



# Conception & Développement Informatique

## Introduction à ASP.NET

*C#, Csharp, .NET Framework, Visual Studio*

**Mickaël DEVOLDÈRE**

MD v1.0.1


**23/04/2018**







<http://www.arfp.asso.fr>

# Introduction à ASP.NET

C#, Csharp, .NET Framework, Visual Studio

Technologies et langages	
	HTML
	CSS
	Javascript
	ASP.Net
	Microsoft SQLServer

Légende des icônes	
	Information complémentaire
	Point d'attention particulier
	Intervention du formateur possible
	Lien vers une ressource externe

## Sommaire

---

ASP.NET Kesako ? .....	3
ASP.NET WebForms .....	3
ASP.NET MVC .....	3
Différences entre WebForms et MVC .....	4
Choisir entre WebForms et MVC pour un projet .....	5
« Application Web » et/ou « Site Web » .....	6
Site web statique .....	6
Site web dynamique .....	6
Les requêtes HTTP .....	7
Paramètres de requête HTTP .....	8
GET .....	9
POST .....	10
Synthèse des différences entre GET et POST .....	10
Historique du document .....	12
Crédits .....	12

## ASP.NET Kesako ?

---

ASP.NET est la plateforme de développement de Microsoft permettant la réalisation d'applications web.

Une plateforme de développement est un ensemble de composants permettant de construire une application (ici, une application web) qui comprend :

- un outil de développement (exemple : Visual Studio)
- une boîte à fonctionnalités pour développer (exemple : Le framework .NET)
- une logique de développement (exemple : respect de la structure d'une application web)
- un langage de programmation (exemple : C#)

ASP propose 2 manières de concevoir une application Web :

- ASP WebForms
- ASP MVC

ASP.NET est le socle de la plateforme de développement de Microsoft pour réaliser des applications web. WebForms et MVC sont deux logiques différentes de développement ASP.NET.

## ASP.NET WebForms

---

La 1<sup>ère</sup> version d'ASP.NET WebForms est sortie en 2002 et en est aujourd'hui (2018) à sa 5<sup>ème</sup> version. C'est une plateforme mature et robuste 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.

Concrètement, tout développeur d'applications Windows C# est capable de produire une application ASP.NET.

ASP.NET WebForms est une logique de développement très puissante pour réaliser des applications web. Elle ajoute une couche d'abstraction permettant de masquer une grande partie de la complexité du travail à réaliser ainsi que toutes les spécificités de la programmation d'applications web.

## ASP.NET MVC

---

ASP.NET MVC est plus récent et a fait son apparition en 2009. La logique de réalisation des applications web s'appuie sur un patron de conception très célèbre, MVC : Modèle Vue Contrôleur.

Là où les WebForms offrent des composants « préfabriqués » afin de simplifier votre vie de développeur, ASP.NET MVC propose un cadre pour la conception d'applications web.

L'utilisation du design pattern « MVC » impose de suivre une ligne directrice qui guidera le développeur dans la réalisation d'une application web.

ASP.NET MVC donne également plus de liberté dans le rendu du HTML en permettant notamment d'utiliser des composants externes (Javascript, CSS, appels AJAX).

## Différences entre WebForms et MVC

---

ASP.NET WebForms et ASP.NET MVC diffèrent sur de nombreux points. Les différences les plus notables sont :

### *Cycle de vie*

---

#### **Webforms**

Le cycle de vie de la requête est fortement basé sur la « page active ».  
Le fonctionnement est géré par des événements sur lesquels vous pouvez vous abonner.

#### **MVC**

Le cycle de vie est différent car il fait intervenir un contrôleur qui va charger des modèles de données puis sélectionner la vue à utiliser pour générer le rendu.

### *Dépendances*

---

#### **Webforms**

Les pages sont composées d'une vue et de code-behind. Ceci a un impact important car la vue et le contrôleur sont fortement dépendants.

#### **MVC**

Le contrôleur sélectionne la vue à afficher. Celle-ci ne dépend que du modèle qu'elle va utiliser pour le rendu.

### *Testabilité*

---

#### **Webforms**

Les pages sont difficiles à instancier en dehors d'un fonctionnement normal. Ceci a pour effet de les rendre impossible à tester de manière automatique.

#### **MVC**

Les contrôleurs sont de « simples » classes que vous pouvez instancier sans contexte HTTP. Elles sont donc plus faciles à tester avec des tests unitaires par exemple.

### *Gestion de l'état*

---

#### **Webforms**

Les contrôles stockent leur état dans le ViewState. Le ViewState est transmis dans chaque page. Il a tendance à grossir très rapidement.

#### **MVC**

MVC ne maintient pas d'état des pages, ni de ViewState. Vous avez le contrôle complet de votre application.

### *Les contrôles visuels*

---

#### **Webforms**

Webforms est fortement basé sur des composants visuels comme Winforms. Ceci implique que le code est généré de manière automatique (sans grand contrôle).

#### **MVC**

MVC n'utilise pas d'évènement ni de code généré. Vous pouvez donc écrire du code HTML propre comme vous le souhaitez.

## Evènements

---

### Webforms

L'extensibilité est gérée par les évènements. Vous devez donc vous abonner pour modifier certains comportements.

### MVC

L'extensibilité est gérée par héritage ou des filtres. Le fonctionnement est simple à comprendre et à déboguer.

## Connaissances HTML, JS et CSS

---

### Webforms

Etant donné la présence de contrôles visuels, vous n'avez pas besoin d'avoir de connaissances approfondies en HTML, JS et CSS pour créer des applications. Les composants vont générer le code pour vous.

### MVC

L'absence de contrôles visuels et de drag & drop nécessite des compétences en développement HTML, CSS et Javascript. En contre-partie, vous pourrez écrire du code jQuery et des requêtes Ajax très simplement.

## Apprentissage

---

### Webforms

Webforms est plus facile à apprendre et permet de créer rapidement des écrans visuels grâce au drag & drop de composants. Le framework est intéressant pour débiter avec le Web quand on connaît déjà WinForms.

### MVC

Le temps d'apprentissage est un peu plus long car il faut bien comprendre les concepts du framework. Une fois que vous aurez bien compris ces concepts, vous pourrez développer très rapidement des applications web modernes.

## Choisir entre WebForms et MVC pour un projet

---

Si vous êtes à l'aise dans le développement d'applications Windows Forms et que le HTML/CSS est un peu obscur pour vous, ASP.NET WebForms est le meilleur choix. Il offre un panel de composants préfabriqués qui vous permettront de développer des pages web aussi rapidement qu'un formulaire Windows.

Si vous êtes à l'aise avec HTML/CSS et que vous souhaitez utiliser des composants externes (jQuery, Bootstrap...) ou personnalisés, ASP.NET MVC est le choix à faire. Avec ASP.NET MVC, les composants d'interface sont à développer entièrement.

A noter qu'il est possible, pour les gros projets, de coupler ASP.NET WebForms et ASP.NET MVC. Ce concept s'appelle « One ASP.NET »

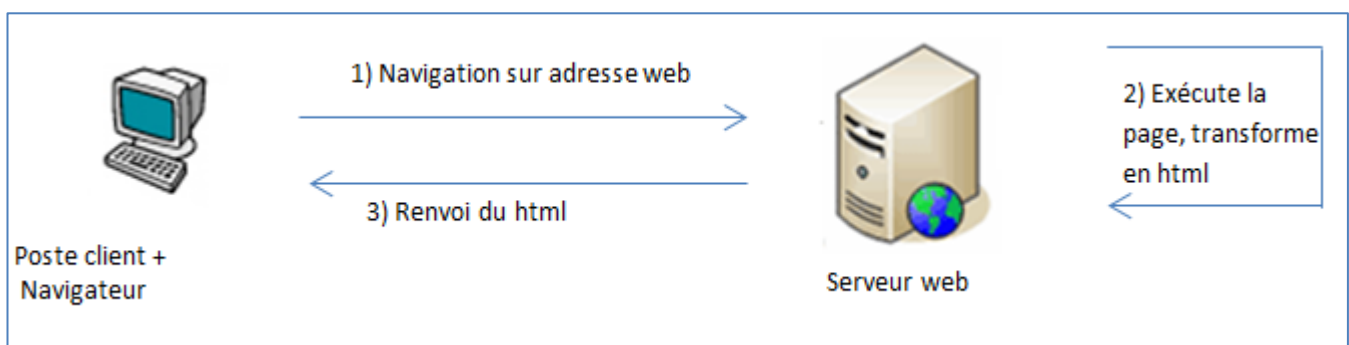
## « Application Web » et/ou « Site Web »

Un site web est un ensemble de fichiers, hébergés sur un serveur et accessibles via un navigateur (Chrome, Firefox, Edge, et pour les plus téméraires... Internet Explorer).

Ces fichiers peuvent être de simples fichiers HTML statiques ou des fichiers exécutables interprétés par un moteur de rendu tel que PHP (Hypertext Preprocessor), JSP (Java Server Pages) ou ASP.NET (Active Server Pages). Un fichier accessible via un serveur web est ce qu'on appelle une « ressource web ».

Un serveur Web est un ordinateur (en général assez puissant) qui héberge des fichiers qui sont accessibles via une adresse internet (exemple : [www.arfp.asso.fr](http://www.arfp.asso.fr)).

Le schéma suivant démontre l'échange entre un navigateur web et un serveur web.



### Site web statique

Un site web statique est un contenu codé avec le langage HTML et mis à disposition « tel quel » sur le serveur web.

Un site web statique ne permet pas de réaliser des opérations aujourd'hui communes : connexion à une base de données, identification d'un utilisateur, rédaction de commentaires etc...

Pour réaliser ce type d'opération, il est nécessaire de développer un « site dynamique ».

### Site web dynamique

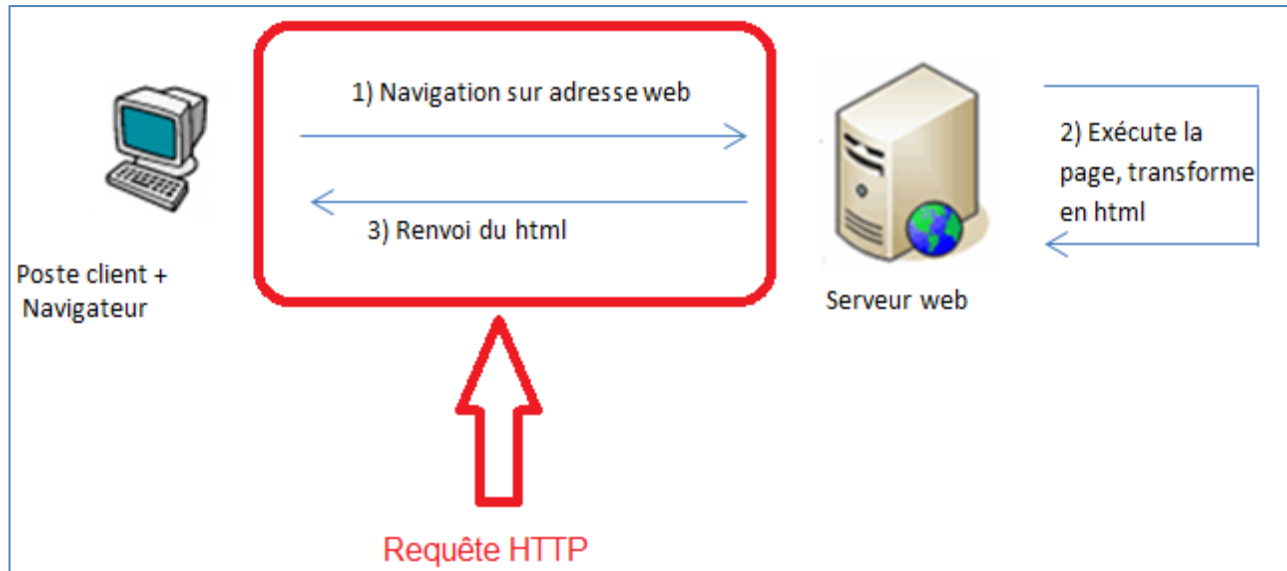
Mis à part quelques antiquités du web ou quelques sites de présentation, les sites web « statiques » se rarifient. Les sites dynamiques sont capables d'exécuter des programmes qui vont être en mesure d'afficher des informations différentes en fonction du besoin. C'est le rôle premier du serveur web, il exécute la page dynamique (qui sera codée en PHP, Java ou C#) à partir de laquelle sera généré du code HTML afin qu'elle soit affichable par le navigateur web.



**Le terme « application web » est l'appellation moderne pour un site web dynamique.**

## Les requêtes HTTP

L'accès à un site web par un navigateur se fait au travers de requêtes HTTP.



Un serveur web a la possibilité de gérer plusieurs types de requêtes HTTP dont les plus importantes sont: GET, POST, PATCH, PUT, DELETE. Ces termes sont des « verbes HTTP ».

Nous nous intéresserons particulièrement à GET et POST. Les verbes HTTP « PATCH, PUT, DELETE » servent généralement aux requêtes effectuées auprès de services Web et ne seront pas traités dans ce document.



**Un service web (ou Web Service) est un protocole d'échange de données entre applications et/ou environnements distribués.**



## Paramètres de requête HTTP

Le format des paramètres GET et POST est sensiblement identique, les données sont transmises sous forme de chaîne de caractères ou chaque paire « clé=valeur » est séparée par le caractère « & ».

Cette chaîne de caractères est communément appelée « Query String ».

Exemples :

QueryString	Nombre Paramètres	Liste Paramètres
« page=articles »	1	Paramètre « page » ayant pour valeur « articles »
« page=articles&id=1 »	2	Paramètre « page » ayant pour valeur « articles » Paramètre « id » ayant pour valeur « 1 »
« page=users&id=1&action=edit »	3	Paramètre « page » ayant pour valeur « users » Paramètre « id » ayant pour valeur « 1 » Paramètre « action » ayant pour valeur « edit »

La différence entre GET et POST réside dans la manière dont les données seront transmises.

## GET

---

Une requête de type « GET » correspond à une requête en lecture seule d'une ressource web. C'est la requête HTTP par défaut et donc celle qui est le plus souvent utilisée.



**Une requête « GET » permet de consulter un contenu à l'infini et ne devrait pas permettre de le modifier.**

Les données d'une requête de type GET sont transmises directement dans l'url, ce qui les expose à la vue de l'utilisateur.

Les paramètres s'ajoutent à la fin de l'url (et sont donc visibles par l'utilisateur) en utilisant un séparateur entre l'url et ses paramètres. Ce séparateur est le point d'interrogation « ? ». Chaque paramètre d'url est une paire « clé=valeur ». Il est possible de définir plusieurs paramètres d'url en les séparant par le caractère « & ».

Exemples :

- <http://localhost/index.php> est un url non paramétré.
- <http://localhost/index.php?id=1> est un url paramétré contenant un paramètre nommé « id » et ayant pour valeur « 1 ».
- <http://localhost/index.php?id=1&page=articles> est un url comprenant 2 paramètres :
  - « id » ayant pour valeur « 1 »
  - « page » ayant pour valeur « articles »

Il est possible de paramétrer « manuellement » une requête de type GET en construisant l'url et en respectant le format du queryString.

## POST

Une requête de type « POST » est généralement utilisée pour soumettre des données au serveur web notamment via des formulaires HTML.



**Une requête « POST » permet d'insérer ou de modifier un contenu et ne devrait pas pouvoir être exécutée à l'infini.**

Les données d'une requête de type POST n'apparaissent pas dans l'url et sont envoyées après les entêtes HTTP.

## Synthèse des différences entre GET et POST

	GET	POST
Longueur maximale	2048 octets (paramétrable au niveau du serveur)	∞
Mise en cache dans le navigateur	✓	✗
Rétention dans l'historique du navigateur	✓	✗
Peut être mis en favori	✓	✗
Lire une ressource	✓	✗
Mettre à jour une ressource	✗	✓
Transmettre des données sensibles	✗	✓

Les différents types de projets ASP.NET sous Visual Studio

Ouvrir Visual Studio (version 2017 ou ultérieure)

--- Fin du document ---

## Historique du document

---

Auteur	Date	Observations
Mickaël DEVOLDÈRE	12/04/2018	Création du document
Mickaël DEVOLDÈRE	24/04/2018	Ajout section « Requêtes http »

## Crédits

---

Ce document est largement inspiré :

- du MOOC d'openclassrooms « Apprendre ASP.NET MVC »
  - o <https://openclassrooms.com/courses/apprendre-asp-net-mvc>
- du MOOC d'openclassrooms « Créez votre site Web avec ASP.NET »
  - o <https://openclassrooms.com/courses/creez-votre-site-web-en-asp-net>

Consultez les liens ci-dessus pour voir la version originale (généralement plus détaillée) de ce document.