Le modèle MVC de Symfony

1. Objectifs

- Identifier et comprendre les rôles de chaque composant du modèle MVC dans Symfony : Modèle, Vue et Contrôleur.

2. Présentation

- Le **MVC (Modèle Vue Contrôleur)** est une méthode d'organisation du développement d'applications Web permettant de séparer les différents concepts résultant de nos pages PHP.
- Symfony, tout comme la plupart des autres frameworks PHP, suit une architecture dite MVC (Modèle-Vue-Contrôleur).
- Le **Modèle-Vue-Contrôleur (MVC)** que le framework Symfony utilise est essentiellement un modèle architectural qui divise une application en trois composants logiques : Modèle, Vue et Contrôleur, comme son nom l'indique. Ces composants gèrent des aspects de développement spécifiques d'une application pour rendre Symfony accessible, puissant et également fournir les outils nécessaires pour exécuter et maintenir des applications volumineuses et robustes.
- Symfony se base effectivement sur le patron de conception MVC, c'est-à-dire modèle-vue-contrôleur.
- Le Modèle interagit avec la base de données, les regroupe, traite et gère les données. Le modèle contient les données et leur logique.
- La Vue s'occupe principalement de faire afficher ce que le modèle renvoie. Ensuite, elle s'occupe de recevoir toute interaction de l'utilisateur. La vue contient la présentation graphique à renvoyer à l'utilisateur.
- Le Contrôleur prend en charge de synchroniser le modèle et la vue. Il capte toutes les activités de l'utilisateur et, en fonction de ces activités, il actionne les changements à effectuer sur l'application. Le contrôleur traite les actions utilisateur (via des requêtes), demande au modèle d'effectuer les changements, puis passe les données à la vue.
- La séparation des composants d'une application en ces trois catégories permet une clarté de l'architecture des dossiers et simplifie grandement la tâche aux développeurs. Ainsi, la figure ci-dessous nous décrit l'architecture MVC de Symfony :

- Le modèle MVC de Symfony:

apcpedagogie.com

- Le **routeur** de l'application va appeler le bon contrôleur en fonction de la route (c'est-à-dire en fonction du verbe utilisé et de l'URI). Dans notre exemple, ce sera le contrôleur responsable de l'inscription d'un utilisateur.
- Le **contrôleur** est responsable de coordonner les actions à effectuer. Ici par exemple, il peut vérifier le contenu de la requête, puis demander au modèle d'inscrire l'utilisateur dans la base de données. Une fois que le modèle aura inscrit l'utilisateur, il pourra préparer la vue à retourner à l'utilisateur.
- Le **modèle** est responsable de la validation de la lecture et de l'enregistrement des données. Dans notre exemple, il va s'assurer que toutes les données pour créer un utilisateur ont bien été fournies. Dans l'affirmative il va créer l'utilisateur dans la base de données et le passer au contrôleur.
- La **vue** est responsable de l'interface graphique. Elle contient la logique pour l'affichage des données qui ont été récupérées dans par le modèle.

3. Application MVC avec Laravel

- **Les Modèles (Models)**
  - Ce que nous appelons un Modèle est en réalité un fichier PHP qui ne fait que gérer les échanges avec la base de données. Lorsque nous avons besoin de lire ou écrire dans la base de données, nous faisons appel au Modèle.
  - Le modèle est la couche représentant les données. On l'appellera parfois logique métier.
  - Le modèle consiste en une série de classes. Si les données sont tirées de la BD, chacune des classes représentera une table.
  - Parmi les fonctionnalités codées dans le modèle, on retrouve les relations entre les tables, les accesseurs et modificateurs, les champs calculés, etc.
- **Les Vues (Views)**
  - La vue est constituée de balises HTML qui représentent ce qui sera affiché à l'écran, c'est une interface utilisateur.
  - En plus des balises HTML, la vue peut utiliser des directives et instructions prévues par le moteur d'affichage afin d'effectuer différentes opérations, comme par exemple tester une condition ou encore boucler dans les données fournies par le modèle.
  - La vue pourra faire appel à des ressources externes, comme des feuilles de style, des fichiers JavaScript, des images, etc.

apcpedagogie.com

    - Sous Laravel, le moteur d'affichage s'appelle Blade.

- **Les Contrôleurs (Controllers)**

    - Véritable tour de contrôle de notre application, le contrôleur a pour fonction de faire l’interface entre les modèles et les vues. Il est chargé de demander les données par l’intermédiaire des modèles, de traiter ces données et de les transmettre aux vues, prêtes à être utilisées.

- **Le routing**

    - Bien qu’indépendant de l’architecture MVC, le routing fait partie intégrante de tous les Frameworks PHP.
    - Dans une architecture classique, nous pointons vers des fichiers :
        - http://monsite.fr/index.php
        - http://monsite.fr/inscription.php
        - http://monsite.fr/login.php
    - …
    - Dans une architecture MVC, nous allons pointer vers des dossiers virtuels appelés routes
        - http://monsite.fr/user/inscription
        - http://monsite.fr/user/login
        - http://monsite.fr/blog/article
    - …
    - Cette architecture offre de nombreux avantages :
        - Protection des fichiers, ceux-ci n’étant plus affichés par le navigateur
        - Des URLs plus simples à mémoriser pour les utilisateurs
        - Amélioration du référencement si les routes contiennent des mots-clés contenus dans la page correspondante

Exemple de liens entre routes, contrôleurs et vues sous Symfony

- Le schéma ci-dessus montre le cheminement du traitement à partir d'un **URL** jusqu'à l'affichage d'une page Web.
    - On y voit, en rouge, que Symfony commence par rechercher une route qui correspond à l'URL (ici : « / »).
    - On y voit en bleu qu'ensuite, Symfony va appeler le contrôleur indiqué dans la route définie. Plus précisément, en vert, on voit le nom de la méthode

apcpedagogie.com

d'action à exécuter dans ce contrôleur (ici : index).
- En rose, on voit que Symfony va afficher la vue mentionnée dans l'action du contrôleur. La vue doit être placée dans un répertoire approprié (par exemple, templates/) et porter le nom spécifié.
- C'est ainsi qu'on obtient à l'écran le visuel de l'URL demandé.

***Source**: https://christianelagace.com/laravel/resume-des-liens-entre-routes-controleurs-et-vues-sous-laravel/*

4. Application

- Pour comprendre comment Symfony intègre la logique MVC, créons un exemple de projet Symfony qui affiche la liste des apprenants d'une école de formation.
- Pour créer un nouveau projet, exécutez la commande ci-dessous dans votre terminal :
- `$ symfony new` **`gestionApprenant`**

> ***Toutes les commandes Symfony doivent être exécutées à la racine du projet, vous devez donc revenir à la racine avec la commande `cd` avant d'exécuter la commande. Pour arrêter le serveur, appuyez sur `ctrl + c` sur votre clavier pour Windows***

- **Modèles :**
  - Créer notre modèle d'apprenant
  - Créons notre premier modèle, **M** en **MVC**, pour notre application. Comme nous l'avons dit, le modèle s'interface généralement avec un stockage de données comme une base de données **MySQL**, par exemple.
  - Dans Symfony, le modèle est généralement une classe dont les propriétés correspondent aux colonnes de la base de données.
  - Dans notre base de données, un apprenant aura les propriétés suivantes :
    - Nom (firstname) - Nom de l'apprenant.
    - Prénom (lastname) - Prénom de l'apprenant.
    - Date de naissance (dteNaissance) - Date de naissance de l'apprenant.
    - Adresse (adress) - Adresse de l'apprenant.
  - Pour créer un modèle dans Symfony, exécutez la commande dans votre terminal : **`$ php bin/console make:entity Apprenant`**
  - Lorsque vous exécutez cette commande, Symfony créera une entité qui sera notre modèle pour notre table d'apprenants dans la base de données.

---

apcpedagogie.com

Sommaire du cours Symfony

Source : https://apcpedagogie.com/le-modele-mvc-de-symfony · téléchargé le 08/11/2026 · © APC Pédagogie · apcpedagogie.com

apcpedagogie.com