

La programmation Orientée Objet : Design Patterns

Table des matières

I. Contexte	3
II. Les designs patterns en PHP	3
A. Les designs patterns en PHP	3
B. Exercice : Quiz	7
III. Le MVC en PHP	8
A. Le MVC en PHP	8
B. Exercice : Quiz	11
IV. Essentiel	11
V. Auto-évaluation	12
A. Exercice	12
B. Test	12
Solutions des exercices	13

I. Contexte

Durée : 1 heure

Prérequis : connaître les notions de base de la POO en PHP

Environnement de travail : un navigateur web, easyPHP et un éditeur de texte (type notepad++ ou Sublim) installés et configurés sur la machine de l'apprenant

Contexte

Que vous travailliez sur des projets personnels ou professionnels, les design patterns sont une compétence essentielle à maîtriser pour créer des applications performantes et de qualité.

L'utilisation des patrons de conception est devenue essentielle dans le développement d'applications modernes et évolutives. Les design patterns sont des solutions éprouvées pour résoudre des problèmes récurrents en programmation. En utilisant des patterns de conception, vous pouvez simplifier et optimiser votre code, améliorer sa maintenabilité, sa fiabilité et sa performance. Les design patterns vous permettent également de suivre les meilleures pratiques en matière de développement logiciel, en vous aidant à concevoir des applications structurées, flexibles et faciles à maintenir.

Enfin, l'utilisation de modèle de conception favorise la réutilisation du code, car ils permettent de modéliser des solutions génériques et adaptables à différentes situations. En somme, l'utilisation de design patterns en PHP est une compétence essentielle pour les développeurs qui cherchent à améliorer la qualité de leur code, leur productivité et leur employabilité.

II. Les designs patterns en PHP

A. Les designs patterns en PHP

Les design patterns en PHP sont des solutions conceptuelles réutilisables aux problèmes courants rencontrés dans le développement de logiciels. Ils permettent aux développeurs de résoudre efficacement les problèmes récurrents en utilisant des solutions optimisées et éprouvées. Ce ne sont pas des algorithmes ou des fichiers de code à intégrer dans un projet, mais plutôt des règles de conception en génie logiciel répondant à une problématique. Les design patterns en PHP sont divisés en trois catégories principales : les patrons de création, de structure et de comportement.

Les patterns de création sont utilisés pour déterminer la manière dont les objets sont créés et initialisés dans une application. Ils offrent une meilleure flexibilité et une meilleure extensibilité pour les objets.

Les patterns de structure sont utilisés pour organiser les objets et les classes en une structure cohérente et facile à comprendre. Ils permettent aux développeurs de concevoir une architecture claire et facilement maintenable pour leur code.

Les patterns de comportement sont utilisés pour gérer le comportement des objets et des classes, définissant comment ils interagissent entre eux et avec le reste de l'application.

Connaître et savoir utiliser les design patterns est crucial pour tout développeur qui souhaite créer des applications efficaces et bien structurées. Les modèles de conception permettent aux développeurs de résoudre efficacement les problèmes courants rencontrés lors du développement de logiciels, en utilisant des solutions optimisées et éprouvées. En comprenant comment les différents patterns fonctionnent, les développeurs peuvent écrire du code de manière plus structurée, modulaire et réutilisable, ce qui facilite la maintenance et la mise à jour des applications.

Les design patterns jouent un rôle crucial dans le respect de principes de développement universels tels que KISS (Keep It Simple, Stupid), DRY (Don't Repeat Yourself) et SOLID (Single Responsibility, Open-Closed, Liskov Substitution, Interface Segregation, Dependency Inversion).

Le principe KISS préconise la simplicité et la clarté du code. Les design patterns favorisent la modularité et l'organisation du code, ce qui facilite sa compréhension et sa maintenance. En suivant les design patterns, on évite les solutions compliquées et on privilégie des structures simples et efficaces.

Le principe DRY vise à éviter la duplication de code. Les design patterns encouragent la réutilisation du code en encapsulant des comportements communs dans des composants réutilisables. Par exemple, le design pattern Factory permet de créer différents objets conformes à une interface commune, réduisant ainsi la duplication de code de création d'objets similaires.

Les principes SOLID sont des directives qui favorisent la conception de logiciels modulaires et extensibles. Les patrons de conceptions aident à respecter ces principes en séparant les responsabilités, en favorisant l'ouverture à l'extension plutôt qu'à la modification, en garantissant la compatibilité des sous-classes avec les superclasses, en définissant des interfaces spécifiques pour chaque client, et en favorisant l'inversion des dépendances.

De plus, les design patterns permettent de créer des applications extensibles et évolutives, car ils offrent une meilleure flexibilité dans la façon dont les objets sont créés et initialisés, organisés et se comportent. Ils permettent également aux développeurs de travailler en collaboration sur un projet, car ils utilisent des structures de code bien définies et facilement compréhensibles.

Enfin, la connaissance des design patterns en PHP est très appréciée dans l'industrie du développement de logiciels, car elle démontre la capacité du développeur à résoudre des problèmes complexes de manière efficace et à écrire un code de qualité professionnelle. Les développeurs qui ont une bonne compréhension des design patterns sont souvent considérés comme des atouts pour les entreprises qui cherchent à développer des applications performantes et maintenables.

Les patterns de création

Les design patterns de création sont un ensemble de modèles de conception qui permettent de créer des objets de manière efficace et structurée dans une application. Ils sont utilisés pour déterminer la manière dont les objets sont créés et initialisés dans une application. Ils permettent aux développeurs de créer des objets de manière efficace et de façon plus structurée, modulaire et réutilisable. Ces patterns offrent une variété d'options pour la création d'objets, en fonction des besoins spécifiques de l'application.

L'un des patterns de création les plus couramment utilisés en PHP est le Singleton Pattern. Ce pattern permet de s'assurer qu'il n'y a qu'une seule instance d'une classe donnée dans une application. Il est principalement utilisé pour contrôler l'accès à une ressource unique, comme une connexion à une base de données. Le Singleton Pattern garantit que l'instance unique d'une classe est partagée entre toutes les parties de l'application qui ont besoin d'y accéder, ce qui peut contribuer à améliorer les performances et la stabilité de l'application.

Un autre pattern de création important en PHP est le Factory Method Pattern. Ce pattern fournit une interface pour créer des objets dans une classe parente, ce qui permet aux sous-classes de modifier le type d'objets qui seront créés. Le Factory Method Pattern est particulièrement utile lorsque l'on travaille avec des classes abstraites ou des interfaces, car il permet aux sous-classes de déterminer quelle implémentation spécifique de l'objet sera utilisée.

Voici un exemple de Factory Method Pattern :

```
1 // Interface du produit
2 interface Product {
3     public function getName(): string;
4 }
5
6 // Implémentation d'un produit concret
7 class ConcreteProduct implements Product {
8     private string $name;
9
10    public function __construct(string $name) {
11        $this->name = $name;
12    }
13
14    public function getName(): string {
```

```

15         return $this->name;
16     }
17 }
18
19 // Interface de la Factory
20 interface ProductFactory {
21     public function createProduct(string $name): Product;
22 }
23
24 // Implémentation de la Factory
25 class ConcreteProductFactory implements ProductFactory {
26     public function createProduct(string $name): Product {
27         return new ConcreteProduct($name);
28     }
29 }
30
31 // Utilisation de la Factory
32 $factory = new ConcreteProductFactory();
33 $product = $factory->createProduct("Exemple");
34 echo $product->getName(); // Affiche "Exemple"

```

Dans cet exemple, nous avons une interface `Product`, elle déclare une méthode `getName()` qui renvoie une chaîne de caractères représentant le nom du produit.

Ensuite, nous avons une classe concrète `ConcreteProduct` qui implémente l'interface `Product`. Elle possède une propriété `$name` de type `string` qui stocke le nom du produit. Le constructeur de `ConcreteProduct` prend en paramètre une chaîne de caractères représentant le nom du produit et l'assigne à la propriété `$name`. La méthode `getName()` de `ConcreteProduct` renvoie le nom du produit.

Nous avons également une interface `ProductFactory` qui définit la méthode `createProduct()` pour créer des produits. Cette interface assure que toutes les Factory respecteront le contrat de création de produits.

Ensuite, nous avons une classe concrète `ConcreteProductFactory` qui implémente l'interface `ProductFactory`. Elle fournit une implémentation de la méthode `createProduct()`. Cette méthode prend en paramètre une chaîne de caractères représentant le nom du produit à créer. À l'intérieur de la méthode, nous instancions un objet `ConcreteProduct` en utilisant le nom fourni, puis nous le retournons.

Dans notre exemple d'utilisation, nous créons une instance de `ConcreteProductFactory` en tant que `ProductFactory`. Ensuite, nous utilisons cette Factory pour créer un produit avec le nom "Exemple" en appelant la méthode `createProduct("Exemple")`. Enfin, nous utilisons la méthode `getName()` du produit créé pour obtenir et afficher son nom.

L'utilisation des types de variables dans cet exemple permet de garantir la cohérence des types de données tout au long du processus de création des produits, améliorant ainsi la robustesse et la sécurité du code.

Le Builder Pattern est un autre pattern de création en PHP qui permet de créer des objets complexes à partir de parties individuelles. Ce pattern est particulièrement utile lorsqu'il est nécessaire de créer des objets avec des configurations différentes à partir de la même base de code. En utilisant le Builder Pattern, les développeurs peuvent créer des objets avec différentes configurations en utilisant une même base de code.

Enfin, le Prototype Pattern est un pattern de création en PHP qui permet de cloner des objets existants pour en créer de nouveaux. Ce pattern est particulièrement utile lorsqu'il est nécessaire de créer des objets qui ont des configurations similaires à des objets existants. En utilisant le Prototype Pattern, les développeurs peuvent cloner des objets existants pour créer de nouveaux objets avec des configurations similaires.

Enfin, nous devons mentionner également le Simple Factory Pattern et le Static Factory Pattern. Le Simple Factory Pattern est un concept simplifié du Factory Method Pattern, où une méthode statique est utilisée pour créer des objets plutôt qu'une classe parente. Le Static Factory Pattern, quant à lui, utilise des méthodes statiques pour créer des objets, plutôt qu'une classe parente.

En utilisant ces différents patterns de création en PHP, les développeurs peuvent créer des objets de manière efficace et structurée, ce qui contribue à améliorer la qualité et la maintenabilité de leurs applications.

Les patterns de structure

Les design patterns de structure sont utilisés pour organiser et structurer le code d'une application, en séparant les responsabilités des différents composants de l'application. Ces patterns permettent de résoudre des problèmes communs liés à la gestion de la complexité du code et de faciliter la maintenance et l'évolutivité de l'application.

Les design patterns de structure sont utilisés pour résoudre des problèmes courants de conception de logiciels tels que la gestion de dépendances, la modularité et l'organisation du code. Voici quelques utilisations courantes des design patterns de structure en PHP :

- Organiser le code : les design patterns de structure aident à organiser le code de manière logique et à le rendre plus facilement compréhensible et maintenable. Par exemple, le design pattern MVC permet de séparer la logique métier, l'interface utilisateur et la gestion des interactions entre les deux.
- Gérer les dépendances : les design patterns de structure permettent de gérer les dépendances entre les différentes parties d'une application. Par exemple, le design pattern Dependency Injection permet de fournir des dépendances à une classe de manière flexible et configurable.
- Favoriser la modularité : les design patterns de structure favorisent la modularité en permettant de créer des composants réutilisables qui peuvent être facilement intégrés dans différentes parties d'une application. Par exemple, le design pattern Adapter permet d'adapter une interface pour une utilisation dans un contexte différent.
- Améliorer la maintenabilité : les design patterns de structure améliorent la maintenabilité en réduisant la complexité du code et en le rendant plus facile à comprendre et à modifier. Par exemple, le design pattern Façade permet de fournir une interface simple pour un sous-système complexe.

En utilisant les design patterns de structure, les développeurs peuvent améliorer la qualité de leur code, accélérer le développement d'applications et réduire les coûts de maintenance à long terme.

Parmi les différents types de patterns de structure, nous retrouvons tout d'abord le MVC. Le MVC (Modèle-Vue-Contrôleur) est un design pattern très courant qui sépare l'interface utilisateur, la logique métier et la gestion des interactions entre les deux. Ce pattern est souvent utilisé pour les applications web, car il permet une meilleure organisation du code et une maintenance plus facile. Laravel, un framework PHP populaire, est un exemple d'application qui utilise le design pattern MVC.

D'autres patterns couramment utilisés incluent l'Adapter Pattern, qui permet de faire communiquer des objets ayant des interfaces incompatibles ; le Bridge Pattern, qui permet de décomposer une classe complexe en deux hiérarchies distinctes pour offrir une plus grande flexibilité dans la conception de l'application ; et le Composite Pattern, qui permet de traiter les objets individuels et les groupes d'objets de la même manière en utilisant une structure en arbre. Les autres patterns incluent le Decorator Pattern, qui permet d'ajouter des fonctionnalités à un objet existant sans modifier son interface, le Façade Pattern, qui fournit une interface unifiée à un ensemble complexe de classes pour faciliter leur utilisation, le Flyweight Pattern, qui permet de partager des objets pour réduire l'utilisation de la mémoire, et le Proxy Pattern, qui permet de contrôler l'accès à un objet en utilisant un objet de proxy qui agit comme une interface intermédiaire.

En utilisant ces différents patterns de structure en PHP, les développeurs peuvent organiser leur code de manière efficace et structurée, ce qui contribue à améliorer la qualité et la maintenabilité de leurs applications. Ces patterns sont des outils essentiels pour les développeurs qui cherchent à améliorer la qualité de leur application web.

Les patterns de comportement

Les design patterns de comportement en PHP sont utilisés pour gérer les interactions entre les objets d'une application et pour définir des algorithmes réutilisables. Ils permettent de résoudre des problèmes de conception courants tels que la gestion de contrôle du flux, la communication entre objets, la gestion des états et la gestion des événements.

Les design patterns de comportement comprennent différents modèles. Chacun de ces modèles résout des problèmes spécifiques, tels que la gestion des algorithmes, la notification des changements, la décomposition de requêtes complexes, l'encapsulation des commandes, la définition de nouvelles opérations, la modélisation des transitions d'état, le stockage temporaire de données, l'accès séquentiel aux éléments d'une collection, la capture et la restauration de l'état d'un objet, et la sélection d'objets en fonction de critères.

L'utilisation des design patterns de comportement en PHP présente de nombreux avantages. Tout d'abord, ces modèles de comportement peuvent améliorer la qualité et la robustesse des applications en permettant aux développeurs de créer des algorithmes réutilisables et facilement intégrables dans différentes parties de l'application. Ils permettent également d'ajouter des fonctionnalités à une application sans avoir à modifier le code existant.

Par exemple, le design pattern Decorator permet d'ajouter des fonctionnalités à une classe existante en utilisant une approche de composition plutôt que d'héritage, ce qui facilite grandement la flexibilité du code.

En outre, les design patterns de comportement aident à gérer les états complexes des objets en définissant des machines à états finis ou des états pour les objets, ce qui simplifie grandement la gestion des états dans les applications.

Enfin, les design patterns de comportement peuvent également simplifier la communication entre les objets en définissant des canaux de communication spécifiques ou en utilisant des observateurs pour notifier les objets lorsqu'un événement se produit.

L'utilisation des design patterns de comportement peut considérablement améliorer la qualité, la flexibilité et la maintenabilité des applications. Parmi ces design patterns, le Strategy Pattern permet de définir une famille d'algorithmes encapsulés dans des classes distinctes, qui peuvent être utilisées de manière interchangeable selon les besoins de l'application. Ce pattern est souvent utilisé pour permettre à l'utilisateur de choisir entre plusieurs algorithmes pour une même tâche, sans avoir à modifier le code de l'application.

L'Observer Pattern est un autre design pattern de comportement, qui permet de définir une relation « *un à plusieurs* » entre objets. Lorsqu'un objet change d'état, tous les objets qui en dépendent sont notifiés et mis à jour automatiquement. Ce pattern est souvent utilisé pour les interfaces graphiques, pour que les widgets réagissent automatiquement aux changements d'état d'autres widgets. Parmi les autres design patterns de comportement populaires, nous retrouvons le Command Pattern, le Visitor Pattern, le State Pattern et l'Iterator Pattern. Chacun de ces patterns offre une solution spécifique à un problème de conception courant, permettant ainsi aux développeurs de créer des applications plus robustes et plus flexibles.

B. Exercice : Quiz

[solution n°1 p.15]

Question 1

Qu'est-ce qu'un design pattern en PHP ?

- ☐ Une solution réutilisable aux problèmes courants rencontrés dans le développement de logiciels
- ☐ Un code source pré-écrit que l'on peut utiliser directement dans son application
- ☐ Une fonctionnalité intégrée dans le langage PHP

Question 2

Les design patterns en PHP se divisent en combien de catégories principales ?

- ☐ Une catégorie
- ☐ Deux catégories
- ☐ Trois catégories

Question 3

Quel est le rôle des patterns de création ?

- ☐ Organiser les objets et les classes en une structure cohérente et facile à comprendre
- ☐ Gérer le comportement des objets et des classes
- ☐ Déterminer la manière dont les objets sont créés et initialisés

Question 4

Quel est le rôle des patterns de structure ?

- ☐ Déterminer la manière dont les objets sont créés et initialisés
- ☐ Organiser les objets et les classes en une structure cohérente et facile à comprendre
- ☐ Gérer le comportement des objets et des classes

Question 5

Quel est le rôle des patterns de comportement ?

- ☐ Déterminer la manière dont les objets sont créés et initialisés
- ☐ Organiser les objets et les classes en une structure cohérente et facile à comprendre
- ☐ Gérer le comportement des objets et des classes

III. Le MVC en PHP

A. Le MVC en PHP

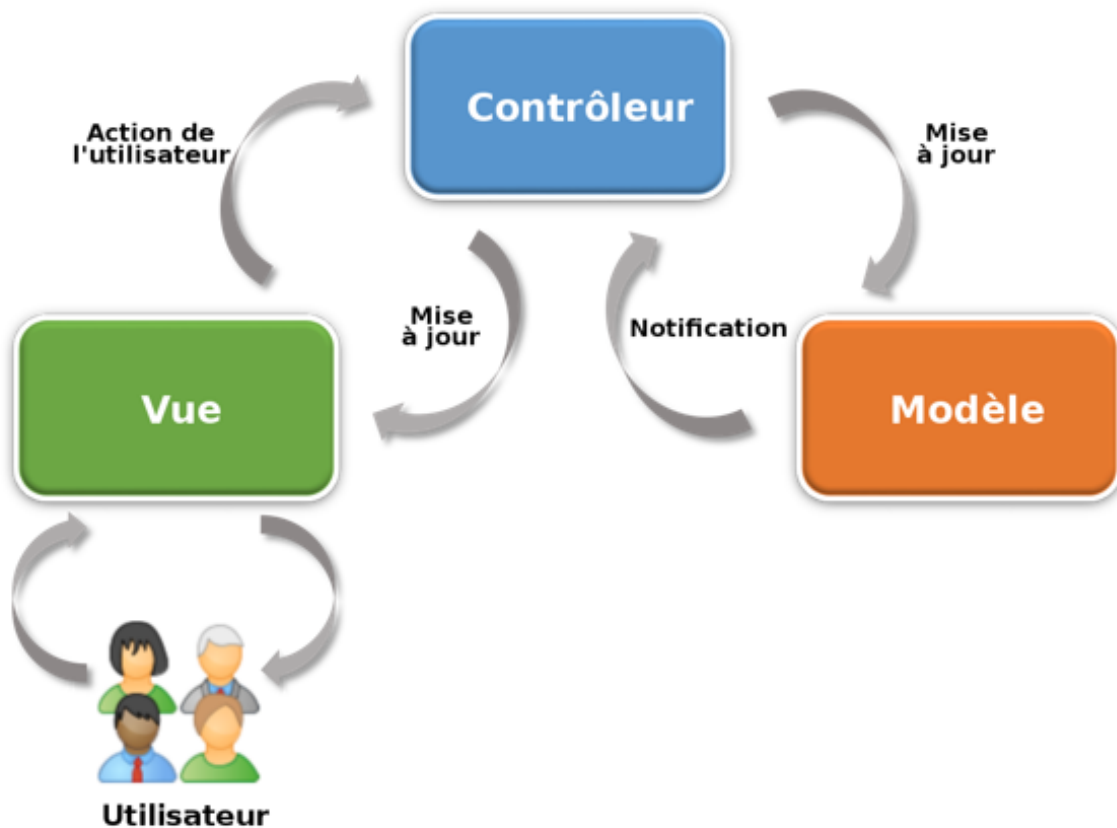
Le pattern MVC en PHP suit le principe de séparation des préoccupations (Separation of Concerns) qui permet de séparer les différentes fonctionnalités de l'application. Cette séparation des préoccupations permet de simplifier le code en le divisant en plusieurs parties et de faciliter la maintenance et la réutilisation du code.

Le Modèle est la couche qui représente les données brutes de l'application. Il contient toutes les requêtes SQL, les interactions avec la base de données et les fonctions qui récupèrent et manipulent les données. Les classes Modèle doivent être simples et contenir uniquement des fonctions liées à la récupération, à la mise à jour et à la suppression des données. Le Modèle doit également être indépendant de la Vue et du Contrôleur.

La Vue est responsable de l'affichage des données et de la gestion de l'interface utilisateur. Elle contient le HTML, le CSS et le JavaScript qui permettent d'afficher les données de manière interactive. Elle ne doit pas contenir de logique métier ou d'accès aux données. La Vue peut être considérée comme une représentation des données sous forme de pages web.

Le Contrôleur est responsable de la gestion des événements de l'application. Il reçoit les entrées de l'utilisateur, les traite et met à jour le Modèle en conséquence. Le Contrôleur est également responsable de la communication avec la Vue pour afficher les résultats de l'interaction utilisateur. Le contrôleur ne doit pas contenir de logique métier ou d'accès aux données.

Lors de la mise en place du pattern MVC en PHP, il est recommandé d'organiser les fichiers en fonction de leur responsabilité. Le Modèle, la Vue et le Contrôleur doivent être stockés dans des dossiers séparés. Les noms de fichiers doivent être descriptifs et correspondre aux fonctions qu'ils remplissent. Par exemple, les fichiers Modèle doivent avoir des noms tels que "UserModel.php" ou "ProductModel.php", tandis que les fichiers Vue peuvent avoir des noms tels que "UserView.php" ou "ProductView.php".



Source : wikimédia¹

Modèle

Le modèle en MVC est responsable de la gestion des données de l'application. Il représente la structure de ces données et fournit des méthodes pour les récupérer, les modifier et les supprimer. Le modèle est généralement lié à une base de données ou à un autre système de stockage de données.

Le rôle principal du modèle est de séparer les données de l'application de la logique de présentation et de contrôle. En d'autres termes, le modèle ne se préoccupe pas de la manière dont les données sont affichées ou de la manière dont les utilisateurs interagissent avec elles, mais plutôt de leur stockage et de leur manipulation.

Le modèle peut également être responsable de la validation des données entrantes pour s'assurer qu'elles sont conformes aux règles et aux contraintes de l'application. Il peut également contenir des méthodes pour effectuer des opérations complexes sur les données, telles que le calcul de statistiques ou la génération de rapports.

Vue

La vue dans le pattern MVC en PHP est responsable de la présentation des données au sein de l'application. Elle récupère les données du modèle puis les affiche à l'utilisateur.

Le rôle principal de la vue est de séparer la logique de présentation de la logique métier de l'application. Elle ne se préoccupe pas de la manière dont les données sont stockées ou manipulées, mais plutôt de la manière dont elles sont présentées à l'utilisateur. Elle peut également être responsable de la gestion des interactions de l'utilisateur avec l'interface utilisateur, telles que la saisie de données ou les clics sur des boutons.

¹ https://commons.wikimedia.org/wiki/File:Model-View-Controller_architectural_pattern.svg

La vue doit être capable de s'adapter aux changements dans le modèle, sans nécessiter de modifications dans le contrôleur ou dans la logique métier. Elle doit également être capable de s'adapter aux différents types de clients, tels que les navigateurs web, les applications mobiles ou les API.

Contrôleur

Le contrôleur dans le MVC est responsable de la gestion des interactions entre l'utilisateur, la vue et le modèle. Il est chargé de traiter les requêtes de l'utilisateur, de récupérer les données nécessaires auprès du modèle, et de les transmettre à la vue pour les afficher.

Le rôle principal du contrôleur est de séparer la logique de présentation et de contrôle de la logique métier de l'application. En d'autres termes, le contrôleur ne se préoccupe pas de la manière dont les données sont stockées ou manipulées, mais plutôt de la manière dont les utilisateurs interagissent avec l'application et de la manière dont les données sont présentées.

Le contrôleur est également responsable de la gestion des actions de l'utilisateur, telles que les soumissions de formulaires ou les clics sur des boutons, et de la redirection de l'utilisateur vers la vue appropriée en fonction de ces actions. Il peut également être responsable de la gestion des erreurs et des exceptions, et de la mise à jour du modèle en fonction des actions de l'utilisateur.

Mise en place du pattern MVC en PHP

Lors de la mise en place du pattern MVC en PHP, il est important d'organiser correctement les fichiers et les dossiers pour faciliter la maintenance et la compréhension du code. Généralement, nous pouvons organiser les fichiers et les dossiers de la manière suivante :

- Un dossier "models" contiendra toutes les classes du modèle,
- Un dossier "views" contiendra toutes les vues de l'application,
- Un dossier "controllers" contiendra tous les contrôleurs de l'application,
- Un fichier index.php servira de point d'entrée de l'application.

Dans le dossier "models", chaque classe représentera une table de la base de données ou une entité de l'application. Les classes du modèle auront pour responsabilité de communiquer avec la base de données et de fournir des méthodes pour manipuler les données.

Dans le dossier "views", chaque vue représentera une page de l'application. Les vues auront pour responsabilité de présenter les données du modèle à l'utilisateur et de gérer les interactions utilisateur.

Dans le dossier "controllers", chaque contrôleur représentera un ensemble d'actions liées à une partie de l'application. Les contrôleurs auront pour responsabilité de récupérer les données du modèle, d'appeler la vue appropriée et de gérer les interactions utilisateur.

Le fichier index.php sera le point d'entrée de l'application et sera responsable d'inclure les fichiers nécessaires pour initialiser l'application. Il aura également pour responsabilité de router les requêtes utilisateur vers les contrôleurs appropriés.

Le pattern MVC est largement utilisé en PHP pour développer des applications web évolutives et maintenables. Il offre une séparation claire de la logique de présentation et de la logique métier de l'application. Ce modèle est particulièrement utile pour les applications de commerce électronique, les applications de gestion de contenu, les applications de médias sociaux et les applications de réservation.

Les applications de commerce électronique nécessitent souvent une base de données complexe avec de nombreuses tables. Le pattern MVC permet de séparer la logique de présentation de la logique métier de l'application, ce qui facilite la gestion des données et l'affichage des résultats à l'utilisateur. Les applications de gestion de contenu ont des besoins similaires en termes de gestion de la base de données et de présentation des données à l'utilisateur, et le pattern MVC est également très utile pour ce type d'application. Les applications de médias sociaux ont des interactions complexes avec les utilisateurs, et le pattern MVC permet de séparer la logique

de présentation de la logique métier de l'application, ce qui facilite la gestion de ces interactions. Enfin, les applications de réservation nécessitent une gestion complexe des données, et le pattern MVC est très utile pour séparer la logique de présentation de la logique métier de l'application.

B. Exercice : Quiz

[solution n°2 p.16]

Question 1

Qu'est-ce que le pattern MVC en PHP ?

- ☐ Un design pattern pour la création de bases de données
- ☐ Un design pattern pour l'organisation du code dans une application PHP
- ☐ Un design pattern pour la création de formulaires en HTML

Question 2

Quel est le rôle du contrôleur dans le pattern MVC en PHP ?

- ☐ Le contrôleur est responsable de la gestion des données de l'application
- ☐ Le contrôleur est responsable de l'affichage des données dans l'interface utilisateur
- ☐ Le contrôleur est responsable de la gestion des interactions de l'utilisateur avec l'application

Question 3

Comment la vue communique-t-elle avec le contrôleur dans le pattern MVC en PHP ?

- ☐ En envoyant des requêtes HTTP au contrôleur
- ☐ En appelant des fonctions du contrôleur directement
- ☐ En utilisant un modèle de données partagé avec le contrôleur

Question 4

Quel est le rôle du modèle dans le pattern MVC en PHP ?

- ☐ Le modèle est responsable de l'affichage des données dans l'interface utilisateur
- ☐ Le modèle est responsable de la gestion des interactions de l'utilisateur avec l'application
- ☐ Le modèle est responsable de la gestion des données de l'application

Question 5

Le pattern MVC est la solution à tous les problèmes de conception d'application en PHP.

- ☐ Vrai
- ☐ Faux

IV. Essentiel

En conclusion, l'utilisation de design patterns en PHP est essentielle pour la conception de logiciels de qualité, maintenables et évolutifs. Les design patterns fournissent des solutions éprouvées pour les problèmes courants dans le développement de logiciels, offrant une structure claire et cohérente pour le code. En utilisant des design patterns, les développeurs peuvent éviter de réinventer la roue et se concentrer sur les aspects uniques de leur application, tout en bénéficiant d'un code plus facile à maintenir et à comprendre.

L'utilisation de design patterns en PHP améliore également la collaboration entre les développeurs, car les design patterns sont largement documentés et connus. Cela facilite la communication et la compréhension entre les membres de l'équipe, ce qui peut accélérer le processus de développement.

En outre, les design patterns permettent aux développeurs d'éviter les pièges courants qui peuvent conduire à des bogues et à une maintenance difficile à long terme. En utilisant les design patterns appropriés, les développeurs peuvent créer des applications robustes et évolutives qui répondent aux besoins de leurs utilisateurs.

Ainsi, l'utilisation de design patterns en PHP est un élément clé de la conception de logiciels de qualité, et tous les développeurs devraient être familiers avec ces concepts. Les design patterns offrent des solutions éprouvées pour les problèmes courants dans le développement de logiciels, tout en améliorant la maintenabilité et la fiabilité du code.

V. Auto-évaluation

A. Exercice

Vous êtes développeur web au sein d'une équipe de 10 personnes et devez réaliser une application de e-commerce pour l'entreprise Canar-store en vue de permettre la vente de canards en ligne. Cette application devant permettre la connexion des utilisateurs, la gestion de leurs paniers ainsi que celle du stock disponible.

Question 1

[solution n°3 p.17]

Lors de la phase de conception, le lead développeur vous demande quel patron de conception serait le plus judicieux à utiliser. Justifiez votre réponse avec au moins deux arguments.

Question 2

[solution n°4 p.17]

Après la mise en production de l'application, le magasin souhaite de se mettre à la vente de poisson plutôt que celle de canard. Le lead développeur vous demande alors s'il est préférable de recommencer le projet depuis le début ou alors de le modifier. Les modifications nécessaires impacteront autant le design de l'application que sa base de données sans pour autant en changer la logique métier.

B. Test

Exercice 1 : Quiz

[solution n°5 p.17]

Question 1

Dans le modèle MVC, les modèles et les vues doivent communiquer directement entre eux.

- ☐ Vrai
- ☐ Faux

Question 2

Quel est le rôle de la vue dans le modèle MVC ?

- ☐ Gérer les données de l'application
- ☐ Contrôler les interactions entre l'utilisateur et l'application
- ☐ Afficher les données à l'écran

Question 3

Les design patterns de création en PHP permettent de créer des objets de manière optimisée et réutilisable.

- ☐ Vrai
- ☐ Faux

Question 4

Le pattern Singleton est un design pattern de création en PHP.

- ☐ Vrai
- ☐ Faux

Question 5


Pourquoi le pattern MVC en PHP est-il utile ?

- ☐ Il facilite la création d'applications PHP en fournissant une structure claire pour l'organisation du code
- ☐ Il permet de créer des applications PHP plus rapidement
- ☐ Il est obligatoire pour toutes les applications PHP

Solutions des exercices


Exercice p. 7 Solution n°1**Question 1**

Qu'est-ce qu'un design pattern en PHP ?

- ☒ Une solution réutilisable aux problèmes courants rencontrés dans le développement de logiciels
- ☐ Un code source pré-écrit que l'on peut utiliser directement dans son application
- ☐ Une fonctionnalité intégrée dans le langage PHP
-  Les design patterns sont des solutions éprouvées et réutilisables pour résoudre les problèmes courants rencontrés dans le développement de logiciels.


Question 2

Les design patterns en PHP se divisent en combien de catégories principales ?

- ☐ Une catégorie
- ☐ Deux catégories
- ☒ Trois catégories
-  Les design patterns en PHP sont divisés en trois catégories principales : les patterns de création, de structure et de comportement.


Question 3

Quel est le rôle des patterns de création ?

- ☐ Organiser les objets et les classes en une structure cohérente et facile à comprendre
- ☐ Gérer le comportement des objets et des classes
- ☒ Déterminer la manière dont les objets sont créés et initialisés
-  Les patterns de création sont utilisés pour déterminer la manière dont les objets sont créés et initialisés dans une application, offrant une meilleure flexibilité et une meilleure extensibilité pour les objets.


Question 4

Quel est le rôle des patterns de structure ?

- ☐ Déterminer la manière dont les objets sont créés et initialisés
- ☒ Organiser les objets et les classes en une structure cohérente et facile à comprendre
- ☐ Gérer le comportement des objets et des classes
-  Les patterns de structure sont utilisés pour organiser les objets et les classes en une structure cohérente et facile à comprendre, permettant une architecture claire et facilement maintenable pour le code.

Question 5


Quel est le rôle des patterns de comportement ?

- ☐ Déterminer la manière dont les objets sont créés et initialisés
- ☐ Organiser les objets et les classes en une structure cohérente et facile à comprendre
- ☒ Gérer le comportement des objets et des classes
-  Les patterns de comportement sont utilisés pour gérer le comportement des objets et des classes, définissant comment ils interagissent entre eux et avec le reste de l'application.

Exercice p. 11 Solution n°2


Question 1

Qu'est-ce que le pattern MVC en PHP ?

- ☐ Un design pattern pour la création de bases de données
- ☒ Un design pattern pour l'organisation du code dans une application PHP
- ☐ Un design pattern pour la création de formulaires en HTML
-  Le pattern MVC en PHP est un design pattern pour l'organisation du code dans une application PHP, qui permet de séparer les responsabilités entre le modèle, la vue et le contrôleur.


Question 2

Quel est le rôle du contrôleur dans le pattern MVC en PHP ?

- ☐ Le contrôleur est responsable de la gestion des données de l'application
- ☐ Le contrôleur est responsable de l'affichage des données dans l'interface utilisateur
- ☒ Le contrôleur est responsable de la gestion des interactions de l'utilisateur avec l'application
-  Le rôle du contrôleur dans le pattern MVC en PHP est de gérer les interactions de l'utilisateur avec l'application, de récupérer les données du modèle et de les transmettre à la vue pour affichage.


Question 3

Comment la vue communique-t-elle avec le contrôleur dans le pattern MVC en PHP ?

- ☒ En envoyant des requêtes HTTP au contrôleur
- ☐ En appelant des fonctions du contrôleur directement
- ☐ En utilisant un modèle de données partagé avec le contrôleur
-  La vue communique avec le contrôleur en envoyant des requêtes HTTP, généralement à l'aide d'un formulaire HTML ou d'un lien cliquable.

Question 4

Quel est le rôle du modèle dans le pattern MVC en PHP ?


- ☐ Le modèle est responsable de l'affichage des données dans l'interface utilisateur
- ☐ Le modèle est responsable de la gestion des interactions de l'utilisateur avec l'application
- ☒ Le modèle est responsable de la gestion des données de l'application
-  Le rôle du modèle dans le pattern MVC en PHP est de gérer les données de l'application, de stocker et de récupérer des données à partir de la base de données ou d'autres sources de données.

Question 5

Le pattern MVC est la solution à tous les problèmes de conception d'application en PHP.

☐ Vrai

☒ Faux

 Bien que le pattern MVC soit une solution courante pour la conception d'applications en PHP, il n'est pas nécessairement la solution à tous les problèmes de conception. Il est important d'évaluer les besoins spécifiques de chaque projet et de choisir le pattern le plus approprié en conséquence.

p. 12 Solution n°3

Pour la réalisation de l'application de e-commerce pour Canar-store, nous utiliserons le design pattern MVC. Cette approche est recommandée pour le développement d'applications web évolutives et maintenables. En effet, le MVC permet de séparer les différentes couches de l'application, facilitant ainsi le travail en équipe et la communication entre les différents développeurs.

La séparation de la logique métier et de la logique de l'application en fait un modèle de choix pour les applications de e-commerce. En utilisant le MVC, les contrôleurs peuvent récupérer les données de la base de données à l'aide des classes du modèle et les transmettre aux vues pour les afficher à l'utilisateur. Cette séparation permet une meilleure organisation du code et une maintenance plus facile de l'application.

En choisissant un design pattern tel que le MVC, il est possible de s'assurer que n'importe quel développeur puisse maintenir l'application sur le long terme. Cette approche favorise également la réutilisation de code, ce qui permet de gagner du temps et de l'efficacité dans le développement de nouvelles fonctionnalités. En somme, le choix du design pattern MVC apportera une certaine rigueur et une structure claire à l'application de e-commerce pour Canar-store.

p. 12 Solution n°4

Dans ce cas précis, il est préférable de modifier l'application existante plutôt que de la recommencer depuis le début. Cette décision est basée sur le fait que le design pattern MVC, utilisé pour concevoir l'application, permet de séparer la logique métier de l'application (le modèle), l'interface utilisateur (la vue) et la gestion des interactions entre les deux (le contrôleur).

Grâce à cette séparation, il est possible de changer le design de l'application (la vue) ou la structure de la base de données (le modèle) sans impacter la logique métier. En effet, cette dernière est gérée par les contrôleurs qui interagissent avec le modèle et la vue en utilisant une interface bien définie.


La modification demandée ici ne représente pas un changement majeur dans la logique métier de l'application. Il s'agit simplement de changer le type de produit vendu, ce qui peut être géré en modifiant la base de données et les vues correspondantes. En faisant cela, la logique métier de l'application reste intacte et le travail déjà effectué sur l'application n'est pas perdu.

Exercice p. 12 Solution n°5**Question 1**

Dans le modèle MVC, les modèles et les vues doivent communiquer directement entre eux.


☐ Vrai

☒ Faux

 Les modèles et les vues communiquent via le contrôleur, qui décide de l'action à effectuer.


Question 2

Quel est le rôle de la vue dans le modèle MVC ?

- ☐ Gérer les données de l'application
- ☐ Contrôler les interactions entre l'utilisateur et l'application
- ☒ Afficher les données à l'écran
-  La vue est responsable de l'affichage des données à l'utilisateur.


Question 3

Les design patterns de création en PHP permettent de créer des objets de manière optimisée et réutilisable.

- ☒ Vrai
- ☐ Faux
-  Les design patterns de création en PHP facilitent la création d'objets de manière optimisée et réutilisable.


Question 4

Le pattern Singleton est un design pattern de création en PHP.

- ☒ Vrai
- ☐ Faux
-  Le pattern Singleton est bien un design pattern de création en PHP.

Question 5

Pourquoi le pattern MVC en PHP est-il utile ?

- ☒ Il facilite la création d'applications PHP en fournissant une structure claire pour l'organisation du code
- ☐ Il permet de créer des applications PHP plus rapidement
- ☐ Il est obligatoire pour toutes les applications PHP
-  Le pattern MVC en PHP est utile, car il fournit une structure claire pour l'organisation du code dans les applications PHP, ce qui facilite la maintenance, la modification et l'évolutivité de l'application. Il ne s'agit pas d'une obligation, mais plutôt d'une bonne pratique recommandée dans le développement d'applications PHP.