Introduction à ASP.NET MVC

Alors, ASP.NET MVC c'est quoi ? Ça va vraiment me permettre de créer des sites web ? En quoi c'est différent du HTML ? Pourquoi utiliser ASP.NET MVC plutôt qu'autre chose ? De quoi ai-je besoin pour commencer à apprendre ASP.NET MVC ? Et comment fonctionne un site web ?

Voilà tout ce que nous allons apprendre dans ce chapitre. Il va vous permettre de vous faire une bonne idée des étapes nécessaires pour réaliser un site web, et des solutions qu'ASP.NET MVC propose pour les franchir. Entrons dans le vif du sujet et découvrons ce qu'est ASP.NET…

Remarquez qu'ASP.NET MVC est utilisable également avec le langage de programmation VB.NET. Les principes sont les mêmes que le C# mais la syntaxe est quand même différente. Si vous connaissez le VB.NET mais pas le C#, vous pourrez sans doute adapter les exemples de code que vous trouverez dans ce cours.

Qu'est-ce qu'ASP.NET ?

Commençons par le commencement : comment ça se prononce ?
ASP.NET peut être prononcé à l'anglaise ("é est-ce pi dot nette"), à la franco-anglaise ("ah est-ce pé dot nette") ou encore à la française ("ah est-ce pé point nette"). À vous de choisir votre prononciation, celle-ci n'ayant aucune incidence sur votre apprentissage.

Et qu'est-ce que c'est donc ?
ASP.NET est la plateforme de développement de Microsoft permettant la réalisation d'applications web.

Même si nous allons réaliser des sites Internet, .NET ce n'est pas l'abréviation d'Internet, mais correspond, vous vous en doutez, au .NET du framework .NET.

Ce qu'on appelle « plateforme » ici est un ensemble de composants permettant de construire une application web. Quand nous construisons une maison, nous avons besoin d'un terrain, d'un chef de chantier, de matériaux de construction, d'un plan de construction, d'outils, et de savoir-faire (et sûrement d'autres choses). Et bien pour réaliser une application web, c'est un peu le même principe, nous avons besoin (entre autres) :

- D'un outil de développement.
- D'une boîte à fonctionnalités pour développer.
- D'une logique de développement.
- D'un langage de programmation.

L'outil de développement est Visual Studio (et comporte une version gratuite du nom de « Visual Studio Express pour le web »). Il est le logiciel qui va nous permettre de réaliser le site web et qui va nous fournir les outils pour orchestrer nos développements. C'est un peu le chef de chantier d'ASP.NET.
La boîte à fonctionnalités c'est le « framework .NET », qui comporte différents types de fonctionnalités :

- des fonctionnalités dites générales qui vont servir à toutes les applications utilisant le framework .NET, que ce soient des applications web ou des applications Windows, de téléphonie, des jeux, etc.
- des fonctionnalités spécifiques à la création de sites web : il s'agit d'ASP.NET. Dans notre chantier, le framework .NET représenterait les matériaux que l'on retrouve dans tout chantier, des briques, du ciment, etc. tandis qu'ASP.NET représenterait un matériau spécifique à un type de construction, par exemple le liner pour construire une piscine.

La logique de développement correspondrait un peu au plan de construction. Elle fournit un schéma directeur permettant la réalisation d'un site web. ASP.NET nous propose deux logiques pour construire une application web : WebForms et MVC. Dans ce cours, nous nous intéresserons à la logique de développement MVC. Enfin, le langage de programmation dans ce cours sera le C#, dont on reparlera un peu plus tard dans ce chapitre.

MVC et WebForms

On peut dire grosso modo qu'ASP.NET se décompose en deux grandes parties (les fameuses logiques!), à savoir :

- ASP.NET WebForms
- ASP.NET MVC

ASP.NET c'est le socle de la plateforme de développement de Microsoft pour réaliser des applications web. WebForms et MVC sont deux logiques différentes de développement s'inscrivant au-dessus du socle d'ASP.NET.

Nous allons étudier dans ce cours ASP.NET MVC et non pas ASP.NET WebForms et je vais vous expliquer pourquoi, mais avant tout qu'est-ce qu'ASP.NET WebForms ?

ASP.NET WebForms a été créé en 2002 et en est aujourd'hui à sa version 5. C'est une plateforme mature qui a fait ses preuves et qui est largement utilisée dans le monde professionnel. À l'origine, ASP.NET WebForms a été créé par Microsoft afin que les développeurs d'applications Windows puissent facilement créer des applications web à partir de leurs connaissances de l'environnement de développement Windows. Ainsi, ASP.NET WebForms dispose de toute une mécanique qui permet d'abstraire le plus possible le modèle web (que nous allons décrire en détail plus loin) afin que les développeurs aient l'impression de se retrouver dans les conditions de développement d'une application Windows. Ceci implique une certaine logique de développement. Les applications ASP.NET WebForms se rapprochent d'un modèle événementiel, c'est-à-dire que chaque composant d'une page est capable de réagir à une action de l'utilisateur. Le modèle ASP.NET WebForms permet également de conserver l'état d'une page, chose que le protocole HTTP ne permet pas.

HTTP est l'acronyme de HyperText Transfer Protocol. C'est un protocole de communication client-serveur qui permet de consulter un site web depuis un navigateur Internet. Le principe est que le navigateur fait une demande au site web via ce protocole et le site web lui renvoie une réponse qui est le contenu de la page que le navigateur souhaite afficher. Ce protocole est sans état, c'est-à-dire que les requêtes sont indépendantes les unes des autres et qu'aucune information propre à une requête n'est conservée.

ASP.NET WebForms est un logique de développement très puissante pour réaliser des applications web. Elle ajoute une couche d'abstraction qui étonne chaque jour ses utilisateurs. Cette couche d'abstraction permet de masquer la complexité du travail à réaliser ainsi que toutes les spécificités de la programmation d'applications web que nous découvrirons plus loin. ASP.NET WebForms fournit également beaucoup de contrôles serveurs permettant de générer du HTML. Par exemple en quelques lignes de codes nous pouvons générer une grille complète affichant des données, gérant les pages multiples, les tris, etc. Mais tout ceci a un coût : cycle de vie de la page complexe, moins de maîtrise du HTML généré, beaucoup d'efforts pour contourner certaines limitations, etc.

ASP.NET MVC est plus récent et a fait son apparition en 2009. Cette façon de penser la réalisation des applications web s'appuie sur un patron de conception très célèbre, MVC (de l'Anglais « model-view-controller »), qui existe depuis 1979, l'année de ma naissance… coïncidence ?

Très connu sous son appellation anglaise « design pattern », un patron de conception constitue une solution éprouvée et reconnue comme une bonne pratique à un problème récurrent dans la conception d'applications informatiques. En général, il décrit une modélisation de classes utilisées pour résoudre un problème précis. Il existe beaucoup de patrons de conceptions. Je décrirai le patron de conception MVC un peu plus loin en détail mais sachez dès à présent que ce patron de conception permet de séparer les données et leurs présentations.
ASP.NET MVC est également une brique qui se rajoute au-dessus d'ASP.NET et qui exploite toute sa plomberie. Ici le principe n'est pas d'offrir une couche d'abstraction permettant de simplifier le travail, mais plutôt un cadre de réalisation des applications web, grâce à ce patron de conception réputé ; une espèce de ligne directrice qui va nous guider dans la réalisation d'une application web. Mais ASP.NET MVC donne également plus de liberté dans le rendu du HTML en permettant notamment d'utiliser des bibliothèques Javascript externes afin d'améliorer l'expérience utilisateur et de créer des applications web 2.0 puissantes, grâce à l'AJAX (que nous étudirons plus en détail en fin de cours).

Pourquoi utiliser ASP.NET MVC ?

Alors, que faut-il étudier ? ASP.NET WebForms ou ASP.NET MVC ?

C'est une question importante qu'il faut se poser ! Il n'y a pas une bonne et une mauvaise façon de concevoir des sites web avec ASP.NET, les deux solutions sont bonnes et dépendent de vos besoins. Le mieux étant bien sûr d'apprendre les deux, mais c'est une autre histoire.

Vous avez besoin de créer votre propre contrôle, pour le réutiliser à plusieurs endroits, ou pour le vendre ? Vous ne connaissez rien au web mais vous savez programmer des applications Windows ?
Alors ASP.NET WebForms est pour vous ! Son modèle événementiel et son abstraction du protocole HTTP vous raviront et vous permettront d'être rapidement efficaces. ASP.NET WebForms a l'avantage de posséder de nombreux contrôles avancés : une grille complexe, un treeview ou plein d'autres choses qui ne nécessitent que très peu de code pour fonctionner et qui sont réutilisables à souhait.

Si vous êtes un pro du développement web et que vous avez déjà fait plein de PHP (ou autre langage du même genre), je vous dirai de foncer sur ASP.NET MVC et de continuer à lire ce cours. Car ASP.NET MVC va se rapprocher beaucoup de ce que vous avez déjà pu faire. Vous aurez quelques particularités spécifiques à apprendre, mais la logique sera globalement la même. Vous aurez une totale maîtrise du modèle HTTP ainsi que du code HTML généré. La maîtrise du HTML généré est également un élément déterminant si vous comptez faire un site qui fonctionne sur des PC mais également sur des tablettes ou des smartphones. Le « responsive design », pratique très à la mode qui consiste à adapter un site web au dispositif de support (smartphone, tablette, ordinateur etc.), implique d'avoir un bon contrôle sur le HTML généré, ce qui est plus compliqué à faire avec ASP.NET WebForms. Par contre, avec ASP.NET MVC il vous faudra écrire vous même tout le HTML nécessaire pour cela.

ASP.NET WebForms bénéficie de toute la maturité d'une plateforme solide, massivement utilisée par les professionnels. ASP.NET MVC, plus récent, est moins mature même s'il en est déjà à sa version 5 et que Microsoft a beaucoup investi sur le sujet. La communauté est plus jeune mais également plus dynamique. Les ressources sont peut-être un peu moins nombreuses, mais ça ce n'est pas grave car vous avez en main la clé pour apprendre ASP.NET MVC.

En tous cas, c'est ce que j'ai choisi de vous apprendre ici : ASP.NET MVC. J'ai personnellement travaillé avec les deux et ils sont tous les deux intéressants. Maintenant, c'est à vous de choisir celui que vous voulez apprendre, mais vous avez des éléments pour décider, et sauf si vous avez opté pour l'apprentissage d'ASP.NET WebForms, je vous propose de continuer ensemble et à découvrir cette formidable plateforme.

One ASP.NET

Je vous ai parlé d'un socle de développement (ASP.NET), et de ses deux logiques différentes pour réaliser un site web (WebForms et MVC). Et bien sachez que nous pouvons mélanger ces deux logiques aussi pour réaliser l'application de nos rêves. Il est tout à fait possible de construire un site web en faisant à la fois du WebForms, à la fois du MVC et à la fois d'autres choses dont je n'ai pas encore parlé (comme Web API que j'aborderai tout à la fin du cours, ou SignalR dont je ne parlerai pas et qui n'a rien à voir avec du dentifrice...).

C'est ce qu'on appelle One ASP.NET. C'est la possibilité d'activer ces briques indépendamment ou ensemble dans nos différents projets. Il s'agit d'une fonctionnalité intéressante qui ne nous lie pas à un socle technique dès la construction du projet.

Nous allons ici nous concentrer sur l'apprentissage de MVC, donc je ne reviendrai pas sur One ASP.NET.

Prérequis

Il n'y a que deux prérequis pour suivre ce cours :

- Le premier, c'est de posséder un ordinateur.

Le plus simple, c'est qu'il soit équipé du système d'exploitation Windows car je vais présenter l'installation des outils gratuits dont nous allons nous servir avec ce système d'exploitation. Ce n'est par contre pas indispensable car il existe une solution

permettant de développer avec ASP.NET MVC sous linux par exemple, grâce à Mono. Son installation et son utilisation ne sont pas décrites dans ce cours mais vous devriez être capable facilement d'utiliser ce cours avec Mono.

- Le second c'est de parler le C#.

Le C# est le langage de développement phare de Microsoft et permet la création d'applications informatiques de toutes sortes. Il est indispensable de connaître un peu le langage de programmation C# afin de pouvoir développer des applications ASP.NET. Comme je vous disais en introduction, son étude n'est pas traitée dans ce cours mais vous pouvez retrouver un cours complet sur C# sur OpenClassrooms à cette adresse .

Pour résumer ce qu'est le C#, on peut dire qu'il s'agit d'un langage orienté objet apparu en même temps que le framework .NET qui n'a cessé d'évoluer depuis 2001. Il permet d'utiliser les briques du framework .NET pour réaliser des applications de toutes sortes et notamment des applications pour ASP.NET, mais aussi pour Windows, Windows Phone, des jeux, etc. C'est le ciment et les outils qui permettent d'assembler les briques de nos applications.

À noter qu'il est également possible de créer des applications ASP.NET avec le langage VB.NET qui est également un langage .NET, comme le C#. La syntaxe est légèrement différente et sera sans doute plus accessible à des personnes ayant déjà développé avec Visual Basic.
Sinon, tout le reste sera abordé dans ce cours. Ce qu'est un site web, ou comment fonctionne le modèle HTTP.

Le Pattern MVC

Maintenant que les bases du web sont révisées et que les outils de développement sont installés, il est grand temps de se mettre à la pratique d'ASP.NET MVC. Parce que du HTML c'est bien beau, mais ce n'est pas avec ça que nous allons créer notre future application web.

Nous allons donc nous plonger dans le vif du sujet en abordant les bases du pattern MVC sur lequel repose ASP.NET MVC. Puis nous appliquerons ces bases avec un petit hello world pour voir du concret à l'écran (c'est quand même ce que je vous ai promis dans ce cours !).

Ne vous inquiétez pas si tout n'est pas clair dès la première lecture, vous aurez tout le loisir d'y revenir ultérieurement.

Préchauffez les cerveaux à 180° et allons-y.

Qu'est-ce que MVC et pourquoi l'utiliser ?

MVC est un patron de conception (*design pattern* en anglais) très répandu pour réaliser des sites web. Ce patron de conception est une solution éprouvée et reconnue permettant de séparer l'affichage des informations, les actions de l'utilisateur et l'accès aux données.

MVC signifie Modèle-Vue-Contrôleur. C'est un modèle qui a été conçu au départ pour des applications dites « client lourd », c'est-à-dire dont la majorité des données sont traitées sur le poste client (par exemple : un traitement de texte comme Word). MVC était tellement puissant pour ces applications « client lourd », qu'il a été massivement adopté comme modèle pour la création d'applications web (dites « client léger »). C'est cette utilisation de MVC pour le web que je vais bien sûr décrire ici.

Vous aurez compris que MVC n'est pas associé à un langage de programmation et qu'il peut être utilisé avec énormément de langages. MVC est beaucoup utilisé avec PHP par exemple ou même JAVA. En général MVC agit comme une couche supplémentaire (qu'on appelle un « framework ») qui vient se greffer par-dessus un langage. C'est le cas pour ASP.NET MVC qui est une couche supplémentaire à ASP.NET.

MVC est donc une bonne solution pour réaliser des applications web et nous allons en détailler tous les avantages, mais commençons d'abord par son unique inconvénient, qui n'en est souvent pas un d'ailleurs : MVC apporte un découpage qui demande une certaine gymnastique mentale et qui multiplie le nombre de fichiers. Donc MVC s'avère souvent **disproportionné pour des tous petits sites** qui n'évoluent jamais. Mais comme il y a peu de sites de ce genre, cet inconvénient est assez minime.

Heureusement, malgré sa relative complexité pour les petits sites, le framework MVC d'ASP.NET apporte beaucoup de simplicité et masque un peu cette complexité pour les petits sites grâce à un mécanisme que je décrirai plus loin qui s'appelle « **convention plutôt que configuration** ».

MVC permet également de concevoir des applications de manière claire et efficace grâce à la séparation des intentions. Les opérations de **maintenance et de mises à jour sont fortement simplifiées**, et on évite plus facilement le syndrome du plat de spaghetti où la moindre modification à un endroit risque de provoquer des débordements à d'autres endroits tellement tout est emmêlé. Par exemple, il est très fréquent que l'affichage d'un site web change beaucoup plus régulièrement que les données que nous souhaitons afficher. Avec une bonne séparation, nous n'aurons qu'à modifier une toute petite partie de code sans toucher au reste qui fonctionne très bien. De même, si une règle métier change - par exemple imaginons qu'un site d'e-commerce souhaite changer ses modalités de livraison - nul besoin de toucher l'interface, les risques d'impacts sont limités.

MVC permet également de répartir plus facilement les tâches entre développeurs. Rares sont les personnes qui sont capables de transformer des règles métiers complexes en code efficace tout en étant expertes en design web, que ce soit pour PC, tablette, ou smartphone... Le découpage permet à un développeur de s'occuper des règles métiers complexes et à un designer de faire du beau HTML ergonomique. Alors, ce découpage, c'est bien beau mais comment ça marche ?

M comme Modèle

Le M de MVC signifie **Modèle**. Il s'agit des données et de l'état de votre application web (par exemple votre panier de courses sur un site d'e-commerce, ou la liste des produits à acheter,...). En général, ces données sont représentées par un ensemble de classes qui permettent d'accéder à une base de données, mais vos données pourraient

tout aussi bien être stockées dans des fichiers plats ou XML, ou même vos classes pourraient utiliser des services web ou autre…

Bref, dans le modèle vous aurez de quoi accéder à vos données. Reprenons par exemple un site d'e-commerce, vous aurez sans doute une classe Client, une classe Commande, une classe Produit, etc. Il y aura des méthodes pour obtenir le détail d'un client précis, retrouver une commande, récupérer les produits par ordre de prix croissant, etc. Toutes ces informations seront stockées dans une base de données, et vous pourrez utiliser pourquoi pas un ORM comme Entity Framework pour faciliter cet accès aux données.

ORM veut dire en anglais object-relational mapping, que l'on peut traduire par « mapping objet-relationnel ». Il s'agit d'une technique de programmation informatique qui permet d'établir des correspondances entre une base de données relationnelle et un modèle objet de classes.
Dans ce cours, nous utiliserons dans un premier temps un modèle de classes très simple avec des données en dur. Puis, plus loin, nous utiliserons l'ORM Entity Framework afin de pousser le modèle dans ses retranchements...

À noter que le modèle ne connaît ni la vue, ni le contrôleur. Sa seule finalité est d'être consulté ou modifié par ces derniers.

V comme Vue

Le V de MVC signifie la **Vue** et traite de ce qu'on voit à l'écran dans le navigateur web. Il s'agira globalement de code HTML et de CSS. Le but de la vue est de présenter les données issues du modèle mais sans les modifier, sans en interpréter le contenu.

Dans un site web, il y a en général plusieurs vues et une vue correspond bien souvent à une unique page. Dans notre site d'e-commerce par exemple, une vue permettra d'afficher les caractéristiques d'un produit, une autre affichera la page d'accueil, une autre encore permettra de visualiser sa commande, etc.

Nous pourrons également avoir plusieurs vues pour représenter les mêmes informations. Il est par exemple tout à fait envisageable de présenter un rapport sous la forme de chiffres dans un tableau, mais également sous la forme de graphiques en barres ou autre…

Vous verrez que la vue, c'est le plus simple à maîtriser, surtout si vous connaissez déjà un peu le HTML. En plus, ASP.NET MVC nous offre toute l'aide qu'il nous faut pour être pleinement efficaces dans la réalisation de nos vues.

C comme contrôleur

Le C de MVC signifie **Contrôleur**. Il fait le lien entre la vue et le modèle.

Le contrôleur gère les interactions avec l'utilisateur et détermine quels traitements doivent être effectués pour une action donnée. D'une manière générale, il va utiliser les données du modèle, les traiter en fonction de l'action de l'utilisateur, et les envoyer à la vue afin qu'elle les affiche.

Nous verrons que le contrôleur est au cœur du système, il interprète et « contrôle » les données venant de l'utilisateur, comme des données venant d'un formulaire ou bien simplement une action faite via une URL.

Une bonne pratique est de faire en sorte que le contrôleur fasse le moins de choses possible. Il doit traiter l'action et choisir la bonne vue. Il est important de comprendre vraiment le rôle de chaque élément car le plus souvent, si vous ne savez pas où mettre quelque chose, vous risquez de le mettre dans le contrôleur…

Schéma récapitulatif

Voici un petit schéma synthétique pour vous rappeler ce qu'est MVC et comment les différentes couches interagissent entre elles :

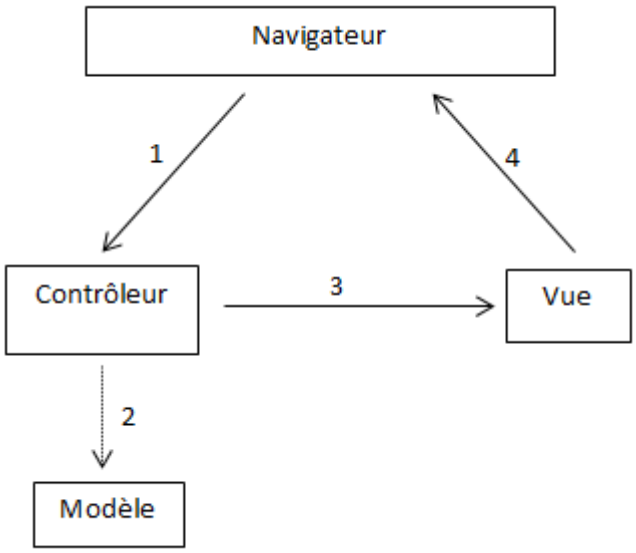

Schéma MVC

1. Action utilisateur via une requête HTTP
2. Consultation et/ou mise à jour du modèle (facultatif)
3. Le contrôleur décide de la vue à afficher
4. La vue renvoi le HTML au navigateur

Nous allons voir tout au long de ce cours comment industrialiser son code grâce à MVC. Ce n'est pas grave si vous n'avez pas complétement saisi l'intérêt de cette architecture, vous verrez au fur et à mesure comment l'appréhender correctement et en tirer des bénéfices.

À noter que je n'en ai pas encore parlé, mais le patron de conception MVC apporte un élément supplémentaire non négligeable : il facilite les tests de son application. Lorsque tout est emmêlé, il est très difficile de tester son application. Il vous faut alors la démarrer, cliquer un peu partout, voir si ça marche ou si ça plante, et essayer de deviner d'où ça vient. Avec MVC, vous pourrez plus facilement tester votre application grâce à la séparation des intentions. Il sera très facile de tester les contrôleurs et le modèle et même d'écrire des tests automatiques permettant de vérifier qu'il n'y a pas de régressions. La vue sera toujours complexe à tester car elle nécessitera une intervention manuelle, mais l'avantage est que vous n'aurez que des contraintes « visuelles » à tester. Toute la logique métier aura préalablement été testée via les tests du modèle et des contrôleurs.

En résumé

- MVC est un patron de conception particulièrement adapté pour réaliser une application web.
- MVC est l'acronyme de Modèle-Vue-Contrôleur : le Modèle contient les données de l'application, la Vue (ou les Vues) contient le code pour afficher les pages de l'application, et le Contrôleur gère les interactions de l'utilisateur en faisant le lien avec le Modèle et la Vue.
- MVC permet une séparation claire des intentions et optimise la création d'une application web ou d'un site, sa maintenance et ses tests automatisés.