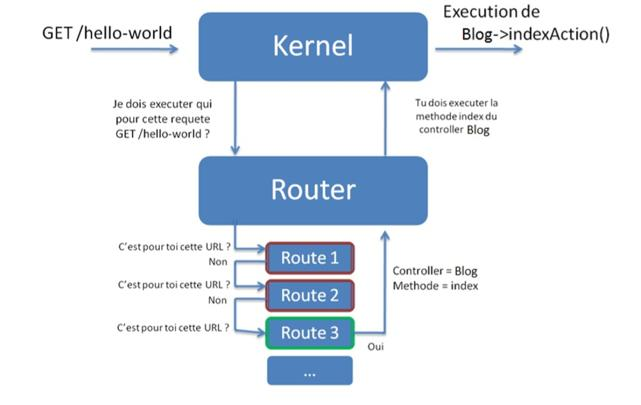
• Les routes se définissent dans un fichier qui se trouve sous

Resources/config/routing.yml du bundle en question.

• Il existe deux types de routes:

1. Route sans paramètres

2. Route avec paramètres



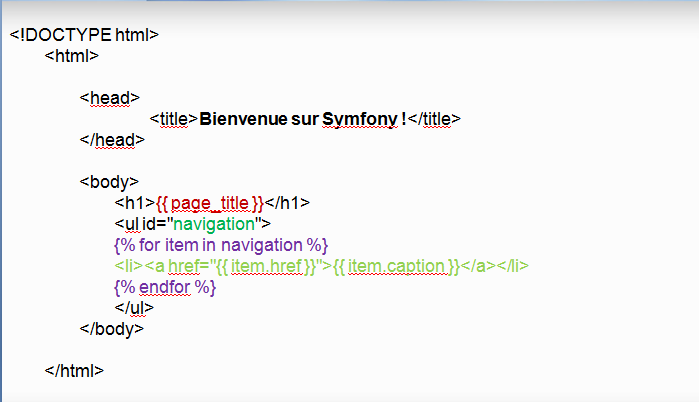
# Twig

Moteur de template :

Un moteur de Template est un script qui permet d‘intégrer des templates, c'est-à- dire des fichiers qui ont pour but d'afficher le contenu de votre page HTML de façon dynamique.

• Symfony2 utilise un moteur de Template encore plus puissant appelé **Twig**.

• Twig vous permet d'écrire des Templates simples et lisibles et, dans bien des aspects, plus puissants que les templates PHP :



**PS**:Le dossier Resources/views représente le répertoire des Templates (ou vues) d’un bundle.

# ORM Docrine : Object Relational Mapping

Technique de programmation faisant le lien entre le monde de la base de données et le monde de la programmation objet

