

Aperçu d'ASP.NET MVC

par Philippe Vialatte (Traduction) ([ma page DVP](#)) ([Blog](#))


Date de publication : mai 2009

Dernière mise à jour :

En savoir plus sur les différences entre les applications ASP.NET MVC et les formulaires Web ASP.NET. Apprendre à décider de quand il est plus approprié de construire une application ASP.NET MVC.

Traduction.....	3
Introduction.....	3
Décider quand créer une application MVC.....	4
Avantages d'une application web basée sur MVC.....	4
Avantages d'une application web basée sur les formulaires Web.....	4
Caractéristiques du Framework ASP.NET MVC.....	5

Traduction

Cet article est la traduction la plus fidèle possible de l'article original :  [ASP.NET MVC Overview](#)

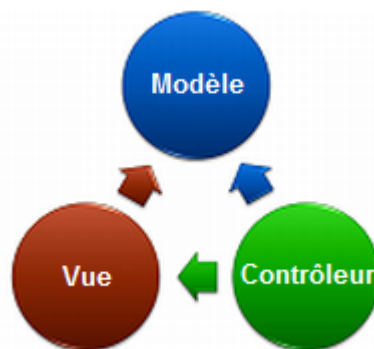
Introduction

Le patron d'architecture (architectural pattern) Modèle-Vue-Contrôleur (MVC) sépare une application en trois composants principaux : le modèle, la vue et le contrôleur. Le Framework ASP.NET MVC fournit une alternative au modèle de formulaires Web ASP.NET, en permettant la création d'applications web basées sur MVC.

Le Framework ASP.NET MVC est un Framework de présentation léger, facilement testable, qui (comme pour les applications utilisant les Formulaires Web) est intégré avec les fonctionnalités ASP.NET, dont les pages maîtres et les mécanismes d'authentification. Le Framework MVC est défini dans l'espace de noms **System.Web.Mvc** et est un élément fondamental, de l'espace de noms **System.Web**.

MVC est un schéma de conception auquel de nombreux développeurs sont habitués. Certains types d'applications Web bénéficieront du framework MVC. D'autres continueront d'utiliser le modèle d'application ASP.NET basé sur les formulaires Web et les postbacks. D'autres types d'applications Web combineront les deux approches, une approche n'excluant pas l'autre.

Le Framework MVC comprend les éléments suivants:



* **Modèle** : les modèles sont les parties de l'application qui mettent en œuvre la logique de l'application de données de domaine.

Souvent, les objets du modèle récupèrent et stockent l'état des objets dans une base de données. Par exemple, un objet **Product** peut récupérer les informations d'une base de données, les exploiter, puis sauvegarder les informations mises à jour dans une table **Products** dans SQL Server.

Dans de petites applications, le modèle est souvent une séparation conceptuelle au lieu d'une séparation physique. Par exemple, si l'application ne fait que lire une série de données et les envoyer à la vue, l'application n'a pas besoin d'avoir une couche modèle physique et des classes associées. Dans ce cas, l'ensemble des données assume le rôle d'un modèle objet.

* **Vues** : Les vues sont les éléments qui affichent l'interface utilisateur (UI). Typiquement, cette interface utilisateur est créée à partir du modèle de données.

Un exemple serait une vue d'édition d'un tableau de produits, affichant des zones de texte, des listes déroulantes et des cases à cocher basées sur l'état actuel d'un objet de type **Product**.

* **Contrôleurs** : Les contrôleurs sont les composants qui gèrent l'interaction avec l'utilisateur, travaillent avec le modèle et, finalement, sélectionnent la vue qui va permettre de faire un rendu de l'interface utilisateur.

Dans une application MVC, la vue ne fait qu'afficher les informations que le contrôleur gère et répond aux entrées et actions de l'utilisateur. Par exemple, le contrôleur traite les valeurs de chaîne de requête et transmet ces valeurs au modèle qui, en retour, effectue les requêtes sur la base de données en utilisant les valeurs.

Le pattern MVC vous aide à créer des applications qui séparent les différents aspects de l'application (traitement des données, logique métier et interface utilisateur), tout en fournissant un couplage lâche entre ces éléments. Le traitement des données fournies par l'utilisateur appartient au contrôleur. La logique métier appartient au modèle. Cette séparation vous permet de gérer la complexité lorsque vous construisez une application, car elle permet de vous concentrer sur un aspect de la mise en œuvre à la fois. Par exemple, vous pouvez vous concentrer sur la vue, sans devoir dépendre de la logique métier.

En plus de gérer la complexité, le modèle MVC rend plus facile de tester les applications qu'il n'est facile de tester un site basé sur les formulaires Web ASP.NET. Par exemple, dans un site Web basé sur les formulaires Web ASP.NET, une classe unique est utilisée à la fois pour gérer l'affichage et pour répondre aux actions de l'utilisateur. Ecrire des tests automatisés pour un site Web basé sur les formulaires ASP.NET peut être complexes, parce que, pour une page de test, vous devez instancier la classe de la page, l'ensemble de ses contrôles enfants, et toutes les classes dont la page dépend dans l'application. Etant donné que de nombreuses classes sont instanciées pour exécuter la page, il peut être difficile d'écrire des tests qui se concentrent sur des parties individuelles de l'application. Tester un site basé sur les formulaires Web ASP.NET peut donc être plus difficile que de tester une application MVC. De plus, des tests ciblant un site basé sur les formulaires Web ASP.NET nécessitent un serveur Web. Le Framework MVC dissocie les éléments et se base sur des interfaces, ce qui permet de tester les composants en isolation par rapport au reste du Framework.

Le couplage lâche entre les trois principales composantes d'une application MVC encourage aussi le développement parallèle. Par exemple, un développeur peut travailler sur les vues, un autre développeur peut travailler sur le code du contrôleur, et un troisième développeur peut se concentrer sur la logique métier dans le modèle.

Décider quand créer une application MVC

Vous devez examiner attentivement s'il y a lieu de mettre en oeuvre une application Web en utilisant le Framework MVC ou le modèle des formulaires Web ASP.NET. Le Framework MVC ne remplace pas le modèle de formulaires Web, vous pouvez utiliser le Framework de votre choix pour vos applications Web. (Si vous avez des applications existantes basées sur les formulaires Web, celles-ci continuent à fonctionner exactement comme elles l'ont toujours fait.)

Avant de décider d'utiliser le Framework MVC ou les formulaires Web pour un site Web spécifique, il faut peser les avantages de chaque approche.

Avantages d'une application web basée sur MVC

Le Framework ASP.NET MVC offre les avantages suivants :

- Elle rend plus facile à gérer la complexité en divisant l'application en un modèle, une vue et un contrôleur.
- Il n'utilise pas le viewstate, ou les contrôles serveurs. Cela rend le Framework MVC idéal pour les développeurs qui veulent un contrôle total sur le comportement de l'application.
- Il utilise un pattern **Contrôleur frontal** (Front Controller), qui traite les requêtes de l'application Web par l'intermédiaire d'un contrôleur unique. Cela vous permet de concevoir une application qui prend en charge une infrastructure de routage riche. Pour de plus amples renseignements, voir **Contrôleur frontal** sur le site Web MSDN.
- Il offre un meilleur support pour le développement dirigé par les tests (TDD).
- Il fonctionne bien pour des applications Web développées et soutenues par de grandes équipes de développeurs et de designers, qui ont besoin d'un degré élevé de contrôle sur le comportement des applications.

Avantages d'une application web basée sur les formulaires Web

Les formulaires Web basée offrent les avantages suivants :

- Ils fournissent un modèle d'événements, qui préserve l'état entre deux requêtes, ce qui permet d'accélérer le développement d'applications Web d'entreprise. Les formulaires Web fournissent des dizaines d'événements qui sont pris en charge dans des centaines de contrôles serveur.
- Ils utilisent un pattern **contrôleur de pages** (Page Controller), qui ajoute des fonctionnalités aux pages individuelles. Pour de plus amples renseignements, voir le **contrôleur de pages** sur le site Web MSDN.
- Ils utilisent le viewstate et des pages basées sur les contrôles serveurs, ce qui peut rendre plus facile la gestion des informations.
- Ils fonctionnent bien pour de petites équipes de développeurs et designers qui veulent profiter du grand nombre de composants disponibles pour développer rapidement des applications.
- En général, ils sont moins complexes pour le développement d'applications, car les composants (la page, les contrôles, etc) sont bien intégrés et nécessitent généralement moins de code que le modèle MVC.

Caractéristiques du Framework ASP.NET MVC

Le Framework ASP.NET MVC fournit les fonctionnalités suivantes :

- Séparation des tâches (traitement des données, logique métier, la logique et l'interface utilisateur), testabilité et développement dirigé par les tests par défaut. Tous les contrats de base dans le Framework MVC sont basés sur des interfaces et peuvent être testés en utilisant des simulacres (Mock Objects), qui sont des objets simulés, imitant le comportement des objets réels de l'application. Vous pouvez en tester de façon unitaire l'application sans avoir à exécuter les contrôleurs dans un processus ASP.NET, ce qui rend les tests unitaires rapides et flexibles. Vous pouvez utiliser n'importe quel Framework de test compatible avec le .NET Framework.
- Un Framework extensible et enfichable. Les composants du Framework ASP.NET MVC sont conçus de telle sorte qu'ils puissent facilement être remplacés ou personnalisés. Vous pouvez utiliser votre propre moteur de vues, vos règles de routage des URL, la sérialisation des paramètres et d'autres composants. Le Framework ASP.NET MVC permet également l'utilisation de Frameworks d'injection de dépendance (Dependency Injection) et d'Inversion de contrôle (Inversion Of Control). L'injection de dépendance permet d'injecter des objets dans une classe, au lieu de compter sur la classe pour créer l'objet. L'inversion de contrôles spécifie que si un objet requiert un autre objet, le premier objet devrait obtenir le deuxième objet à partir d'une source extérieure, par exemple, un fichier de configuration. Cela rend les tests plus faciles.
- Un outil de mappage d'URL puissant, qui vous permet de créer des applications dont les URL sont compréhensibles. Les URL n'ont pas à inclure les extensions du nom de fichier, et sont destinées à soutenir des modèles de désignation qui fonctionnent bien pour l'optimisation des moteurs de recherche (SEO) et pour l'adressage au format REST (representational state transfer).
- Support des balises existantes dans ASP.NET (fichiers .aspx), contrôles utilisateur (fichiers .ascx) et pages maîtres (fichiers .master), en tant que modèles de vues. Vous pouvez utiliser les fonctionnalités existantes d'ASP.NET avec le Framework ASP.NET MVC, telles que des pages maîtres imbriquées, des expressions en ligne (<% =%>), les contrôles serveur déclaratifs, les templates, le data-binding, la localisation, etc.
- Support des fonctionnalités existantes d'ASP.NET. ASP.NET MVC vous permet d'utiliser des dispositifs tels que les formulaires d'authentification et l'authentification Windows, les autorisations sur les URL, les fonctionnalités d'appartenance à un groupe et de rôle, le mécanisme de cache de données et de code produit, les sessions et la gestion du profil, la surveillance de la santé (health monitoring), le système de configuration et l'architecture des fournisseurs.