



UNIVERSITÉ
CAEN
NORMANDIE

Framework MVC

Introduction à Symfony et Révision de l'Architecture MVC

Présentation de Symfony

- Framework PHP pour le développement web
- Historique et popularité
- Importance dans l'écosystème PHP

Framework PHP pour le développement web

Symfony est l'un des frameworks PHP les plus populaires pour le développement web. Il fournit une structure solide pour créer des applications web, des API et des microservices.

Historique et popularité

- **2005** : Lancement de Symfony par SensioLabs.
- **2011** : Symfony2 introduit le concept de "composants", des bibliothèques réutilisables pour PHP.
- **Popularité** : De nombreuses entreprises et projets open-source utilisent Symfony, attestant de sa robustesse et de sa fiabilité.

Importance dans l'écosystème PHP

Symfony joue un rôle crucial dans l'écosystème PHP :

- **Composants** : Les composants Symfony sont largement utilisés en dehors du framework lui-même.
- **Standardisation** : Symfony a contribué à la standardisation de nombreuses pratiques en PHP, comme l'utilisation de l'auto-chargement PSR-4.
- **Intégrations** : Grâce à sa flexibilité, Symfony s'intègre facilement avec d'autres outils et bibliothèques PHP.

Qu'est-ce que Symfony ?

Symfony est un framework PHP de développement web et une collection de composants PHP réutilisables. Il offre des outils et des fonctionnalités pour faciliter le développement d'applications web robustes et complexes.

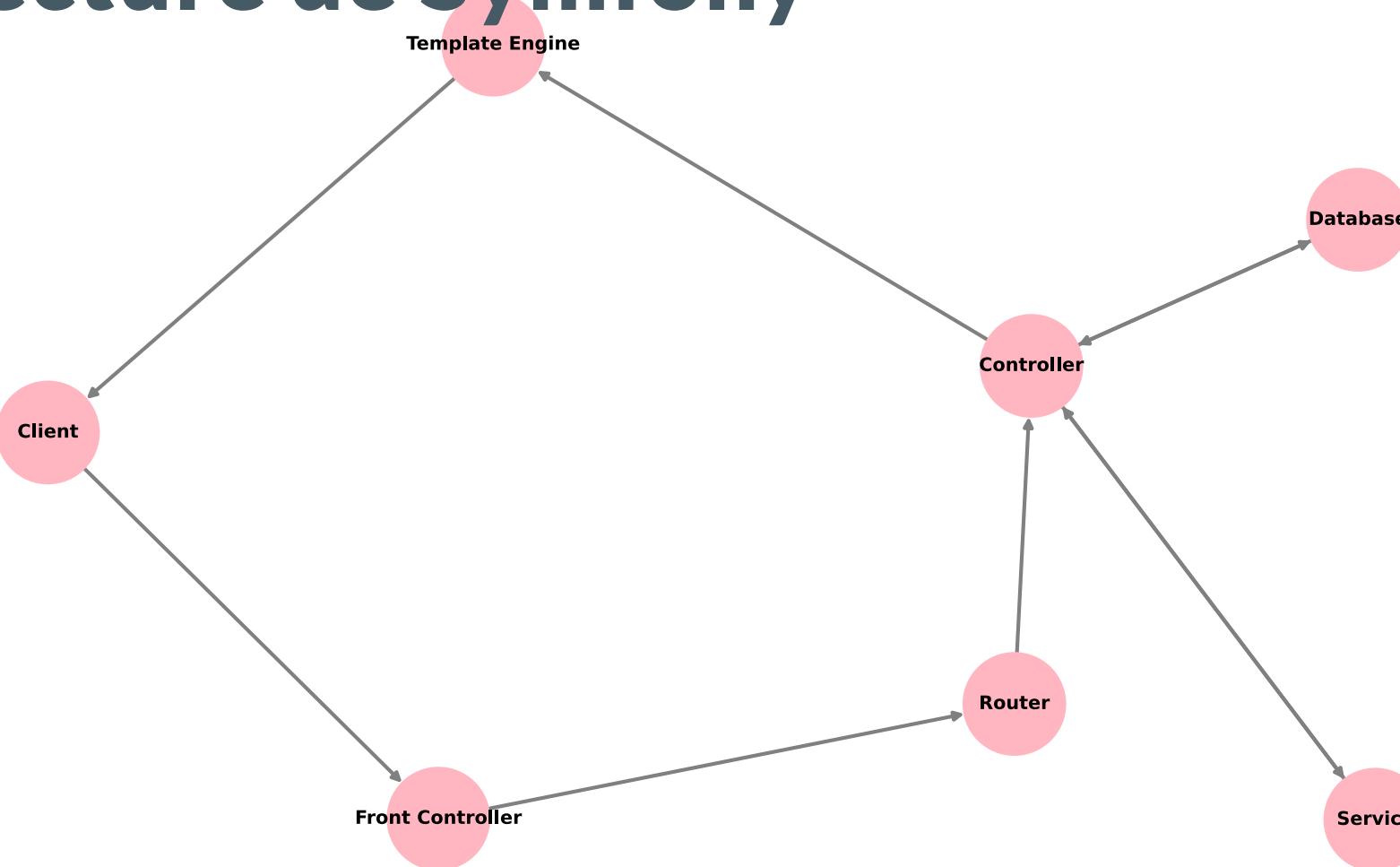
Pourquoi choisir Symfony ?

Symfony est réputé pour sa modularité, sa performance et sa flexibilité. Il fournit une architecture solide, une grande communauté de développeurs et de nombreuses solutions pour des problèmes courants, tels que le routage, la gestion des formulaires, l'authentification et bien plus encore.

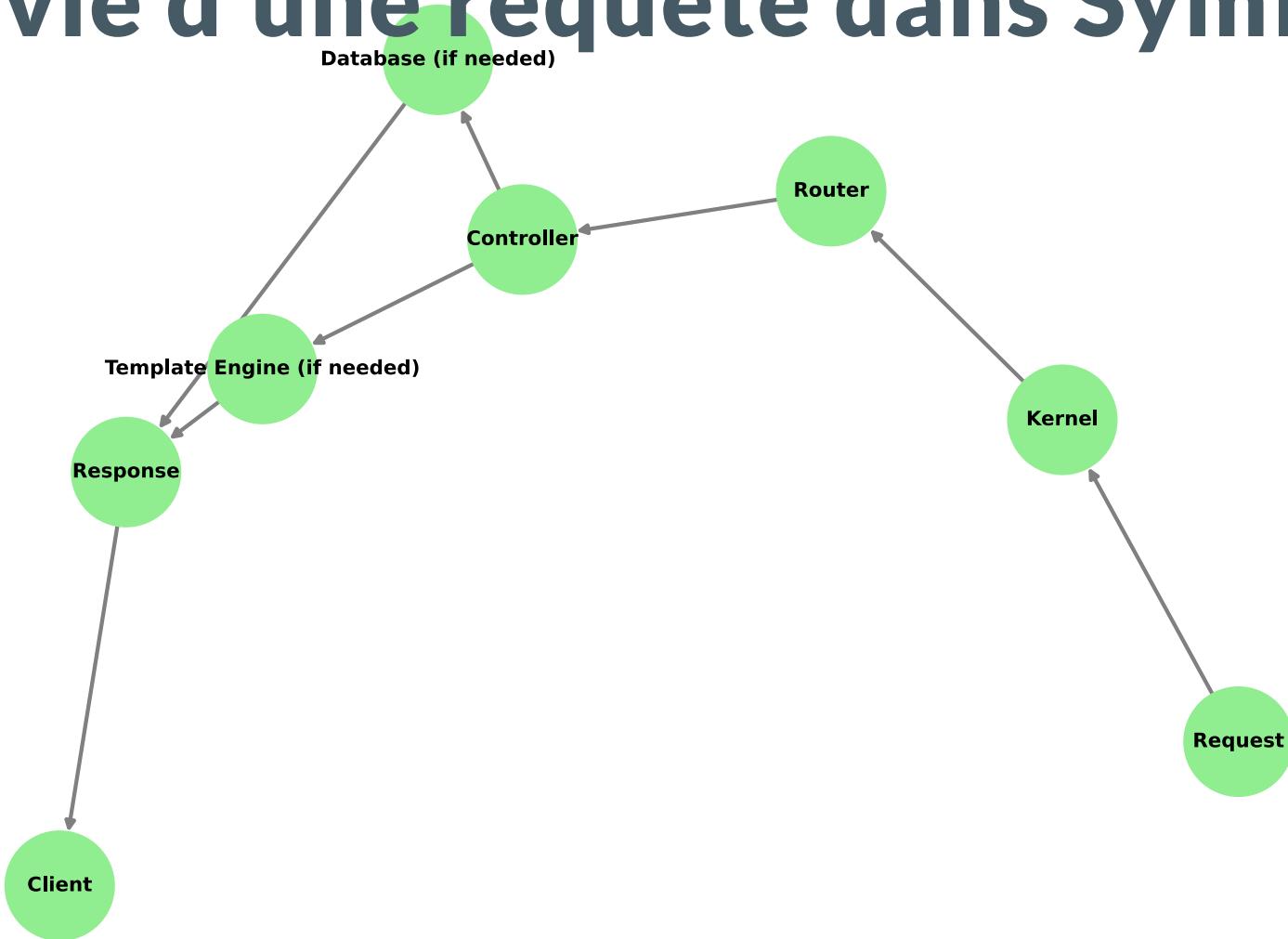
Avantages de Symfony

- **Modularité** : Utilisez uniquement ce dont vous avez besoin.
- **Performance** : Optimisé pour une exécution rapide.
- **Sécurité** : De nombreuses fonctionnalités pour protéger votre application.
- **Communauté** : Une grande communauté de développeurs et de nombreuses ressources disponibles.

Architecture de Symfony



Cycle de vie d'une requête dans Symfony



En résumé

Symfony est un choix puissant pour le développement d'applications web en PHP. Grâce à ses outils et sa communauté, il permet de construire des applications solides et maintenables.

Qu'est-ce qu'un framework ?

Définition d'un framework

Un framework (ou cadre d'application) est un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel (architecture).

Pourquoi utiliser un framework ?

- **Rapidité de développement** : Évite de "réinventer la roue"
- **Maintenabilité** : Code plus organisé et standardisé
- **Sécurité** : Les frameworks populaires bénéficient de mises à jour régulières
- **Communauté** : Documentation, support et extensions

Framework vs. Bibliothèque

- **Framework** : Il dicte l'architecture de votre application (inversion de contrôle).
- **Bibliothèque** : Vous l'intégrez à votre application et lappelez quand nécessaire.

Schéma : Application sans framework

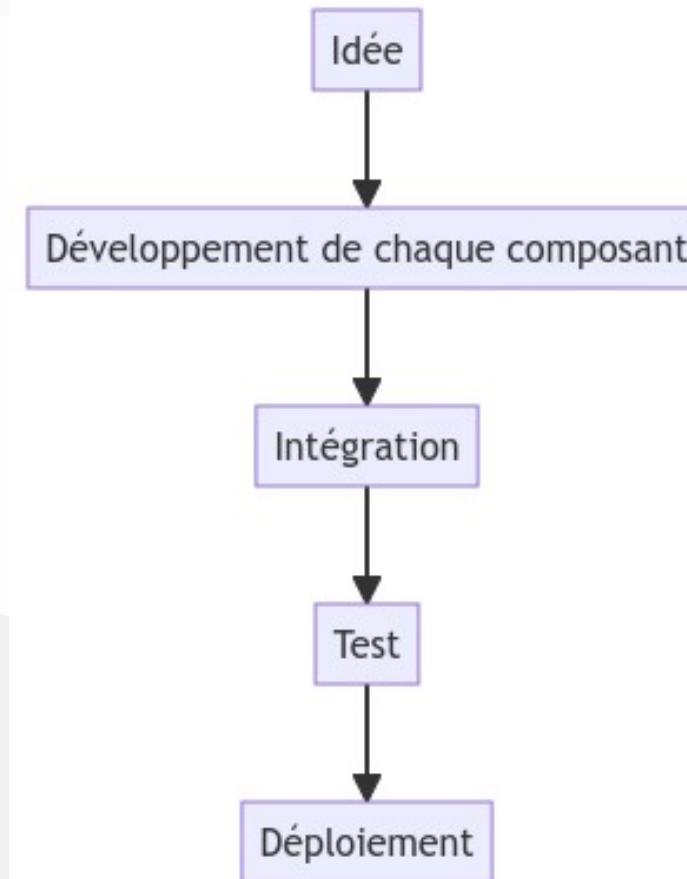
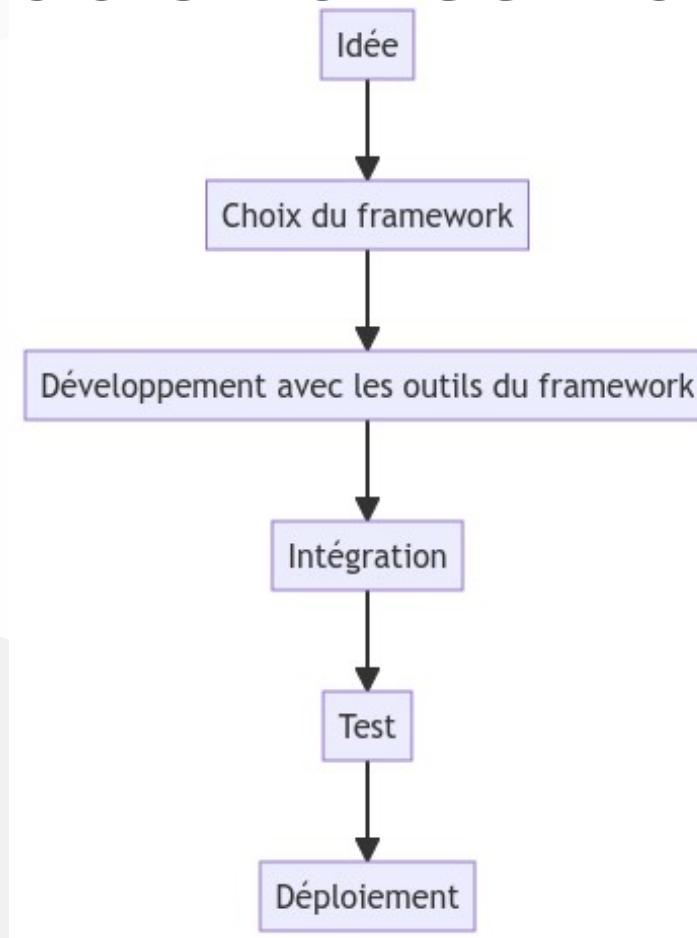


Schéma : Application avec framework



Les avantages en un coup d'œil

- **Moins de code à écrire**
- **Des pratiques standardisées**
- **Meilleure sécurité**
- **Support et communauté**

Questions/Réponses

Des questions sur les frameworks ?

Qu'est-ce que le MVC ?

- **MVC** : Modèle-Vue-Contrôleur
- Un pattern de conception pour séparer la logique de l'application

Définition

MVC signifie Modèle-Vue-Contrôleur. C'est un motif de conception (design pattern) utilisé dans le développement logiciel pour séparer une application en trois parties interconnectées.

Modèle

- Représente les données et la logique métier
- Gère la persistance (ex: interactions avec la base de données)

Les trois composants du MVC

- 1. Modèle (Model)** : Représente la logique métier et les données.
- 2. Vue (View)** : Affiche les données à l'utilisateur.
- 3. Contrôleur (Controller)** : Gère les entrées de l'utilisateur.

Vue

- Présente les données à l'utilisateur
- Affichage et mise en forme

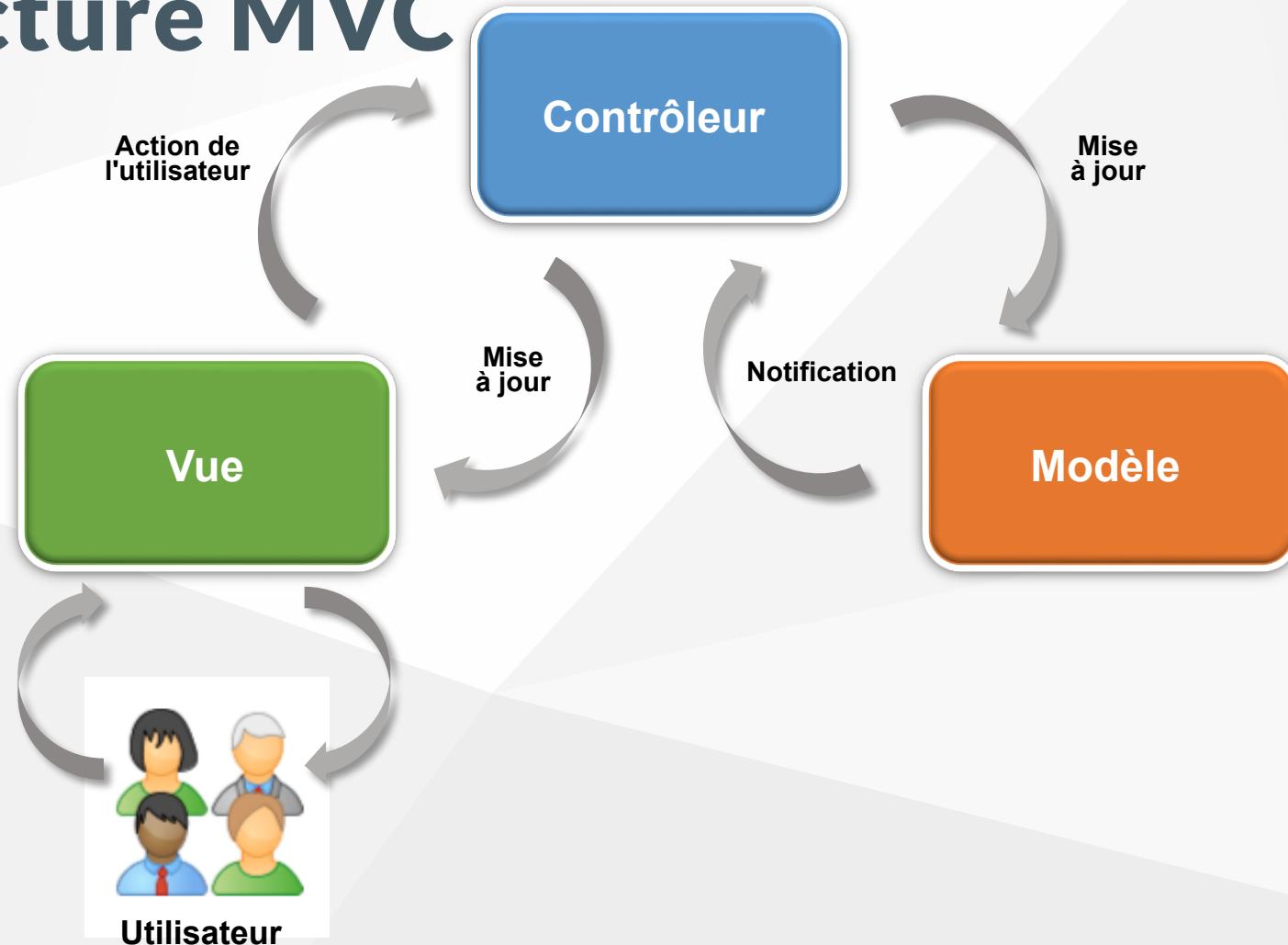
Contrôleur

- Gère les interactions utilisateur
- Met à jour le modèle et la vue en conséquence

Pourquoi utiliser le MVC ?

- **Séparation des préoccupations** : Chaque composant du MVC a un rôle bien défini.
- **Maintenabilité** : Il est plus facile de mettre à jour, de tester et de déboguer des applications structurées en MVC.
- **Réutilisabilité** : Les composants peuvent être réutilisés dans différentes parties de l'application ou dans d'autres applications.
- **Flexibilité** : Les modifications apportées à un composant n'affectent pas les autres.

Architecture MVC



En résumé

Le motif de conception MVC offre une structure claire pour le développement d'applications, permettant une séparation propre et une organisation du code. Cela facilite la maintenance, le test et l'évolutivité des applications.

Comment Symfony implémente le MVC ?

- Structure d'un projet Symfony typique
- Comment Symfony gère-t-il le MVC ?

Une organisation claire et modulaire

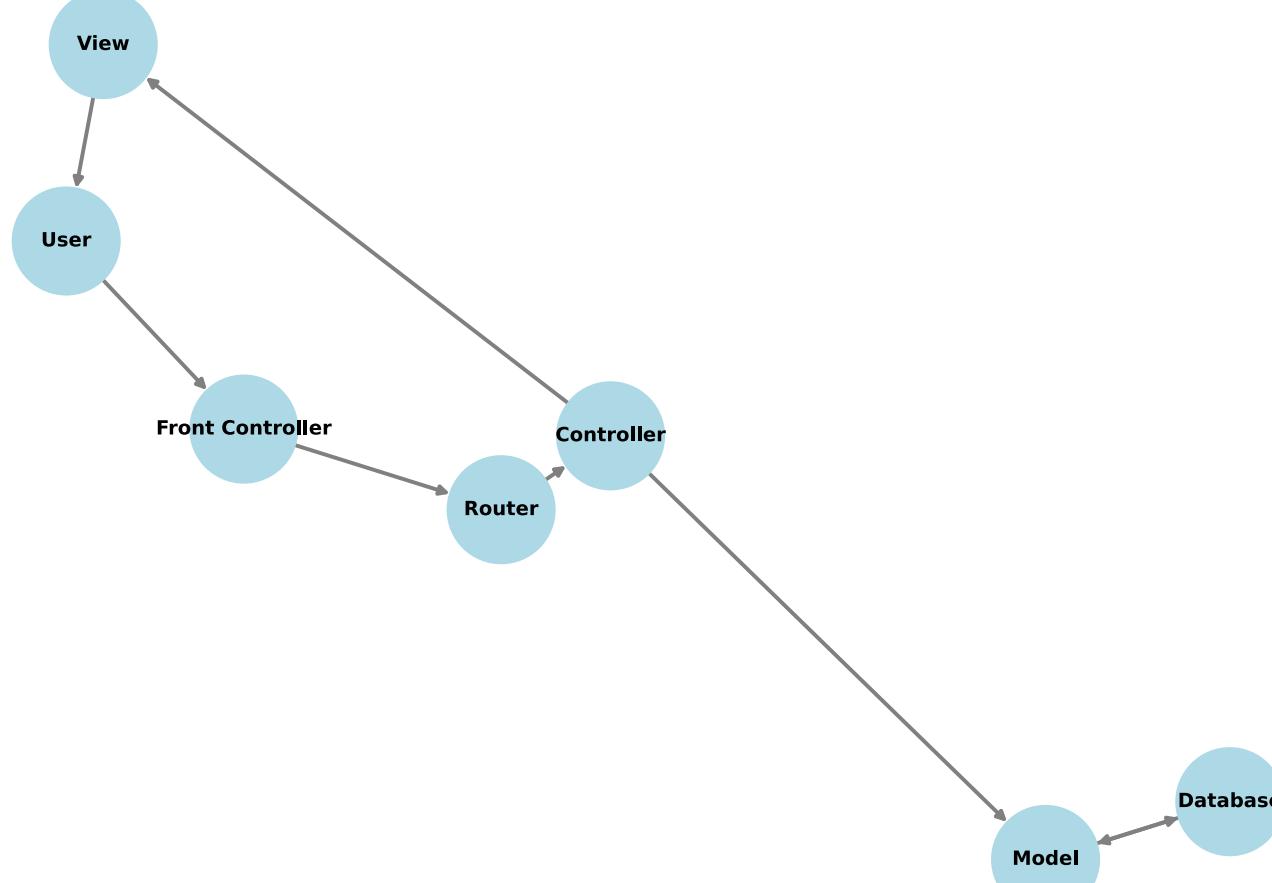
```
your_project/
├── assets/
├── bin/
│   └── console
├── config/
│   ├── packages/
│   └── services.yaml
├── migrations/
└── public/
    ├── build/
    └── index.php
src/
├── Kernel.php
├── Command/
├── Controller/
├── DataFixtures/
├── Entity/
├── EventSubscriber/
├── Form/
├── Repository/
├── Security/
└── Twig/
templates/
tests/
translations/
var/
    ├── cache/
    └── log/
vendor/
```

Comment Symfony gère-t-il le MVC ?

Symfony et le MVC

Bien que Symfony suive le principe du MVC, il le fait avec quelques variations. Symfony utilise une architecture MVC étendue avec des composants supplémentaires pour plus de flexibilité.

Architecture MVC de Symfony



Points clés de l'implémentation de Symfony

1. **Front Controller** : Toutes les requêtes passent par ce point d'entrée.
2. **Router** : Il détermine quel contrôleur doit traiter la requête.
3. **Controller** : Contient la logique pour traiter la requête et renvoyer une réponse.
4. **Model** : Interagit avec la base de données.
5. **View** : Prépare la sortie pour l'utilisateur.

Pourquoi cette implémentation ?

- **Flexibilité** : L'architecture permet d'introduire facilement des composants supplémentaires.
- **Réutilisabilité** : Les composants sont indépendants et peuvent être réutilisés dans différents projets.
- **Maintenabilité** : L'organisation claire du code facilite les mises à jour et la maintenance.

Conclusion

Symfony apporte une touche moderne au motif de conception MVC traditionnel, offrant une structure flexible et puissante pour le développement d'applications web complexes.

Questions/Réponses

- Des questions sur Symfony ?
- Des questions sur le MVC ?

Fin du CM

Merci pour votre attention !
À la prochaine séance !