COURS DE DEVELOPPEMENT WEB

Objectif du cours

Il est vrai que l'évolution de nouvelles technologies de l'information et de la communication, influence le système d'information (entreprise) dans l'optique d'aider les gestionnaires à optimiser leurs services et leurs visibilités sur internet.

C'est ainsi que notre cours fixe comme objectif :

- ✓ De permettre aux étudiants de bénéficier des connaissances sur les concepts de base liés web ;
- ✓ De faciliter les futurs ingénieurs à la maitrise du développement web en utilisant ASP.NET et d'autres outils de base.
- ✓ D'aider les étudiants à manipuler les langages de construction des pages ainsi que la création d'un projet à défendre.

Chapitre 1 : GENERALITE SUR LE WEB

Dans ce chapitre nous allons nous consacrer essentiellement aux théoriques. En expliquant brièvement les concepts de base ; et puis donner les significations des principales sémantiques et quelques notions auxquelles nous ferons allusion dans le cadre de notre cours.

Le web est considéré comme un service de l'internet, or l'internet est un réseau international. Il est aussi un ensemble des données reliées par des liens hypertextes, sur Internet. On parle aussi de Surfer sur le web ou sur internet.

- **A quoi correspondent les différentes phases de cette évolution ?**

Le web est sans nul doute une technologie majeure du 21ème siècle. Et si sa nature, sa structure et son utilisation ont évolué au cours du temps, force est de constater que cette évolution a également profondément modifié nos pratiques commerciales et sociales.

Pour mieux comprendre les enjeux et les différentes phases de cette évolution, nous allons nous livrer à un exercice de synthèse, qui ne se veut en aucun cas exhaustif, mais qui devrait vous fournir quelques clés de compréhension.

- Le **web 1.0**, encore appelé **web traditionnel**, est avant tout un web statique, centré sur la distribution d'informations. Il se caractérise par des sites orientés produits, peu d'intervention des utilisateurs.
- Le **web 2.0,** ou **web social**, change totalement de perspective. Il privilégie la dimension de partage et d'échange d'informations et de contenus (textes, vidéos, images ou autres). Il voit l'émergence des réseaux sociaux, des smartphones et des blogs. Le web se démocratise et se dynamise. L'avis du consommateur est sollicité en permanence et il prend goût à cette socialisation virtuelle. Toutefois, la prolifération de contenus de qualité inégale engendre une infobésité difficile à contrôler.
- Le **web 3.0**, aussi nommé **web sémantique**, vise à organiser la masse d'informations disponibles en fonction du contexte et des besoins de chaque utilisateur, en tenant compte de sa localisation, de ses préférences, etc.
- Le **web 4.0**, évoqué par certains comme le **web intelligent**, effraie autant qu'il fascine, puisqu'il vise à immerger l'individu dans un environnement (web) de plus en plus prégnant. C'est un terrain d'expérimentation où tous ne sont pas (encore) prêts à s'aventurer !

SECTION 1 : CONCEPTS DE BASE

A. Conception de site web

La *conception* de site *web* ou *web* design est la *conception* de l'interface *web* : l'architecture interactionnelle, l'organisation des pages.

Quelles sont étapes de la conception d'un site Web ?

1. Rencontre initiale et analyse des besoins. À cette étape, on travaille en étroite collaboration avec le client. ...
2. Planification du projet **Web** ;
3. La mise en œuvre ;
4. La **conception** et la réalisation ;
5. Contrôles de qualité ;
6. Mise en ligne ;
7. Promotion et suivi du **site Web**.

Conseils concrets pour créer un site web réussi

1. Définissez l'objectif de votre site.

Soyez clair et précis sur ce que vous espérez atteindre avec votre site avant de commencer. Que sont supposés faire vos visiteurs sur votre site ? Cela peut être par exemple acheter sur votre e-boutique, s'inscrire à votre newsletter, trouver votre adresse/contact, etc… Ainsi, si vous êtes par exemple infirmier(e) ou professeur(e) de yoga, l'objectif principal sera la prise de contact via un formulaire.

2. Déterminez votre cible.

(Et ne trichez pas en disant que c'est "tout le monde" ;). Si vous savez précisément qui vous essayez d'atteindre, vous pouvez créer un site qui cible les besoins de vos visiteurs. Définir pour qui est fait votre site vous permettra de créer plus facilement un site efficace et réussi.

3. Faites tenir votre navigation sur une ligne.

Au-delà de sept éléments dans votre navigation, vous risquez de perdre l'attention de vos visiteurs. Le plus important est de faire tenir votre navigation sur une ligne.

Une navigation réduite rend l'apparence du site beaucoup plus claire et mieux organisée, et vos visiteurs trouveront plus rapidement l'information qu'ils recherchent.

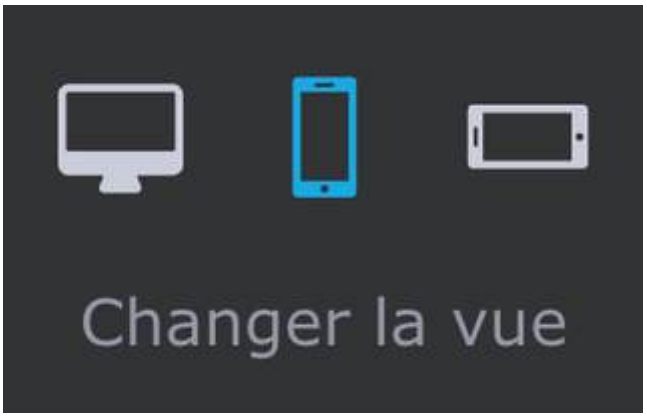
4. Remplissez vos paramètres SEO : (Search Engine Optimization) est l'acronyme qui signifie « Optimisation pour les moteurs de recherche » en français.

Beaucoup de gens ignorent cette étape en pensant que le SEO est bien trop compliqué pour eux. Or vous n'avez pas besoin d'être un expert en référencement pour remplir les paramètres SEO de votre site – et cela fera vraiment la différence. Si vous avez bien rempli vos Titre de site, Titre de page et les Descriptions de pages pour chacune de vos pages, alors vous pouvez vous féliciter ! Si ce n'est pas encore le cas, nous vous expliquons ici comment faire.

5. Investissez dans un nom de domaine.

En choisissant de passer à une version payante, vous obtiendrez un nom de domaine personnalisé et vous n'aurez aucune publicité Jimdo sur votre site. C'est un gage de sérieux et de professionnalisme.

6. Vérifiez l'affichage de votre site sur mobile.



Sur Jimdo, tous les designs sont désormais responsives. Ainsi votre site web s'adaptera automatiquement à la taille de l'écran de votre visiteur. Mais il est toujours bon de vérifier en utilisant le mode Aperçu – et de s'assurer que votre police de caractère est assez grande et donc bien lisible.

7. Utilisez des photos de qualité.

Pour marquer/faire bonne impression, rien n'est plus efficace que des images bien conçues et en haute résolution. Si vous venez de fabriquer un magnifique bijou par exemple, alors mettez-le vraiment en scène – grâce à une photo de qualité et à un arrière-plan adapté. Un conseil : fabriquer une boîte à lumière simplement pour réaliser de belles photos de vos produits !

8. Pensez à investir dans un logo.

Les avantages d'un logo personnalisé sont souvent sous-estimés. Pourtant c'est un élément essentiel de l'image de marque, que vous pourrez utiliser notamment pour vos réseaux sociaux. Cela vaut donc le coup d'investir un peu dans un logo pour votre projet. Un conseil : les logos sur fond transparent (voir ci-dessous) rendront particulièrement bien dans la partie Header de votre site.

Et si vous avez un budget limité, nous vous recommandons le site 99designs qui propose des offres abordables.

9. N'oubliez pas les icônes.

Saviez-vous qu'il existe des quantités d'icônes originaux disponibles gratuitement en ligne ? Cela paraît trop beau pour être vrai, mais de nombreux sites proposent des icônes à télécharger. Il vous suffit ensuite de les insérer sur votre site grâce à l'élément Image.

10. Ne cachez pas vos informations de contact.

Votre site web peut être le moyen le plus simple de commencer une relation avec vos nouveaux clients. Placez vos coordonnées dans la sidebar de votre site ou dans le bas de page, afin que celles-ci apparaissent sur chacune de vos pages – vos visiteurs pourront ainsi vous contacter sans avoir à chercher longtemps sur votre site où se trouve votre adresse e-mail !

B. Internet

// est un réseau informatique mondial qui permet à ses utilisateurs, appelés internautes de communiquer entre eux à l'échelle de la planète.

Il est aussi l'ensemble de réseaux mondiaux interconnectés qui permet à des ordinateurs et à des serveurs de communiquer efficacement au moyen d'un protocole de communication commun (IP ou probole internet). Ses principaux services sont le Web, le FTP, la messagerie et les groupes de discussion.

- **WWW** : Le **Web** est le terme communément employé pour parler du World Wide **Web**, ou WWW, traduit en **français** par la toile d'araignée mondiale. Il fait référence au système hypertexte fonctionnant sur le réseau informatique mondial Internet. Quelle est la signification du Web ?

Étymologie. (Années 1990) De l'anglais Web , lui-même ellipse de World Wide Web , composé de worldwide (« mondial ») et de web (« **toile d'araignée** »). Cette toile symbolise les hyperliens entre les ressources du Web

- PORT : c'est un **point virtuel où les connexions réseau commencent et se terminent**. Les ports sont basés sur des logiciels et gérés par le système d'exploitation d'un ordinateur.

Quels sont les types de ports en informatique ?

Ports d'un ordinateur

Un port HDMI qui permet par exemple de relier l'ordinateur à une TV moderne. Un port Ethernet pour brancher un câble donnant accès à un réseau et à Internet. Un port VGA pour connecter l'ordinateur à un projecteur. Des ports USB 2.0 pour brancher une souris, un disque dur externe, une clé USB…

1. Les ports matériels sont des **ports informatiques sur lesquels vous pouvez voir et connecter un appareil**. (non traité dans ce cours).

2. Les ports logiciels sont basés sur des logiciels et sont principalement utilisés lorsque deux programmes souhaitent échanger des données entre eux.

Quels sont les différents numéros de port ?

Il existe 65 535 numéros de port possibles, mais tous ne sont pas utilisés couramment. Voici quelques-uns des ports les plus couramment utilisés, ainsi que le protocole réseau qui leur est associé :

- **Ports 20 et 21 :** Protocole de transfert de fichiers (FTP). Le protocole FTP permet de transférer des fichiers entre un client et un serveur.
- **Port 22 :** Secure Shell (SSH). SSH est l'un des nombreux protocoles de tunneling qui créent des connexions réseau sécurisées.
- **Port 25 :** Simple Mail Transfer Protocol (SMTP). Le SMTP est utilisé pour le courrier électronique.
- **Port 53 :** domain Name System (DNS). Le DNS est un processus essentiel pour l'Internet moderne ; il fait correspondre les noms de domaines lisibles par l'homme aux adresses IP lisibles par la machine, ce qui permet aux utilisateurs de charger des sites Web et des applications sans avoir à mémoriser une longue liste d'adresses IP.
- **Port 80 :** Protocole de transfert hypertexte (HTTP). HTTP est le protocole qui rend le World Wide Web possible.
- **Port 123 :** Network Time Protocol (NTP). NTP permet aux horloges des ordinateurs de se synchroniser entre elles, un processus essentiel pour le chiffrement.
- **Port 179 :** Border Gateway Protocol (BGP). BGP est essentiel pour établir des routes efficaces entre les grands réseaux qui composent Internet (ces grands réseaux sont appelés systèmes autonomes). Les systèmes autonomes utilisent BGP pour diffuser les adresses IP qu'ils contrôlent.
- **Port 443 :** HTTP Secure (HTTPS). HTTPS est la version sécurisée et cryptée de HTTP. Tout le trafic Web HTTPS passe par le port 443. Les services réseau qui utilisent HTTPS pour le chiffrement, tels que DNS over HTTPS, se connectent également à ce port.
- **Port 500 :** Internet Security Association and Key Management Protocol (ISAKMP), qui fait partie du processus de mise en place de connexions sécurisées IPsec .

- **Port 3389 :** Protocole de bureau à distance (RDP). Le protocole RDP permet aux utilisateurs de se connecter à distance à leur ordinateur de bureau depuis un autre périphérique.

L'Internet Assigned Numbers Authority (IANA) tient à jour la liste complète des numéros de port et des protocoles qui leur sont attribués.

Qu'est-ce qu'un numéro de port ?

Les ports sont normalisés sur tous les périphériques connectés au réseau, et un numéro est attribué à chaque port. La plupart des ports sont réservés à certains protocoles, par exemple, tous les messages Hypertext Transfer Protocol (HTTP) vont au port 80. Alors que les adresses IP permettent aux messages d'aller vers et depuis des périphériques spécifiques, les numéros de port permettent de cibler des services ou des applications spécifiques au sein de ces périphériques.

- **URL** : **Une URL** (pour uniform resource locator, signifiant littéralement localisateur de ressources uniforme) est une chaîne de caractères décrivant l'emplacement d'une ressource.

C. Site Web

Outre le gain de temps et d'argent, un site web permet de communiquer, à tout moment, et en tout lieu. Le site étant accessible 24h/24, de n'importe où dans le monde, la communication d'une entreprise prend une forme continue, ininterrompue.
Un **site web** est l'ensemble des pages **web** et des ressources inter reliées entre elles par des liens hypertextes, auxquelles l'internaute peut accéder par une adresse **web** appelée Url, le tout enregistré sous le même nom de domaine.

Un site internet, également appelé site web, est un ensemble de pages de textes, d'images et/ou de vidéos, reliées entre elles par des liens hypertextes également appelés liens internes. Ces éléments permettent de générer un affichage public pour les internautes désirant obtenir des réponses à leurs questions.

Page web : c'est Un document qui peut être affiché par un navigateur web (tel que Mozilla Firefox, Google Chrome, Microsoft Internet Explorer ou Edge ou encore Safari). On parle aussi simplement de « pages ».

Il faut savoir qu'il existe 2 types de site web :

- **Statique**

Un *site web statique* est un site web composé de pages web construites en utilisant uniquement HTML, CSS et d'autres technologies web sans la base de données.

Une page web statique est une page web dont le contenu ne varie pas en fonction des caractéristiques de la demande, c'est-à-dire qu'à un moment donné tous les internautes qui demandent la page reçoivent le même contenu

- Dynamique

Quand je parle de **site dynamique**, je fais référence à un site web construit avec des fonctionnalités dynamiques qui permettent de séparer le contenu du contenant. **Cette méthode de conception permet à l'éditeur d'ajouter / modifier facilement ses contenus lui-même, sans toucher à la mise en page du site.**

Ce type de site web est particulièrement pertinent pour les entreprises ou organismes où plusieurs personnes vont être appelées à éditer du contenu sur le site. Comparativement à un site plus conventionnel, **il s'agit de sites personnalisés, conçus sur mesure selon les besoins du client**.

Par exemple, vous pourriez avoir des membres, et votre site pourrait afficher du contenu spécifique selon le membre qui visite (affichage de son nom, des contenus auxquels il a accès etc.), ou votre site pourrait contenir des types de contenus personnalisés (documents, membres du CA, témoignages, événements, etc.) qui s'affichent à des endroits sur le site, définis selon des critères (catégorie, étiquette, type de visiteur, page etc.)… Avec cette manière de construire un site, le ciel est *presque* la limite.

Des sites dynamiques ET faciles à manipuler

J'entends déjà certains me dire : « Mais Capucine, ça va coûter trop cher » ou « Je ne veux pas de site codé à la main et être dépendant d'un programmeur pour le modifier ». Ce sont effectivement des considérations légitimes, puisqu'il fut un temps (très peu lointain) où ces affirmations étaient véridiques. Par contre, l'univers de la création web évolue très vite, et de nouveaux outils de développement nous permettent dorénavant de créer des sites dynamiques plus facilement, et plus rapidement. Sans oublier qu'il est relativement facile, pour un administrateur moindrement habile avec les technologies du web, d'apprendre à modifier les modèles de page et toutes les fonctionnalités du site.

- Catégorie de sites

8 catégories de sites Internet

1 - **Site vitrine**. Présente l'entreprise ou la marque ainsi que son activité. ...

2 - **Site catalogue**. Présente l'entreprise ou la marque ainsi que son activité et tous ses produits détaillés. ...

3- **Site informatif** : Regroupe des données sur un sujet commun.

exemple : Amorino

4 - Site marchand ou E-commerce : Boutique en ligne dynamique avec une gestion des contenu et l'intégration d'un paiement sécurisé.

exemples : Cafés Folliet

5 - Site institutionnel : Décrit une organisation, ses activités et ses valeurs. Le site institutionnel donne toutes les informations pratiques nécessaires à ses clients ou à ses bénéficiaires. Il permet également de mettre en relation les acteurs économiques comme les collectivités et les associations.

exemple : ADEME

6 - L'intranet : Accessible uniquement par le personnel d'une même entreprise ou d'une direction, l'intranet met à disposition et partage des informations professionnelles.

exemple : CAUE de l'Isère

7- Mini-site - Jeux concours - événementiel

Un mini-site est rattaché à un site institutionnel ou à une marque et permet de créer une dynamique et faire la promotion d'un événement particulier.

exemple : CAUE de l'Isère

8- Application Web

Une application web est un programme s'exécutant directement dans un navigateur web. Consultable en ligne, il n'est pas nécessaire d'installer un logiciel sur votre ordinateur, pour accéder à l'application il faut cependant avoir une connexion à Internet (plusieurs applications proposées sur Chrome Web Store ou Marketplace de Mozilla). Une application web peut être un jeu, un logiciel de traitement de contacts, un fil d'actualités, un moteur de recherche, un système de gestion des contenus, etc.

Les exemples les plus connus d'application Web sont : Google Map et Facebook

exemple : Geolink

D. Portail web et Navigateur

1. Portail web

C'est un site web disposant d'outils et des ressources. Un portail web est un site web qui offre une porte d'entrée commune à un large éventail de ressources et de services accessibles sur Internet et centrés sur un domaine d'intérêt ou une communauté particulière.

2. Navigateur

C'est un logiciel conçu pour consulter et afficher le World Wide Web. Techniquement, **c'est** au minimum un client HTTP. Les **navigateurs** internet **sont** les logiciels nécessaires à l'affichage des sites internet. Un **navigateur** Internet n'est pas un moteur de recherche. Les 5 **navigateurs** les plus populaires **sont** Google Chrome, Safari, Mozilla Firefox, Opera et Microsoft Edge.

E. Protocole

Un protocole informatique est un ensemble de règles qui régissent les échanges de données ou le comportement collectif de processus ou d'ordinateurs en réseaux ou d'objets connectés.

- **Https ou Http**

  HyperText Transfer Protocol Secure (***HTTPS***) est une extension de HyperText Transfer Protocol (HTTP). Il est utilisé pour une communication sécurisée sur un réseau informatique et est largement utilisé sur Internet. C'est aussi la variante avec authentification et chiffrement de ce protocole.

Hypertext Transfer Protocol (HTTP) (ou protocole de transfert hypertexte en français) est un protocole de la couche application servant à transmettre des documents hypermédias, comme HTML.

Les verbes du http

- **GET** : La méthode GET demande une représentation de la ressource spécifiée. Les requêtes GET doivent uniquement être utilisées afin de récupérer des données.
- **HEAD** : La méthode HEAD demande une réponse identique à une requête GET pour laquelle on aura omis le corps de la réponse (on a uniquement l'en-tête).
- **POST** : La méthode POST est utilisée pour envoyer une entité vers la ressource indiquée. Cela entraîne généralement un changement d'état ou des effets de bord sur le serveur.
- **PUT** : La méthode PUT remplace toutes les représentations actuelles de la ressource visée par le contenu de la requête.
- **DELETE** : La méthode DELETE supprime la ressource indiquée.
- **CONNECT** : La méthode CONNECT établit un tunnel vers le serveur identifié par la ressource cible.
- **OPTIONS** : La méthode OPTIONS est utilisée pour décrire les options de communications avec la ressource visée.
- **TRACE** : La méthode TRACE réalise un message de test aller/retour en suivant le chemin de la ressource visée.
- **PATCH** : La méthode PATCH est utilisée pour appliquer des modifications partielles à une ressource.

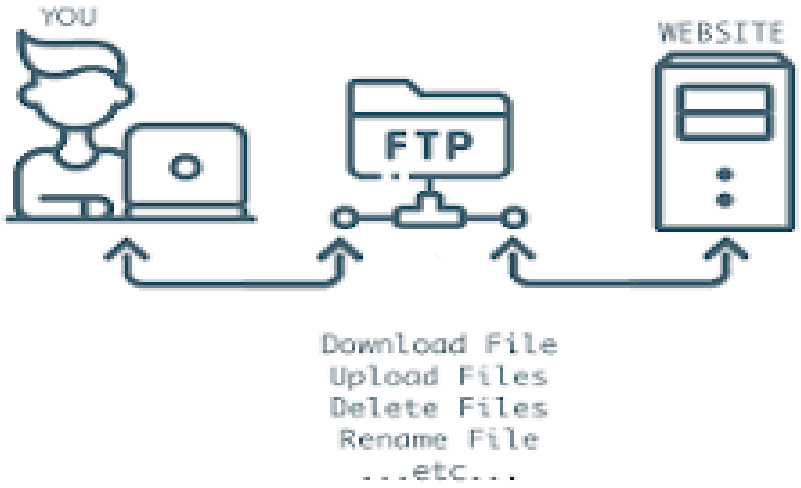
- FTP (file transfert Protocol)

C'est un logiciel utilisé dans le transfert de fichiers entre deux ordinateurs. De nos jours, le serveur FTP est principalement utilisé par les webmasters qui gèrent des sites Internet pour la mise en ligne, la modification ou encore la sauvegarde de contenus.

File Transfer FTP, est un 

Protocol (protocole de transfert de fichier), ou protocole de communication destiné au partage de fichiers sur un réseau TCP/IP. Il permet, depuis un ordinateur, de copier des fichiers

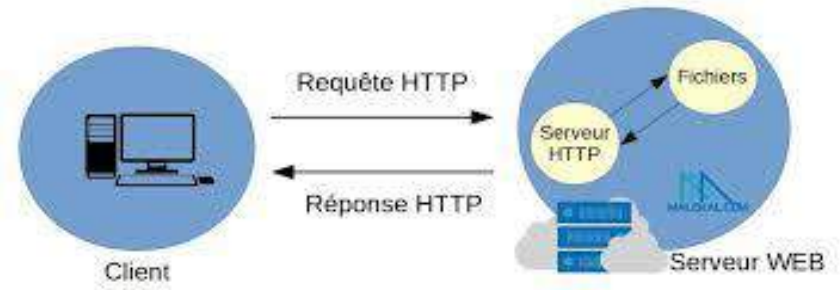
vers un autre ordinateur du réseau, ou encore de supprimer ou de modifier des fichiers sur cet ordinateur.



F. Serveur web

Un serveur web est un ordinateur connecté à Internet et sur lequel sont hébergés des sites web, composés de pages HTML (le serveur web, également appelé serveur HTTP, peut également être composé d'un groupe d'ordinateurs). Il est aussi un logiciel utilisé pour servir des ressources à travers le protocole http.

La fonction principale d'un serveur Web est de stocker et délivrer des pages web qui sont généralement rendues en HTML (HyperText Markup Language,). Le protocole de communication HyperText Transfer Protocol (HTTP) permet le dialogue via le réseau avec le logiciel client, généralement un navigateur web.



Les principaux **serveurs Web** sont Apache (le **serveur Web** le plus répandu), IIS (Internet Information **Server**) de Microsoft et Nginx (prononcé engine X) de NGINX. Il existe d'autres **serveurs Web**, notamment le **serveur** NetWare de Novell, Google **Web Server** (GWS) et la gamme des **serveurs** Domino d'IBM.

LAMP, WAMP, MAMP - Ils ne font en réalité que référence à une pile de serveurs Web **Apache-MySQL-PHP** sur Windows, Linux et Mac. Mystère résolu. Mais attendez… XAMPP est un peu différent des autres.

XAMPP

Pour configurer une pile de serveurs Web, nous devrons normalement installer Apache, MySQL, puis PHP individuellement. Il existe de nombreux tutoriels sur la façon de le faire en ligne, ou en utilisant cherchant le logiciel compatiple pour votre système entre Mamp pour mac, Wamp pour windows et Lamp pour Lunix. Tout cela peut parraitre dûr à trouver, mais heureusement, quelqu'un a tout compilé dans un seul package d'installation appelé **XAMPP** - Disponible sur Windows, Linux et Mac.

- XAMPP signifie *Cross-Platform (X)*, Apache, MariaDB, P HP et Perl.
- **Qu'est-ce que MariaDB ?** Fondamentalement, un peu d'histoire ici. MySQL était autrefois open-source, puis Oracle l'a repris. Les développeurs d'origine de MySQL avaient des inquiétudes quant à la reprise d'une entreprise et ont ainsi créé une «spin-off» appelée MariaDB.
- **MariaDB** est toujours hautement compatible avec MySQL.
- **Perl** est encore un autre langage de programmation.

- **XAMPP** est l'un des choix les plus populaires si vous souhaitez configurer rapidement un serveur Web.

a. Sortes des serveurs web

Sortes serveurs web appelés aussi http sont :

- Le **NCSA HTTPd** est un serveur **HTTP** développé au National Center for Supercomputing Applications de 1993 à 1998.
- Apache HTTP Server de la Apache Software Foundation, successeur du NCSA HTTPd ;
- Apache Tomcat de l'Apache Software Foundation, évolution de Apache pour J2EE(**Java EE** signifie **Java** édition entreprise pour les applications qui s'exécutent sur des serveurs, par exemple des sites Web.); Google Web Server de Google

- **Serveur apache**

Le logiciel libre Apache HTTP Server (Apache) est un serveur HTTP créé et maintenu au sein de la fondation Apache. Jusqu'en avril 2019, ce fut le serveur HTTP le plus populaire du World Wide Web. Il est distribué selon les termes de la licence Apache.

G. Hébergeur web

Il est une entreprise qui fournit l'hébergement sur Internet de systèmes informatiques divers et variés à des personnes ou entités qui ne désirent pas le faire par leurs propres moyens.

Dans le domaine informatique, un hébergeur représente un ordinateur central qui fournit des services d'hébergement à certains postes de travail. Dans un contexte plus vaste, l'hébergement informatique fait donc appel à un grand serveur pour le public sur le net.

Lors de la création d'un site web, il est indispensable de souscrire à un **service d'hébergement web**. Toutefois, le choix d'un hébergeur peut être un vrai casse-tête lorsque l'on débute sur Internet.

Quelles sont les principaux critères à prendre en compte ? Comment choisir un hébergeur adapté à vos besoins ? Voici les réponses.

Le type d'hébergement

On distingue 3 types d'hébergement web :

- **L'hébergement mutualisé :** vous partagez un serveur avec d'autres sites ;
- **L'hébergement dédié :** vous êtes le seul utilisateur de votre serveur ;
- **Le cloud :** vous disposez d'un serveur virtuel pouvant être redimensionné en fonctions de vos besoins.

Le premier est la solution la moins coûteuse, mais aussi la moins performante : elle ne sera pas forcément adaptée à un site à fort trafic, par exemple, mais conviendra tout à fait pour un petit blog personnel.

L'hébergement dédié, plus onéreux, permet de profiter à plein de votre serveur pour de meilleures performances.

Le cloud, enfin, est idéal pour les sites ayant un trafic très important et très variable, puisque c'est la solution la plus flexible, mais aussi souvent la plus chère.

Les possibilités d'intégration

Si vous utilisez un CMS (système de gestion de contenu) ou une plateforme e-commerce, alors il est souhaitable que votre hébergement soit parfaitement adapté à la solution choisie. La plupart des hébergeurs permettent ainsi de simplifier l'installation des sites utilisant les CMS les plus populaires (WordPress, Prestashop, Joomla…)

Les ressources disponibles

Les hébergeurs web proposent les plus souvent différentes offres plus ou moins adaptées en fonction du niveau de performance recherché pour le site.

Par exemple, un petit blog personnel n'aura pas les mêmes besoins qu'un site e-commerce référençant des milliers de produits. Lors du choix d'une offre d'hébergement, soyez particulièrement attentif à l'espace disque proposé ainsi qu'au nombre de bases de données inclues.

Le support technique

Le support fourni par votre hébergeur est au moins aussi important que l'hébergement web en lui-même, surtout si vos connaissances techniques sont limitées. En cas de problème avec votre site ou serveur, la réactivité et l'efficacité de votre hébergeur peut faire toute la différence.

Les adresses mail

La plupart des hébergeurs proposent également un service permettant de créer des adresses mail à partir du nom de domaine du site. Veillez à choisir une formule fournissant assez d'adresses, en fonction du nombre de personnes qui en auront besoin.

La sécurité

L'hébergeur web a aussi une responsabilité importante en matière de sécurité. Renseignez-vous sur sa fiabilité et sa capacité à protéger vos données en cas de piratage, de panne, d'intempéries, etc.

La sauvegarde

La plupart des hébergeurs permettent d'effectuer régulièrement une sauvegarde de votre site afin de pouvoir revenir à une version antérieure en cas de problème.Bien qu'il existe d'autres moyens de faire un backup de votre site (comme des plugins WordPress), assurez-vous que votre service hébergement web inclut cette fonctionnalité.

H. Domaine

Un nom de domaine est, dans le système de noms de domaine DNS, un identifiant de domaine internet. Un domaine est un ensemble d'ordinateurs reliés à Internet et possédant une caractéristique commune.

Pour implémenter un site sur internet, il faut toujours chercher un nom de domaine.

I. Framework

Un Framework est une boite à outils pour un développeur web. Frame signifie cadre et work se traduit par travail. Un Framework contient des composants autonomes qui permettent de faciliter le développement d'un site web ou d'une application.

En programmation informatique, un Framework est un ensemble cohérent de composants logiciels structurels qui sert à créer les fondations ainsi que les grandes lignes de tout ou partie d'un logiciel, c'est-à-dire une architecture.

Par exemple, un Framework peut inclure des classes, variables et fonctions prédéfinies.

J. Librairie ou bibliothèque

Qu'est-ce qu'une bibliothèque ?

En informatique et en sciences de données, une bibliothèque, également connue sous le nom de library ou librairie logicielle, désigne un ensemble de fonctions (comme print par exemple) et de classes utilitaires (regroupent des constantes ou des méthodes). En informatique, une bibliothèque logicielle est une collection de routines, qui peuvent être déjà compilées et prêtes à être utilisées par des programmes.

G. Programmation web

C'est la programmation informatique qui permet d'éditer des sites web. Elle permet la création d'applications, destinées à être déployées. Le développement Web est le travail impliqué dans le développement d'un site Web pour Internet ou un intranet. Le développement Web peut aller du développement d'une simple page statique unique de texte brut à des applications Web complexes, des entreprises électroniques et des services de réseaux sociaux.

- **Technologie web**

Les technologies web sont un ensemble de protocoles et spécifications qui composent et sont utilisés par le World Wide Web et ses normes.

Les **technologies Web** font référence aux divers outils et techniques utilisés dans le processus de communication entre différents types

SECTION 2 : ASP.NET (ACTIVE SERVER PAGES)

1. Qu'est-ce que ASP.NET ?

C'est un **Framework de développement** sophistiqué de Microsoft côté serveur. Les développeurs utilisent ASP.NET pour mettre en œuvre des sites Web dynamiques, des applications Web et des services basés sur le Web. Après des décennies de développement, le Framework se poursuit aujourd'hui sous le nom d'ASP.NET Core.

2. L'histoire du développement d'ASP.NET

Développé à l'origine par Microsoft, ASP.NET **fait désormais partie de la .NET Foundation.** Alors que les premières versions étaient encore publiées en tant que logiciel propriétaire, l'ASP.NET moderne est un projet open source.

L'ASP.NET utilisé aujourd'hui est le résultat d'un processus de développement qui s'est étendu sur plusieurs décennies. **ASP est devenu ASP.NET, puis ASP.NET Core**. Dans le cadre des progrès réalisés par la technologie Web, le Framework a radicalement changé au fil des ans. Intéressons-nous tout d'abord à l'historique du développement d'ASP.NET :

L'histoire du développement d'ASP.NET

Développé à l'origine par Microsoft, ASP.NET **fait désormais partie de la .NET Foundation.** Alors que les premières versions étaient encore publiées en tant que logiciel propriétaire, l'ASP.NET moderne est un projet open source.

L'ASP.NET utilisé aujourd'hui est le résultat d'un processus de développement qui s'est étendu sur plusieurs décennies. **ASP est devenu ASP.NET, puis ASP.NET Core**. Dans le cadre des progrès réalisés par la technologie Web, le Framework a radicalement changé au fil des ans. Intéressons-nous tout d'abord à l'historique du développement d'ASP.NET :

Technologie	Environnement de développement	Version actuelle	Extensions de fichiers
ASP	Windows	3.0 / 2000-02	.asp
ASP.NET	Windows	4.8 / 2019-04	.aspx/.aspx.cs, .aspx.vb etc.
ASP.NET Core	multiplatformes	5.0 / 2020-11	.cshtml, .vbhtml, etc.

Conseil

Créez votre propre page d'accueil sans aucune connaissance en programmation ! Rien de plus simple avec l'outil de création My Website de IONOS. Active Server Pages (ASP) : premier langage de script de Microsoft côté serveur

L'« Active Server Pages » (ASP) original fut le premier langage de script côté serveur publié par Microsoft. Active Server Pages a permis la **création dynamique de pages Web sur le serveur** à partir de 1996. Alors que d'autres langages de script s'appuient sur Linux comme système d'exploitation et sur un serveur Web open source tel qu'Apache, Microsoft a intégré les Active Server Pages au « Internet Information Server » (IIS) fonctionnant sous Windows.

La fonctionnalité de base **d'ASP est à peu près comparable à celle de PHP ou des « Java Server Pages » (JSP)**. Des fichiers modèles sont utilisés dans les trois technologies. Les fichiers modèles contiennent des extraits de code exécutables qui sont intégrés au contenu HTML statique. Le code est écrit entre des balises spéciales pour le séparer du code HTML environnant. Les langages VBScript, JScript et PerlScript, disponibles à l'époque et spécifiques à Microsoft, étaient utilisés comme langages de programmation.

Lorsqu'un navigateur accède à une URL (« Request »), le code est exécuté sur le serveur. L'exécution génère du contenu HTML qui est inséré dans les structures HTML prédéfinies. Le résultat global est un **document composé de contenu HTML statique et dynamique** ; celui-ci est transmis au navigateur en tant que réponse (« Response ») et affiché à l'utilisateur. Toute modification du contenu de la page, par exemple par une entrée de l'utilisateur, nécessitait un cycle complet de requête-réponse et donc un rechargement de la page. Voici un exemple de code ASP :

```
<p>
  The server's current time:
  <%
    Response.Write Now()
  %></p>
```

ASP n'était pas encore un Framework. Il s'agissait bien plus d'une **collection dissociée de plusieurs objets** permettant, une fois associés, de constituer un site Web dynamique :

- Application
- Request

- Response
- Server
- Session
- ASPError

3. ASP.NET : des Active Server Pages au Framework

ASP.NET, le successeur de l'ASP classique, a été introduit vers 2003. La collection dissociée d'objets a été remplacée par le **framework .NET** utilisé **comme sous-structure**. Cela a permis d'abstraire des processus fréquemment requis, tels que l'authentification et l'autorisation des utilisateurs ou encore l'accès aux bases de données. De manière générale, ASP.NET est à peu près comparable aux frameworks Java tels que « Struts » ou « Spring ».

ASP.NET a inclus les « Web Forms » comme caractéristique essentielle. L'approche des Web Forms **a permis aux développeurs Windows expérimentés de programmer des sites Web dynamiques**. Les mécanismes sous-jacents du Web étaient cachés aux développeurs et ces derniers pouvaient continuer à utiliser les flux de travail et les environnements de développement qui leur étaient familiers. Des outils visuels de développement rapide d'applications (RAD) spécifiques à Microsoft ont en particulier été utilisés.

Les Web Forms ont permis aux développeurs Windows de se lancer rapidement dans la programmation Web. Dans le même temps, cette approche limitait cependant le degré de contrôle sur les pages HTML livrées.ASP.NET Un modèle de développement alternatif a rapidement été ajouté avec ASP.NET MVC. Ce modèle suit le **modèle établi de « Model View Controller » (MVC)** et permet une séparation plus nette des préoccupations. Basé sur le framework pionnier « Ruby on Rails », ASP.NET MVC offrait une fonctionnalité permettant d'« échafauder » un projet.

Aujourd'hui**, ASP.NET a été remplacé par son évolution « ASP.NET Core »**. Dans le langage courant, les deux noms sont cependant souvent utilisés comme synonymes.

- **ASP.NET Core : Nouveau développement en open source**

Avec la sortie d'ASP.NET Core, l'organisation du Framework a changé. Dans le cadre de l'ouverture progressive de Microsoft, le développement d'ASP.NET Core a été placé

sous l'égide **de la .NET Foundation**. Le code source du projet est disponible sous une licence open source.

Sur le plan technique, ASP.NET Core est une refonte d'ASP.NET 4.0. **Les** composants d'ASP.NET qui ont été développés de manière organique ont été **fusionnés**. Le développement de projets ASP.NET Core et l'hébergement en dehors de l'écosystème Windows ont été rendus possibles. Sous Windows, ASP.NET Core est basé sur le Framework .NET ; les autres systèmes d'exploitation ont, quant à eux, recours au Framework .NET Core.

Dans le droit fil du développement continu de la technologie Web, ASP.NET Core est paré pour le Cloud. En plus du serveur Internet Information Server (IIS) traditionnellement utilisé par Microsoft, des environnements de **serveurs ouverts et des conteneurs** sont utilisés comme environnement d'hébergement ASP.NET. Le framework prend en charge le code côté client et les approches modernes de la programmation réactive. ASP.NET Core se place ainsi au même niveau que les frameworks basés sur JavaScript tels que React.

À quels projets ASP.NET convient-il ?

Des projets Web de toutes sortes peuvent être mis en œuvre avec ASP.NET Framework, tels que des **sites Web dynamiques et des applications Web**, y compris les « Single Page Apps » (SPA). Des services Web tels que des API et des systèmes de communication en temps réel peuvent également être implémentés. Au fil des ans, différentes approches ont été utilisées à des fins diverses.

4. Modèles de programmation dans ASP.NET

Avec l'avancée des technologies Web depuis les années 2000, de nouvelles approches du développement Web ont été intégrées à ASP.NET. Vous pouvez les considérer comme une **boîte à outils bien organisée** : différents outils sont disponibles pour différents types de projets. En fonction des besoins, plusieurs approches peuvent être combinées dans un même projet. Avec l'apparition d'ASP.NET Core, les nombreux modèles de programmation développés de manière organique ont été fusionnés permettant ainsi une simplification dans le choix des approches appropriées.

a. Web Forms ASP.NET

Les Web Forms classiques permettent d'assembler des **pages à partir de composants prédéfinis**. Pour ce faire, un kit de construction de formulaire visuel est utilisé ; les

différents composants sont positionnés par glisser-déposer. Cette possibilité était particulièrement intéressante pour les développeurs ayant une expérience de la programmation Windows. Ils ont pu utiliser les outils qui leur étaient familiers pour le « Rapid Application Development » (RAD), à ceci près que le produit final n'était pas une application Windows, mais un site Web dynamique.

L'approche Web Forms est basée sur le **modèle code-behind de Microsoft** qui renforce la séparation des préoccupations :

- Les **fichiers modèles** avec l'extension .aspx définissent la structure HTML d'une page et contiennent des espaces réservés pour le contenu généré dynamiquement
- La **logique d'application proprement dite est stockée dans un fichier distinct** avec l'extension *.aspx.cs* ou *.aspx.vb*. C'est le fichier code-behind qui lui donne son nom.
- Le client récupère le fichier *.aspx* disponible sous une URL. Les **composants statiques et dynamiques sont combinés** sur le serveur et le document HTML résultant est livré au client.
- Lorsque l'utilisateur saisit des données sur la page, celles-ci sont transférées **vers la même URL via une requête GET ou POST** et sont traitées par le fichier code-behind.

Pour la mise en œuvre de la logique de l'application, une approche orientée objet est utilisée : le fichier code-behind définit une classe dérivée. Le langage de programmation utilisé est généralement C# ou Visual Basic. Il est intéressant de noter que le **fichier code-behind est précompilé une fois**. Cela permet une exécution plus rapide et une plus grande résistance aux erreurs lors de l'appel de la page.

b. ASP.NET MVC

Contrairement à l'ASP d'origine, les Web Forms ont constitué un pas en avant vers la séparation des préoccupations. Avec ASP, le HTML statique et les composants de code étaient mélangés dans un seul et même fichier. Les Web Forms ont permis la séparation entre le modèle et le fichier code-behind. Avec ASP.NET MVC, ASP.NET s'est vu doter d'un autre modèle de programmation qui permet un **développement Web basé sur le modèle Model View Controller (MVC).**

Le modèle MVC sépare la logique de l'application (« Model »), le modèle de présentation (« View » et l'interaction avec l'utilisateur (« Controller »). L'un des

avantages de l'approche MVC est que les préoccupations individuelles peuvent être testées plus facilement. De plus, la séparation des préoccupations permet d'utiliser différents contrôleurs. Par exemple, au lieu d'envoyer toutes les entrées de l'utilisateur vers une seule URL et de recharger la page, on utilise AJAX via jQuery ce qui permet de **recharger certaines parties de la page**. Avec ASP.NET Core MVC, la tradition d'ASP.NET MVC se poursuit dans la version actuelle du framework.

c. Pages Web ASP.NET

Les Web Forms ASP.NET et ASP.NET MVC conviennent parfaitement à la réalisation de sites Web complexes. Si vous avez besoin de plusieurs pages avec des composants réutilisables, les deux modèles de programmation peuvent s'y prêter. Mais que faire si cela est superflu ? Supposons que vous vouliez créer un site Web simple, composé d'une seule ou de quelques pages. Vous disposez de quelques composants dynamiques, **mais l'accent est davantage mis sur une mise en page sophistiquée** que sur une logique d'application complexe et le traitement des entrées de l'utilisateur. Dans ce cas, il serait exagéré de définir vos propres classes ou de viser une répartition selon le modèle MVC.

Les pages Web ASP.NET constituent un bon choix dans les cas où la logique de l'application est moins importante que la mise en page et la conception sophistiquéeASP.NET. Comme pour l'ASP ou le PHP classique, une **combinaison de structures HTML statiques et de composants de code dynamiques** a lieu dans un fichier. Pour cela, une syntaxe spéciale est utilisée. Les pages Web ASP.NET sont particulièrement adaptées à la création de pages de renvoi.

d. API Web ASP.NET

Les modèles de programmation présentés jusqu'à présent visent tous à générer du contenu HTML destiné à des utilisateurs humains. En outre, ASP.NET Framework contient également des modèles qui servent à fournir une infrastructure pour les projets Web. L'API Web ASP.NET est un **modèle de programmation permettant de créer des API REST.** L'accès aux points de terminaison de l'API se fait via AJAX. Pour la transmission des données, on utilise le format JSON ou XML.

e. ASP.NET WebHooks

ASP.NET WebHooks est une implémentation du modèle WebHooks. Les WebHooks vous permettent de **publier et de vous abonner à des événements** qui ont

lieu dans un système. Il peut s'agir, par exemple, de l'ajout d'un fichier ou de la réception d'un paiement. Un abonné enregistre le changement à suivre auprès du système de publication. Pour ce faire, il transmet une URL, le WebHook à qui il doit son nom. Lorsque l'événement enregistré se produit, le système de publication appelle le WebHook ; l'abonné est informé de l'événement.

SignalR

SignalR est un framework pour la **communication en temps réel entre le client et le serveur**. Le framework est basé sur la norme WebSockets et permet le transfert bidirectionnel de données. Les navigateurs qui ne prennent pas en charge les WebSockets sont pris en charge par des mécanismes de secours. SignalR est souvent utilisé pour mettre en œuvre des services de chat basés sur un navigateur et des logiciels de vidéoconférence.

Nouveaux modèles de programmation dans ASP.NET Core

ASP.NET Core est le successeur d'ASP.NET. Le framework ASP.NET Core a été réécrit, mais est hautement compatible avec son prédécesseur. **D**ans la version Core, les composants d'ASP.NET qui étaient séparés ont été fusionnés**.** De plus, certains composants ont été nouvellement développés et ont conservé leurs noms existants. Par exemple, le framework SignalR existe à la fois en version ASP.NET et ASP.NET Core. Jetons un œil aux nouveaux développements majeurs d'ASP.NET Core.

ASP.NET Core MVC : Sites Model View Controller Sites pilotés par API

ASP.NET Core MVC **combine les fonctionnalités d'ASP.NET MVC et de l'API Web ASP.NET**. Cela permet de développer des applications Web hautement dynamiques avec une interface utilisateur et une API sous-jacente modulaires. Le framework .NET Core est utilisé comme base commune. Les approches familières du développement API de .NET peuvent être transférées au développement MVC et vice versa.

Pages Razor : Poursuite du développement des pages Web ASP.NET

Il faut noter que razor est considéré comme un moteur de Template dans l'environnement et la technologie ASP.NET.

Les pages Razor occupent un créneau similaire à celui des anciennes pages Web ASP.NET. Elles peuvent être utilisées dans les cas où le **modèle Model-View-Controller représenterait une surcharge inutile**. Si vous souhaitez par exemple créer une

page de renvoi, vous pouvez la mettre en œuvre comme une page Razor sans trop d'efforts. La syntaxe Razor est utilisée pour créer le modèle de page. Comme d'habitude dans l'univers .NET, C# et Visual Basic peuvent être utilisés comme langages de programmation.

Jetons un œil à un exemple de page Razor. Il est à noter qu'un signe @ initial est utilisé à la place des balises de code d'ouverture et de fermeture habituelles des langages de modèle :

```
@page
@model HelloWorldModel

@if (Name != null) {
   <p>Hello dear @Name</p>
}
<form method="post">
   <p>
      <label asp-for="Name"></label>
      <input class="form-control" asp-for="Name" />
   </p>
   <input type="submit" value="Say Hello" />
</form>
```

Programmation réactive avec Blazor : « .NET in the browser »

Le framework Blazor repose sur la syntaxe Razor susmentionnée ; en fait, Blazor est l'acronyme de « Browser + Razor ». Comme son nom l'indique, Blazor se concentre sur le **navigateur en tant qu'environnement d'exécution**. Avec Razor Pages, le traitement de l'interaction de l'utilisateur a lieu sur le serveur. Blazor permet une programmation réactive, dans laquelle les composants individuels de la page dans le navigateur réagissent dynamiquement aux changements. De ce point de vue, Blazor est à peu près comparable aux technologies React, Angular et Vue.

Voici un exemple simple de programmation réactive avec Blazor. Nous lions la valeur d'un champ de saisie à la variable Name. La syntaxe Razor est utilisée pour cela :

```
@page "/"
<h1>A Blazor example</h1>
<p>Welcome to Blazor, @Name.</p>
<input bind="@Name" type="text" class="form-control" placeholder="Name" />
```

Outre la programmation réactive, une autre fonctionnalité de Blazor est très intéressante : les **langages .NET peuvent être compilés via WebAssembly pour une exécution dans le navigateur**. C'est la raison pour laquelle on appelle parfois Blazor l'approche « .NET in the browser ». L'avantage est qu'il n'est pas nécessaire d'écrire du JavaScript pour le code côté client. Au lieu de cela, le développement se fait en C# ou Visual Basic ; le code peut accéder aux composants familiers du framework .NET.

Quels sont les avantages et les inconvénients d'ASP.NET ?

ASP.NET ou ASP.NET Core offre un environnement mature pour le développement d'une grande variété de projets Web. Le champ d'application comprend les langages de programmation, les éditeurs de code et les IDE ainsi que les outils de développement et un écosystème florissant de packages disponibles gratuitement. De nos jours, des **méthodes modernes telles que la programmation réactive, WebSockets et WebAssembly** sont utilisées.

Le plus grand inconvénient à l'utilisation d'ASP.NET était traditionnellement son étroite relation avec Microsoft et le Vendor Lock-in qui en découlait. Avec l'ouverture progressive vers l'open source, cette question est désormais moins préoccupante.

Quels sont les avantages d'ASP.NET ?

Les développeurs familiarisés avec la **programmation dans l'écosystème Microsoft** sont ceux qui gagnent le plus à utiliser ASP.NET. Ils peuvent facilement accéder aux langages, outils et workflows qui leur sont familiers. Le framework mature .NET est utilisé comme base d'ASP.NET. Cela signifie que des composants adaptés sont disponibles pour une grande variété d'applications. C'est un avantage considérable lorsqu'il s'agit de mettre en œuvre des applications complexes rapidement et avec des résultats fiables.

Le framework .NET a une structure modulaire et contient le « Common Language Runtime » (CLR) en tant qu'environnement d'exécution. Cela permet **l'utilisation de divers langages de programmation**, pour autant qu'ils soient conformes à la norme « Common Language Infrastructure » (CLI). En plus des langages classiques orientés objet C# et Visual Basic, les langages CLI développés par Microsoft comprennent le nouveau langage fonctionnel F#. Les langages CLI peuvent être exécutés en tant que code côté client dans le navigateur via Blazor et WebAssembly.

À l'origine, ASP.NET était un logiciel propriétaire sous le contrôle de Microsoft. Aujourd'hui, c'est un projet open source sous l'égide de la .NET Foundation. Au cours de cette ouverture, le **gestionnaire de paquets NuGet** et le dépôt public associé ont été mis en place. Comparable à NPM ou RubyGems, ASP.NET met ainsi à disposition des développeurs un écosystème de paquets librement utilisables.

Quels sont les inconvénients d'ASP.NET ?

Le plus grand avantage d'ASP.NET, c'est-à-dire l'utilisation de **l'écosystème propre à Microsoft**, est également le plus grand inconvénient de cette technologie. En effet, le développement Web repose en grande partie sur des plates-formes, des langages et des formats libres et ouverts. Les développeurs qui plongent pour la première fois dans l'univers .NET sont confrontés à une variété déconcertante de versions et de modèles de programmation.

Traditionnellement, toute personne souhaitant développer avec le framework ASP.NET **ne pouvait le faire que sous Windows**. Depuis des années, Microsoft fait un effort concerté pour s'ouvrir aux normes largement utilisées et à l'open source. Avec l'apparition d'ASP.NET Core, le développement fonctionne désormais également sur les trois grandes familles de systèmes d'exploitation Windows, macOS et Linux. En outre, il est désormais possible de conserver l'environnement de développement dans un conteneur Docker.

L'ASP.NET classique ne pouvait être hébergé que sur la **technologie de serveur de Microsoft, Internet Information Server (IIS).** C'est un inconvénient majeur par rapport aux autres frameworks Web, qui fonctionnent tous également sous Linux. L'hébergement ASP.NET a des exigences particulières et n'est pas disponible auprès de tous les fournisseurs. Même s'il était possible d'utiliser d'autres serveurs Web avec « Mono » en tant qu'implémentation .NET gratuite, le véritable changement n'est intervenu qu'avec l'apparition de .NET Core et de l'« Open Web Interface for .NET » (OWIN). En effet, OWIN permet de découpler une application ASP.NET du serveur Web sous-jacent, levant ainsi l'un des principaux obstacles à l'utilisation d'ASP.NET.

Quelles sont les exigences liées à l'environnement d'hébergement APS.NET ?

Le framework ASP.NET est spécifique à l'environnement d'hébergement. Les autres frameworks Web basés sur les langages PHP, Java, Python ou JavaScript

fonctionnent tous sur des serveurs Linux. Traditionnellement, seul l'hébergement d'applications ASP.NET **nécessitait Windows comme système d'exploitation du serveur**.

En outre, le serveur Web Internet Information Services (IIS) de Microsoft était obligatoire. Pour héberger une application ASP.NET sur des serveurs Windows, il convient de choisir l'hébergement géré Windows. Avec ce modèle d'hébergement, les serveurs sont constamment entretenus par le fournisseur et alimentés avec des mises à jour logicielles.

L'hébergement d'applications ASP.NET Core est beaucoup plus simple. Outre IIS, elles peuvent être **hébergées sur divers environnements de serveurs**. On utilise le serveur Web intégré Kestrel, qui fonctionne sous Windows, Linux et macOS. Les serveurs Web populaires tels que Nginx et Apache peuvent être configurés en tant que Reverse Proxy et utilisés avec Kestrel. L'alternative Kestrel HTTP.sys est également disponible sous Windows.

5. Visual Studio

Visual Studio est un environnement de développement intégré (IDE). Il permet de programmer à l'aide de différents langages dont Visual Basic, C# et C++.

Visual Studio n'est pas un outil gratuit. Par contre, les abonnés au MSDN (Microsoft Developer Network) peuvent l'utiliser gratuitement.

Microsoft .NET Framework

Microsoft .NET Framework est une architecture d'application (framework) pour la création d'applications de nouvelle génération. Le Framework .NET inclut des produits visant principalement à rendre les applications facilement accessibles sur Internet.

ASP.NET : (Se prononce A S P point net, acronyme de Active Server Pages for .NET)

Contrairement à la croyance populaire, ASP.NET n'est pas un langage de programmation. Il s'agit d'un ensemble de classes basées sur le Framework .NET visant le développement d'applications Web et de services Web XML. Ces classes sont utilisables avec les différents langages de programmation supportés par le Framework .NET, dont Visual Basic et C#. En fait, ASP.NET est un modèle de programmation qui permet au programmeur de se concentrer sur la logique d'affaires du programme plutôt que sur la « plomberie ».

ASP.NET WebForms vs ASP.NET MVC

Il existe deux principales architectures de développement ASP.NET. La première à avoir été utilisée est l'architecture utilisant les WebForms ou, en français, les formulaires Web. C'est généralement de cette architecture qu'il est question quand on parle d'ASP.NET. On y utilise des composants comme <asp:GridView>, <asp:DropDownList>, <asp:TextBox>, etc. Chacun des composants possède de nombreux événements que le programmeur peut utiliser pour poser une action donnée. Le cycle de vie de la page Web est un aspect important de la programmation ASP.NET WebForms.

ASP.NET MVC a vu le jour en 2009. La programmation avec cette architecture est tout à fait différente de celle avec les WebForms et les deux ne sont pas compatibles ensemble. Un site Web sera donc écrit soit en ASP.NET WebForms, soit en ASP.NET MVC. **C# (Se prononce C sharp).**

Chapitre 2 : DEVELOPPEMENT DE FRONT-END ET BACKEND

Dans ce chapitre nous allons parler de frontend appelé aussi front-office, qui est la partie visible du site web, celle avec laquelle les utilisateurs interagissent. Il s'oppose par nature au back end qui est la partie du site ou de l'application que les utilisateurs ne peuvent pas voir : Les langages de programmation sont différents pour chaque spécialité. Puis nous finirons par la partie backend ou back office.

Section 1 DEVELOPPEMENT DE FRONT-END

Le développement web frontal correspond aux productions HTML, CSS et JavaScript d'une page internet ou d'une application qu'un utilisateur peut voir et avec lesquelles il peut interagir directement.

Pour bien commencer, il faut avoir quelques prérequis sur le HTML, CSS et Le framework Bootstrap ainsi que d'un environnement développement intégré (EDI) appelé Visual studio.

1.1. Environnement développement intégré (EDI)

Pour développer des application web avec Asp.net, il faut avoir l'idée sur l'EDI Visual studio. Microsoft Visual Studio est une suite de logiciels de développement pour Windows et mac OS conçue par Microsoft. La dernière version s'appelle Visual Studio 2022. **Langages de programmation :** C++, C#, Visual Basic .NET, J#

Logo de VS

1.1.1. Présentation de Visual studio

Pour utiliser Visual studio, il faut d'abord installer le logiciel en respectant les configurations suivantes :

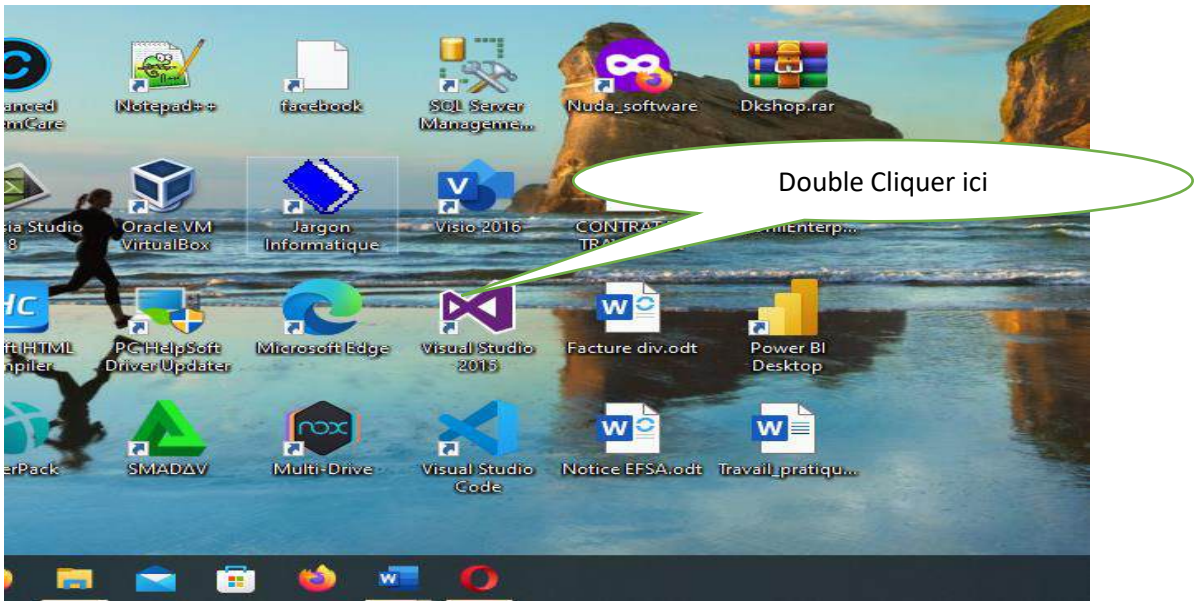
Visual Studio Professional 2015 with Update 3 Hardware Requirements • 1.6 GHz or faster processor • 1 GB of RAM (1.5 GB if running on a virtual machine) • 10 GB of available hard disk space • 5400 RPM hard disk drive • DirectX 9-capable video card (1024 x 768 or higher resolution)	Windows 10 Windows 8.1 Windows 8 Windows 7 SP 1 Windows Server 2012 R2 Windows Server 2012 Windows Server 2008 R2 SP1

Dans votre machine, il faut veiller sur les différents framework, par exemple 3.5, et 4.6.

- **Procédure pour le faire** : Sélectionnez Démarrer > Panneau de configuration > Programmes > Programmes et fonctionnalités. Sélectionnez Activer ou désactiver des fonctionnalités Windows. Si la fonctionnalité n'est pas déjà installée, sélectionnez Microsoft. NET Framework 3.5.1 et cliquez sur OK.
- Veuillez en sort que la machine soit connectée sur internet

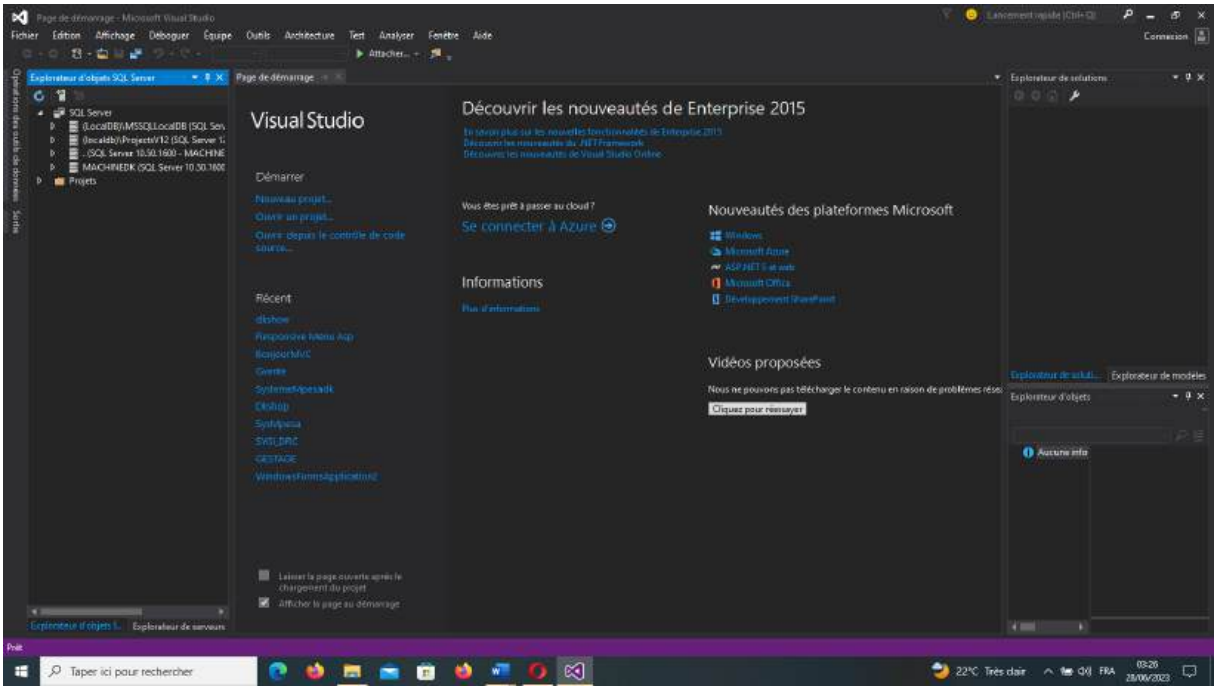
Pour installer Microsoft Visual Studio pour notre cas 2015 mais vous pouvez utiliser toutes les versions que vous vouliez.

- Télécharger Microsoft Visual Studio dans ce lien :
- https://visualstudio.microsoft.com/fr/vs/older-downloads/
- Puis installer ou chercher le setup
- A la fin de l'installation, lancer le logiciel pour afficher cette fenêtre



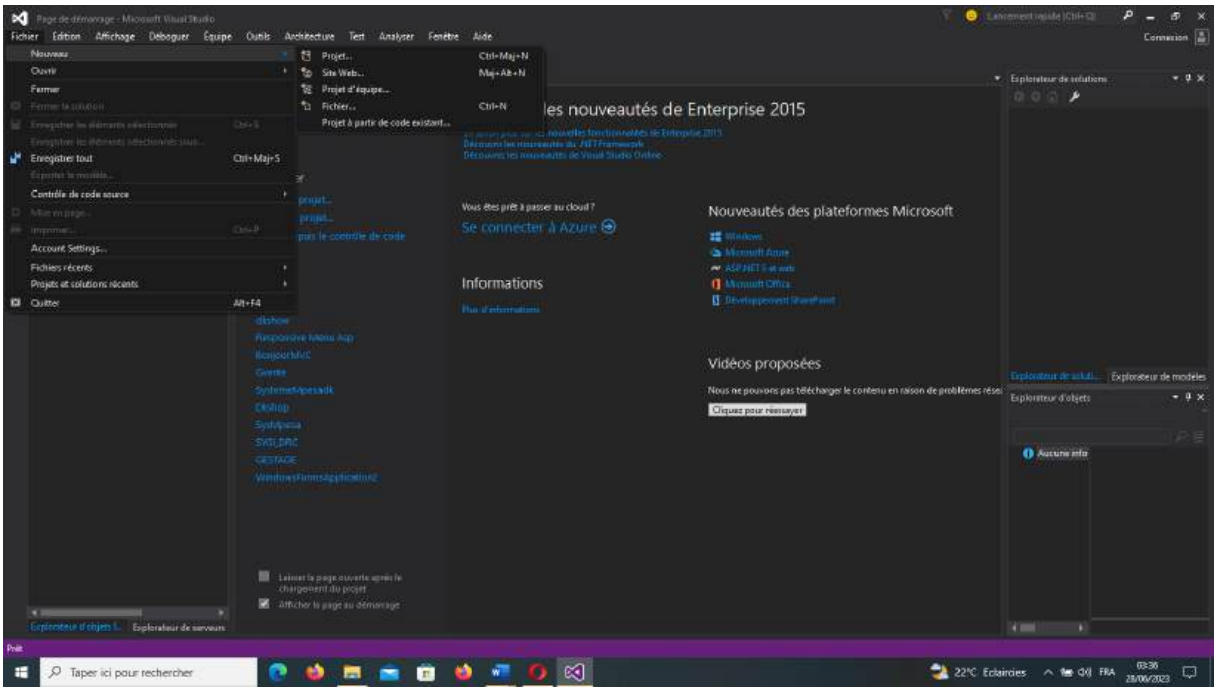
Page d'accueil du logiciel

1. La page de démarrage :



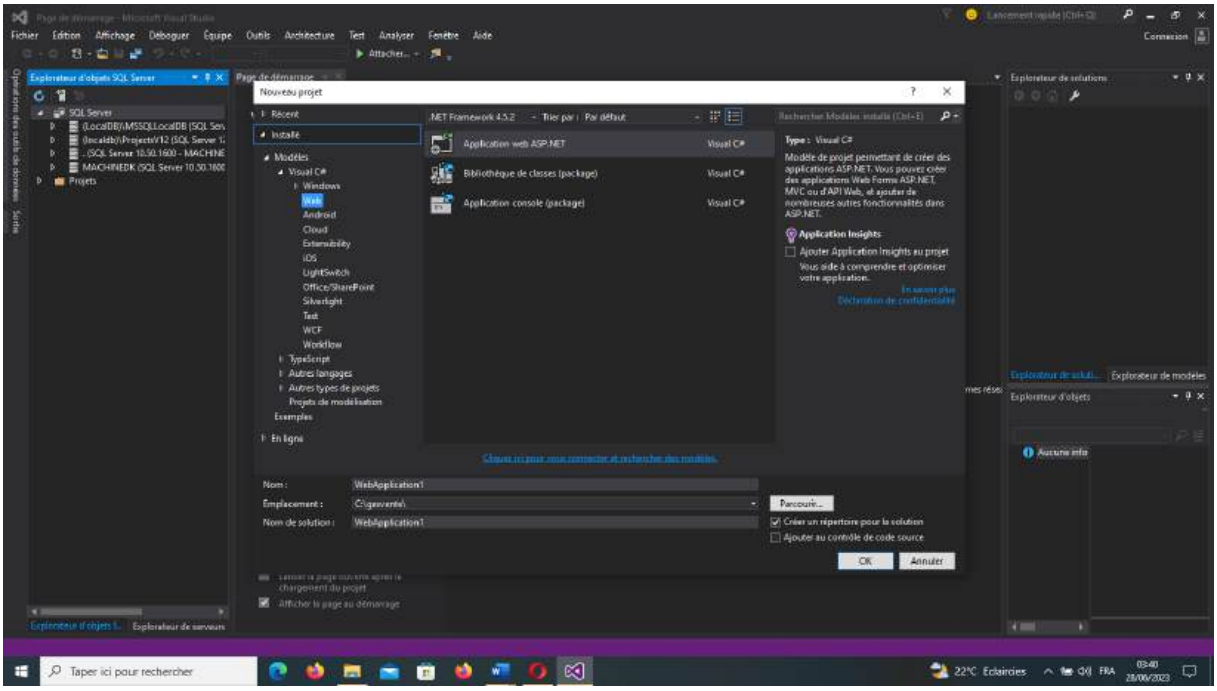
2. Première de chose à faire dans cet environnement est créé un projet

- Pour créer un projet, cliquer sur fichier/nouveau/projet ou directement nouveau projet



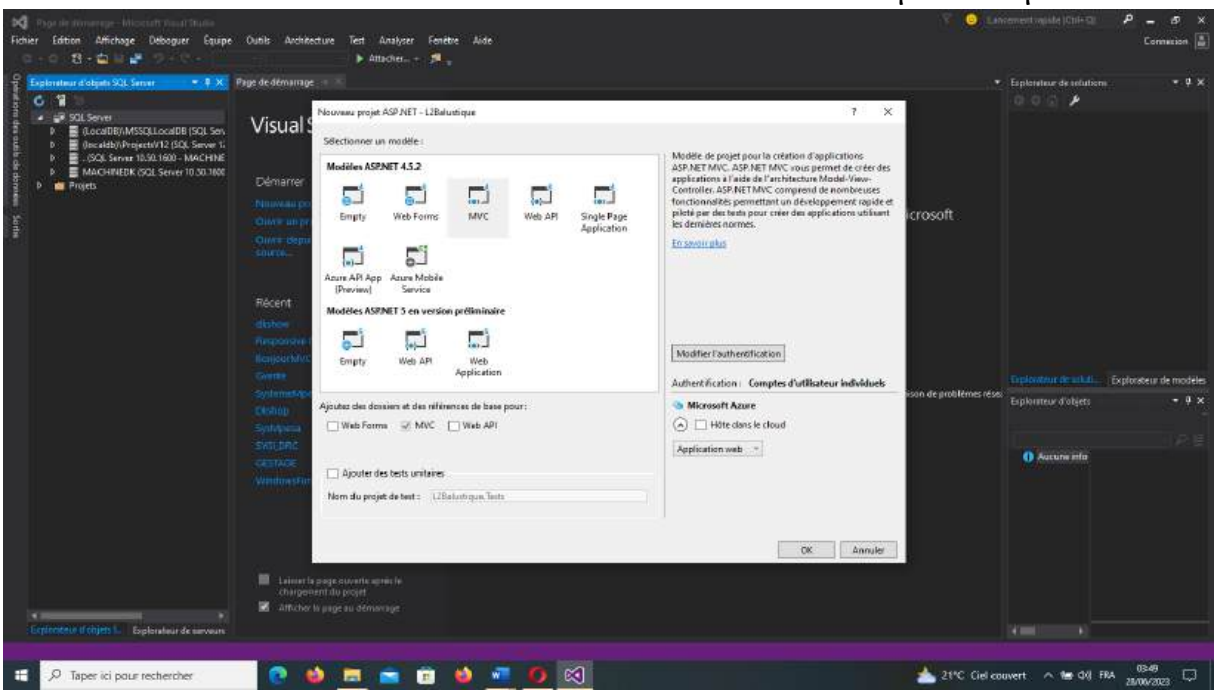
- Sélectionnez modèle web pour le langage Visual C-Sharp
- Puis cliquer sur application web
- Saisir le nom du projet « L2Balustique », si possible choisir l'emplacement du projet en cliquant sur parcourir.

- ok.

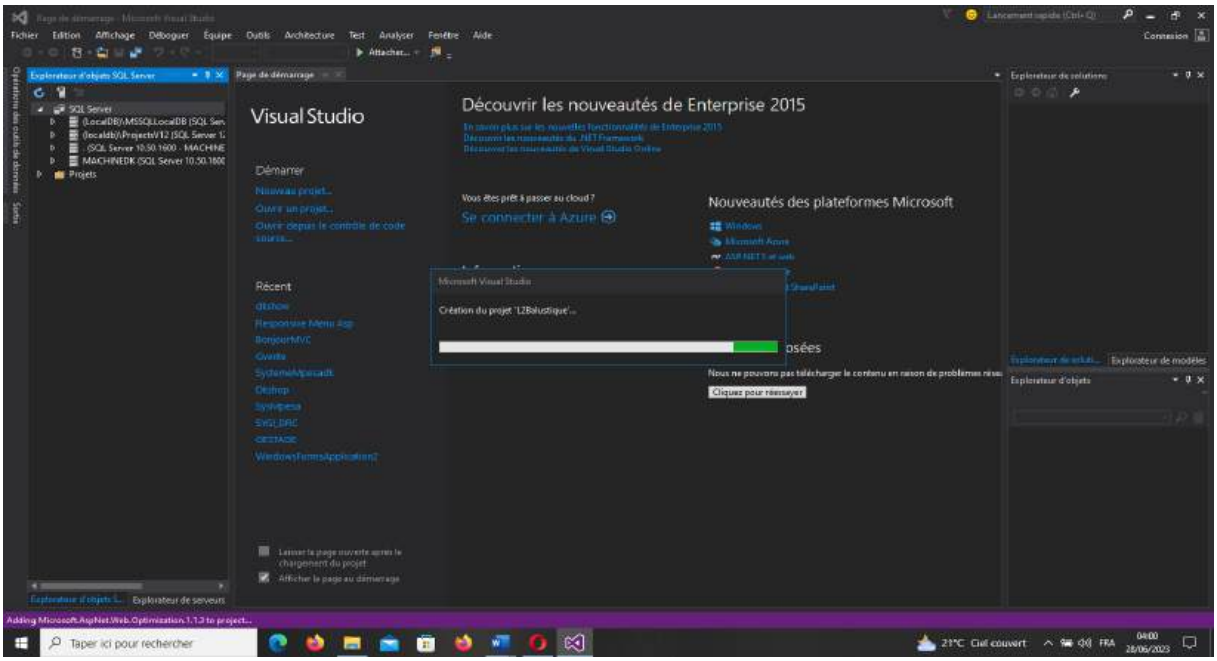


Une fenêtre s'affiche pour choisir le modèle. Par défaut nous allons trouver le projet dans MCV. Nous pouvons laisser par defaut dans l'optique d'obtenir un modele organiser en modele,contrôle et vues.

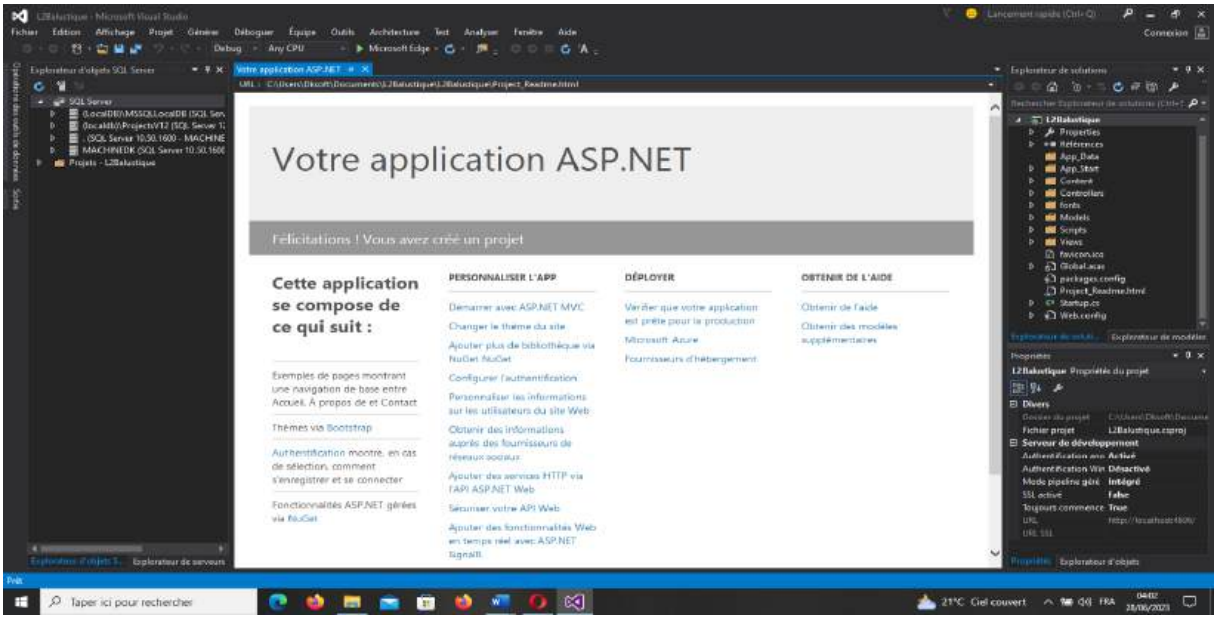
Verifier si la référence de base est MVC seulement puis cliquer sur ok



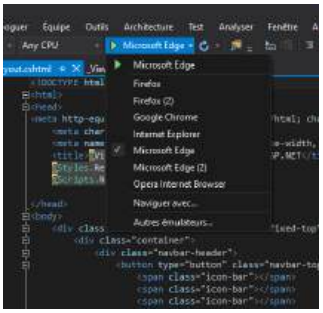
Veuillez attendre le chargement du projet



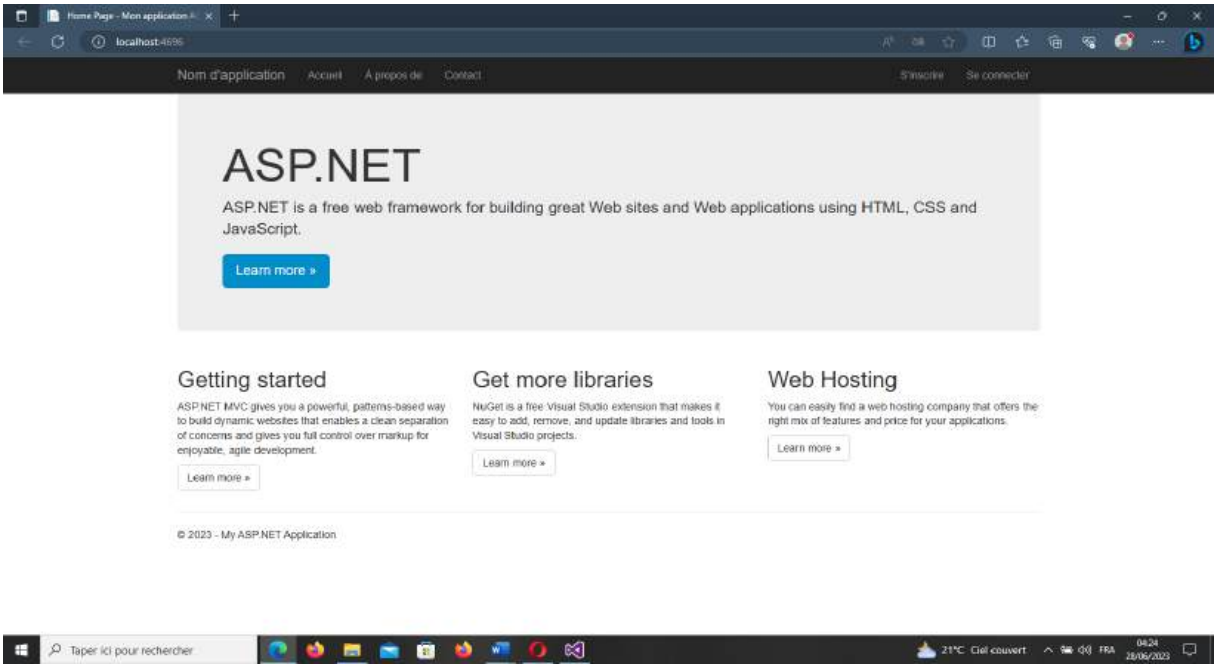
Le projet est créé, nous pouvons maintenant construire notre application



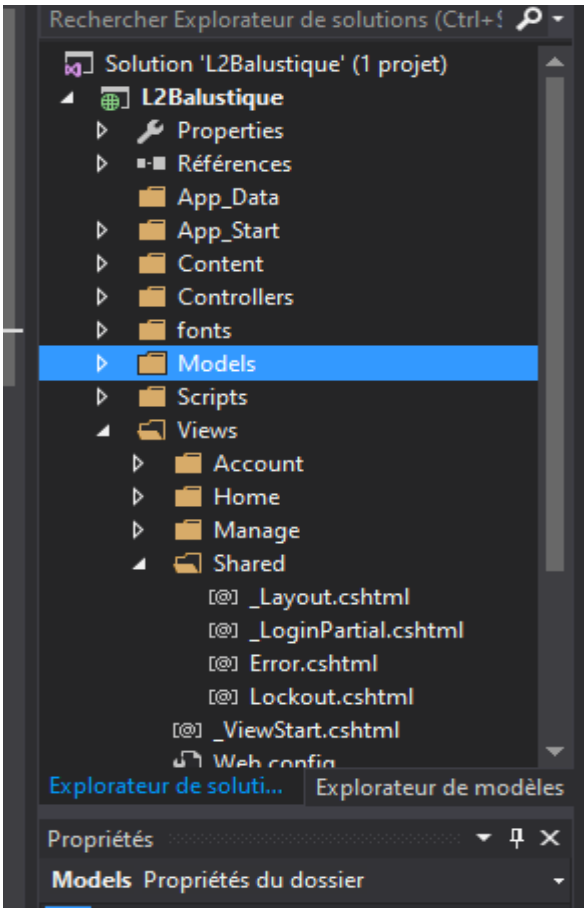
Pour lancer notre premier test, il faut sélectionner un navigateur là où se trouve le bouton exécuté



Voilà le résultat de notre page de démarrage du projet, qui n'est qu'un layout (appelé aussi mise en page ou disposition) principal.



1.1.2. Structure de dossier du projet MVC



- Le "Account" se trouvant dans le dossier Views : C'est un dossier qui contient des fichiers de connexion et de sécurité
- Le "App_Data" : C'est un dossier qui contient des bases de données et fichiers de données
- Le "Images" : C'est un dossier qui contient des images
- Le "Scripts" : C'est un dossier qui contient des scripts de navigateur
- Le "Shared" : c'est un dossier qui contient des fichiers communs (like layout and style files ou en français comme les fichiers de mise en page et de style)
- Les "App_start" : vous devez remarquer qu'un nouveau dossier App_Start est présent dès la création du projet. Celui-ci contient un ensemble de classes utilisées pour l'initialisation de l'application, dans la méthode Application_Start du fichier Global.asax!
- Pour d'autres dossier comme : Models, controllers et views facilitent streucture app.

1.2. NOTION SUR LANGAGE HTML ET CSS EN ASP.NET

1.2.1. Rappel sur le langage HTML et CSS

1. **HTML :** (Internet) Langage informatique qui décrit les éléments d'une page Web à l'aide de balises, en vue de sa publication dans le World Wide Web.

HTML est l'abréviation de HyperText Markup Language, soit en français « langage hypertexte de balisage ». Ce langage a été créé en 1991 et a pour fonction de structurer et de donner du sens à du contenu.

CSS signifie Cascading StyleSheets, soit « feuilles de style en cascade ». Il a été créé en 1996 et a pour rôle de mettre en forme du contenu en lui appliquant ce qu'on appelle des styles.

Il ne faut JAMAIS utiliser le HTML pour faire le travail du CSS :

- HTML est utilisé pour baliser un contenu, c'est à dire pour le structurer et lui donner du sens. Le HTML sert, entre autres choses, à indiquer aux navigateurs quel texte doit être considéré comme un paragraphe, quel texte doit être considéré comme un titre, que tel contenu est une image ou une vidéo.
- CSS est utilisé pour appliquer des styles à un contenu, c'est-à-dire à le mettre en forme.

Ainsi, avec le CSS, on pourra changer la couleur ou la taille d'un texte, positionner tel contenu à tel endroit de notre page web ou ajouter des bordures ou des ombres autour d'un contenu.

Les versions actuelles du HTML et du CSS sont HTML5 et CSS3. Il faut savoir que, contrairement aux idées reçues, ce sont encore des versions non finalisées. Cela signifie que ces versions sont toujours en développement et qu'elles sont toujours en théorie sujettes à changements et à bugs.

La version 5 d'HTML, tout comme la version 3 de CSS introduisent de nombreuses nouvelles fonctionnalités longtemps attendues par les développeurs et il serait donc dommage de s'en priver.

Parmi celles-ci, l'insertion simplifiée de vidéos et de contenus audio et de nouvelles balises améliorant la sémantique (on va y revenir, pas d'inquiétude !) pour mieux structurer nos pages en HTML ou encore la possibilité de créer des bordures arrondies en CSS. Pour coder en HTML et en CSS, nous n'avons besoin que d'un éditeur de text :

- NotePad++,Vs code, et Visual studio sous Windows;
- Komodo, pour Mac ou Linux;
- TextWrangler, pour Linux.

1.2.2 Les bases en HTML et CSS

1 Les bases du HTML

a. Éléments, balises et attributs

Il y a trois termes dont vous devez absolument comprendre le sens en HTML. Ce sont les termes élément, balise et attribut. Les éléments, tout d'abord, vont nous servir à définir des objets dans notre page. Grâce aux éléments, nous allons pouvoir définir un paragraphe (élément p), des titres d'importances diverses (éléments h1, h2, h3, h4, h5 et h6) ou un lien (élément a). Les éléments sont constitués de balises. Dans la majorité des cas, un élément est constitué d'une paire de balises : une balise ouvrante et une balise fermante.

Les balises reprennent le nom de l'élément et sont entourées de chevrons. La balise fermante possède en plus un slash qui précède le nom de l'élément. Cependant, certains éléments ne sont constitués que d'une balise qu'on appelle alors balise orpheline. C'est par exemple le cas de l'élément br qui va nous servir à créer un retour à la ligne.

Notez que, dans le cas des balises orphelines, le slash se situe après le nom de l'élément. Notez également que ce slash n'est pas obligatoire et que certains développeurs l'omettent volontairement. Je vous conseille cependant, pour des raisons de propreté de code et de compréhension, de mettre le slash dans un premier temps.

Les attributs, enfin, vont venir compléter les éléments en leur donnant des indications ou des instructions supplémentaires. Par exemple, dans le cas d'un lien, on va se servir d'un attribut pour indiquer la cible du lien, c'est à dire vers quel site le lien doit mener.

Notez que les attributs se placent toujours à l'intérieur de la balise ouvrante d'un élément (ou de la balise orpheline le cas échéant).

Dans

l'exemple ci-dessus, on discerne l'élément a composé :

- d'une balise ouvrante elle-même composée d'un attribut href ;
- d'une ancre textuelle ;
- d'une balise fermante.

L'attribut href (initiales de « Hypertexte Reference ») sert à indiquer la cible de notre lien, en l'occurrence le site Wikipédia. L'ancre textuelle correspond au texte entre les balises. Ce sera le texte sur lequel vos visiteurs devront cliquer pour se rendre sur Wikipédia et également l'unique partie visible pour eux. Voici ce que les visiteurs de votre site verront :

2. Structure de base d'une page en HTML5

En programmation comme dans beaucoup d'autres disciplines, vous l'aurez compris, il y a des règles. Ainsi, toute page écrite en HTML5 doit comporter une certaine structure, un « squelette » qui sera toujours le même. Ce squelette est bien évidemment constitué de divers éléments.

Tout d'abord, toute page HTML5 doit commencer par la déclaration de ce qu'on appelle un « doctype ». Le doctype, comme son nom l'indique, sert à indiquer le type du document. Dans notre cas, le type de document est HTML. On écrira donc :

```
<!DOCTYPE html>
```

Notez bien le point d'exclamation, obligatoire, au début de cet élément un peu particulier. Ensuite, pour que notre page HTML5 soit valide, il va nous falloir indiquer trois nouveaux éléments : html, head (« en-tête ») et body (« corps de page »). L'élément html va contenir toute la page.

L'élément head contiendra entre autres, le titre de la page, l'encodage utilisé et des meta-data (des données invisibles pour les utilisateurs mais essentielles – nous en reparlerons plus tard). L'élément body contiendra tout le contenu de notre page (paragraphes, images, tableaux, etc.). Voilà où nous en sommes rendus pour le moment :

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>@ViewBag.Title - Mon application ASP.NET</title>
```

```
</head>
<body>

</body>

</html>
```

Mais ce n'est pas fini ! Pour que la page soit valide, l'élément head doit lui-même contenir un élément title, qui correspond au titre de la page et le meta charset qui prendra comme valeur le type d'encodage choisi. Pour les langues latines, nous choisirons généralement la valeur « utf-8 ».

Et voilà, vous venez, sans vous en rendre compte, de créer votre première page valide en HTML5 ! Retenez bien ce schéma, il sera toujours le même quelque soit la page HTML5 que vous créerez.

Le HTML permet d'imbriquer des éléments les uns à l'intérieur des autres. C'est même l'une des possibilités qui font toute sa force. Dans l'exemple précédent, par exemple, notre élément title était à l'intérieur de notre élément head, lui-même contenu dans un élément html Toutefois, si le HTML permet d'imbriquer des éléments les uns dans les autres, vous devrez toujours faire bien attention de refermer les balises dans le bon ordre : le dernier élément ouvert est toujours le premier refermé. Autre bonne pratique : l'indentation. Indenter son code, c'est tout simplement l'aérer en ajoutant des espaces au début de certaines lignes afin de le rendre plus lisible pour vous comme pour les autres.

Il n'y a pas de règle absolue concernant l'indentation, certains accentuant plus ou moins le retrait en début de ligne. Pour ma part, j'utilise une tabulation (la touche à gauche du a) à chaque fois que j'ouvre une nouvelle balise à l'intérieur d'un élément. Cela permet de bien hiérarchiser son code et de voir immédiatement quel élément est imbriqué dans quel autre.

Troisième et dernière bonne pratique : commenter son code.

Commenter son code, c'est tout simplement y ajouter des commentaires. Ces commentaires sont spéciaux : ils ne seront pas visibles par vos visiteurs (à moins que ceux-ci n'affichent le code source de la page).

Voici comment on écrit un commentaire en HTML :

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ma première page en HTML</title>
    <meta charset="utf-8"/>
  </head>
  <body>
    <h1>Salut, je suis un gros titre !</h1>
    <p>Et moi je suis un paragraphe</p>
    <!--Moi, je suis un commentaire... Personne
    ne me verra à moins d'afficher le code
    source de cette page...-->
  </body>
</html>
```

3 Listes ordonnées et non-ordonnées

Les listes servent à ordonner du contenu, à lui donner un ordre cohérent. Visuellement, les listes en HTML correspondent à ce que vous créez lorsque vous utilisez des puces dans un document ms Word :

- Élément numéro 1
- Élément numéro 2
- Élément numéro 3

En HTML, les listes vont avoir deux grands intérêts : on va pouvoir les utiliser pour créer des menus ou, dans leur forme brute, pour mieux présenter du contenu pour un blog par exemple. Il existe trois grands types de listes en HTML : les listes ordonnées, les listes non-ordonnées et un dernier type un peu particulier : les listes de définition. Nous allons commencer avec les listes ordonnées et non-ordonnées.

La différence entre les listes ordonnées et non-ordonnées est que les listes ordonnées possèdent un aspect de subordination, d'ordre logique, de classement tandis que ce n'est pas le cas pour les listes non-ordonnées. Pour créer une liste non-ordonnée, on va avoir besoin de deux nouveaux éléments : l'élément ul (abréviation de unordered list), qui va contenir toute la liste et l'élément li (pour list item) que l'on va utiliser pour créer chaque élément de la liste.

3. Listes de définitions

Dernier grand type de listes que nous allons voir ensemble, les listes de définition son utilisées pour définir des termes. Pour créer une liste de définition, il va nous falloir

utiliser l'élément dl (pour definition list), l'élément dt (pour definition term) et l'élément dd pour la définition en soi.

Il n'y a qu'une règle à respecter lorsque l'on crée une liste de définitions : vous devez toujours placer l'élément dt avant l'élément dd, c'est-à-dire le terme à définir avant sa définition. Cela est assez intuitif et ne devrait donc pas vous poser de problème.

En revanche, il est tout à fait possible d'associer plusieurs termes à une définition ou même plusieurs définitions à un même terme.

```
<!DOCTYPE html>
    <html>
        <head>
            <title>L'univers merveilleux des listes</title>
            <meta charset="utf-8"/>
        </head>
        <body>
            <h1>Les listes de définition</h1>
            <dl>
                <dt>HTML</dt>
                <dd>HTML signifie HyperText Markup Language. Créé en 1991, le HTML...</dd
            </dl>
        </body>
    </html>
```

4. Listes imbriquées

HTML nous offre la possibilité d'imbriquer les listes les unes dans les autres très simplement. Pour imbriquer des listes, il suffit de commencer une liste, puis d'en ouvrir une seconde à l'intérieur d'un élément de la première (on peut évidemment imbriquer plus de deux listes en répétant le même processus).

Voici une illustration, en imbriquant par exemple une liste non-ordonnée à l'intérieur d'une liste ordonnée :

```
<!DOCTYPE html>
    <html>
        <head>
            <title>L'univers merveilleux des listes</title>
            <meta charset="utf-8"/>
        </head>
        <body>
            <h1>Les listes imbriquées</h1>
            <ol>
                <li>Natation
                    <ul>
                        <li>Maillot</li>
                        <li>Bonnet</li>
                        <li>Lunettes</li>
                    </ul>
                </li>
                <li>Vélo</li>
                <li>Course à pied</li>
            </ol>
        </body>
    </html>
```

5. Liens internes et liens externes

Tout d'abord, il faut savoir qu'il existe différents types de liens. Pour l'instant, nous allons nous concentrer sur les eux types les plus « classiques » : les liens internes et les liens externes.

Quelle différence entre ces deux types de liens ? Un lien interne est un lien créé entre deux pages d'un même site web tandis qu'un lien externe est un lien menant d'un site web vers un autre site web.

Dans tous les cas, pour créer un lien, nous allons utiliser l'élément « a » accompagné de son attribut href (pour Hypertext Reference) qui sert à indiquer la cible (c'est à dire la destination) du lien.

Quel que soit le type de liens créés, la seule chose qui va changer va être ce que l'on va indiquer en valeur pour l'attribut href. Commençons donc avec les liens internes. Nous avons trois cas à distinguer :

1. La page à partir de laquelle on fait un lien et celle vers laquelle on fait un lien se trouvent dans le même dossier. Dans ce premier cas, il suffit de préciser le nom de la page dans l'attribut href.
2. La page vers laquelle on souhaite faire un lien se trouve dans un sous-dossier. Dans ce cas, il faudra en tenir compte et inclure le nom du sous-dossier dans la valeur de l'attribut href.
3. La page vers laquelle on veut faire un lien se trouve dans un dossier parent. Dans ce cas, nous devrons rajouter « ../ » dans la valeur de l'attribut href.

On a créé quatre pages en HTML (avec une structure minimale pour chacune d'entre elles afin de les rendre valide). On a placé les pages page1.html et page2.html dans le même dossier, la page page3.html dans un sous-dossier relativement à ma page page1.html et la page page4.html dans un dossier parent par rapport à ma page page1.html. On va donc maintenant créer des liens de ma page page1.html vers mes pages page2.html, page3.html et page4.html grâce à l'élément a et à l'attribut href :

```
<!DOCTYPE html>
<html>
    <head>
        <title>Liens internes</title>
        <meta charset="utf-8"/>
    </head>
    <body>
        <!--Lien entre 2 pages situées dans un même dossier-->
        <p>Ce lien vous transporte de la page 1 vers la <a href="page2.html">page 2</a></p>

        <!--Lien vers une page située dans un sous-dossier nommé "sous-dossier"-->
        <p>Ce lien vous amène de la page 1 vers la <a href="sous-dossier/page3.html">page 3</a></p>

        <!--Lien vers une page située dans un dossier parent nommé "dossier-parent"-->
        <p>Ce lien vous amène de la page 1 vers la <a href="../page4.html">page 4</a></p>
    </body>
</html>
```

A noter qu'il existe pour les liens internes et externes des attributs facultatifs qui peuvent modifier le comportement par défaut de ces liens. C'est par exemple le cas de l'attribut target : l'attribut target permet de choisir si vous voulez que la cible de votre lien s'ouvre

dans une nouvelle fenêtre / nouvel onglet ou pas. Pour que la cible de votre lien s'ouvre dans une nouvelle fenêtre ou un nouvel onglet, on attribuera la valeur _blank à l'attribut target. Un exemple immédiatement en image

Attribut à retenir donc, car celui-ci peut s'avérer très utile dans de nombreux cas (lorsque vous ne voulez pas que vos visiteurs quittent votre site par exemple). Notez en revanche que vous ne pouvez pas choisir si le lien va s'ouvrir dans un nouvel onglet ou dans une nouvelle fenêtre.

6. Les autres types courants de liens

Les liens internes et externes sont très certainement les types de liens les plus courants, mais c'est loin d'être les seuls ! En effet, on peut utiliser les liens pour faire bien d'autres choses.

Commençons avec les liens de type ancre. Les liens de type ancre sont des liens menant à un autre endroit d'une même page web. Ils peuvent être utile dans le cas d'une page web très longue pour donner à vos visiteurs un accès rapide à une section en particulier par exemple.

Vous comprendrez qu'il va donc tout d'abord nous falloir rajouter quelques lignes de textes dans notre page HTML pour pouvoir tester les ancres (sinon, on n'en verra pas l'effet). Pour créer une ancre, on commence par rajouter un attribut id à une balise ouvrante HTML à l'endroit où l'on veut envoyer le visiteur. On peut attribuer n'importe quelle valeur à cet attribut, le mieux étant de choisir une valeur qui fasse sens.

Ensuite, on crée le lien cliquable en soi qui va amener à notre id. Pour cela, on utilise toujours notre élément a avec son attribut href (on ne réinvente pas la roue à chaque fois), mais cette fois ci on va devoir placer un dièse avant d'écrire la valeur de l'attribut href. La valeur inscrite pour l'attribut href doit être strictement la même que celle donnée à notre id.

1.3. Les bases du CSS

1. Sélecteurs, propriétés et valeurs

Pour rappel, le CSS sert à modifier l'apparence de nos pages web en appliquant des styles au contenu en HTML. Un sélecteur, tout d'abord, va servir à déterminer à quel(s) élément(s) HTML ou à quel type d'éléments on souhaite appliquer un style particulier. Si l'on souhaite appliquer un style particulier à tous nos paragraphes, par exemple, on utilisera le sélecteur « p ».

Une propriété va nous servir à modifier le style d'un élément en ciblant un critère bien particulier comme la taille d'un texte, sa police ou sa couleur par exemple. Une valeur, enfin, va venir compléter une propriété et va en déterminer le comportement. Pour la propriété servant à changer la couleur d'un texte par exemple, la valeur va être la nouvelle couleur à appliquer.

Dans cet exemple, nous utilisons le sélecteur simple « p », ce qui signifie que nous souhaitons appliquer un style particulier à tous les paragraphes de nos pages.

Nous utilisons les propriétés « color » (qui sert à modifier la couleur d'un texte) et « font-size » (pour changer la taille d'un texte). Cela signifie donc que nous travaillerons sur la couleur et la taille de nos paragraphes.

```
P{
Color : blue;
Font-size :18px ;
}
```

Enfin, nous indiquons que nous voulons que tous nos paragraphes s'affichent en bleu grâce à la valeur « blue » et que notre texte ait une taille de 18px avec la valeur « 18px ». Notez d'ores-et-déjà la syntaxe de notre première déclaration en CSS. On entoure les propriétés et les valeurs avec des accolades et on place un point-virgule après avoir spécifié une valeur pour chacune de nos propriétés. Chaque propriété est séparée de sa valeur par un deux-points.

- Où écrire le CSS ?

Nous avons trois possibilités pour écrire notre CSS. L'une d'elles est préférable aux deux autres et nous allons immédiatement, voir pourquoi. Nous pouvons écrire le CSS :

- A l'intérieur de l'élément head de notre document HTML ;
- Dans la balise ouvrante des éléments de notre fichier HTML ;
- Dans un fichier portant l'extension .css séparé.

Pour des raisons de performances du code, de clarté et d'économie de temps, je vous recommande vivement d'utiliser la dernière méthode dès que cela est possible. Voyons ensemble comment cela se passe en pratique pour chacune de ces trois méthodes. Commençons avec la première façon : écrire son code CSS dans l'élément head de notre page HTML. Pour faire cela, il suffit d'insérer un élément style dans notre élément head et de placer nos déclarations CSS à l'intérieur de cet élément style comme ceci

```
<!DOCTYPE html>
    <html>
        <head>
            <title>Illustrations CSS</title>
            <meta charset="utf-8"/>
            <style>
                p{
                    color:blue;
                    font-size:16px;
                }
            </style>
        </head>

        <body>

        </body>
    </html>
```

Deuxième méthode maintenant : écrire du CSS dans la balise ouvrante d'un élément HTML. Pour faire cela, nous allons devoir utiliser un attribut style et lui affecter en valeur nos propriétés CSS :

```
<!DOCTYPE html>
    <html>
        <head>
            <title>Illustrations CSS</title>
            <meta charset="utf-8"/>
        </head>

        <body>
            <p style="color:blue;font-size:16px;">Un paragraphe</p>
        </body>
    </html>
```

Vous remarquerez que l'on respecte la syntaxe du CSS à l'intérieur de l'attribut style en utilisant les deux points et les points virgules. Attention cependant : dans notre premier cas, on utilisait l'élément style tandis que dans le cas présent, style est un attribut.

Troisième et dernière méthode enfin (la méthode recommandée) : écrire le code CSS dans un fichier séparé. Pour faire cela, nous allons déjà devoir ouvrir un nouveau fichier dans notre éditeur de texte et l'enregistrer au format « .css ». Vous pouvez le nommer « style.css ». Pensez bien à enregistrer ce fichier dans le même dossier que votre fichier HTML dont vous souhaitez modifier le style, sinon vous risquez d'avoir des problèmes. Une fois que vous avez fait cela, retournez sur votre page HTML. Nous allons maintenant devoir lier nos deux fichiers HTML et CSS. On va faire cela à l'aide d'un élément link que nous allons placer dans l'élément head de cette manière :

```
<head>
    <title>Illustrations CSS</title>
    <meta charset="utf-8"/>
    <link rel="stylesheet" href="style.css"/>
</head>
```

L'élément link est représenté sous forme de balise orpheline et doit être accompagné de ses deuxattributs « rel » et « href ». L'attribut rel sert à préciser le style du fichier lié (dans notre cas c'est une feuille de style, donc «stylesheet » en anglais). L'attribut href, que vous connaissez déjà, sert à faire le lien en soi.

Si le fichier avait été placé dans un dossier parent ou dans un sous-dossier par rapport à notre fichier HTML, il aurait fallu refléter cela dans la valeur de notre attribut href. Finalement, nous n'avons plus qu'à écrire notre code CSS dans le fichier style.css.

Jusqu'à présent, nous n'avons manipulé que des sélecteurs que l'on appelle « simple », car ils correspondent à des éléments HTML seuls et sans attributs (par exemple le sélecteur p). Ce type de sélecteur doit être préféré tant que possible pour des raisons d'optimisation et de performance du code. En effet, ils requièrent moins de code et sont donc moins gourmands en énergie que des sélecteurs plus complexes. Votre page mettra ainsi moins de temps à charger. Le problème reste qu'on est quand même très limité avec des sélecteurs simples : comment faire pour appliquer un style différent à deux éléments de même type, deux paragraphes par exemple ?

Pour cela que l'on a créé les attributs class et id.

Class et Id sont deux attributs HTML qui ont été créés pour pouvoir appliquer différents styles à des éléments de même type. Class permet également de faire l'inverse et d'appliquer le même style à différents éléments choisis.

Premièrement on se place dans la balise ouvrante d'un élément HTML, on écrit le nom de notre attribut (class ou id), et on lui donne une valeur cohérente.

Cette valeur ne devrait contenir ni de caractères spéciaux (accents et autres) ni d'espace. Par exemple :

```
HTML
<!DOCTYPE html>
    <html>
        <head>
            <title>Illustrations CSS</title>
            <meta charset="utf-8"/>
        </head>

        <body>
            <p class="para_1">Un paragraphe</p>
            <p id="para_2">Un autre paragraphe</p>
        </body>
    </html>
```

Ensuite, on retourne sur le fichier CSS. On va devoir commencer notre déclaration par un point là où on a utilisé un attribut class et par un dièse si l'on a utilisé l'attribut id. Après le point ou le dièse, on écrit la valeur de l'attribut en question pour former notre sélecteur. Enfin, on écrit le code CSS voulu. Voilà ce que ça donne en pratique :

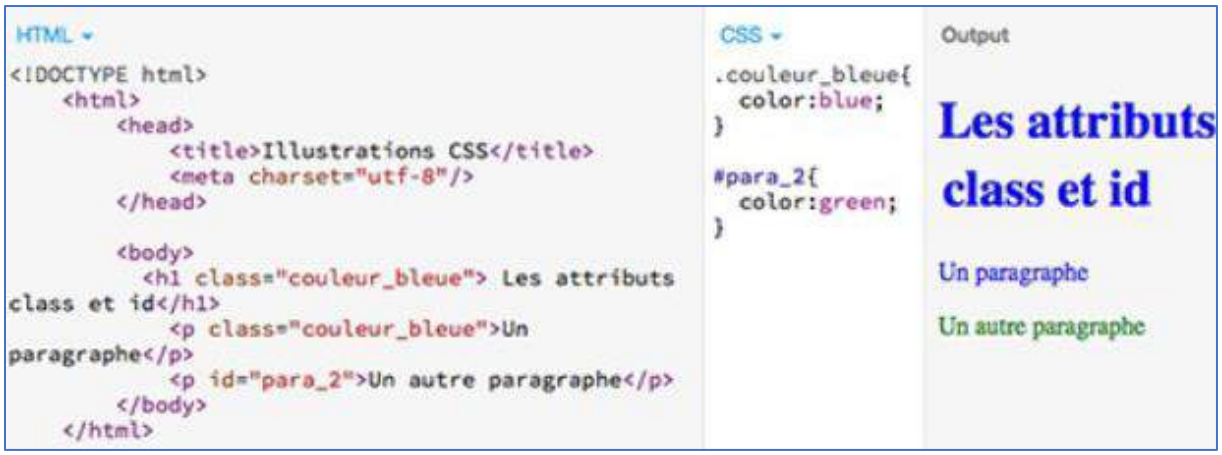
```
HTML
<!DOCTYPE html>
    <html>
        <head>
            <title>Illustrations CSS</title>
            <meta charset="utf-8"/>
        </head>

        <body>
            <p class="para_1">Un paragraphe</p>
            <p id="para_2">Un autre paragraphe</p>
        </body>
    </html>
```

```
CSS
.para_1{
  color:blue;
}

#para_2{
  color:green;
}
```

Nous pouvons maintenant appliquer un style différent à chaque élément HTML grâce aux attributs class et id.

Pourquoi avoir créé deux attributs pour faire la même chose ? En fait, il existe une différence notable entre class et id : un attribut id avec une valeur précise ne peut être utilisé qu'une fois dans une page, au contraire de class.

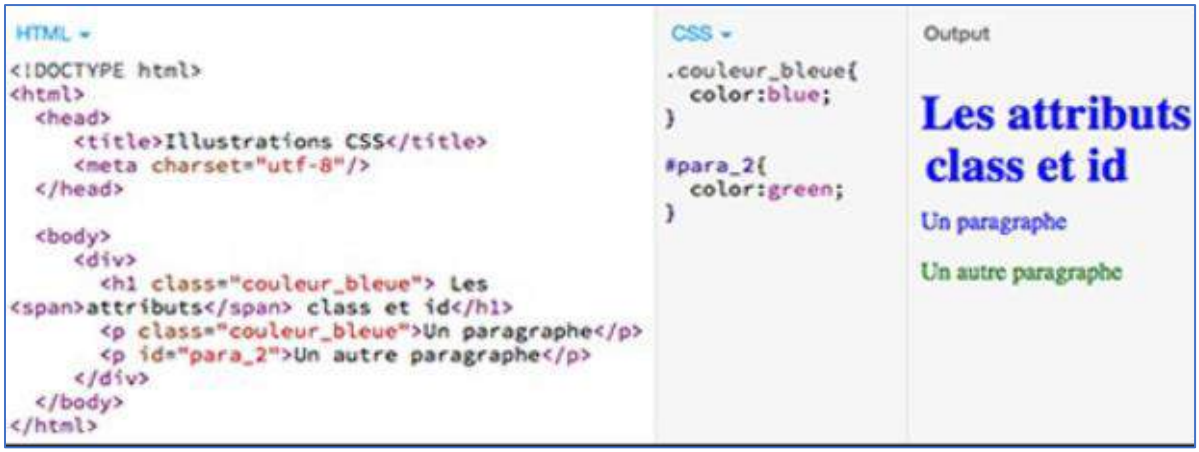
Id sera donc utilisé pour des éléments uniques dans une page web, comme le logo de votre site par exemple. En revanche, on peut utiliser plusieurs attributs class identiques (c'est à dire ayant la même valeur) par page. C'est d'ailleurs une des méthodes que nous utiliserons pour appliquer un même style à différents éléments :



Notez que, dans l'exemple précédent, utiliser deux attributs class n'est pas la meilleure solution. Nous voilà déjà un peu moins limités. Cependant, nous ne pouvons pour le moment appliquer un style qu'à un contenu entre balises. Effectivement, on ne pourrait pas appliquer de style particulier au mot « attributs » de notre titre dans l'exemple précédent. Pour remédier à cela, on a inventé les deux éléments HTML div et span.

2. Les éléments div et span

Les éléments div et span ne possèdent aucune valeur sémantique, ce qui va à l'encontre même du rôle du HTML. Ainsi, vous ne devez les utiliser que lorsque vous n'avez pas d'autre choix. Les éléments div et span vont nous servir de containers. Nous allons nous en servir pour entourer des blocs de code et ainsi pouvoir attribuer des styles particuliers à ces blocs. L'utilisation des éléments div et span est très simple : il suffit d'entourer le bloc de code voulu avec une paire de balises ouvrante et fermante div ou span comme cela :



Généralement, on attribuera une class ou un id à div et span afin de pouvoir différencier nos différents div et span dans notre page. Ainsi, on peut désormais appliquer un style particulier à n'importe quel bout de code dans notre page HTML.

Tout comme pour class et id, il existe une différence entre div et span : div est un élément de type block tandis que span est un élément de type inline.

3. Les éléments de type block et inline

En HTML, tout élément est soit de type block, soit de type inline. Par exemple, div est un élément de type block tandis que span est un élément de type inline. Les éléments de type block sont fondamentalement différents des éléments de type inline en HTML et il est essentiel de bien comprendre les différences entre ces deux types si vous voulez un jour créer un site Internet, ne serait-ce que pour des raisons de mise en page.

Les éléments de type block…	Les éléments de type inline…
Commencent sur une nouvelle ligne	S'insèrent dans une ligne
Occupent toute la largeur disponible	Occupent seulement la largeur nécessaire
Peuvent être imbriqués les uns dans les autres et contenir des éléments de type inline	Peuvent être imbriqués les uns dans les autres mais ne peuvent pas contenir d'éléments de type block

Afin que vous compreniez bien la différence entre les deux types d'éléments, voyons ensemble quelques exemples d'éléments de type inline ou block pour que vous puissiez observer leur comportement.

Eléments de type block	Eléments de type inline
p	em
h1, h2, h3, h4, h5, h6	strong
ol, ul,dl	mark
li	a
table	img

4. Les sélecteurs avancés

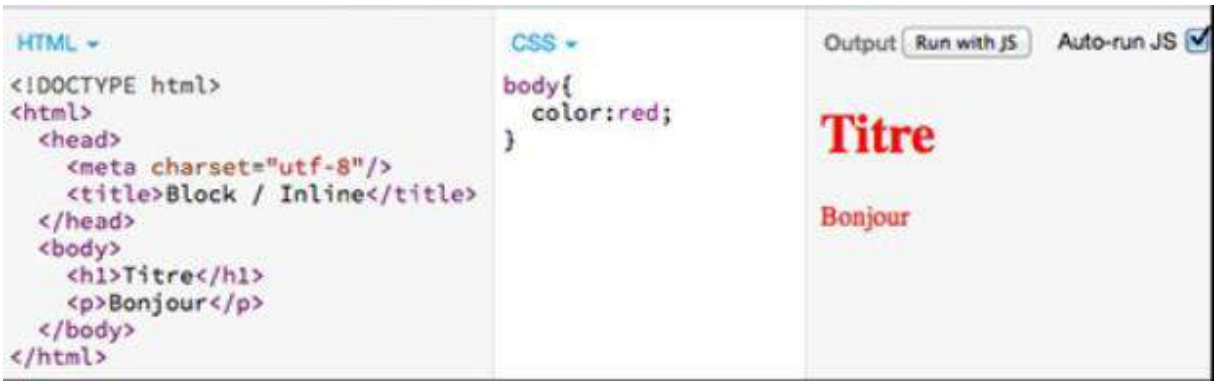
Les sélecteurs avancés sont l'une des grandes forces du CSS. En effet, grâce à eux, nous allons pouvoir cibler du contenu très précisément et assez simplement. Il faut savoir qu'il existe de très nombreux sélecteurs avancés en CSS et on ne présentera que les plus utiles :

Ce sélecteur sert à…	Et il s'écrit comme cela…
Sélectionner tous les éléments (sélecteur universel)	*
Sélectionner deux éléments A et B	A, B
Sélectionner un élément B contenu dans un élément A	A B
Sélectionner le premier élément B suivant un élément A	A + B
Sélectionner tous les éléments A possédant un attribut particulier	A[nom de l'attribut]
Sélectionner tous les éléments A possédant un attribut particulier avec une valeur	A[nom de l'attribut* = « valeur »]
Sélectionner tous les éléments A possédant un attribut particulier avec une valeur précise	A[nom de l'attribut = « valeur »]

5. La notion d'héritage

L'héritage est une notion centrale et fondamentale du CSS. L'héritage signifie que tout élément HTML va hériter des styles de ses parents (c'est le fameux « cascading »).

En HTML, si un élément A est inclus dans un élément B ; l'élément A s'appellera l'enfant et l'élément B sera le parent de l'élément A. Ainsi, si l'on applique un style à l'élément B, l'élément A en héritera automatiquement.



Ici, l'élément body est le parent des éléments h1 et p, puisque les éléments h1 et p sont bien contenus dans l'élément body. Ainsi, lorsqu'on applique un style à l'élément body (ici, mettre le texte en rouge), les éléments p et h1 héritent automatiquement de ce style. Que se passe-t-il lorsque l'on donne deux ordres contradictoires à un élément parent et à son enfant en CSS (par exemple, donner une couleur rouge au parent et bleue à l'enfant) ?

Le CSS possède ici sa logique et le style appliqué sera celui le plus proche de l'élément en question. Cela signifie que si on applique un style à un élément enfant, c'est bien ce style qui lui sera appliqué.

5. Formater du Texte & Positionner des Éléments

- ### Les Propriétés de Type « Font- »

La propriété font-size

Nous utiliserons la propriété font-size lorsque nous voudrons modifier la taille d'un texte. Cette propriété accepte deux types de valeurs : des valeurs de type absolu (en pixel ou en point), ou relatif (en em, ex ou en pourcentage).

Les valeurs de type « relatives » sont appelées de la sorte car elles permettent au texte de s'adapter relativement aux préférences de vos visiteurs. En clair, si vous fixez la taille d'un texte à 100%, ce texte pourra avoir des tailles différentes selon les réglages faits par vos visiteurs sur leurs navigateurs. Ce type de valeur présente des avantages indéniables, et notamment le fait que tous vos visiteurs devraient être capables de lire vos écrits sans difficulté. De plus, le texte peut également s'adapter relativement aux autres éléments de votre page web.

L'utilisation de la propriété font-size, avec des valeurs relative et absolue. Notez qu'on utilisera les notations px pour pixel, pt pour point et % pour pourcentage.

La propriété font-style

La propriété font-style permet de fixer l'inclinaison d'un texte. La propriété font-style accepte 4 valeurs différentes :

• Normal (valeur par défaut) ;

• Italic (change le texte en italique) ;

• Oblique (penche le texte) ;

• Inherit (hérite des propriétés de l'élément parent)

La propriété font-weight

La propriété font-weight permet de fixer le poids d'un texte. Cette propriété accepte 6 valeurs différentes :

- Normal (la valeur par défaut) ;
- Lighter (version allégée de la police) ;

- Bold (la police est en gras) ;
- Bolder (la police est encore plus en gras) ;
- Une centaine compris entre 100 et 900 (du plus léger au plus gras) ;
- Inherit (hérite des styles de ses parents).

La propriété font-family

La propriété font-family permet de choisir la police du texte. Dans tous les cas, nous déclarerons plusieurs polices (on parle de « famille » de polices, d'où le nom de cette propriété) afin de s'assurer qu'au moins une des polices mentionnées soit supportée par vos visiteurs

La propriété color

Il existe plusieurs façons de gérer la couleur d'un texte.

Première façon de changer la couleur d'un texte : en attribuant un nom de couleur (en anglais) en valeur de la propriété color.

black (#000000)	silver (#C0C0C0)	gray (#808080)	white (#FFFFFF)
maroon (#800000)	red (#FF0000)	purple (#800080)	fuchsia (#FF00FF)
green (#008000)	lime (#00FF00)	olive (#808000)	yellow (#FFFF00)
navy (#000080)	blue (#0000FF)	teal (#008080)	aqua (#00FFFF)

Les Propriétés de Type « Text- »

- **L'alignement d'un texte**

Pour modifier l'alignement d'un texte, nous allons utiliser la propriété text-align.

Cette propriété peut prendre cinq valeurs différentes :

- Left : le texte sera aligné sur la gauche ; valeur par défaut ;
- Center : le texte sera centré ;
- Right : le texte sera aligné sur la droite ;
- Justify : le texte sera justifié ;
- Inherit : hérite des propriétés de l'élément parent.

Le centrage ou l'alignement se fait toujours par rapport à l'élément parent le plus proche du texte. Dans l'exemple suivant, on voit bien que mon paragraphe « pdiv » est aligné à droite de son élément parent (c'est-à-dire le div qui fait lui-même 100px de large) et non pas de la page. Le second paragraphe, n'ayant pour parent que l'élément body, est donc bien lui centré sur la page.

La propriété text-decoration

On peut modifier la décoration d'un texte grâce à la propriété text-decoration parmi six valeurs pour cette propriété :

- Underline : le texte sera souligné ;
- Overline : une ligne apparaîtra au-dessus du texte ;
- Line-through : le texte sera barré ;
- Blink : le texte clignotera (attention, ne fonctionne pas sur tous les navigateurs) ;
- Inherit ;
- None : pas de décoration, comportement par défaut

La propriété text-transform

On utilise la propriété text-transform pour modifier l'aspect des caractères d'un texte (majuscules ou minuscules).

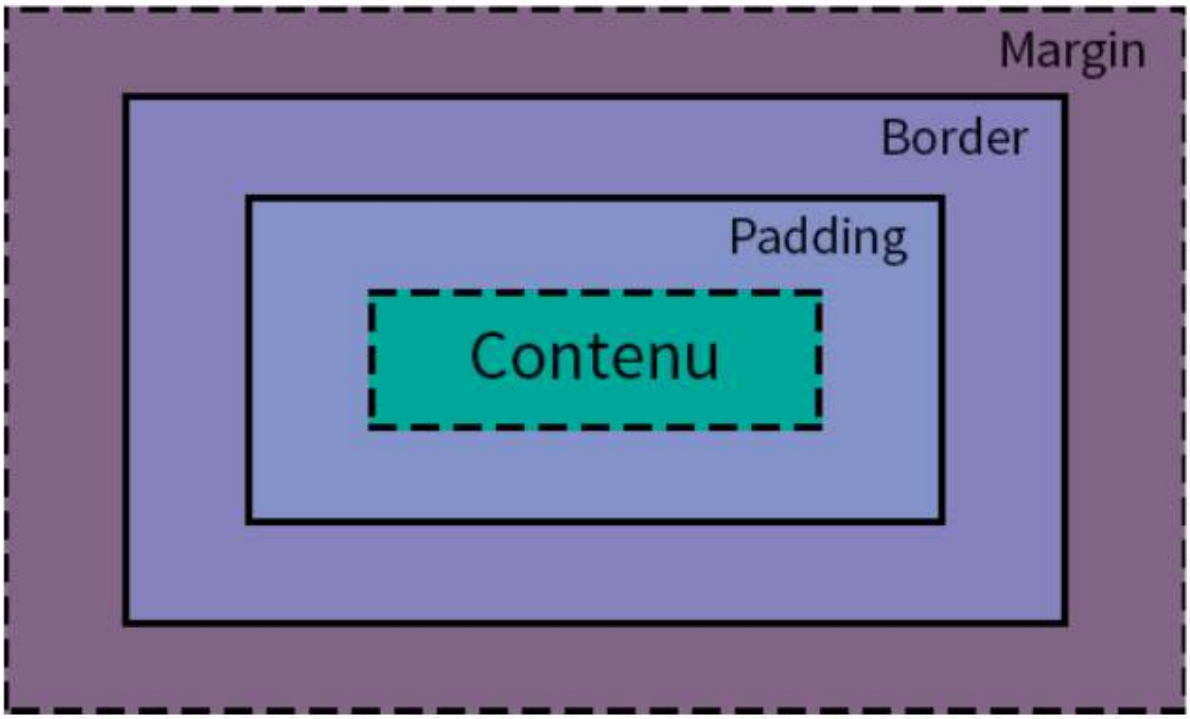
Nous pouvons choisir parmi cinq valeurs :

- Uppercase : transforme tout le texte en majuscules ;
- Lowercase : met tout le texte en minuscules ;
- Capitalize : met uniquement la première lettre de chaque mot en majuscule ;
- None : pas de transformation ;
- Inherit : hérite des styles de l'élément parent.

6. Le Modèle des Boîtes

Le modèle des boîtes est un concept essentiel : « tout élément d'une page est une boîte rectangulaire et peut avoir un padding, une marge et des bordures ». Cela mérite d'être répété : tout élément, qu'il soit un élément de type block ou de type inline, est une boîte rectangulaire d'un élément forment ce qu'on appelle le modèle des boîtes.

Les propriétés permettant d'indiquer la longueur, la largeur, la marge, le padding et les bordures



La première boîte est définie par la longueur et la largeur d'un élément. La padding, ou marge intérieure, forme ensuite la seconde boîte. Puis viennent les bordures qui constituent la troisième boîte. Enfin, la marge extérieure vient former la quatrième et dernière boîte.

7. Hauteur et largeur d'un élément

Tout élément possède une hauteur et une largeur par défaut. La hauteur d'un élément est déterminée par son contenu. Par exemple, des paragraphes d'une ligne ou de deux lignes n'occuperont pas la même hauteur. La largeur par défaut d'un élément est avant tout déterminée par son type (block ou inline) puis par son contenu si l'élément est de type inline. En effet, rappelez-vous que les éléments de type block occupent automatiquement toute la largeur disponible. Pour modifier la hauteur d'un élément, on utilise la propriété height à laquelle on attribue une valeur en px, % ou égale à auto dans la grande majorité des cas.

En utilisant la valeur auto, on laisse le navigateur de nos visiteurs décider de la hauteur que doitprendre l'élément visé. Cela est très utile dans le cas où l'on veut conserver les proportions d'une image tout en l'adaptant à la taille de l'écran de nos visiteurs. Pour modifier la largeur d'un élément, on utilise cette fois la propriété width. Cette propriété prend les mêmes types de valeurs que height.

Les bordures et les bordures arrondies

Il existe de nombreuses sortes de bordures dont certaines sont plus ou moins bien supportées par certains navigateurs. Pour créer des bordures et les personnaliser, nous allons avoir besoin de trois propriétés :

- Border-width, qui va définir l'épaisseur de la bordure (valeur en px) ;
- Border-style, qui va définir le style de la bordure ;
- Border-color, qui va définir la couleur de la bordure (accepte les mêmes valeurs que la propriété « color »).

La propriété border-style peut prendre de nombreuses valeurs différentes. Les valeurs les plus utilisées sont solid, dotted (pointillé) et dashed (tiret).

Les marges intérieures

Pour définir les marges intérieures d'un élément, nous utiliserons la propriété padding.

On peut considérer qu'un élément HTML possède toujours une bordure. Celle-ci peut être explicite, c'est-à-dire matérialisée à l'aide des propriétés CSS vues précédemment ou implicite (invisible). La propriété padding va définir l'espace entre l'élément en soi et sa bordure. Cette propriété doit être utilisée uniquement dans ce but, et jamais pour positionner des éléments dans une page ou les uns par rapport aux autres. On donnera généralement une valeur en px à padding. Notez que l'on peut définir des espacementsdifférents pour chaque marge intérieure de nos éléments en utilisant les propriétés padding-right, padding-bottom, padding-left et padding-top.

Les marges extérieures

Pour définir la taille des marges extérieures, c'est-à-dire de l'espace à l'extérieur des bordures d'un élément, nous allons utiliser la propriété margin. Contrairement à la propriété padding, la propriété margin peut tout-à-fait être utilisée pour positionner des éléments dans une page ou les uns par rapport aux autres. Nous attribuerons généralement des valeurs en px ou en % à cette propriété. Tout comme la propriété padding, nous allons pouvoir des marges différentes de chaque côté de nos éléments avec les propriétés margin-right, margin-bottom, margin-left et margin-top.

Notez que les valeurs par défaut des marges intérieures et extérieures peuvent légèrement différer d'un navigateur à un autre. Cela peut impacter le design général de votre site pour certains de vos visiteurs. Afin de s'assurer que chaque visiteur verra un

résultat conforme à nos attentes, nous pouvons utiliser un « reset CSS » pour notre padding et notre margin.

Dans ce cas-là, c'est très simple, il suffit par exemple d'appliquer un padding et une margin avec des valeurs égales à zéro à notre élément body. Ensuite, on précisera les différentes marges souhaitées à nos éléments enfants.

Faire flotter un élément

Pour aligner des éléments les uns par rapport aux autres, on peut les faire « flotter ». Pour faire flotter un élément, nous utiliserons la propriété float avec les valeurs suivantes : left, right, none ou inherit. Un élément flottant va sortir du schéma naturel (du « flow ») d'une page web pour venir se placer contre le bord gauche ou droit de l'élément qui le contient ou contre le bord de la page. Lorsque l'on fait flotter un élément, les éléments après l'élément flottant vont venir se positionner à côté de celui-ci.

on remarque que l'élément strong, contenu dans l'élément p2, va venir se placer dans le coin à droite de son élément parent (l'élément p2 donc).

L'élément p1 va lui se placer à gauche dans la page tandis que l'élément p2 va se placer à droite ; à côté de l'élément p1. Généralement, on utilisera plutôt la propriété float sur des éléments de type inline comme des images par exemple. En effet, cette propriété peut être la cause de problème d'affichages lorsqu'elle est mal utilisée sur des éléments de type block.

Si l'on veut qu'un élément suivant un élément flottant vienne se placer sous cet élément flottant, il faudra utiliser la propriété clear. Celle-ci accepte trois valeurs : left, right ou both :

- left : un élément va se placer en dessous après un float left ;
- right : un élément va se placer en dessous après un float right ;
- both : un élément va se placer en dessous après un float left ou un float right.

La propriété display

La propriété display est une propriété extrêmement puissante : elle permet de changer le type d'un élément de block à inline ou d'inline à block.

Cette propriété supporte quatre valeurs différentes qui correspondent aux différents types d'éléments possibles : inline, block, inline-block et none. La nouveauté ici est le type inline-block. Ce nouveau type ne peut être donné à un élément que grâce à la propriété display. Il va être un mix des types inline et block.

Un élément de type inline-block se comporte de cette façon : l'élément en soi (contenu et boîtes) se comporte comme un type block tandis que le contenu seulement se comporte comme un type inline. Pour le dire plus simplement, un élément de type inline-block se comportera comme un élément de type inline excepté que l'on va pouvoir contrôler précisément sa hauteur et sa largeur.

Gestion du Background

- **Ajouter de la couleur ou une image pour le fond**

Pour ajouter une couleur de fond, nous allons utiliser la propriété background-color. Cette propriété accepte les mêmes valeurs que la propriété color que nous avons vu précédemment, à savoir des valeurs de type nom de couleur, hexadécimale ou RGB.
On peut également ajouter une image de fond. Pour ce faire, on va cette fois utiliser la propriété background-image. On lui donne en valeur l'url de l'image qui, en l'occurrence, prend la forme d'un chemin relatif comme nous avions vu pour les liens.
Tout d'abord, il va nous falloir une page HTML et une page CSS que nous allons lier entre elles et enregistrer dans le même dossier pour plus de simplicité.

```
body {
width :200px
height :100px
Background-image :url(''rachel.png'') ;
}
```

Position et répétition du fond

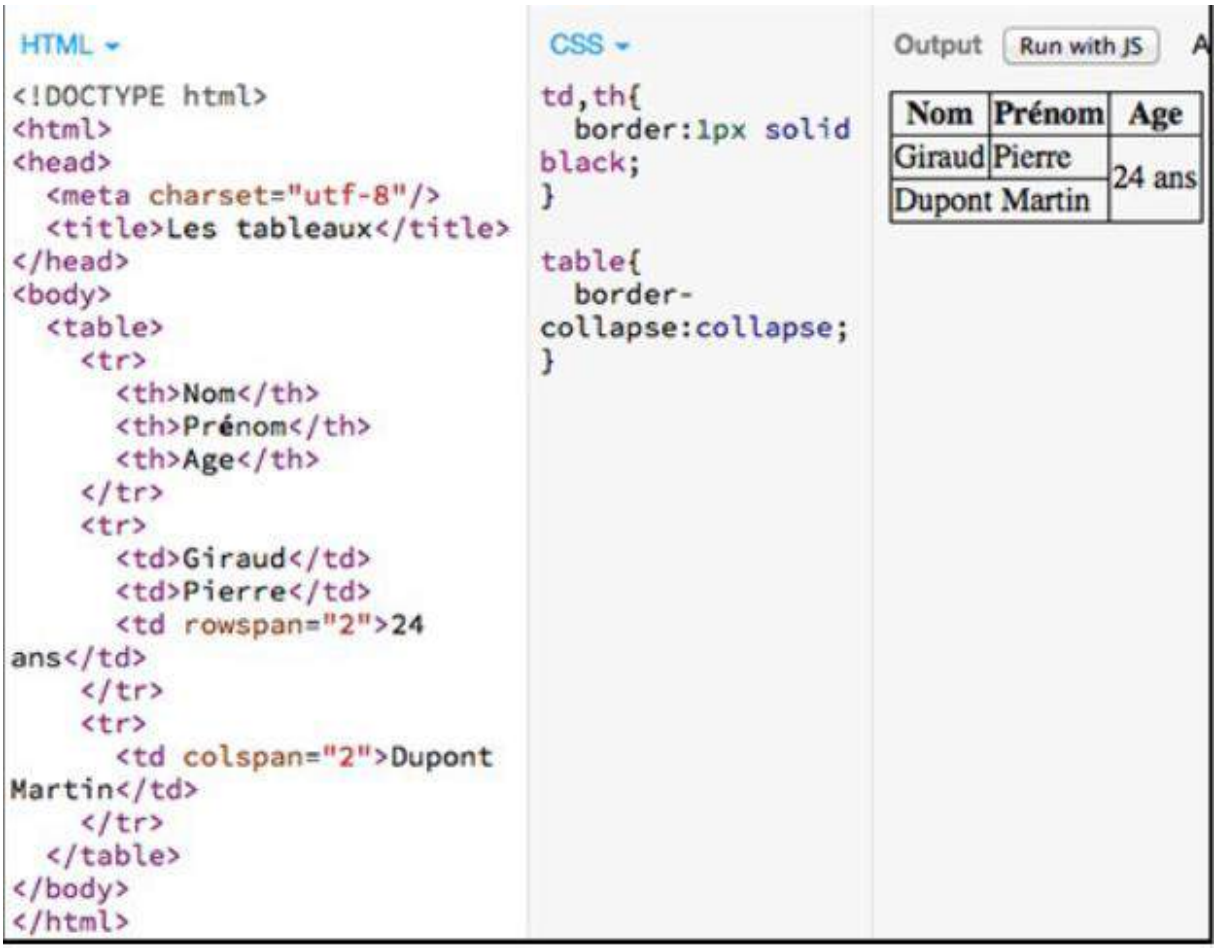
La propriété background-repeat nous permet de gérer la répétition de notre fond. Cette propriété accepte quatre valeurs :

- Repeat : le fond se répète horizontalement et verticalement, c'est le comportement par défaut ;
- Repeat-x : le fond ne se répète qu'horizontalement ;
- Repeat-y : le fond ne se répète que verticalement ;
- No-repeat : le fond ne se répète pas.

7. Table et Combiner des cellules

Pour combiner des cellules, on va utiliser les attributs HTML colspan et rowspan.

L'attribut colspan va nous permettre de combiner des cellules appartenant à différentes colonnes dans une même colonne tandis que l'attribut rowspan va nous permettre de combiner des cellules provenant de différentes lignes. Chacun de ces deux attributs accepte un nombre entier en valeur qui indique le nombre de cellules qui doivent être collées entre elles.



Ici, nous avons fusionné les cellules des deux colonnes « Nom » et « Prénom » en une pour « Dupont Martin » et les cellules de deux lignes pour « 24 ans ».

Les Formulaires

1. Introduction aux formulaires

Les formulaires sont certainement le moyen le plus simple et le plus utilisé pour recueillir des données à propos de vos utilisateurs. En cela, ils sont essentiels et incontournables. Cependant, nous touchons là aux limites du HTLM. En effet, si l'on peut créer des formulaires en HTML, en ne peut pas en revanche stocker ni utiliser les données du formulaire avec ce langage.

- Créer le squelette d'un formulaire

Pour créer notre formulaire, nous allons devoir tout d'abord utiliser l'élément form avec deux attributs : method et action. L'attribut action, tout d'abord, va servir à indiquer où les informations recueillies dans le formulaire doivent être envoyées pour être traitées. 'attribut method va lui spécifier de quelle manière on va envoyer ces données.

On peut choisir entre deux valeurs : GET et POST. En utilisant la valeur get, les données vont transiter via l'URL de la page ce qui ne sera pas le cas si l'on utilise la valeur post.

Il faut savoir que la valeur get est assez limitée par rapport à post. En effet, avec get, on est limité dans le nombre d'informations que l'on peut envoyer et surtout les informations sont visibles lors de l'envoi, ce qui est problématique si l'on envoie un mot de passe par exemple.

C'est pourquoi il vaut mieux utiliser la valeur post, qui ne possède pas ces inconvénients.

Créer un formulaire simple

Nous allons commencer par créer un formulaire très simple, demandant simplement un pseudo et un mot de passe à l'utilisateur. Pour capturer des données textuelles simples comme un pseudo par exemple, on utilise soit l'élément input (pour des textes courts), soit l'élément textarea (pour des textes longs).

Dans notre cas, nous allons utiliser l'élément input. Cet élément prend forcément un attribut « type ». La valeur de l'attribut type correspond au type de données demandées. On a le choix entre text, password, date, email, tel, number, time, color et url.

Toutes ces valeurs ont été créées pour des raisons de sémantique. A noter également que l'affichage par défaut de chaque champ de votre formulaire pourra être légèrement différent selon la valeur choisie pour l'attribut type. Il faut également préciser un deuxième attribut à l'élément input qui est l'attribut name. On lui donnera la valeur que l'on souhaite en essayant de rester cohérent. Cet attribut va nous être très utile pour traiter les données de notre formulaire.

A noter que l'élément input est représenté sous forme d'une balise orpheline comme aussi :

- Textearia
- Select
- Checkbox
- Iframe

1.4 NOTION DE BOOTSTRAP

Bootstrap est une collection d'outils utiles à la création du design (graphisme, animation et interactions avec la page dans le navigateur, etc.) de sites et d'applications web. C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript en option. C'est l'un des projets les plus populaires sur la plate-forme de gestion de développement

1. Le responsive design

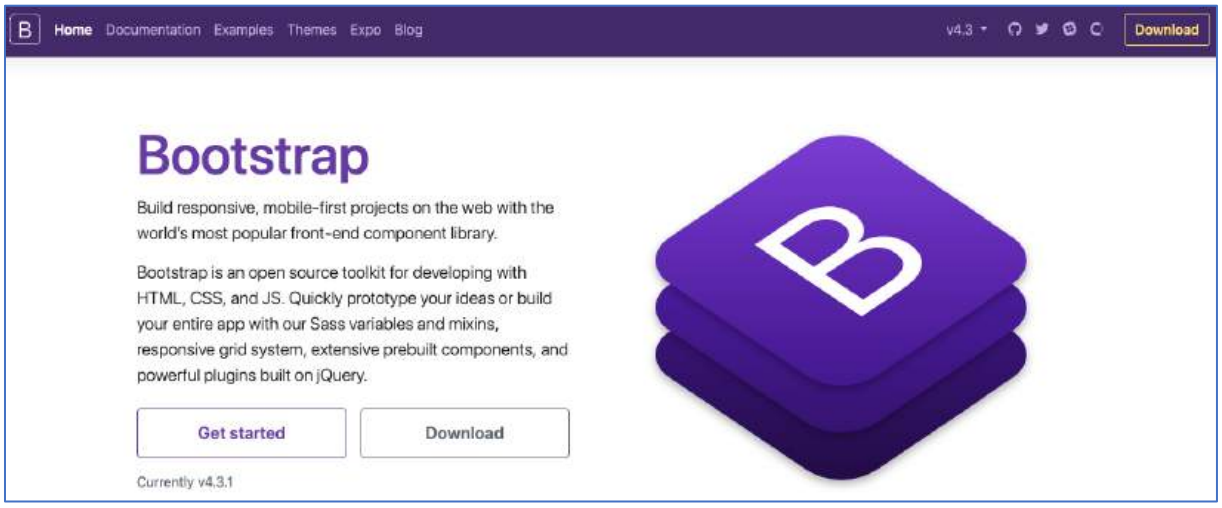
Lorsque l'on parle de responsive design, on parle généralement de l'ensemble des techniques nous permettant de créer un site qui pourra s'adapter en fonction de la taille de l'écran de vos visiteurs. Aujourd'hui, nous disposons de trois moyens pour créer un site pouvant s'adapter à différents terminaux :

1. Créer un site dédié pour chaque terminal différent (un site pour mobile, un site pour tablettes, un pour ordinateur, etc.)
2. Créer des applications mobiles natives (pour Android, iPhone, etc.)
3. Créer une version responsive de votre site.

Chaque méthode possède des avantages et des inconvénients, ainsi que des prix divers. Dans notre cas, nous allons nous intéresser à la dernière : la création d'une version responsive. Pour créer la version responsive de notre site, nous allons utiliser ce qu'on appelle Bootstrap. On va ainsi, grâce à Bootstrap, pouvoir modifier le style de chaque élément de notre site web afin de l'adapter à l'écran de vos visiteurs. La détection de la taille de l'écran se fait évidemment automatiquement.

Vous êtes un développeur frontal qui en a assez d'écrire sans cesse des syntaxes CSS ? C'est le moment de commencer à utiliser Bootstrap ! Cet article traitera des avantages de l'utilisation du framework web et de la manière de l'intégrer correctement dans votre projet.

2. C'est quoi Bootstrap ?



Bootstrap *est* une infrastructure de développement frontale, gratuite et open source pour la création de sites et d'applications Web.

Vous connaissez probablement les fonctionnalités des frameworks. Leur collection de syntaxes spécifiques aux tâches permet aux développeurs de créer des sites web beaucoup plus rapidement, car ils n'ont pas à se soucier des commandes et des fonctions de base. Malgré cela, il y a eu un manque de cohérence dû à l'utilisation intensive des bibliothèques qui a exigé un changement. Bootstrap a répondu à l'appel.

Le framework frontal à code source ouvert a été créé à l'origine par Mark Otto et Jacob Thornton pour accélérer et faciliter le développement de sites web frontaux. Il contient toutes sortes de modèles de conception basés sur HTML et CSS pour diverses fonctions et composants tels que la navigation, le système de grille, les carrousels d'images et les boutons.

Bien que Bootstrap fasse gagner du temps aux développeurs en leur évitant de devoir gérer les modèles de façon répétitive, son objectif premier est de créer des sites réactifs. Il permet à l'interface utilisateur d'un site web de fonctionner de manière optimale sur toutes les tailles d'écran, que ce soit sur des téléphones à petit écran ou des ordinateurs de bureau à grand écran.

Les développeurs n'ont donc pas besoin de créer des sites spécifiques à un appareil et de limiter leur public.

En raison de sa popularité, de plus en plus de communautés Bootstrap émergent. Elles constituent un lieu privilégié pour les développeurs et les concepteurs qui peuvent y échanger leurs connaissances et discuter des derniers correctifs du framework.

3. Les 3 fichiers primaires de Bootstrap

Comme **Bootstrap** est constitué d'une collection de syntaxes qui remplissent des fonctions spécifiques, il est logique que le framework contienne différents types de fichiers. Voici les trois principaux fichiers qui gèrent l'interface utilisateur et les fonctionnalités d'un site web.

Bootstrap.css

Bootstrap.css est un framework CSS qui organise et gère la mise en page d'un site web. Alors que le HTML gère le contenu et la structure d'une page web, le CSS s'occupe de la mise en page du site. Pour cette raison, les deux structures doivent coexister pour effectuer une action particulière.

Grâce à ses fonctions, le CSS vous permet de créer un aspect uniforme sur autant de pages web que vous le souhaitez. Dites adieu aux heures d'édition manuelle juste pour changer la largeur d'une bordure.

Avec le CSS, il suffit de renvoyer les pages web au fichier CSS. Toute modification nécessaire peut être effectuée dans ce seul fichier.

Les fonctions du CSS ne se limitent pas aux seuls styles de texte car elles peuvent être utilisées pour formater d'autres aspects de la page web tels que les tableaux et les mises en page d'images.

Bootstrap.js

Ce fichier est la partie centrale de Bootstrap. Il est constitué de fichiers **JavaScript** qui sont responsables de l'interactivité du site web.

Pour gagner du temps en évitant d'écrire de nombreuses fois des syntaxes JavaScript, les développeurs ont tendance à utiliser jQuery. Il s'agit d'une bibliothèque JavaScript multiplateforme à code source ouvert très répandue qui permet d'ajouter diverses fonctionnalités à un site web.

Voici quelques exemples de ce que jQuery peut faire :

- Effectuer des requêtes Ajax comme la soustraction dynamique de données d'un autre emplacement
- Créer des widgets à l'aide d'une collection de plugins JavaScript
- Créer des animations personnalisées en utilisant les propriétés CSS
- Dynamiser le contenu du site

Alors qu'un Bootstrap avec des propriétés CSS et des éléments HTML peut fonctionner parfaitement, il a besoin de jQuery pour créer un design réactif. Sinon, vous ne pouvez utiliser que les parties nues et statiques du CSS.

Glyphicons

Les icônes font partie intégrante de la partie frontale d'un site web. Elles sont souvent associées à certaines actions et données dans l'interface utilisateur. Bootstrap utilise des glyphes pour répondre à ce besoin.

Bootstrap comprend un jeu de Glyphicons Halflings qui a été autorisé à être utilisé gratuitement. La version gratuite a un aspect standard mais est adéquate pour les fonctions essentielles. Si vous souhaitez trouver des icônes plus élégantes, **Glyphicons** vend divers ensembles hauts de gamme qui auront sans aucun doute un meilleur aspect sur des sites web de niche. Vous pouvez également télécharger gratuitement des icônes individuelles et thématiques sur divers sites web tels que **Flaticon**, **GlyphSearch** et **Icons8**. Certaines icônes peuvent être modifiées par le CSS pour changer leur apparence, tandis que d'autres ont un aspect par défaut. Utilisez les icônes qui répondent le mieux aux besoins de votre site.

4. Comment utiliser Bootstrap

Pour avoir une meilleure idée de la façon d'utiliser bootstrap, jetez un coup d'œil à l'exemple ci-dessous

```
<html lang="en">

<head>
<meta charset="utf-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1"
/>
<title>Bootstrap 101 Template</title>
<link href="css/bootstrap.min.css" rel="stylesheet" />
</head>

<body>
<h1>Hello, world!</h1>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3
/jquery.min.js"></script>
<script src="js/bootstrap.min.js"></script>
</body>

</html>
```

Codage des caractères pour le document HTML. Dans ce cas, UTF-8 correspond à l'Unicode.

meta charset="utf-8"

Indique le jeu de caractères qui est utilisé pour écrire le site web.

meta http-equiv="X-UA-Compatible"

Détermine la version d'Internet Explorer qui doit rendre la page. En utilisant le mode Edge, il est réglé pour utiliser le mode le plus élevé disponible.

meta name="viewport"

Assure que la page a un rapport de 1 : 1 avec la taille de la fenêtre de visualisation.

link href="css/bootstrap.min.css" rel="stylesheet"

Ceci est la partie où nous ajoutons le CSS principal Bootstrap.

src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"

Charge jQuery via Google CDN. Il est préférable de le charger depuis le CDN via HTTP car les fichiers peuvent être mis en cache pour un an.

src="js/bootstrap.min.js

Ajoute le JavaScript principal de Bootstrap. Cette syntaxe doit toujours être inférieure à la syntaxe jQuery pour fonctionner correctement. Le processus d'ajout peut être effectué via l'URL de **Google** ou par téléchargement manuel. Commencez avec Bootstrap, le framework le plus populaire au monde pour créer des sites réactifs et mobiles, avec jsDelivr et un modèle de page de démarrage.

Veuillez consulter les sites suivants :

https://getbootstrap.com/docs/3.3/components/#navbar

https://www.w3bai.com/fr/bootstrap/default.html#gsc.tab=0

Composants

Plus d'une douzaine de composants réutilisables conçus pour fournir une iconographie, des listes déroulantes, des groupes de saisie, une navigation, des alertes et bien plus encore.

Remplacez la marque de la barre de navigation par votre propre image en remplaçant le texte par un <img>. Étant donné que la marque .navbar a son propre rembourrage et sa propre hauteur, vous devrez peut-être remplacer certains CSS en fonction de votre image.

```
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <div class="navbar-header">
      <a class="navbar-brand" href="#">
        <img alt="Brand" src="...">
      </a>
    </div>
  </div>
</nav>
```

Bootstrap comprend un premier système de grille fluide réactif et mobile qui s'adapte de manière appropriée jusqu'à 12 colonnes à mesure que la taille de l'appareil ou de la fenêtre d'affichage augmente. Il comprend des classes prédéfinies pour des options de mise en page faciles, ainsi que de puissants mixins pour générer des mises en page plus sémantiques.

Introduction

Les systèmes de grille sont utilisés pour créer des mises en page à travers une série de lignes et de colonnes qui hébergent votre contenu. Voici comment fonctionne le système de grille Bootstrap :

Les lignes doivent être placées dans un .container (largeur fixe) ou .container-fluid (pleine largeur) pour un alignement et un remplissage corrects.

Utilisez des rangées pour créer des groupes horizontaux de colonnes.

Le contenu doit être placé dans des colonnes, et seules les colonnes peuvent être des enfants immédiats des lignes.

Des classes de grille prédéfinies telles que .row et .col-xs-4 sont disponibles pour créer rapidement des dispositions de grille. Moins de mixins peuvent également être utilisés pour des mises en page plus sémantiques.

Les colonnes créent des gouttières (espaces entre le contenu des colonnes) via le rembourrage. Ce rembourrage est décalé en lignes pour la première et la dernière colonne via une marge négative sur .rows.

La marge négative est la raison pour laquelle les exemples ci-dessous sont en retrait. C'est ainsi que le contenu des colonnes de la grille est aligné avec le contenu hors grille.

Les colonnes de la grille sont créées en spécifiant le nombre de douze colonnes disponibles que vous souhaitez couvrir. Par exemple, trois colonnes égales utiliseraient trois .col-xs-4.

Si plus de 12 colonnes sont placées dans une seule ligne, chaque groupe de colonnes supplémentaires sera, comme une unité, enveloppé sur une nouvelle ligne.

Les classes de grille s'appliquent aux appareils dont la largeur d'écran est supérieure ou égale aux tailles des points d'arrêt et remplacent les classes de grille ciblées sur les appareils plus petits. Par conséquent, par ex. l'application d'une classe .col-md-* à un élément affectera non seulement son style sur les appareils moyens, mais également sur les grands appareils si une classe .col-lg-* n'est pas présente.

Exemple de base

Les contrôles de formulaire individuels reçoivent automatiquement un style global. Tous les éléments textuels <input>, <textarea> et <select> avec .form-control sont définis sur width : 100 % ; par défaut. Enveloppez les étiquettes et les contrôles dans .form-group pour un espacement optimal.

```
<form>
  <div class="form-group">
    <label for="exampleInputEmail1">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1"
placeholder="Email">
  </div>
  <div class="form-group">
    <label for="exampleInputPassword1">Password</label>
```

```
    <input type="password" class="form-control" id="exampleInputPassword1"
placeholder="Password">
  </div>
  <div class="form-group">
    <label for="exampleInputFile">File input</label>
    <input type="file" id="exampleInputFile">
    <p class="help-block">Example block-level help text here.</p>
  </div>
  <div class="checkbox">
    <label>
      <input type="checkbox"> Check me out
    </label>
  </div>
  <button type="submit" class="btn btn-default">Submit</button>
</form>
```

Choix

Utilisez l'une des classes de boutons disponibles pour créer rapidement un bouton stylé.

```
<!-- Standard button -->
<button type="button" class="btn btn-default">Default</button>

<!-- Provides extra visual weight and identifies the primary action in a set of buttons --
>
<button type="button" class="btn btn-primary">Primary</button>

<!-- Indicates a successful or positive action -->
<button type="button" class="btn btn-success">Success</button>

<!-- Contextual button for informational alert messages -->
<button type="button" class="btn btn-info">Info</button>

<!-- Indicates caution should be taken with this action -->
<button type="button" class="btn btn-warning">Warning</button>

<!-- Indicates a dangerous or potentially negative action -->
<button type="button" class="btn btn-danger">Danger</button>

<!-- Deemphasize a button by making it look like a link while maintaining button
behavior -->
<button type="button" class="btn btn-link">Link</button>
```

Il faut noter, avant de bien utiliser Bootstrap, veuillez exploiter sa documentation pour plus de faciliter. Mais retenait que le framework proposer de classe préparer pour une utilisation rapide et facile.

Exemple :

<button class="form-control">Valider </button>

Ici nous avons utilisé une balise bouton avec comme classe Bootstrap Form-control qui permet la mise en forme du bouton.

SECTION 2 : DEVELOPPEMENT DU BACKEND

1. Qu'est-ce qu'un backend ?

Le backend est la partie d'un logiciel que les utilisateurs ne peuvent pas voir ou avec laquelle ils ne peuvent pas interagir et qui contient toutes les fonctionnalités.

Le back-end est la partie d'une application qui exécute les différentes tâches pour lesquelles l'application est conçue. C'est le lieu où a lieu le travail administratif qui garantit que tout se déroule sans accroc. Le backend stocke les données et les codes qui interprètent les syntaxes du programme. C'est le moteur qui traite les demandes d'information et délivre les résultats aux terminaux clients. Le backend est géré par l'administrateur et inaccessible à l'utilisateur de l'application.

2.1. NOTION SUR RAZOR

Il vrai que l'utilisation de langages html, css et du framework Bootstrap nous aident dans l'organisation de la structure des pages web (interfaces de l'application) mais en asp.net une nuance veut que ces langages cités ci-haut soit manipuler avec la syntaxe razor.

Razor est une **syntaxe utilisée pour créer des pages web dynamiques avec les langages C# ou Visual Basic.NET**, qui a été intégrée à Visual Studio en 2011. Razor propose une structure simple de génération de vue et a été intégrée dans ASP.NET MVC 3 et le jeu d'outils WebMatrix

Razor est une syntaxe de programmation ASP.NET utilisée pour créer des pages Web dynamiques avec les langages de programmation C# ou VB.NET. Razor était en développement en juin 2010 et a été publié pour Microsoft Visual Studio 2010 en janvier 2011.

Principales Razor Règles de syntaxe pour C #

- Razor blocs de code sont enfermés dans @ {...}
- Expressions inline (variables and functions) commencer par @
- Les instructions de code se terminent par virgule
- Les variables sont déclarées avec le mot-clé var
- Les chaînes sont placées entre guillemets
- Code C # est sensible à la casse
- fichiers C # ont l'extension .cshtml

```
<!-- Single statement block -->
@{ var myMessage = "Hello World"; }

<!-- Inline expression or variable -->
<p>The value of myMessage is: @myMessage </p>
```

```
<!-- Multi-statement block ou Bloc multi-instructions-->
@{
var greeting = "Welcome to our site!";
var weekDay = DateTime.Now.DayOfWeek;
var greetingMessage = greeting + " Today is: " + weekDay;
}
<p>The greeting is: @greetingMessage </p>
```

Dans cet exemple : nous avons déclaré une variable greeting puis affecter une valeur string "Welcome to our site!"; ainsi que deux variables weekDay et greetingMessage. Dans weekDay nous avons affecté la date et greetingMessage récupère les valeurs concaténées « Salutation, Aujourd'hui est le jour de la semaine ».

Pour afficher, il faut appeler la variable en utilisation le signe @ avant le nom de la variable comme : @greetingMessage

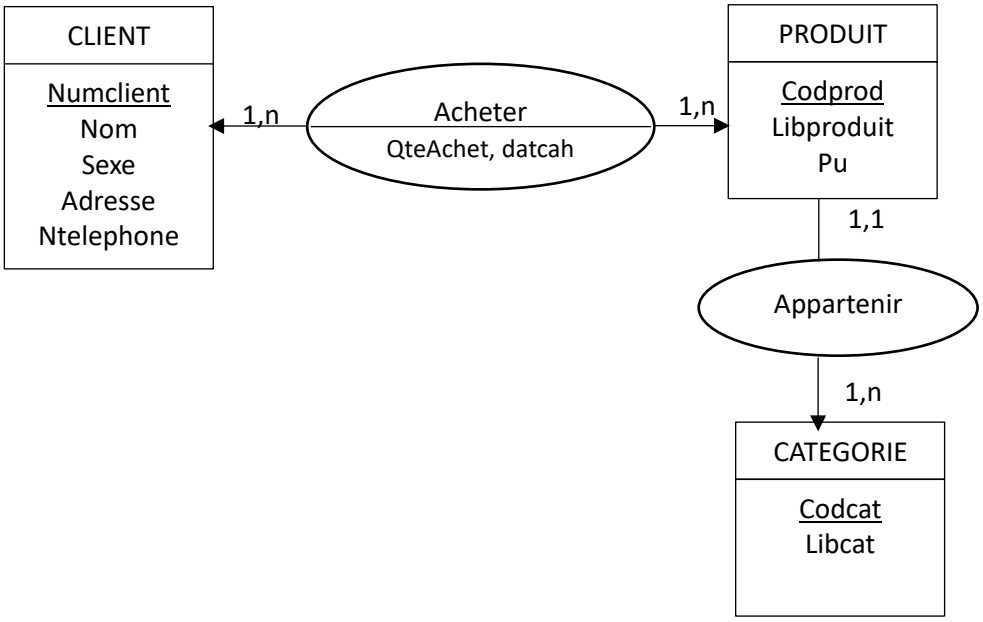
Il faut noter, la maitrise de langage C# ou de Vb facilite l'utilisation de razor mais en respectant la syntaxe.

2.2. IMPLEMENTATION DU PROJET MVC

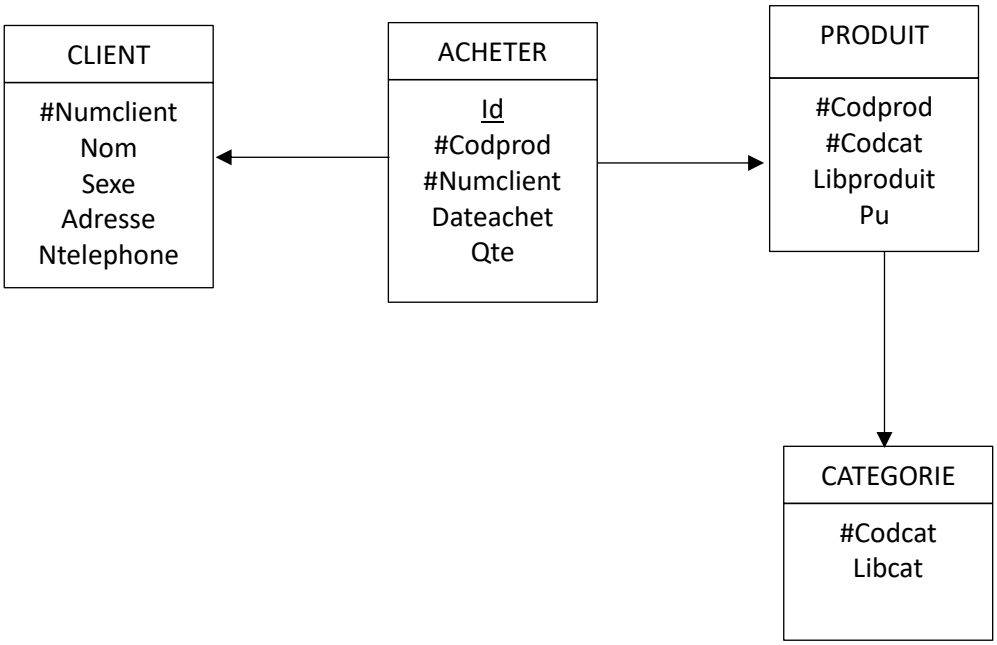
Pour concevoir une application web avec asp.net MVC, il faut disposer d'une structure de données qui vous servira de base de données. Pour notre cas nous allons créer un modèle conceptuel de données avec 3 entités :

- Client
- Produit
- Catégorie

a. Présentation du schéma



b. Passage du MCD au MLD



Nous avons maintenant notre modèle logique de données, notre cas nous allons directement créer le schéma relationnel. Pour les amours de Merise, il faut respecter les étapes par exemples produire un MLDV.

c. Schéma relationnel

Présentation des Tables, tellement que nous allons utiliser SGBD SQL SERVEUR

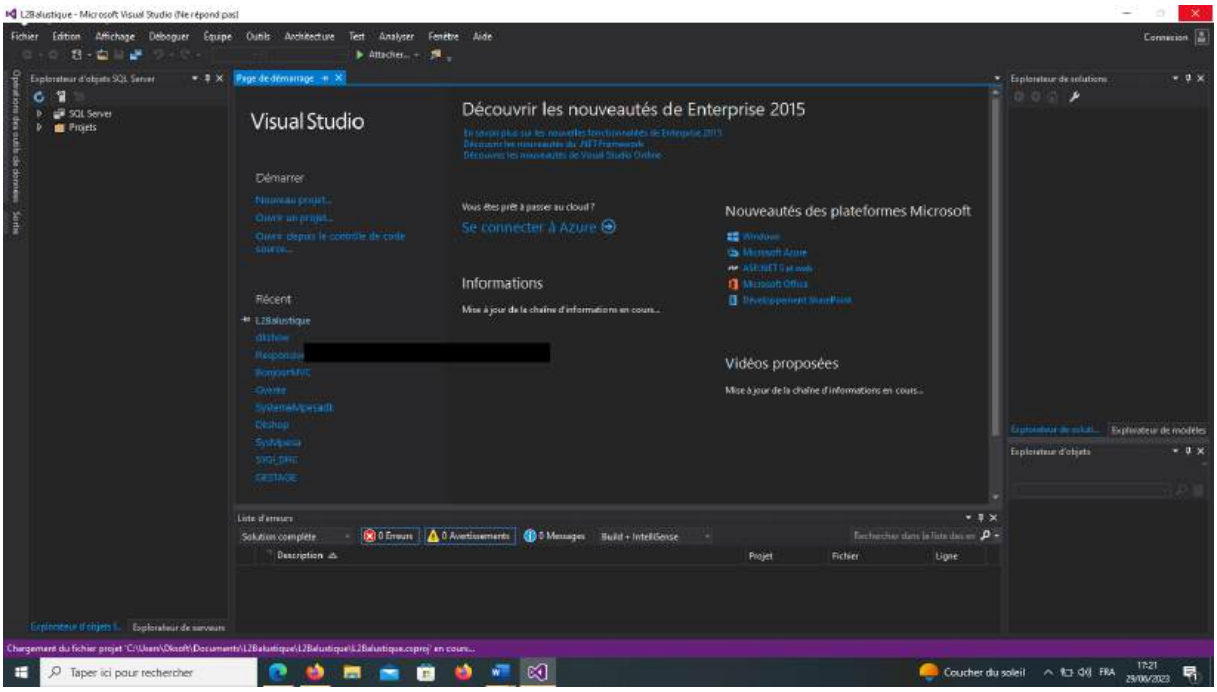
Nom de la table	Attributs
CATEGORIE	#Codcat varcher(5), Libcat varchar(30)
PRODUIT	#Codprod varchar(5), #Codcat varcher(5), Libprod varchar(30), pu numérique,
CLIENT	#Numclient int, Nom varchar(30), Sexe varchar(1), Adresse varchar(100), Numtelephone varchar(15)
ACHETER	#Id int, #Codprod varchar(5),#Numclient int, Dateachet date, Qte int

C'est partant de ce schéma que nous allons créer notre base de données. Pour l'environnement de SGBD, sa dépendra de caractéristique de votre machine, si vous avez un PC performant avec SQL serveur installé pas de problème utiliser votre SQL pour créer la base de données mais sinon veuillez utiliser le SQL express de Visual studio que nous allons exploiter dans le cadre de notre support.

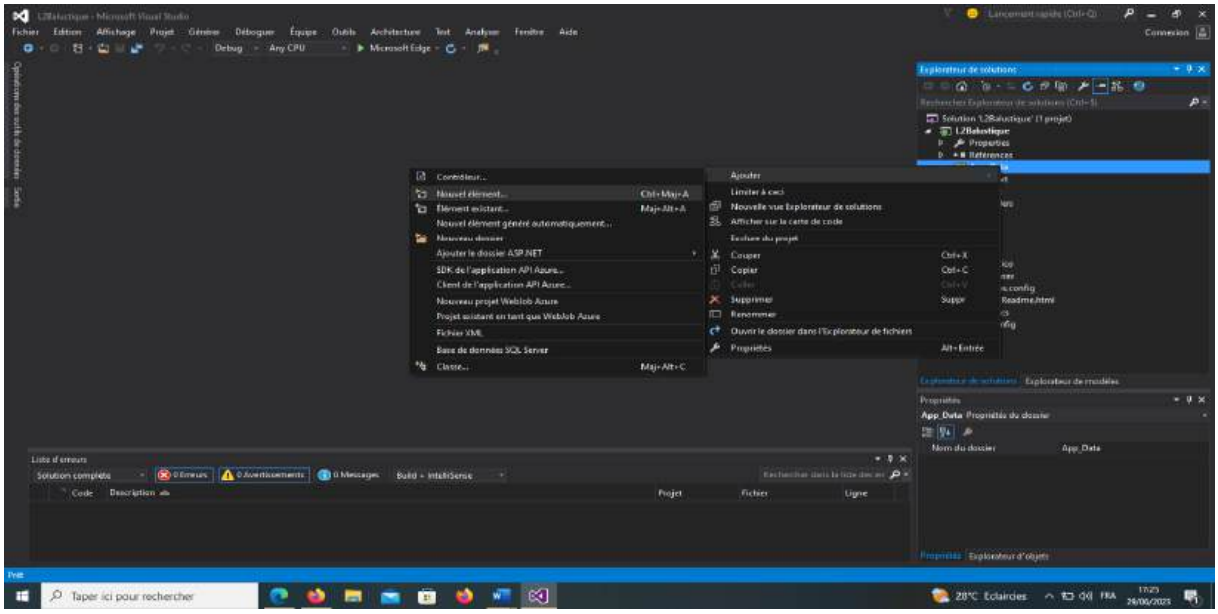
2.2.1. Création de la base de données avec SQL Express

Procédure :

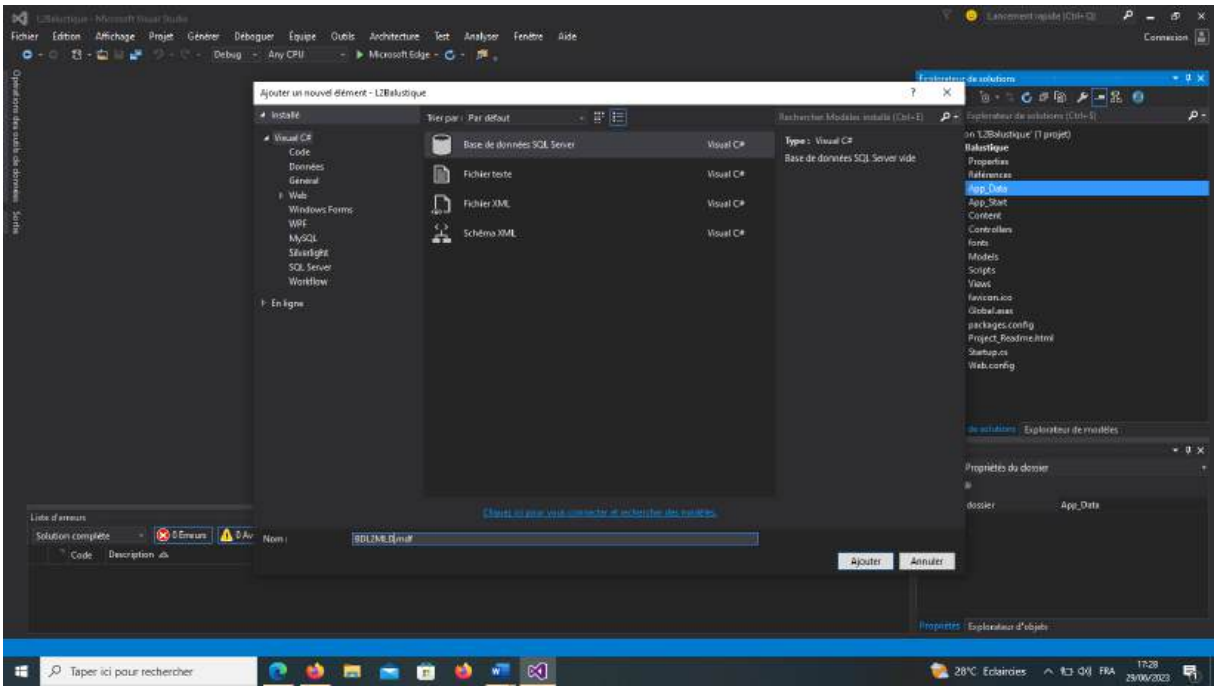
- Lancer le projet déjà créer et cliquer sur nom du projet dans récent, sinon il faut créer le projet d'abord en suivant la procédure démontrée ci -haut :



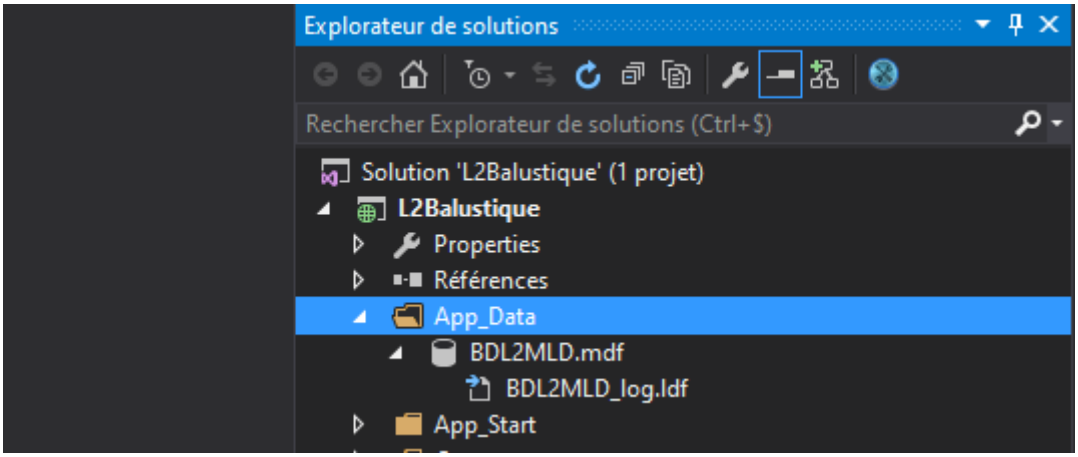
- Dans le dossier App_DATA cliquer droit sur ajouter/Nouvel élément



- Sélectionnez l'élément Base de données SQL Server qui créer un fichier dans le dossier App_data avec l'extension .mdf, puis saisie nom de la base de données ex : BDL2MLD.mdf et cliquer sur ajouter.



Apres cette opération vous allez constater que le dossier App_data contiendra deux fichiers mdf et log.

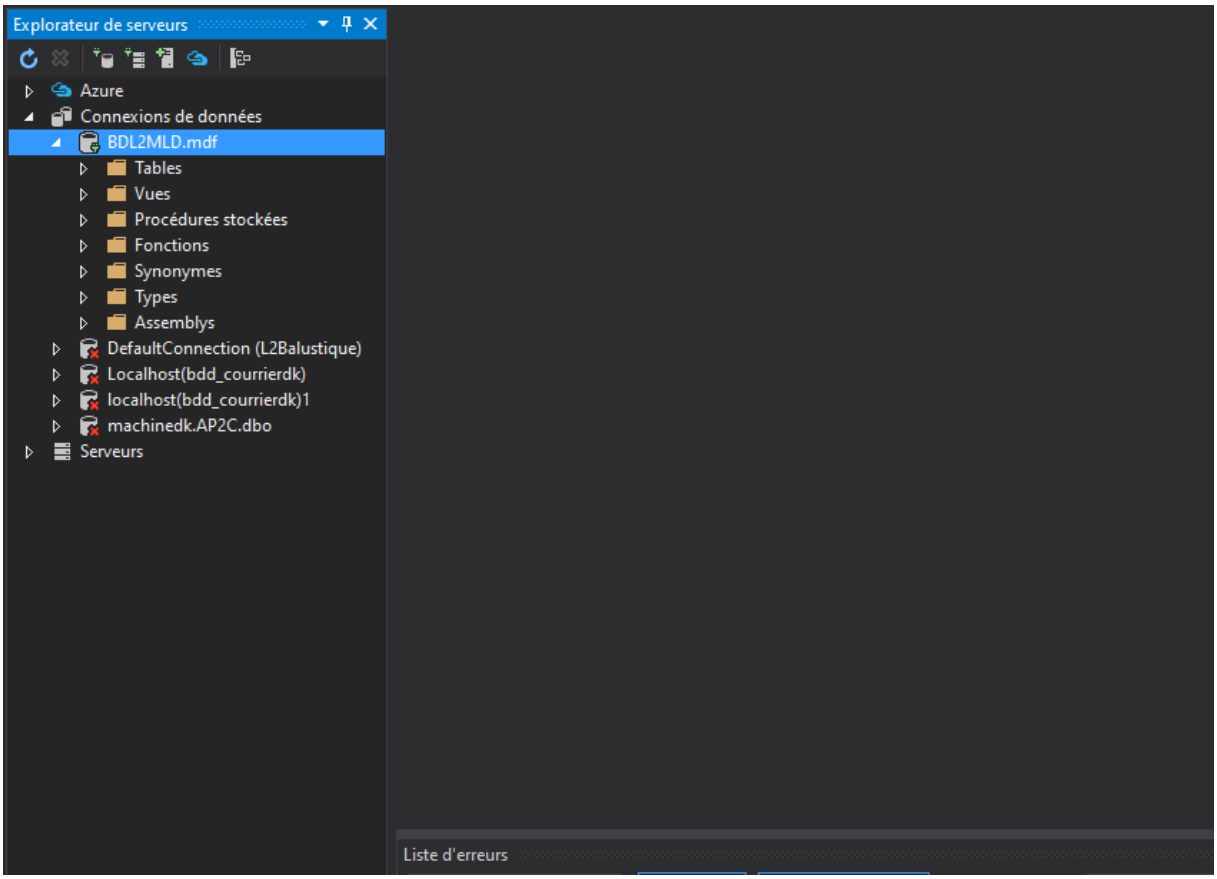


2.2.2. Création des tables et relations

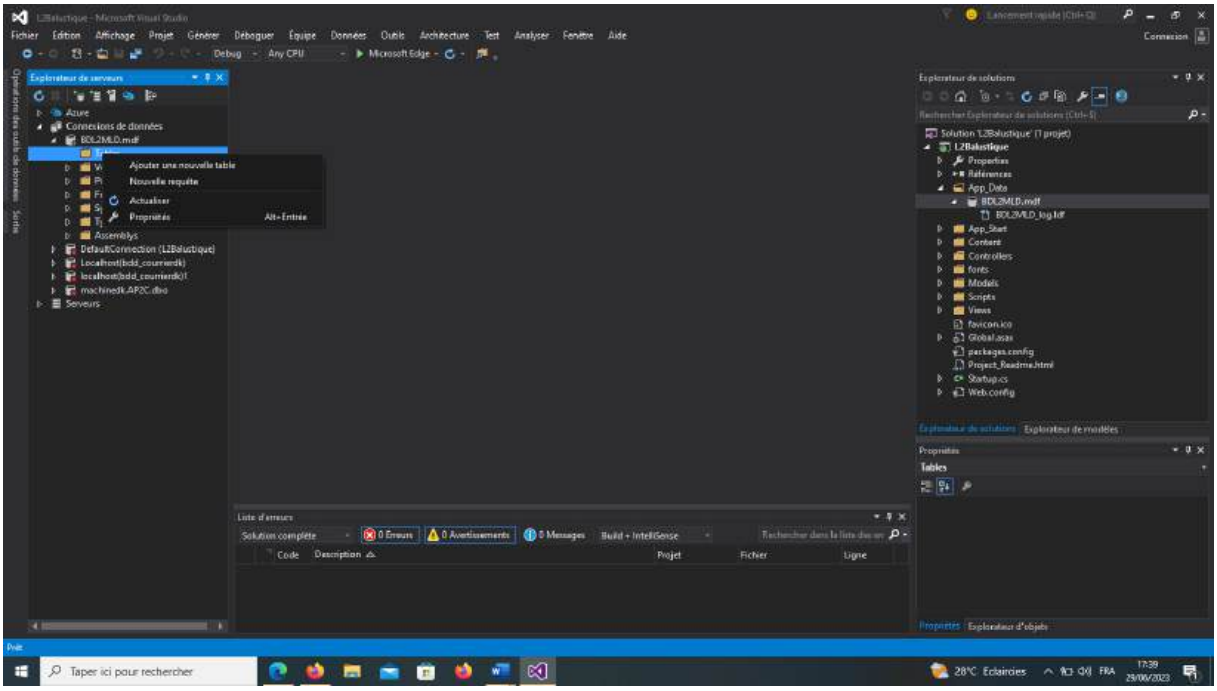
A. CREATION DES TABLES

Procédure :

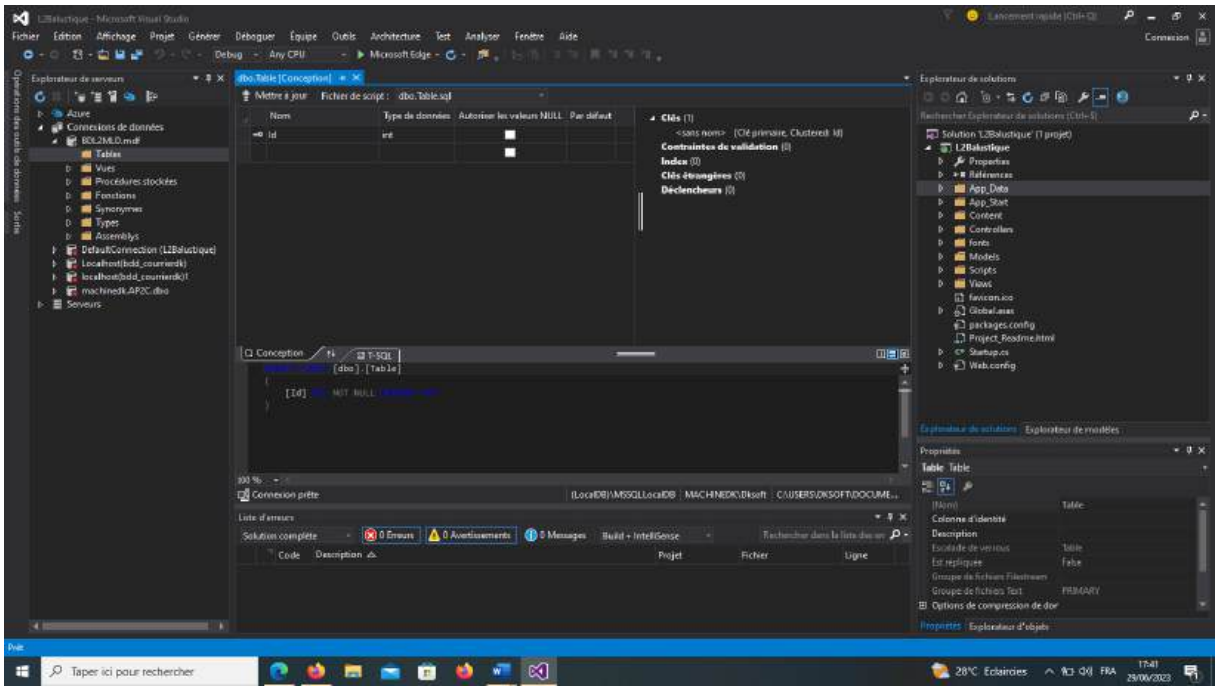
- Double cliquer sur le nom de votre base de données puis à gauche vous aurez ceci :



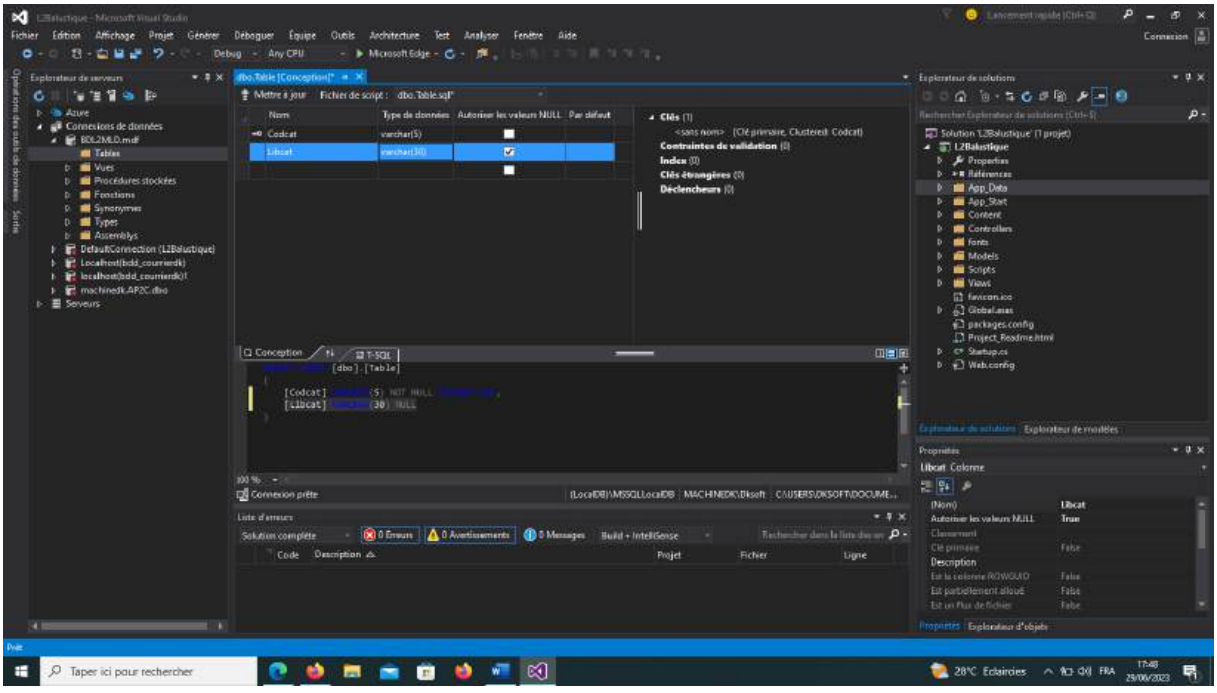
- Il faut maintenant créer les tables en cliquant droit sur le nom de la table puis sur Ajouter une nouvelle table



- Une fenêtre ci-après va s'afficher :

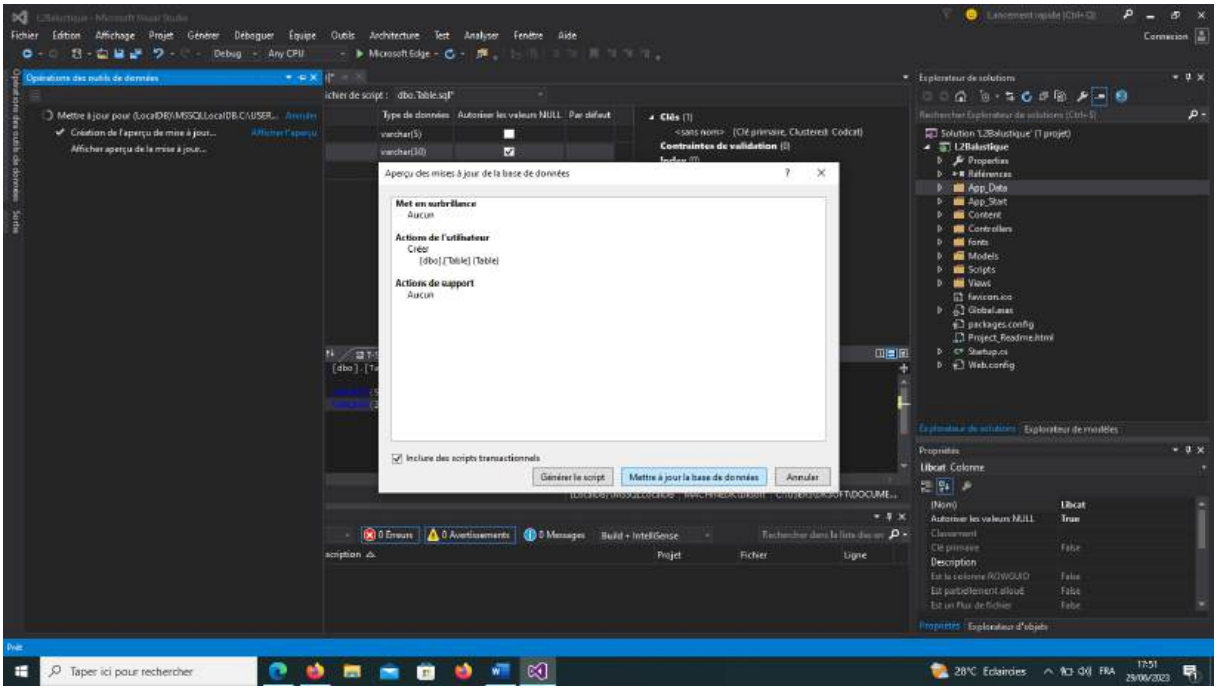


- La fenêtre est constituée de deux parties : coté graphique et TSQL (Transact-SQL) est un ensemble d'extensions de programmation de Sybase et Microsoft qui ajoutent plusieurs fonctionnalités au langage de requête structuré (SQL), notamment le contrôle des transactions, la gestion des exceptions et des erreurs, le traitement des lignes et les variables déclarées.
- Utilisons maintenant notre schéma relationnel pour créer les tables moi je commence par table catégorie, client, produit et à la fin table relationnelle acheter.

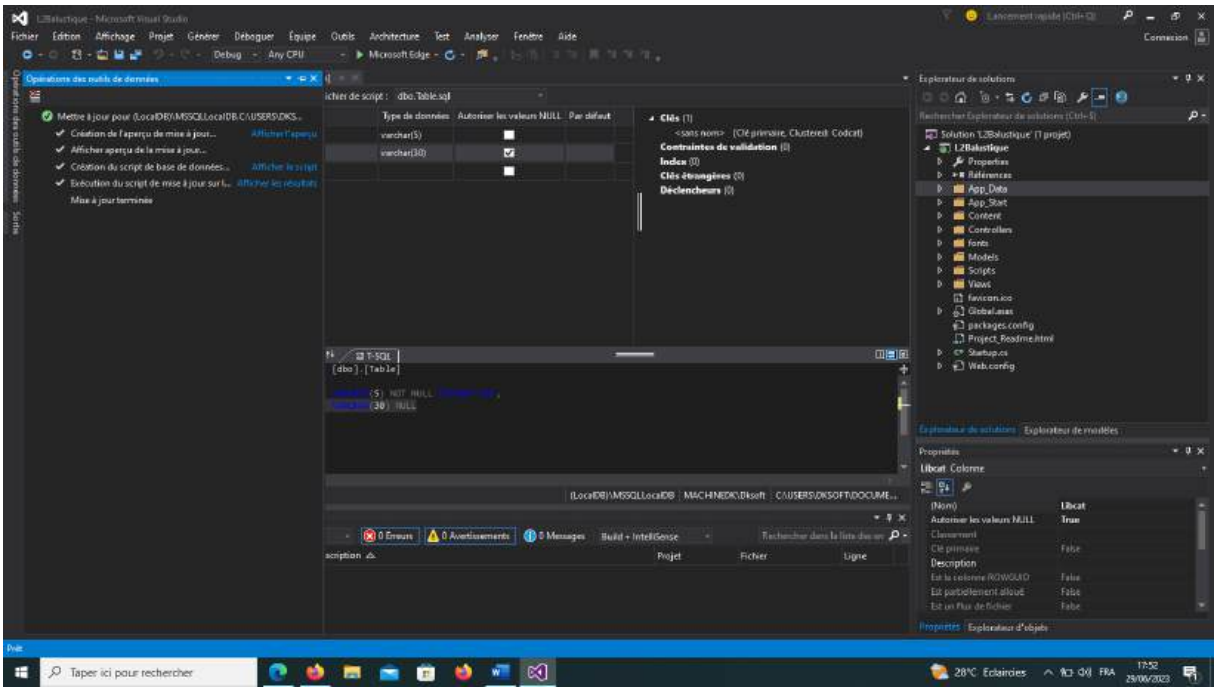


- Pour enregistrer la table cliquer sur Mettre à jour la base de données /ok

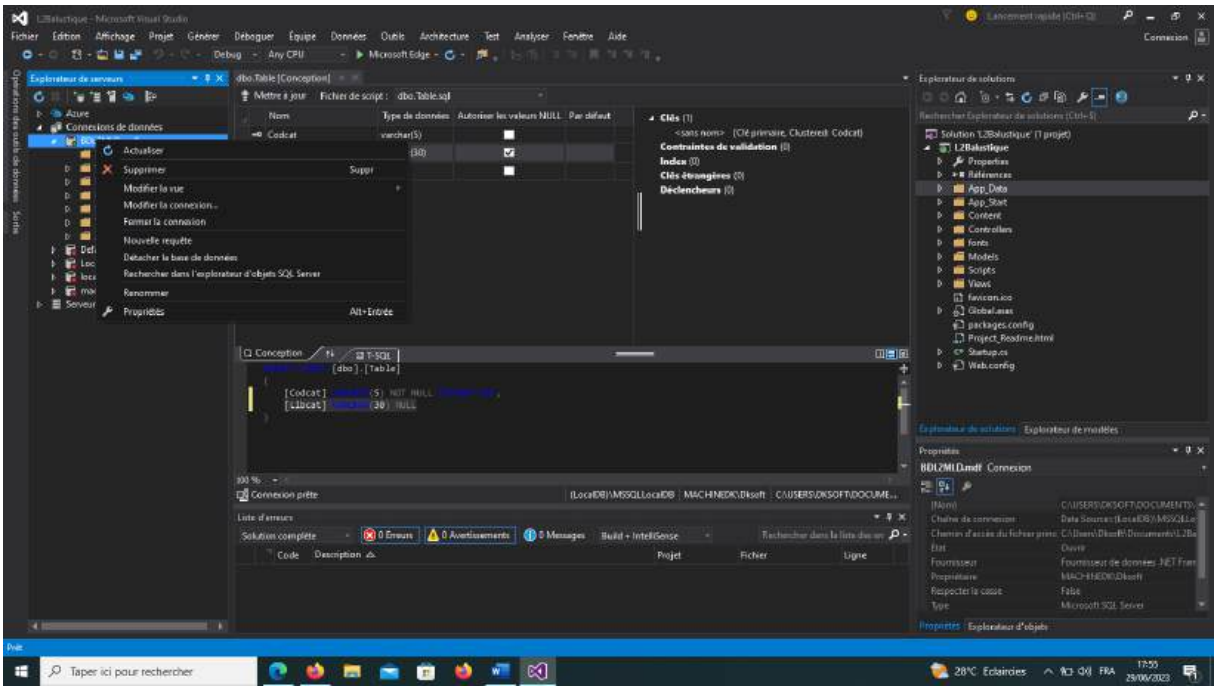
- Vous pouvez aussi créer la table en mode TSQL si vous êtes fort



- Cliquer sur mettre à jour la base de données mais avant ça cliquer sur le code Tsql renommée la table par exemple : Catégorie.



- Pour voir la table créée cliquer sur opération des outils de données
- Puis cliquer droit sur le nom de la table et gauche sur actualiser

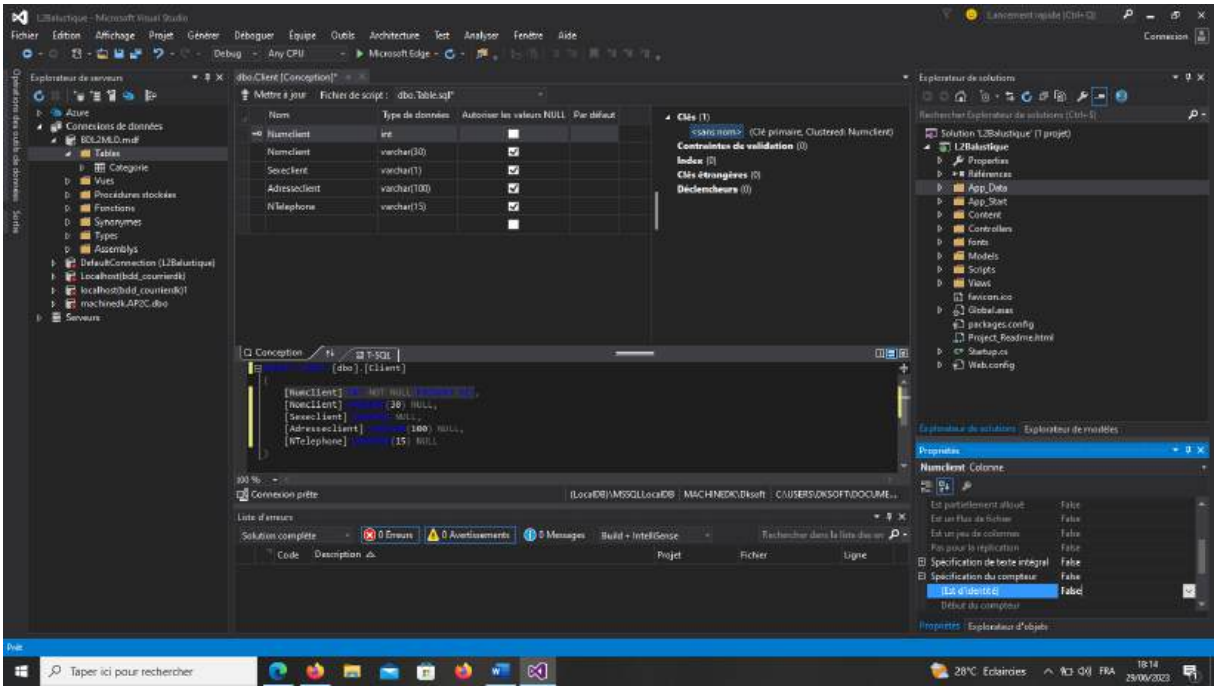


NB : S'il y a erreur de la création de la table, il faut savoir qu'on peut modifier la structure ou supprimer.

- Pour modifier double sur le nom de table ;
- Pour supprimer cliquer droit sur le nom de table puis sur supprimer.

- Le cas de la Table contenant le champ avec identifiant incrémentale

- Il faut sélectionner le nom du champ par exemple numéro client
- Aller dans propriété puis chercher spécification du compteur mets « true »



Ou soit faite le code suivant :

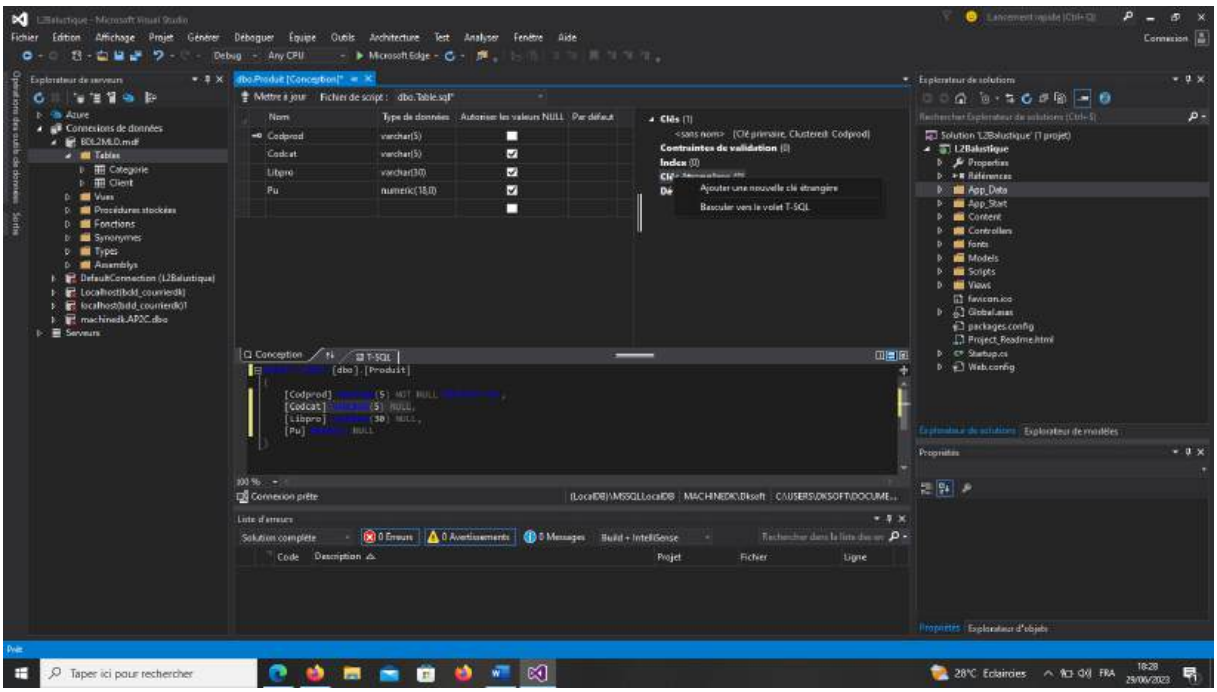
```
CREATE TABLE [dbo].[Client]
(
	[Numclient] INT NOT NULL PRIMARY KEY IDENTITY,
    [Nomclient] VARCHAR(30) NULL,
    [Sexeclient] VARCHAR NULL,
    [Adresseclient] VARCHAR(100) NULL,
    [NTelephone] VARCHAR(15) NULL
)
```

a. Création de relation entre table

Dans le cas où la table contient une clé estrangère, il faut cliquer droit là où écrit clés étrangères, puis gauche sur ajouter une nouvelle clé étrangère puis faites entrer. Une ligne d'instruction s'ajouter dans la partie TSQL comme suit

CONSTRAINT [FK_Produit_ToTable] FOREIGN KEY ([Column]) REFERENCES [ToTable]([ToTableColumn]),

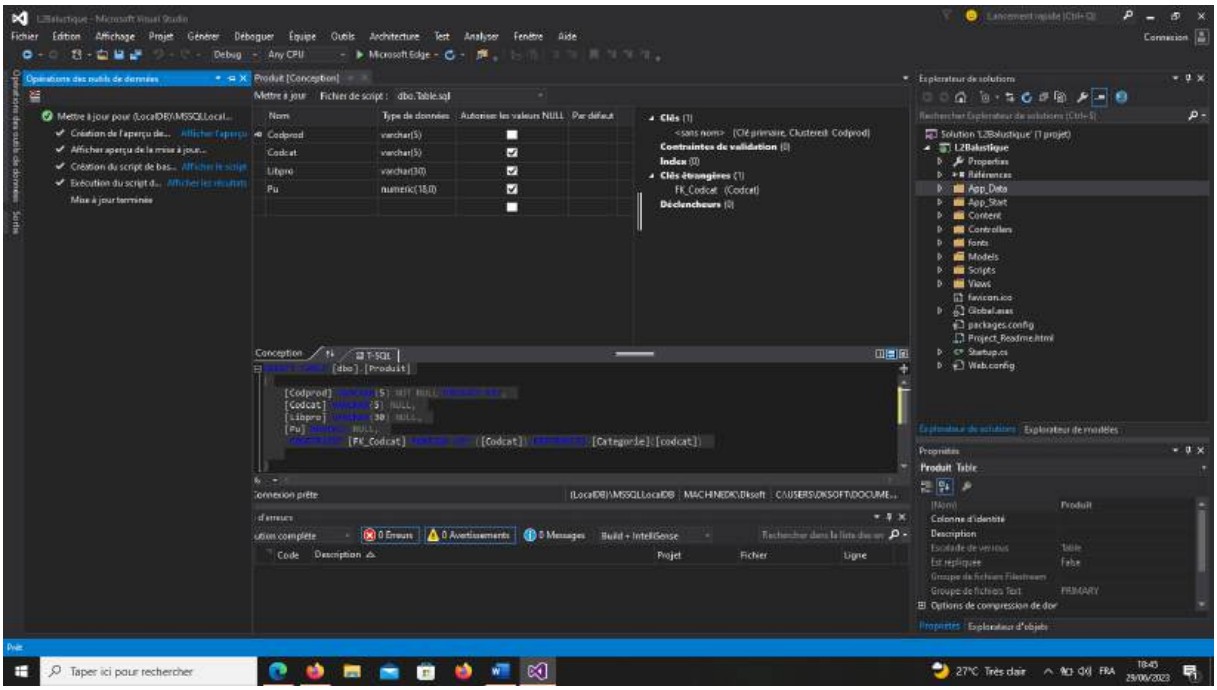
- Fk_Produit_toTable doit devenir fk_codcat pour représenter le champ de la table produit
- FOREIGN KEY ([Column]) : remplacer seulement column par codcat en se rassurant que la codcat est bien ecrit comme dans la table catégorie
- REFERENCES [ToTable]([ToTableColumn]) : remplacer Totable par Catégorie et toTablecolumn par le nom du codcat.



```
CREATE TABLE [dbo].[Produit]
(
	[Codprod] VARCHAR(5) NOT NULL PRIMARY KEY,
    [Codcat] VARCHAR(5) NULL,
    [Libpro] VARCHAR(30) NULL,
```

```
    [Pu] NUMERIC NULL,
     CONSTRAINT [FK_Codcat] FOREIGN KEY ([Codcat]) REFERENCES
[Categorie]([codcat])

)
```

- Après cette opération cliquer sur mettre à jour

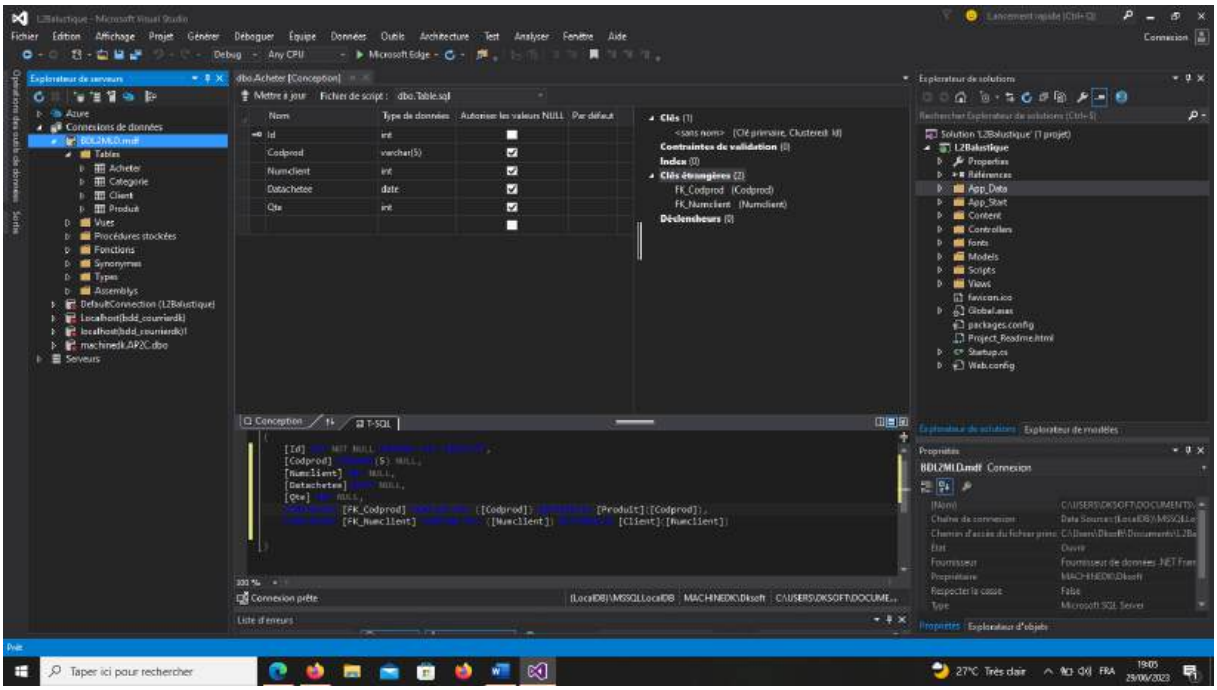


Pour terminer, nous allons créer la table acheter qui est une table relationnelle, la logique reste la même avec la table produit mais ici nous allons ajouter un champ ID qui sera auto incrément pour faciliter les opérations de Suppression et modification même de rechercher.

Nous allons ajouter deux clés étrangères dans la table acheter mais le nombre dépendra des rubriques de la table relationnelle.

NB : Avant de créer une table, il faut d'abord commencer par renommer la table, par défaut vous auriez create table [dbo].[Table] mais vous changer la table par un nom exemple acheter, vous allez constater qu'il a plus d'erreur.

```
CREATE TABLE [dbo].[Table]
(
        [Id] INT NOT NULL PRIMARY KEY
)
```



La creation de table est terminée. Un conseil chaque fois que vous auriez besoin de rentrer dans la base de données, il faut doubli cliquer sur le nom de la base de données qui se trouve dans le dossier App_Data. Le fichier de votre base de données se trouve dans le projet directement. Mais pour c'est qui vont utiliser sqlserveur il faut créer la base de données dans le dossier du projet.

2.2.2. Création du modèle entity framework

A. Vue d'ensemble de l'Entity Framework

Entity Framework est un ensemble de technologies dans ADO.NET qui prend en charge le développement d'applications logicielles orientées données. Les architectes et les développeurs d'applications orientées données ont eu du mal à atteindre deux objectifs très différents. Ils doivent modéliser les entités, les relations et la logique des problèmes métier qu'ils résolvent, et ils doivent également travailler avec les moteurs de données utilisés pour stocker et récupérer les données. Les données peuvent couvrir plusieurs systèmes de stockage, chacun avec ses propres protocoles ; même les applications qui fonctionnent avec un seul système de stockage doivent équilibrer les exigences du système de stockage avec les exigences d'écriture d'un code d'application efficace et maintenable.

Entity Framework permet aux développeurs de travailler avec des données sous la forme d'objets et de propriétés spécifiques à un domaine, tels que des clients et des adresses de clients, sans avoir à se préoccuper des tables et des colonnes de base de données sous-jacentes où ces données sont stockées. Avec Entity Framework, les développeurs peuvent travailler à un niveau d'abstraction plus élevé lorsqu'ils traitent des données, et peuvent créer et maintenir des applications orientées données avec moins de code que dans les applications traditionnelles. Étant donné que Entity Framework est un composant du .NET Framework, les applications Entity Framework

peuvent s'exécuter sur n'importe quel ordinateur sur lequel le .NET Framework à partir de la version 3.5 SP1 est installé.

1. Donner vie aux modèles

Une approche de conception courante et de longue date lors de la création d'une application ou d'un service consiste à diviser l'application ou le service en trois parties : un modèle de domaine, un modèle logique et un modèle physique. Le modèle de domaine définit les entités et les relations dans le système en cours de modélisation. Le modèle logique d'une base de données relationnelle normalise les entités et les relations dans des tables avec des contraintes de clé étrangère. Le modèle physique traite les capacités d'un moteur de données particulier en spécifiant les détails de stockage tels que le partitionnement et l'indexation.

Le modèle physique est affiné par les administrateurs de base de données pour améliorer les performances, mais les programmeurs écrivant du code d'application se limitent principalement à travailler avec le modèle logique en écrivant des requêtes SQL et en appelant des procédures stockées. Les modèles de domaine sont généralement utilisés comme un outil pour capturer et communiquer les exigences d'une application, souvent sous forme de diagrammes inertes qui sont visualisés et discutés dans les premières étapes d'un projet, puis abandonnés. De nombreuses équipes de développement ignorent la création d'un modèle conceptuel et commencent par spécifier des tables, des colonnes et des clés dans une base de données relationnelle.

Entity Framework donne vie aux modèles en permettant aux développeurs d'interroger des entités et des relations dans le modèle de domaine (appelé modèle conceptuel dans Entity Framework) tout en s'appuyant sur Entity Framework pour traduire ces opérations en commandes spécifiques à la source de données. Cela libère les applications des dépendances codées en dur sur une source de données particulière.

Lorsque vous travaillez avec Code First, le modèle conceptuel est mappé au modèle de stockage dans le code. Entity Framework peut déduire le modèle conceptuel en fonction des types d'objets et des configurations supplémentaires que vous définissez. Les métadonnées de mappage sont générées pendant l'exécution en fonction d'une combinaison de la façon dont vous avez défini vos types de domaine et des informations de configuration supplémentaires que vous fournissez dans le code. Entity Framework génère la base de données selon les besoins en fonction des métadonnées. Pour plus d'informations, voir Création d'un modèle.

Lorsque vous travaillez avec Entity Data Model Tools, le modèle conceptuel, le modèle de stockage et les mappages entre les deux sont exprimés dans des schémas XML et définis dans des fichiers qui ont des extensions de nom correspondantes :

Le langage de définition de schéma conceptuel (CSDL) définit le modèle conceptuel. CSDL est l'implémentation d'Entity Framework du modèle de données d'entité. L'extension de fichier est .csdl.

Le langage de définition de schéma de magasin (SSDL) définit le modèle de stockage, également appelé modèle logique. L'extension de fichier est .ssdl.

Le langage de spécification de mappage (MSL) définit les mappages entre le stockage et les modèles conceptuels. L'extension de fichier est .msl.

Le modèle de stockage et les mappages peuvent changer selon les besoins sans nécessiter de modifications du modèle conceptuel, des classes de données ou du code d'application. Étant donné que les modèles de stockage sont spécifiques au fournisseur, vous pouvez travailler avec un modèle conceptuel cohérent sur diverses sources de données.

Entity Framework utilise ces fichiers de modèle et de mappage pour créer, lire, mettre à jour et supprimer des opérations sur des entités et des relations dans le modèle conceptuel avec des opérations équivalentes dans la source de données. Entity Framework prend même en charge le mappage des entités du modèle conceptuel aux procédures stockées dans la source de données. Pour plus d'informations, consultez les spécifications CSDL, SSDL et MSL.

2. Mapper des objets sur des données

La programmation orientée objet pose un défi pour interagir avec les systèmes de stockage de données. Bien que l'organisation des classes reflète fréquemment l'organisation des tables de bases de données relationnelles, l'ajustement n'est pas parfait. Plusieurs tables normalisées correspondent fréquemment à une seule classe, et les relations entre les classes sont souvent représentées différemment des relations entre les tables. Par exemple, pour représenter le client d'une commande client, une classe Order peut utiliser une propriété qui contient une référence à une instance d'une classe Customer, tandis qu'une ligne de table Order dans une base de données contient une colonne de clé étrangère (ou un ensemble de colonnes) avec une valeur qui correspond à une valeur de clé primaire dans la table Customer. Une classe Customer peut avoir une propriété nommée Orders qui contient une collection d'instances de la classe Order, tandis que la table Customer d'une base de données n'a pas de colonne comparable. Entity Framework offre aux développeurs la possibilité de représenter les relations de cette manière ou de modéliser plus étroitement les relations telles qu'elles sont représentées dans la base de données.

Les solutions existantes ont tenté de combler cet écart, fréquemment appelé "inadéquation d'impédance", en ne mappant que des classes et des propriétés orientées objet sur des tables et des colonnes relationnelles. Au lieu d'adopter cette approche traditionnelle, Entity Framework mappe les tables relationnelles, les colonnes et les contraintes de clé étrangère dans les modèles logiques aux entités et aux relations dans les modèles conceptuels. Cela permet une plus grande flexibilité à la fois dans la définition des objets et dans l'optimisation du modèle logique. Les outils Entity Data Model génèrent des classes de données extensibles basées sur le modèle conceptuel. Ces classes sont des classes partielles qui peuvent être étendues avec des membres supplémentaires que le développeur ajoute. Par défaut, les classes générées pour un modèle conceptuel particulier dérivent des classes de base qui fournissent des services

pour matérialiser des entités en tant qu'objets et pour suivre et enregistrer les modifications. Les développeurs peuvent utiliser ces classes pour travailler avec les entités et les relations en tant qu'objets liés par des associations. Les développeurs peuvent également personnaliser les classes générées pour un modèle conceptuel. Pour plus d'informations, voir Travailler avec des objets.

3. Accéder aux données d'entité et les modifier

Plus qu'une simple solution de mappage objet-relationnel, Entity Framework consiste fondamentalement à permettre aux applications d'accéder et de modifier des données représentées sous forme d'entités et de relations dans le modèle conceptuel. Entity Framework utilise les informations du modèle et des fichiers de mappage pour traduire les requêtes d'objets par rapport aux types d'entités représentés dans le modèle conceptuel en requêtes spécifiques à la source de données. Les résultats de la requête sont matérialisés dans des objets gérés par Entity Framework. Entity Framework fournit les méthodes suivantes pour interroger un modèle conceptuel et renvoyer des objets :

LINQ aux entités. Fournit la prise en charge de LINQ (Language-Integrated Query) pour interroger les types d'entités définis dans un modèle conceptuel. Pour plus d'informations, consultez LINQ aux entités.

Entité SQL. Un dialecte SQL indépendant du stockage qui fonctionne directement avec les entités du modèle conceptuel et qui prend en charge les concepts Entity Data Model. Entity SQL est utilisé à la fois avec des requêtes d'objet et des requêtes exécutées à l'aide du fournisseur Entity Client. Pour plus d'informations, consultez Vue d'ensemble d'Entity SQL.

Entity Framework inclut le fournisseur de données Entity Client. Ce fournisseur gère les connexions, traduit les requêtes d'entité en requêtes spécifiques à la source de données et renvoie un lecteur de données que Entity Framework utilise pour matérialiser les données d'entité en objets. Lorsque la matérialisation d'objet n'est pas requise, le fournisseur Entity Client peut également être utilisé comme un fournisseur de données ADO.NET standard en permettant aux applications d'exécuter des requêtes Entity SQL et de consommer le lecteur de données en lecture seule renvoyé. Pour plus d'informations, consultez Fournisseur EntityClient pour Entity Framework.

Les outils de modèle de données d'entité peuvent générer une classe dérivée de System.Data.Objects.ObjectContext ou System.Data.Entity.DbContext qui représente le conteneur d'entités dans le modèle conceptuel. Ce contexte d'objet fournit les fonctionnalités de suivi des modifications et de gestion des identités, de la concurrence et des relations. Cette classe expose également une méthode SaveChanges qui écrit des insertions, des mises à jour et des suppressions dans la source de données. Comme les requêtes, ces modifications sont effectuées soit par des commandes générées automatiquement par le système, soit par des procédures stockées spécifiées par le développeur.

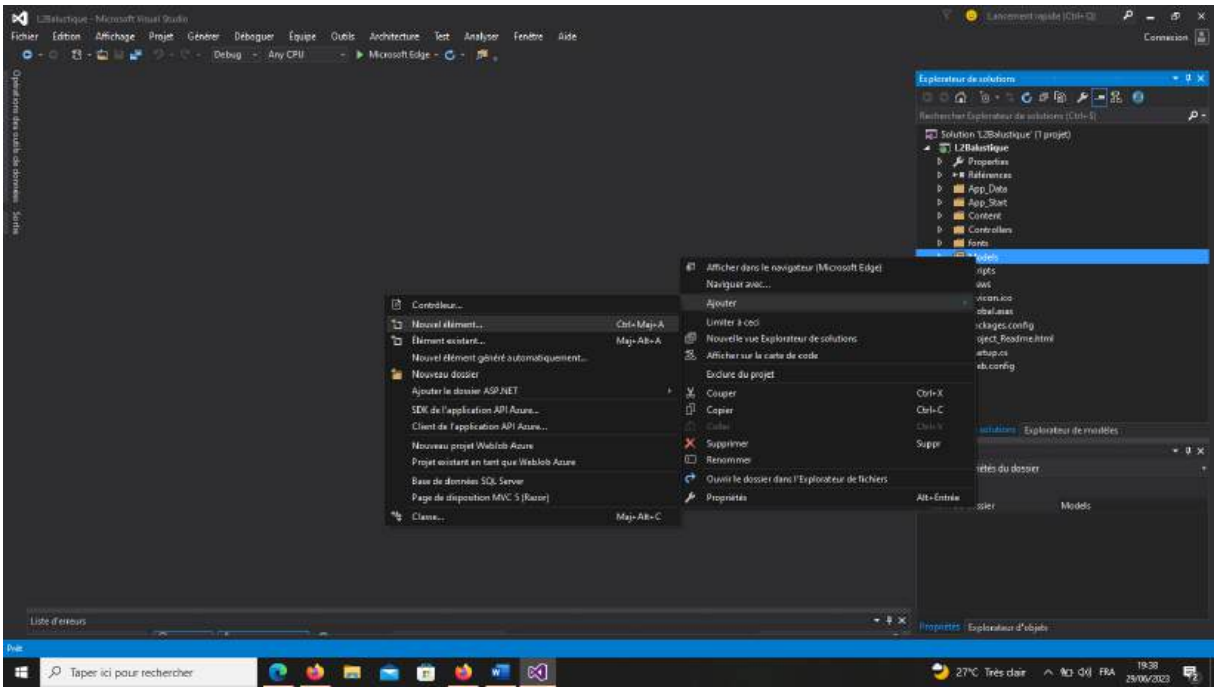
4. Fournisseurs de données

Le fournisseur EntityClient étend le modèle de fournisseur ADO.NET en accédant aux données en termes d'entités conceptuelles et de relations. Il exécute des requêtes qui utilisent Entity SQL. Entity SQL fournit le langage de requête sous-jacent qui permet à EntityClient de communiquer avec la base de données. Pour plus d'informations, consultez Fournisseur EntityClient pour Entity Framework. Entity Framework inclut un fournisseur de données SqlClient mis à jour qui prend en charge les arborescences de commandes canoniques. Pour plus d'informations, consultez SqlClient pour Entity Framework.

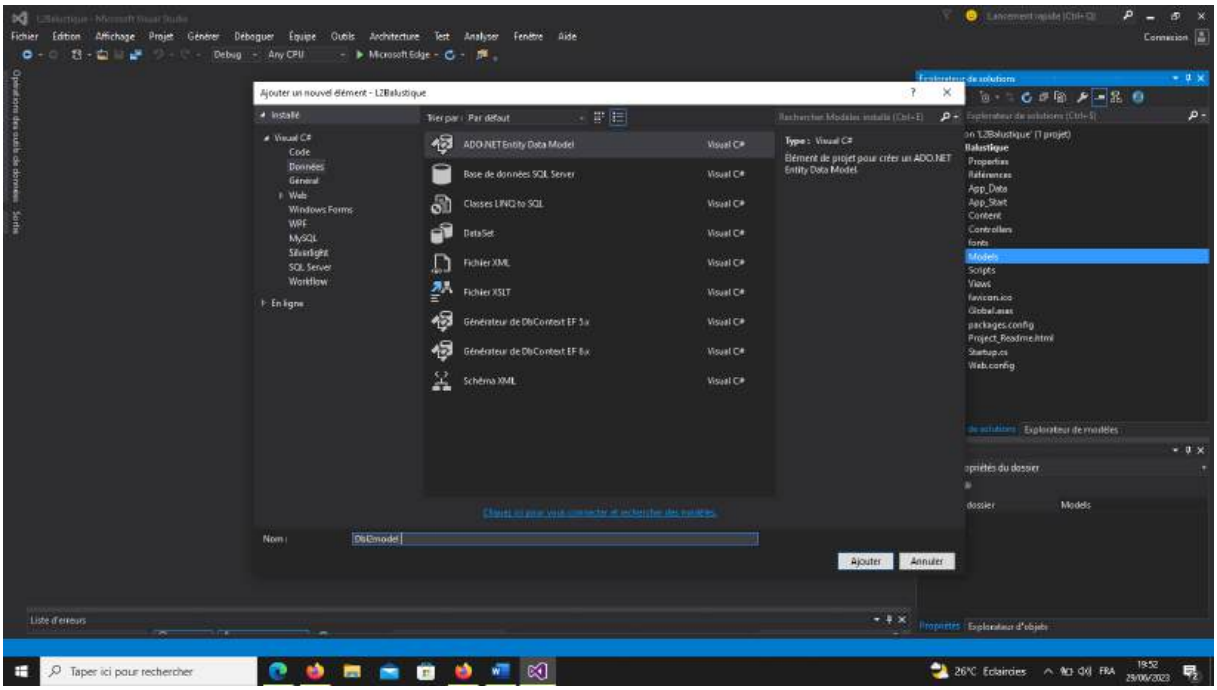
Outils de modèle de données d'entité Avec le runtime Entity Framework, Visual Studio inclut les outils de mappage et de modélisation. Pour plus d'informations, voir Modélisation et mappage.

B. CREATION DU MODELE

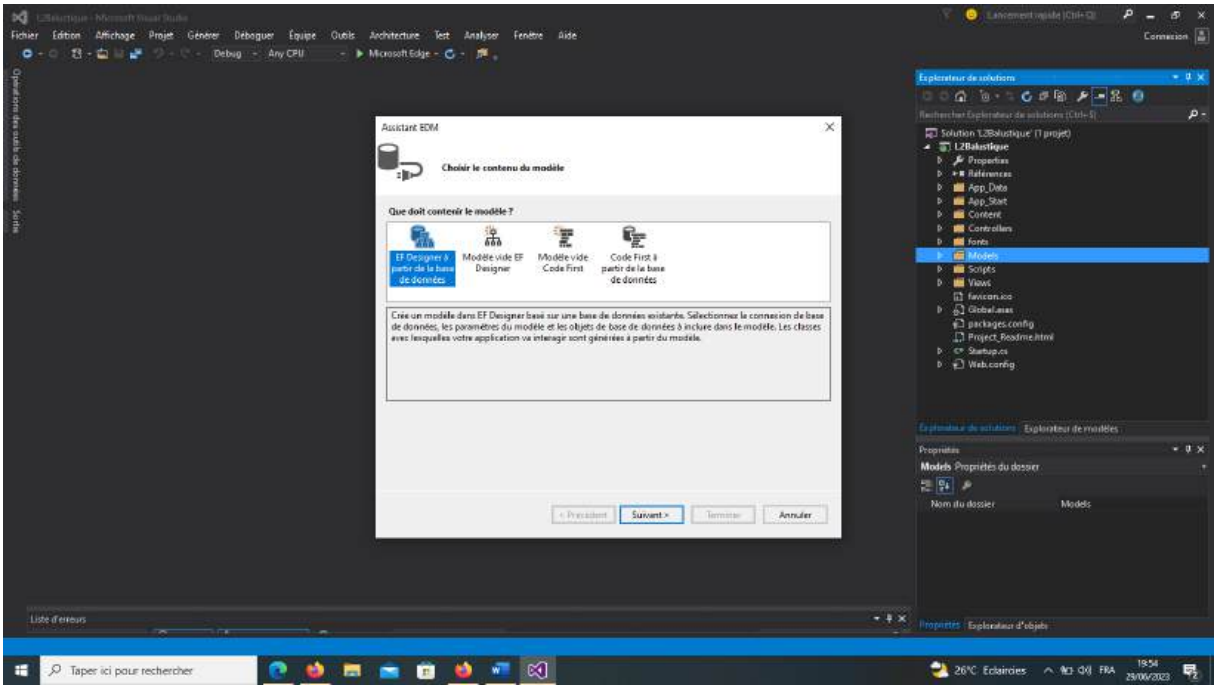
Pour créer le modèle, il faut cliquer droit sur le dossier model du projet puis cliquer gauche sur ajouter/nouvel élément. Mais avant tout fermer d'abord la connexion en cliquant droit puis gauche sur s en être l'opération des outils de données.



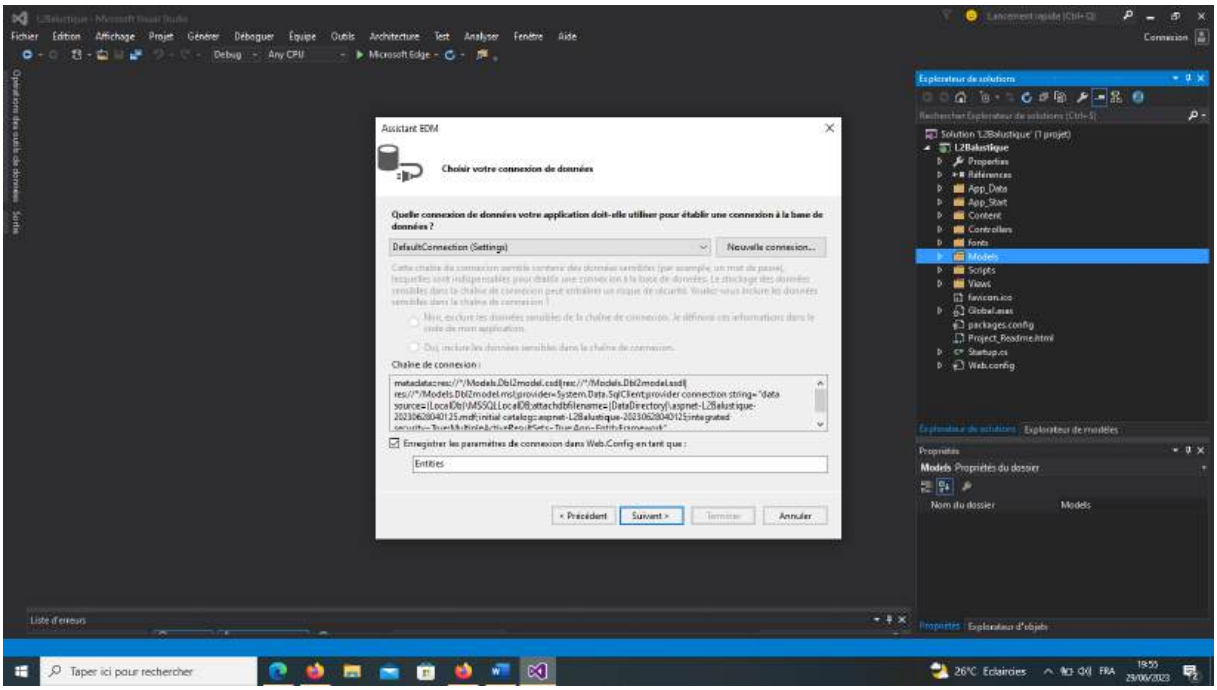
- Sélectionnez données puis cliquer sur ADO.NET Entity Data Model
- Donner un nom à votre model par exemple Dbl2model puis cliquer sur ajouter



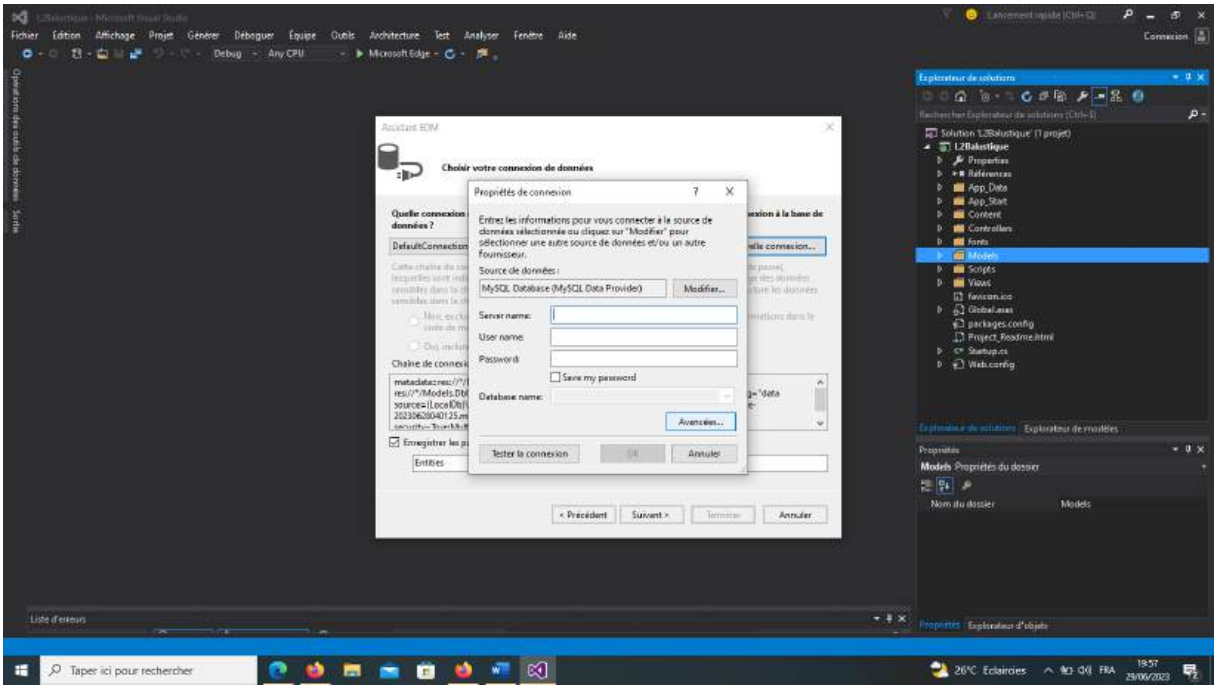
- Cliquer sur EF designer à partir de la base de données



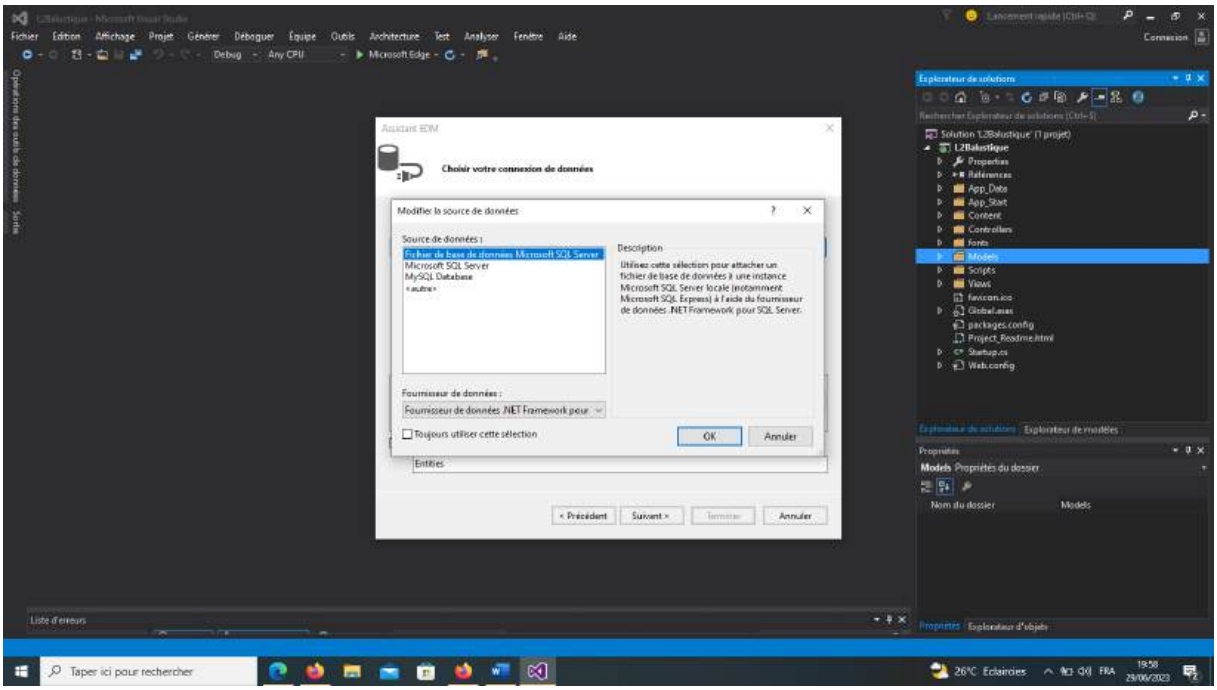
- Cliquer sur suivant



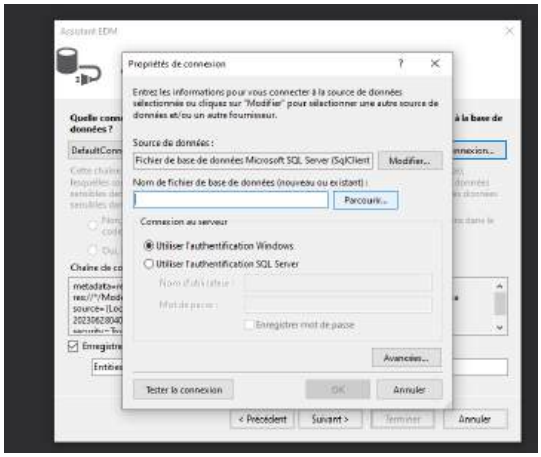
- Cliquer sur nouvelle connexion

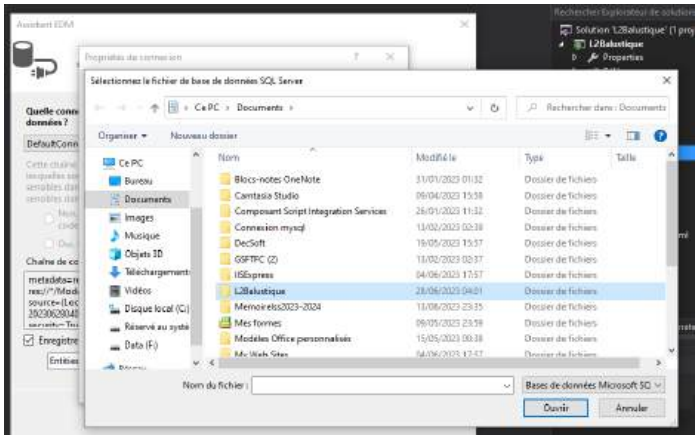


- Cliquer sur modifier et sélectionnez fichier de la base de données MS SQLserveur/OK

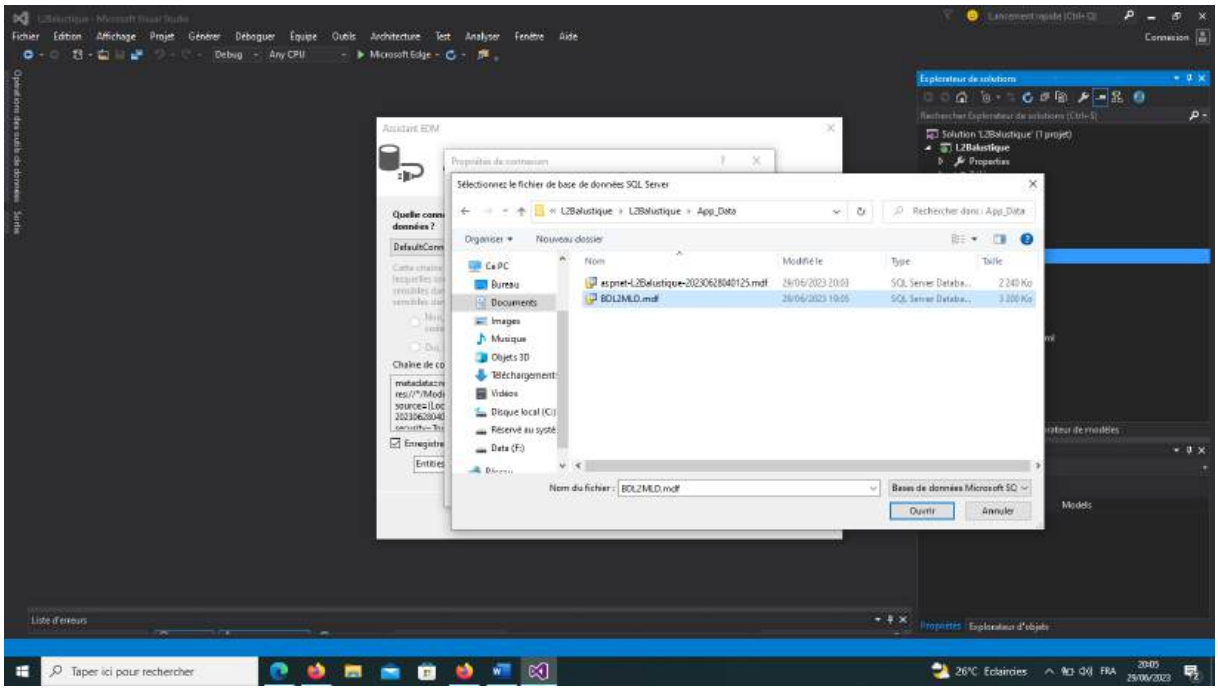


- Cliquer ok puis sur parcourir pour chercher le nom du fichier de la base de données dans le dossier App_Data.

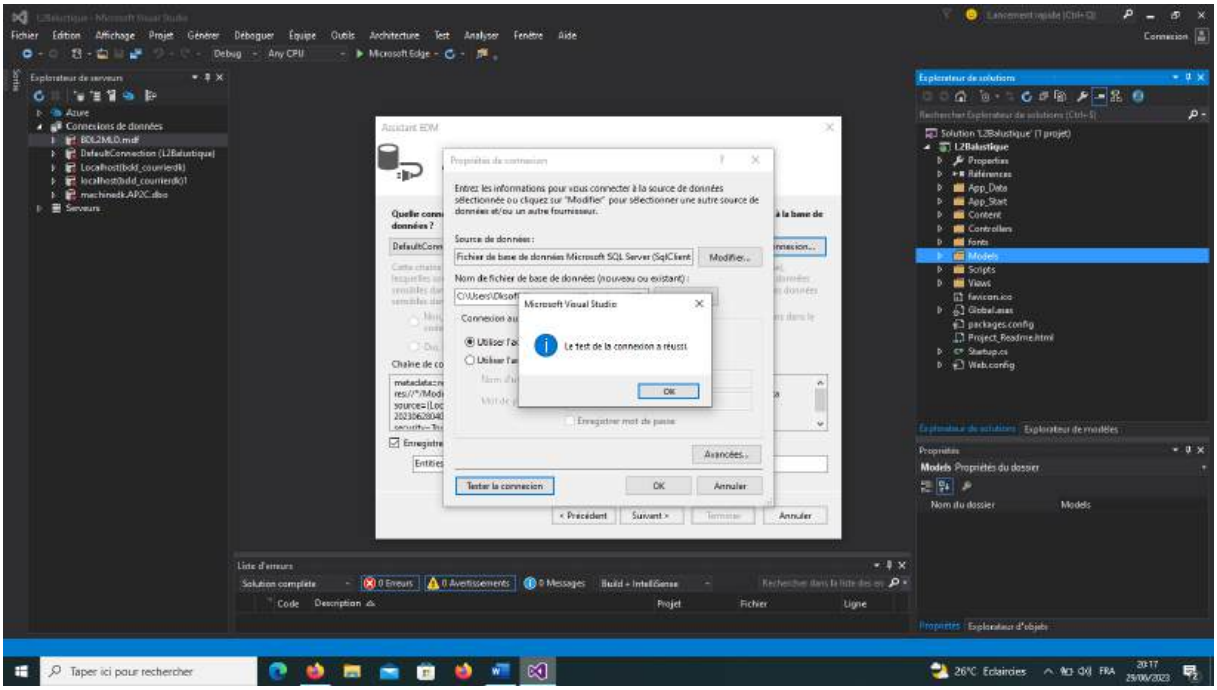




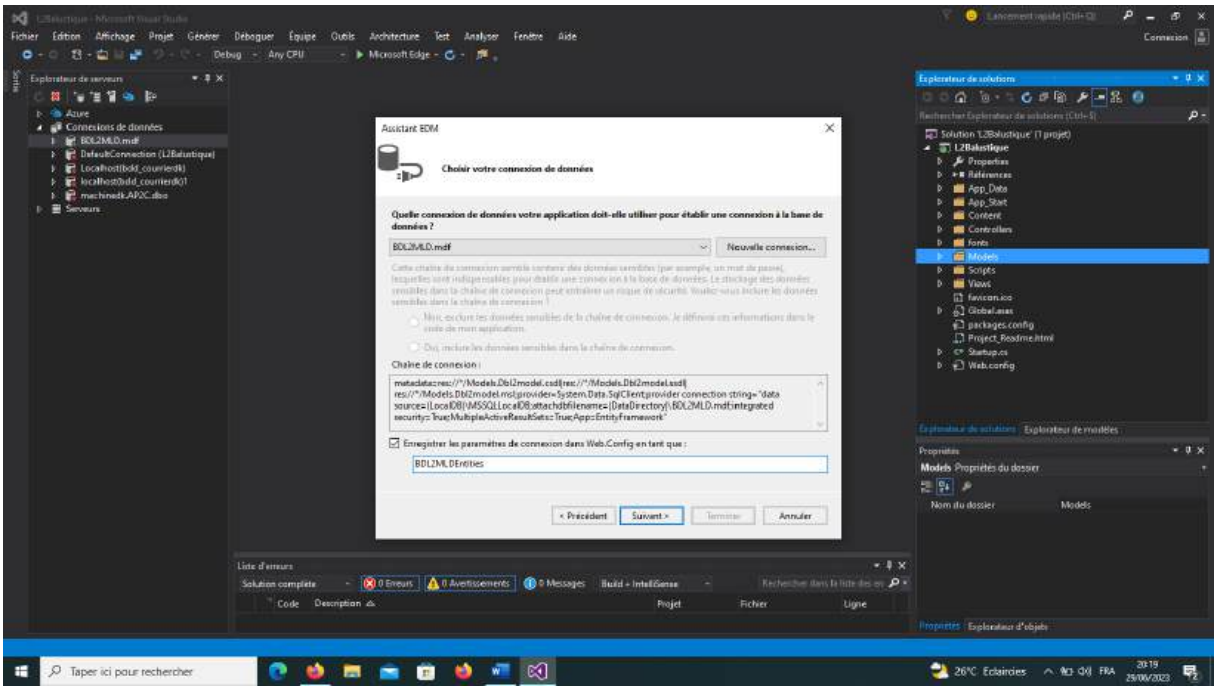
- Sélectionnez le dossier du projet puis chercher le dossier App_data, l'intérieur se trouve le fichier de la base de données



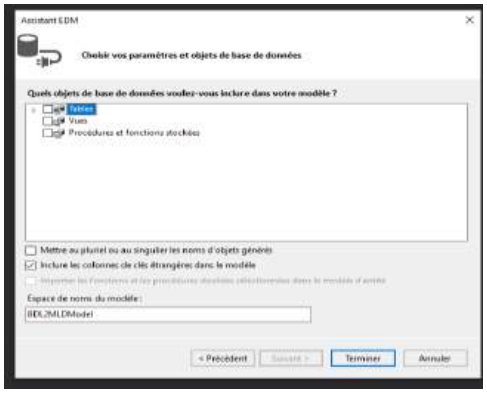
- Sélectionnez le fichier puis cliquer sur ouvrir
- Tester la connexion en cliquant sur ok



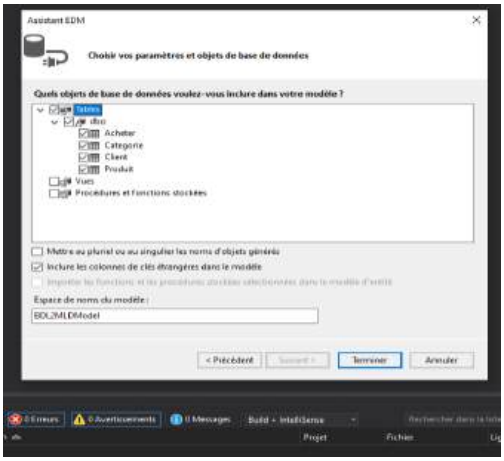
- Cliquer encore sur ok



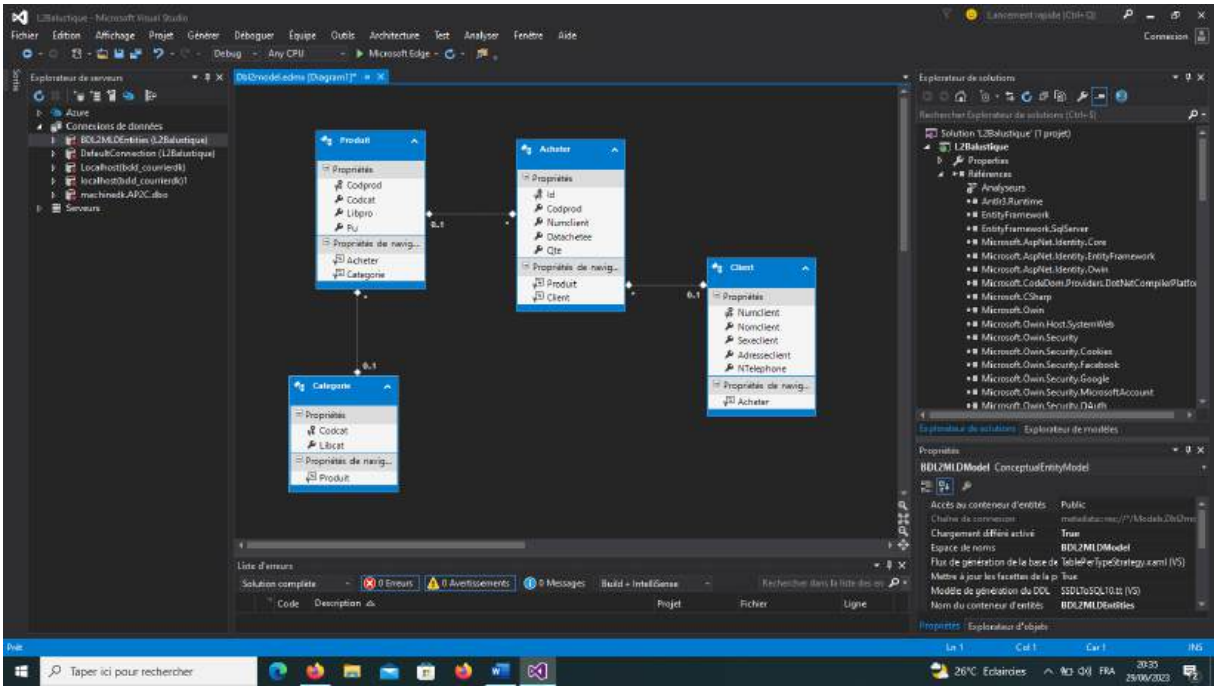
- Nous avons maintenant un model entity nommé BDL2MLDEntites
- Cliquer sur suivant



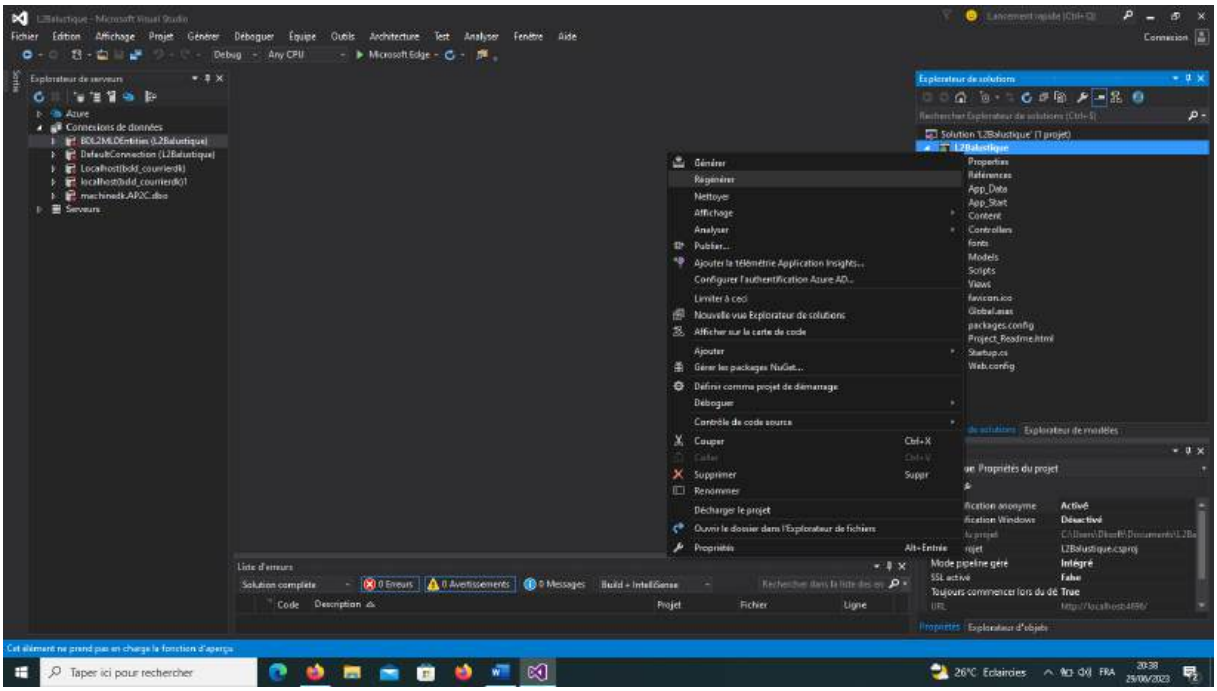
- Sélectionnez les tables de votre model et cliquer sur terminer mais si vous disposez de procédure stockées.



- Après chargement un schéma des objets s'affiche



- A la fin de l'opération, il faut régénérer un projet pour optimiser le projet.



- Vérifier s'il y a l'erreur, sinon ok
- Mais s'il y a les erreurs veuillez désactiver votre antivirus ou le pare feu

2.2.3. Création d'un Controller

Un contrôleur contient la logique de contrôle de flux pour une application MVC ASP.NET. Un contrôleur détermine la réponse à renvoyer à un utilisateur lorsqu'un utilisateur effectue une demande de navigateur. Un contrôleur n'est qu'une classe (par exemple, une classe Visual Basic ou C#)

a. Présentation du routage ASP.NET

Une requête de navigateur est mappée à une action de contrôleur via une fonctionnalité de l'infrastructure ASP.NET appelée *routage ASP.NET*. ASP.NET routage est utilisé par l'infrastructure MVC ASP.NET pour *acheminer* les requêtes entrantes vers les actions du contrôleur.

ASP.NET routage utilise une table de routage pour gérer les requêtes entrantes. Cette table de routage est créée lorsque votre application web démarre pour la première fois. La table de routage est configurée dans le fichier Global.asax. Le fichier MVC Global.asax par défaut se trouve dans listing 1.

Référencement 1 - Global.asax

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace MvcApplication1
{
    // Note: For instructions on enabling IIS6 or IIS7 classic mode,
    // visit https://go.microsoft.com/?LinkId=9394801

    public class MvcApplication : System.Web.HttpApplication
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                "Default",                                          // Route name
                "{controller}/{action}/{id}",                       // URL with parameter
                new { controller = "Home", action = "Index", id = "" }  // Parameter defaults
            );

        }

        protected void Application_Start()
        {
            RegisterRoutes(RouteTable.Routes);
        }
    }
}
```

Lorsqu'une application ASP.NET démarre pour la première fois, la méthode Application_Start() est appelée. Dans la liste 1, cette méthode appelle la méthode RegisterRoutes() et la méthode RegisterRoutes() crée la table de routage par défaut.

La table de routage par défaut se compose d'un itinéraire. Cette route par défaut divise toutes les requêtes entrantes en trois segments (un segment d'URL est tout ce qui se trouve entre les barres obliques). Le premier segment est mappé au nom d'un contrôleur, le deuxième segment est mappé à un nom d'action et le dernier segment est mappé à un paramètre passé à l'action nommée Id.

Considérez par exemple l'URL suivante :
/Product/Details/3
Cette URL est analysée en trois paramètres tels que ceux-ci :
Contrôleur = Produit
Action = Détails

ID = 3

L'itinéraire par défaut défini dans le fichier Global.asax inclut les valeurs par défaut pour les trois paramètres. Le contrôleur par défaut est Accueil, l'action par défaut est Index et l'ID par défaut est une chaîne vide. Avec ces valeurs par défaut à l'esprit, réfléchissez à la façon dont l'URL suivante est analysée :

/Employee
Cette URL est analysée en trois paramètres tels que ceux-ci :
Contrôleur = Employé
Action = Index
ID =
Enfin, si vous ouvrez une application MVC ASP.NET sans fournir d'URL (par exemple, http://localhost) , l'URL est analysée comme suit :
Contrôleur = Accueil
Action = Index
ID = La requête est acheminée vers l'action Index() sur la classe HomeController.

La route Default inclut des valeurs par défaut pour les trois paramètres. Si vous ne fournissez pas un contrôleur, alors la valeur par défaut sera prise à savoir Home. Pour l'action il s'agira d'Index et pour le paramètre Id, il s'agira d'une chaine vide.

Regardons quelques exemples pour voir comment la route Default lie des URL à des actions contrôleur. Imaginez que vous saisissiez l'adresse suivante dans votre navigateur :
/Home

À cause des paramètres par défaut de la route Default, saisir l'URL précédente causera l'appel de l'action Index du contrôleur HomeController.

HomeController.cs

```
using System.Web.Mvc;
namespace MvcApplication1.Controllers
{
    [HandleError]
    public class HomeController : Controller
    {
        public ActionResult Index(string id)
        {
            return View();
        }
    }
}
```

La classe HomeController précédente inclut une méthode Index() qui accepte un unique paramètre Id. L'URL /Home causera l'appel de cette méthode en passant une chaine vide au paramètre Id.

À cause de la façon dont le framework MVC invoque les actions contrôleur, l'URL /Home correspond également à la méthode Index() de la classe HomeController.

```
using System.Web.Mvc;

namespace MvcApplication1.Controllers
{
    [HandleError]
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

La méthode Index() présentée juste avant n'accepte aucun paramètre. L'URL /Home aura pour effet d'appeler cette méthode. L'URL /Home/Index/3 appellera aussi cette méthode, le paramètre Id étant ignoré, tout simplement.

L'URL /Home correspond également à la méthode Index() de la casse HomeController suivante :

```
using System.Web.Mvc;
namespace MvcApplication1.Controllers
{
    [HandleError]
    public class HomeController : Controller
    {
        public ActionResult Index(int? id)
```

```
        {
            return View();
        }
    }
}
```

Dans le code précédent, la méthode Index() avec un paramètre de type entier. Mais parce que ce paramètre est de type nullable, la méthode Index() peut être appelée sans générer d'erreur. Au contraire, appeler la méthode Index() (du code suivant) avec l'URL /Home causera une exception parce que le paramètre Id n'est pas nullable et sera donc manquant.

b. Comprendre les résultats des actions

Une action de contrôleur renvoie quelque chose appelé un résultat d'action. Un résultat d'action est ce qu'une action de contrôleur renvoie en réponse à une requête du navigateur.

Le framework ASP.NET MVC prend en charge plusieurs types de résultats d'action, notamment :

ViewResult -Représente le HTML et le balisage. EmptyResult - Représente aucun résultat.

RedirectResult - Représente une redirection vers une nouvelle URL. JsonResult - Représente un résultat JavaScript Object Notation qui peut être utilisé dans une application AJAX.

- JavaScriptResult - Représente un script JavaScript.
- ContentResult - Représente un résultat textuel.
- FileContentResult - Représente un fichier téléchargeable (avec le contenu binaire).
- FilePathResult - Représente un fichier téléchargeable (avec un chemin).
- FileStreamResult - Représente un fichier téléchargeable (avec un flux de fichiers).

Tous ces résultats d'action héritent de la classe ActionResult de base. Dans la plupart des cas, une action de contrôleur renvoie un ViewResult. Par exemple, l'action du contrôleur Index dans le Listing 2 renvoie un ViewResult.

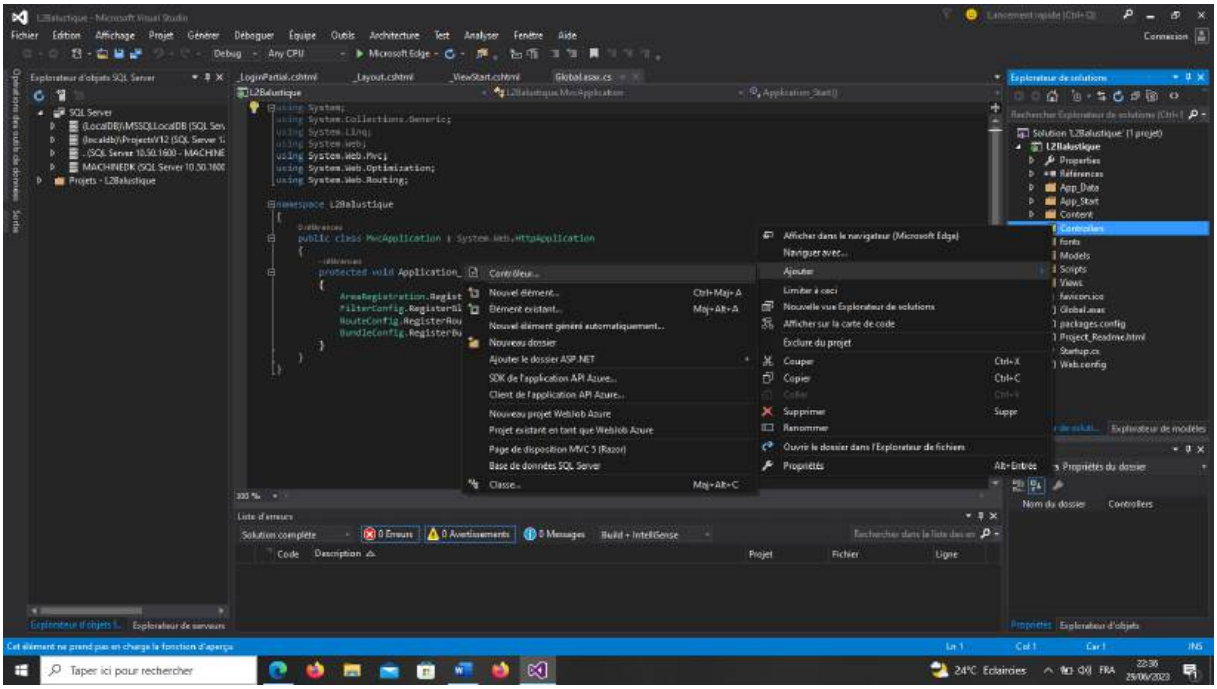
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Mvc.Ajax;
namespace MvcApplication1.Controllers
{
    public class BookController : Controller
    {
        public ActionResult Index()
        {
            // Add action logic here
return View();
        }
    }
}
```

c. Comment créer un controllers

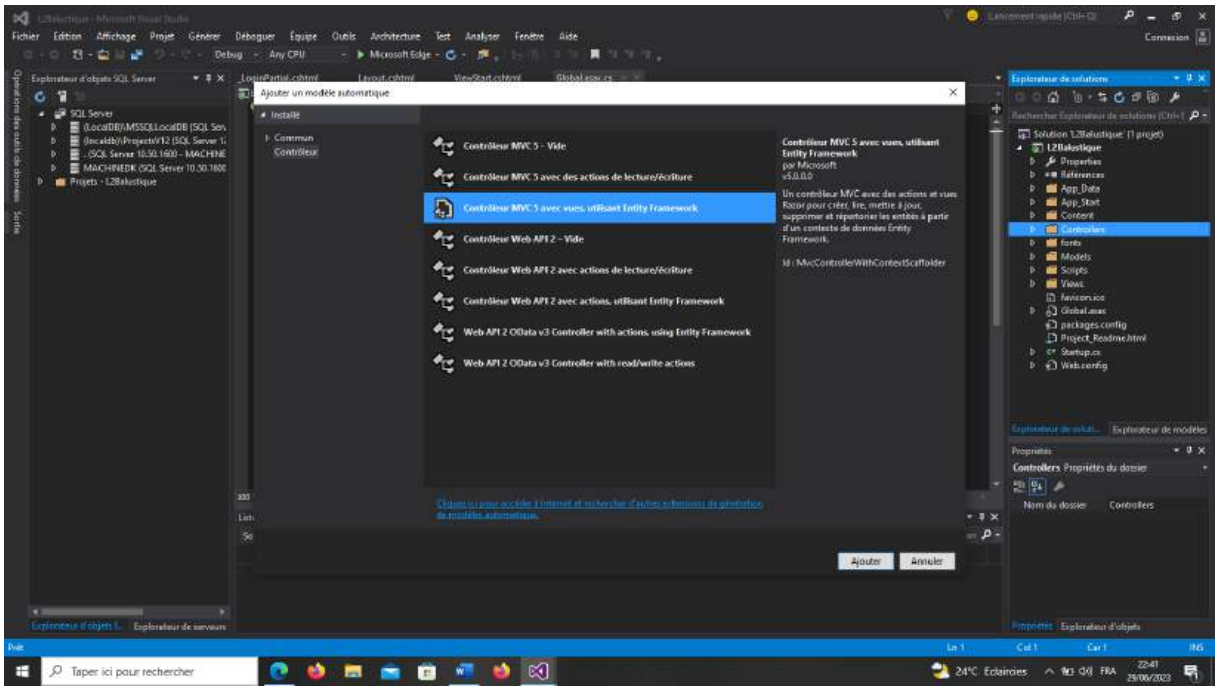
Pour créer un contrôleur, il faut noter que chaque contrôleur donne vie à une vue.

Procédure :

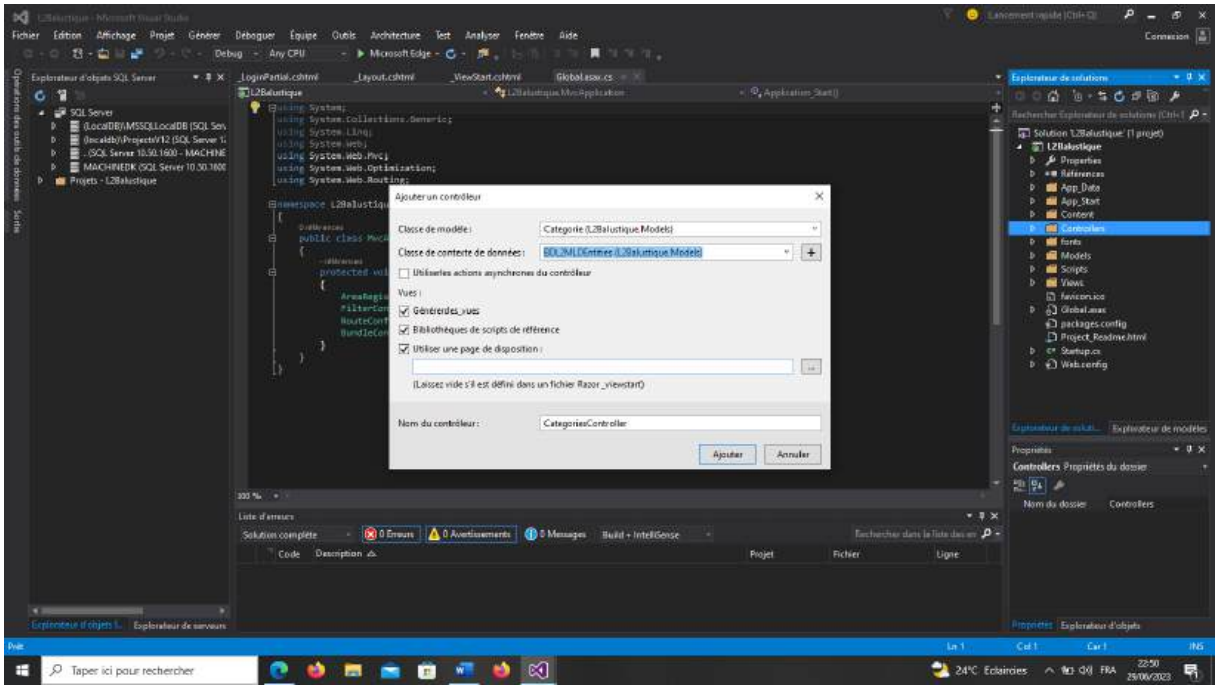
- Cliquer droit sur le dossier controllers/ajouter
- Cliquer sur contrôleur



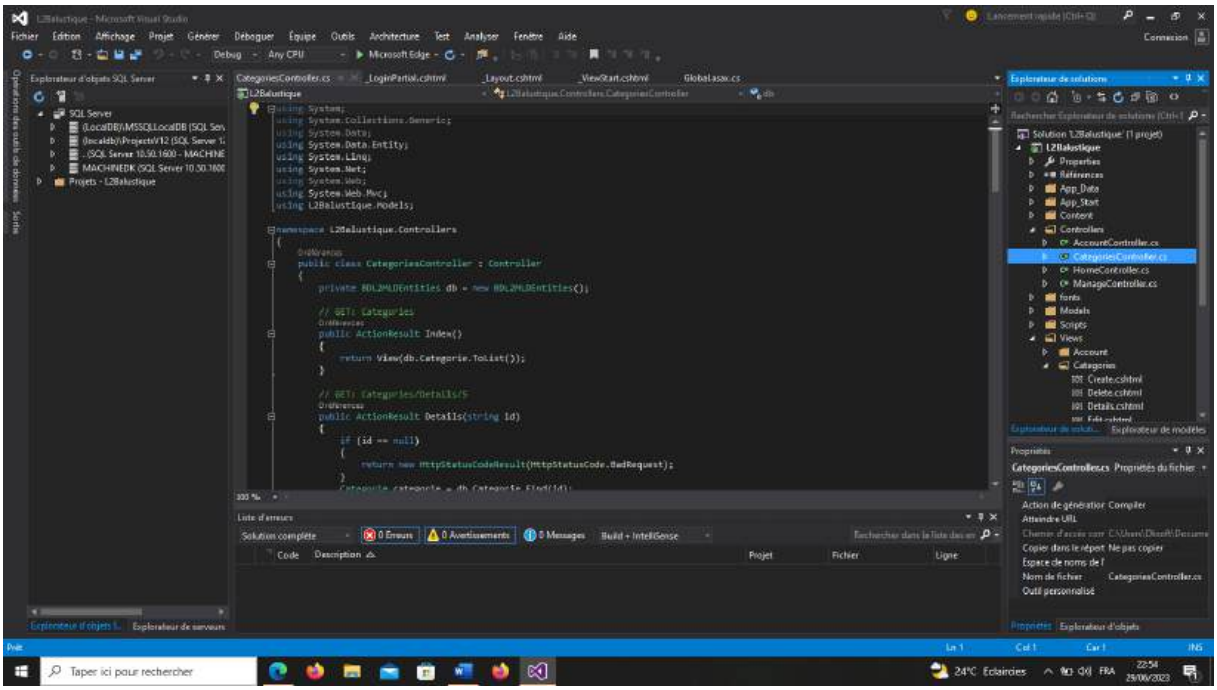
- Sélectionnez trois contrôleurs MVC 5 avec vues puis cliquer utilisant entity framework.



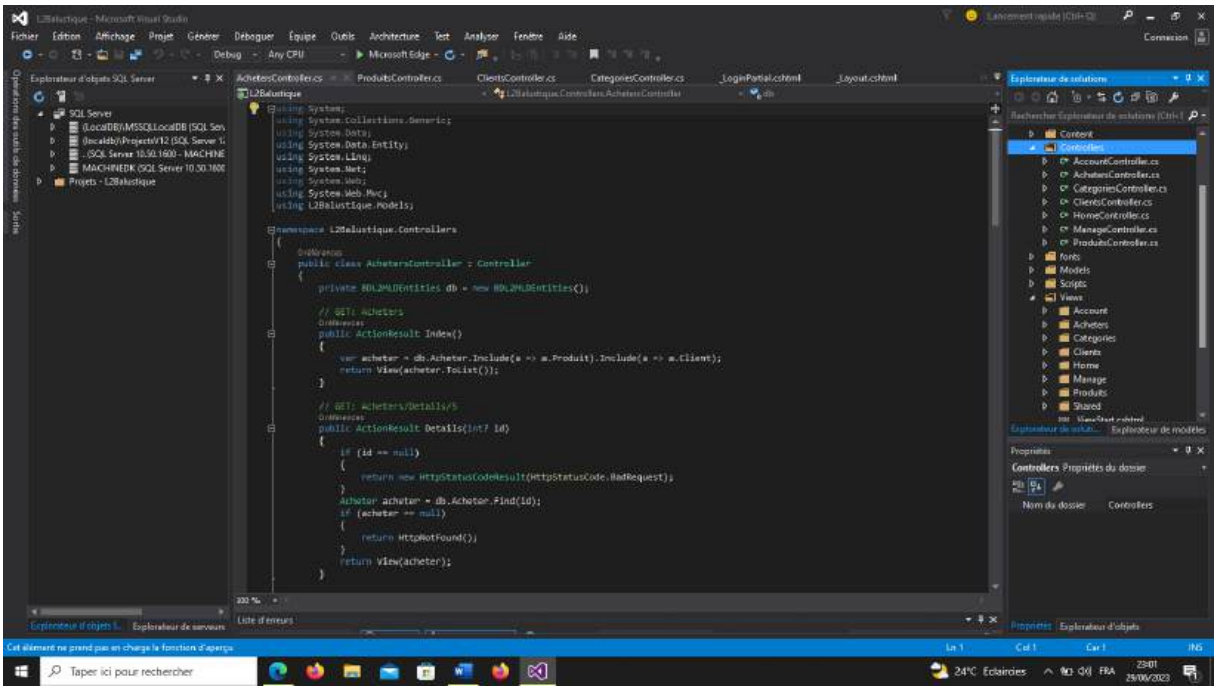
- Sélectionnez le nom de votre classe modèle et classe contexte de données. Cette opération concerne tous les objets du modelé, dans notre cas nous allons prendre catégorie comme exemple de notre support mais vous devez faire pour tous les objets.



- Puis cliquer sur ajouter
- Après chargement le Visual studio nous génère la vue et le contrôleur avec les différentes méthodes (actions).



Apres la création de tous les contrôleurs, vous auriez l'image suivante :



- Il faut retenir que le contrôleur détient un formisme suivant :

  Exemple : CategoriesControllers.css

- Tandis que la vue se présente de la manière suivante :

  Exemple dossier Categories avec les fichiers suivants : Creation.cshtml ; Delete.cshml, Edit.cshtml, Details.cshtml et index.cshtml.

Pour les différents fichiers, il faut retenir ceci :

- Creation.cshtml : la page permettant l'ajout des informations ;
- Delete.cshml : la page permettant la suppression de l'information ;
- Edit.cshtml : la page permettant la modification de l'information ;
- Details.cshtml : la page permettant l'afficher les détails des informations
- index.cshtml : la page permettant l'afficher des informations sous forme d'un tableau.

2.2.4. Notion de vues

Le but de ce point est de vous fournir une brève introduction aux vues d'ASP.NET MVC, aux vues de données et aux HTML helpers (Les Helpers HTML sont des méthodes qui renvoient une chaîne). A la fin vous devriez être capable de créer de nouvelles vues, de passer des données d'un controleur vers une vue, et d'utiliser des HTML helpers pour générer du contenu dans une vue.

Listing 1 - HomeController.cs

```
using System.Web.Mvc;

namespace MvcApplication1.Controllers
{
    [HandleError]
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult Details()
        {
            return RedirectToAction("Index");
        }
    }
}
```

Vous pouvez déclencher la 1ère action, l'action Index(), en tapant l'url suivante dans la barre d'adresse de votre navigateur: /Home/Index.
Vous pouvez déclencher la 2nde action, l'action Details(), en tapant l'url suivante dans la barre d'adresse de votre navigateur: /Home/Details.

L'action Index() retourne une vue. La plupart des actions que vous créez retourneront des vues. Cependant, une action peut retourner des résultats d'action d'autres types. Par exemple, l'action Details() retourne une RedirectToActionResult qui redirige la requête entrante vers l'action Index().

L'action Index() contient l'unique ligne de code suivante: View(); Cette ligne de code retourne une vue qui doit être située à l'endroit suivant sur votre serveur web: \Views\Home\Index.aspx.

Le chemin de la vue est obtenu à partir du nom du controleur et du nom de l'action du controleur. Si vous préférez, vous pouvez spécifier directement la vue. La ligne de code suivante retourne une vue appelée "Fred": View("Fred");

Quand cette ligne de code est exécutée, une vue du chemin suivant est retournée: \Views\Home\Fred.aspx

a. Utiliser des HTML Helpers pour générer le contenu d'une vue

Pour rendre plus facile l'ajout de contenu dans une vue, vous pouvez tirer avantage de ce qu'on appelle un HTML Helper. Un HTML Helper, typiquement, est une méthode qui génère une chaine de caractères. Vous pouvez utiliser des HTML Helpers pour générer des éléments HTML standards tels que des zones de saisie, des liens, des listes déroulantes ou des cases à cocher.
Par exemple, la vue dans Listing 4 tire avantage de 3 HTML Helpers -- le BeginForm(),le TextBox() et le Password() helper -- pour générer un formulaire de connection.

Listing 4 - \Views\Home\Login.aspx

```
<%@ Page Language="C#" Inherits="System.Web.Mvc.ViewPage" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>Login Form</title>
</head>
<body>
    <div>
            <% using (Html.BeginForm())
               { %>
                <label for="UserName">User Name:</label>
                <br />
                <%= Html.TextBox("UserName") %>

                <br /><br />

                <label for="Password">Password:</label>
                <br />
                <%= Html.Password("Password") %>

                <br /><br />

                <input type="submit" value="Log in" />
            <% } %>
    </div>
</body>
</html>
```

b. Les Layouts ou disposition (mise en page)

Razor apporte, avec les layouts, une alternative au concept des Masterpages en ASP.NET, qui ne sont pas supportées par Razor. Pour ceux qui n'ont pas l'habitude

d'ASP.NET, ou qui n'ont jamais utilisé les Masterpages, le principe est de regrouper, dans une page spécifique, les éléments communs de mise en page pour le site (tels que le menu, l'entête, le pied de page, ou autres scripts JavaScript utilisés par toutes les pages). Par défaut, lors de la création d'un site Web utilisant le moteur de vue Razor le fichier de Layout suivant est créé automatiquement :

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
   <meta charset="utf-8" />
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>@ViewBag.Title - Mon application ASP.NET</title>
   @Styles.Render("~/Content/css")
   @Scripts.Render("~/bundles/modernizr")

</head>
<body>
   <div class="navbar navbar-inverse navbar-fixed-top">
      <div class="container">
         <div class="navbar-header">
            <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-
collapse">
               <span class="icon-bar"></span>
               <span class="icon-bar"></span>
               <span class="icon-bar"></span>
            </button>
            @Html.ActionLink("Nom d'application", "Index", "Home", new { area = "" }, new { @class =
"navbar-brand" })
         </div>
         <div class="navbar-collapse collapse">
            <ul class="nav navbar-nav">
               <li>@Html.ActionLink("Accueil", "Index", "Home")</li>
               <li>@Html.ActionLink("À propos de", "About", "Home")</li>
               <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
            </ul>
            @Html.Partial("_LoginPartial")
         </div>
      </div>
   </div>
   <div class="container body-content">
      @RenderBody()
      <hr />
      <footer>
         <p>&copy; @DateTime.Now.Year - My ASP.NET Application</p>
      </footer>
   </div>

   @Scripts.Render("~/bundles/jquery")
   @Scripts.Render("~/bundles/bootstrap")
   @RenderSection("scripts", required: false)
</body>
</html>
```

Si l'on regarde de plus près la page, on se rend compte que ce qui la rend spéciale est l'appel à la méthode @RenderBody(). Cette méthode, qui ne peut être appelée qu'une seule fois dans la page, va permettre de déterminer l'endroit où le code de la page utilisant le Layout va être rendu.

Par exemple, si on a la vue suivante (la vue Home.cshtml créée par défaut) :

```
@inherits System.Web.Mvc.WebViewPage
@{
    View.Title = "Home Page";
    LayoutPage = "~/Views/Shared/_Layout.cshtml";
}

<h2>@View.Message</h2>
<p>
 To learn more about ASP.NET MVC visit <a href="http://asp.net/mvc" title="ASP.NET MVC Website">http://asp.net/mvc</a>.
</p>
```

Et que je modifie le fichier _Layout.cshtml ainsi :

```
<div id="main" style="background-color:#ccc">
    @RenderBody()
    <div id="footer">
    </div>
</div>
```

c. ViewStart

Il peut vite être lassant d'avoir, en haut de chaque page, une ligne indiquant le fichier de layout à utiliser. Pour remédier à cela, Razor propose un mécanisme supplémentaire, la vue **_ViewStart**, spécifiquement prévue pour gérer l'association des pages avec un layout.

ViewStart est en effet appelée avant chaque rendu d'une vue se trouvant dans son répertoire ou un de ses sous-répertoires. Dans sa plus simple expression, une page ViewStart aura le contenu suivant :

```
@{
 Layout = "~/Views/Shared/_Layout.cshtml";}
```

Il est aussi possible d'utiliser cette vue pour sélectionner le layout à utiliser, par exemple, on pourrait, pour utiliser un layout spécifique pour les pages d'édition, utiliser le code suivant :

```
@{
    if
(ViewContext.Controller.ValueProvider["controller"].RawValue.StartsWith("Ed
it"))
    {
        Layout = "~/Views/Shared/_EditLayout.cshtml";
    }
    else
    {
        Layout = "~/Views/Shared/_Layout.cshtml";
    }
}
```

d. Sections

Le fait de n'avoir qu'un seul appel possible à RenderBody va certainement faire lever les sourcils de ceux qui utilisaient des masterpages ayant plusieurs zones de rendu possibles (plusieurs ContentPlaceHolders).

Razor apporte une solution élégante à cette problématique, par l'intermédiaire des sections. Une section est une zone de la page qui va être définie ou redéfinie au niveau de la vue qui appelle le layout. Par exemple, supposons que nous voulions, au niveau d'une vue spécifique, par exemple la vue Home, afficher un pied de page particulier, nous pourrions faire, dans notre vue _Layout :

```
<div id="main">
  @RenderBody()
  @RenderSection("PiedDePage")
// .
```

Et, dans notre vue Home.cshtml :

```
@{
   Layout = "~/Views/Shared/_Layout.cshtml";
}

@section PiedDePage {
   <div>Mon pied de page pour la page Home</div>
}
```

Ceci étant dit, toute vue appelant _Layout va dorénavant devoir avoir une section Footer, au risque de remonter une exception. Nous avons le choix de rendre cette section optionnelle. Pour cela, il nous suffit de modifier l'appel à RenderSection, en mettant l'attribut required à false. Cela se passe comme cela :

```
@RenderSection("PiedDePage", required: false)
```

Cela est bien joli, mais pour vraiment couvrir les fonctionnalités de la masterpage, il nous manque toujours la possibilité d'avoir du contenu par défaut. Pour cela, on va tester l'existence de la section sur laquelle on travaille, et appliquer un rien de logique conditionnelle dans notre layout.

Notre layout aura dorénavant le contenu suivant :

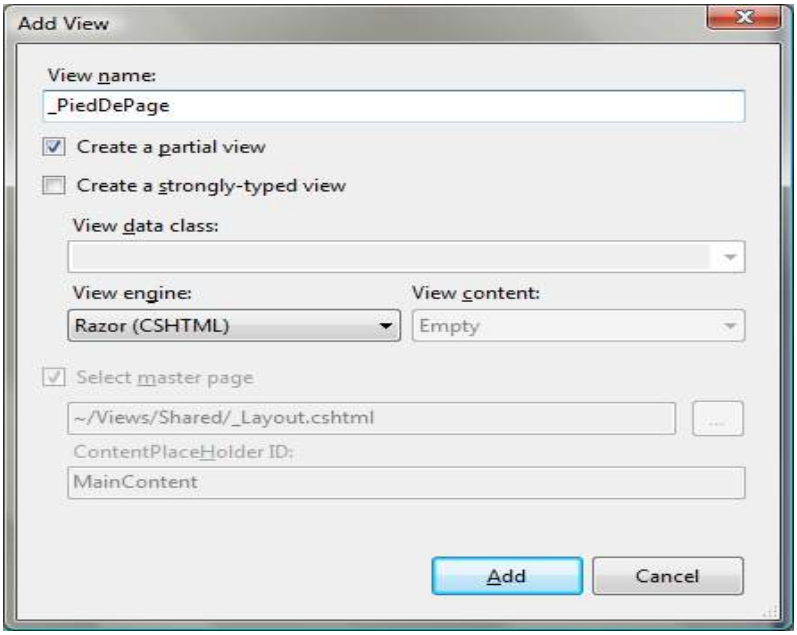
```
@RenderBody()

@if (IsSectionDefined("PiedDePage")) {
   @RenderSection("PiedDePage")
}
else {
   <div>Mon pied de page par défaut </div>}
```

e. Vues partielles avec Razor

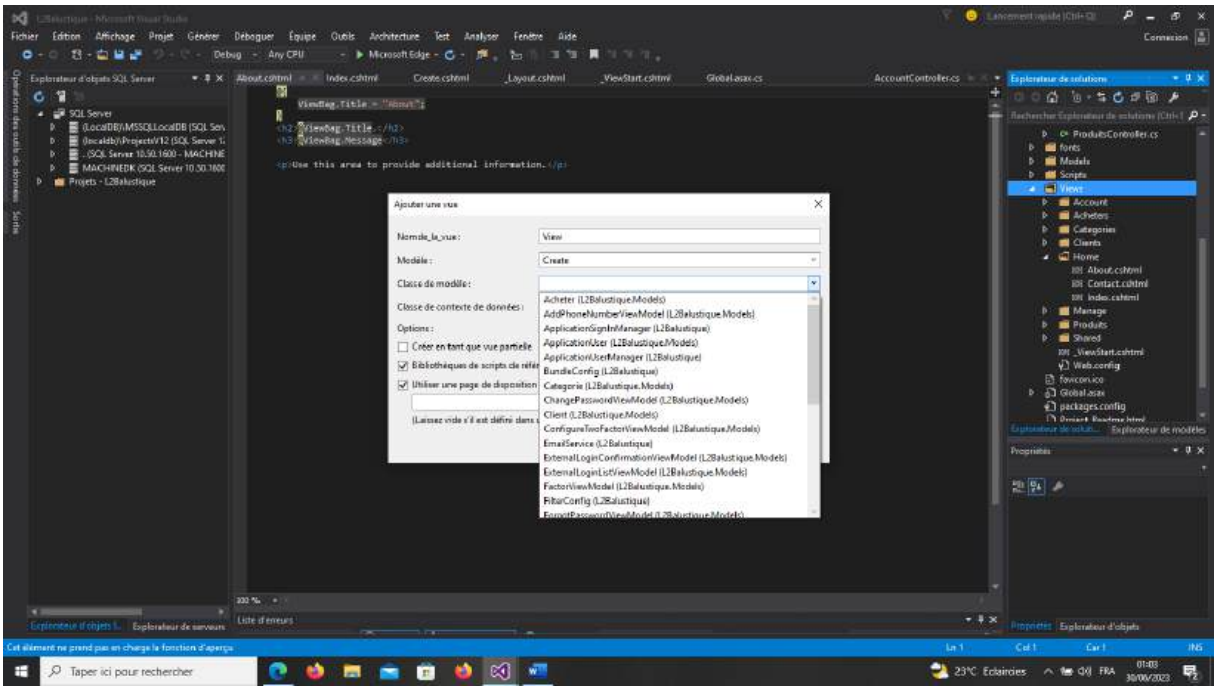
L'exemple que nous venons de voir, bien que fonctionnel, est relativement tiré par les cheveux. En effet, dans la "vraie vie" d'un projet, un pied de page sera généralement soit géré directement depuis le layout, soit déporté dans une vue partielle. En effet, tout comme lorsque l'on utilise le moteur de rendu aspx, Razor permet de créer et de manipuler des vues partielles, et introduit une syntaxe spécifique pour ces vues.

On va donc transformer notre pied de page en vue partielle, et utiliser la méthode RenderPage pour l'appeler.

Pour cela, on va commencer par créer un nouveau fichier cshtml



- Pour créer une vue, procéder de la manière suivante
- Donner un nom à a vue et sélectionnez un modèle qui peut être un objet ou empty (vide) puis cliquer sur ajouter.



Notez que nous avons préfixé le nom de la vue par un caractère souligné (underscore). Cette convention, héritée de WebPages, indique que PiedDePage est une classe partielle, qui ne peut pas être rendue directement. Pour conserver la démonstration la plus simple possible, on va commencer avec une vue partielle qui n'a pas besoin de données supplémentaires, et dont le code est décrit ci-après :

```
<div>Mon pied de page par défaut </div>
```

Oui, c'est tout, rien de plus compliqué que ça.

Notre fichier de layout, lui, va être modifié ainsi :

```
<div id="main">
   @RenderBody()
   @RenderPage("~/Views/Shared/_PiedDePage.cshtml")
// ...
```

Notez que vous arriveriez exactement au même résultat en utilisant la méthode RenderPartial, avec le code suivant : {@Html.RenderPartial("_PiedDePage");}

2.2.5. Mise en forme et appel des vues

a. Appel des vues

Pour faire appel à une vue, il faut passer par un lien ou un bouton appelant une méthode (action d'un contrôleur)

- **Routage conventionnel**

Le routage conventionnel est utilisé avec les contrôleurs et les vues. La route default :

C#

```
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");
```

Le précédent est un exemple *d'itinéraire conventionnel*. Il est appelé *routage conventionnel*, car il établit une *convention* pour les chemins d'URL :

- Le premier segment de chemin, {controller=Home}, est mappé au nom du contrôleur.
- Le deuxième segment, {action=Index}, correspond au nom de l'action .
- Le troisième segment, {id?} est utilisé pour un facultatif id. {id?} le ? rend facultatif. id est utilisé pour mapper à une entité de modèle.

À l'aide de ce default itinéraire, le chemin d'URL :

- /Products/List mappe à l'action ProductsController.List .
- /Blog/Article/17 mappe à BlogController.Article et le modèle lie généralement le id paramètre à 17.

Ce mappage :

- Est basé uniquement sur **les noms du** contrôleur et des actions.
- N'est pas basé sur des espaces de noms, des emplacements de fichiers sources ou des paramètres de méthode.

L'utilisation du routage conventionnel avec l'itinéraire par défaut permet de créer l'application sans avoir à créer un nouveau modèle d'URL pour chaque action. Pour une application avec des actions de style CRUD , ayant une cohérence pour les URL entre les contrôleurs :

- Permet de simplifier le code.
- Rend l'interface utilisateur plus prévisible.

Avertissement

Dans id le code précédent est défini comme facultatif par le modèle d'itinéraire. Les actions peuvent s'exécuter sans l'ID facultatif fourni dans le cadre de l'URL. En règle générale, quand id est omis de l'URL :

- id est défini sur 0 par liaison de modèle.
- Aucune entité n'est trouvée dans la base de données correspondant id == 0 à . **Le routage d'attributs** fournit un contrôle affiné pour créer l'ID requis pour certaines actions et non pour d'autres. Par convention, la

documentation inclut des paramètres facultatifs tels que **id** lorsqu'ils sont susceptibles d'apparaître dans une utilisation correcte.

La plupart des applications doivent choisir un schéma de routage de base et descriptif pour que les URL soient lisibles et explicites. La route conventionnelle par défaut {controller=Home}/{action=Index}/{id?} :

- Prend en charge un schéma de routage de base et descriptif.
- Est un point de départ pratique pour les applications basées sur une interface utilisateur.
- Est le seul modèle d'itinéraire nécessaire pour de nombreuses applications d'interface utilisateur web. Pour les applications d'interface utilisateur web plus volumineuses, une autre route utilisant Areas est souvent tout ce qui est nécessaire.

MapControllerRoute et MapAreaRoute :

- Attribuez automatiquement une valeur de **commande** à leurs points de terminaison en fonction de l'ordre dans lequel ils sont appelés.

Routage des points de terminaison dans ASP.NET Core :

- N'a pas de concept d'itinéraires.
- Ne fournit pas de garanties de classement pour l'exécution de l'extensibilité, tous les points de terminaison sont traités en même temps.

Activez la journalisation pour voir comment les implémentations de routage intégrées, comme Route, établissent des correspondances avec les requêtes.

Le routage des attributs est expliqué plus loin dans ce document.

Plusieurs routes conventionnelles

Plusieurs routes conventionnelles peuvent être configurées en ajoutant d'autres appels à MapControllerRoute et MapAreaControllerRoute. Cela permet de définir plusieurs conventions ou d'ajouter des routes conventionnelles dédiées à une action spécifique, par exemple :

```
app.MapControllerRoute(name: "blog",
            pattern: "blog/{*article}",
            defaults: new { controller = "Blog", action = "Article" });
app.MapControllerRoute(name: "default",
           pattern: "{controller=Home}/{action=Index}/{id?}");
```

L'itinéraire blog dans le code précédent est un **itinéraire conventionnel dédié**. Il s'agit d'un itinéraire conventionnel dédié pour les raisons suivantes :

- Il utilise le routage conventionnel.
- Il est dédié à une action spécifique.

Étant donné que controller et action n'apparaissent pas dans le modèle d'itinéraire "blog/{*article}" en tant que paramètres :

- Ils peuvent uniquement avoir les valeurs { controller = "Blog", action = "Article" } par défaut .
- Cet itinéraire est toujours mappé à l'action BlogController.Article.

/Blog, /Blog/Articleet /Blog/{any-string} sont les seuls chemins d'URL qui correspondent à l'itinéraire du blog.

L'exemple précédent :

- blog l'itinéraire a une priorité plus élevée pour les correspondances que l'itinéraire default , car il est ajouté en premier.
- Il s'agit d'un exemple de routage de style Slug où il est courant d'avoir un nom d'article dans le cadre de l'URL.

✓ **Résolution des actions ambiguës**

Lorsque deux points de terminaison correspondent via le routage, le routage doit effectuer l'une des opérations suivantes :

- Choisissez le meilleur candidat.
- Levée d'une exception.

Par exemple :

```
public class Products33Controller : Controller
{
    public IActionResult Edit(int id)
    {
        return ControllerContext.MyDisplayRouteInfo(id);
    }

    [HttpPost]
    public IActionResult Edit(int id, Product product)
    {
        return ControllerContext.MyDisplayRouteInfo(id, product.name);
    }
}
```

Le contrôleur précédent définit deux actions qui correspondent :

- Chemin d'accès de l'URL /Products33/Edit/17

- Données de routage { controller = Products33, action = Edit, id = 17 }.

Il s'agit d'un modèle classique pour les contrôleurs MVC :

- Edit(int) affiche un formulaire pour modifier un produit.
- Edit(int, Product) traite le formulaire publié.

Pour résoudre l'itinéraire correct :

- Edit(int, Product) est sélectionné lorsque la requête est un http POST.
- Edit(int) est sélectionné lorsque le verbe HTTP est autre chose. Edit(int) est généralement appelé via GET.

, HttpPostAttribute[HttpPost] est fourni pour le routage afin qu'il puisse choisir en fonction de la méthode HTTP de la requête. Le HttpPostAttribute fait Edit(int, Product) une meilleure correspondance que Edit(int).

Il est important de comprendre le rôle des attributs tels que HttpPostAttribute. Des attributs similaires sont définis pour d'autres verbes HTTP. Dans le routage conventionnel, il est courant que les actions utilisent le même nom d'action lorsqu'elles font partie d'un formulaire d'affichage, envoi de flux de travail de formulaire. Par exemple, consultez Examiner les deux méthodes d'action Modifier.

Si le routage ne peut pas choisir un meilleur candidat, un AmbiguousMatchException est levée, répertoriant les points de terminaison correspondants multiples.

- ✓ Noms de routage réservés

Les mots clés suivants sont des noms de paramètres d'itinéraire réservés lors de l'utilisation de contrôleurs ou Razor de pages :

- action
- area
- controller
- handler
- page

L'utilisation page comme paramètre d'itinéraire avec le routage d'attribut est une erreur courante. Cela entraîne un comportement incohérent et confus avec la génération d'URL.

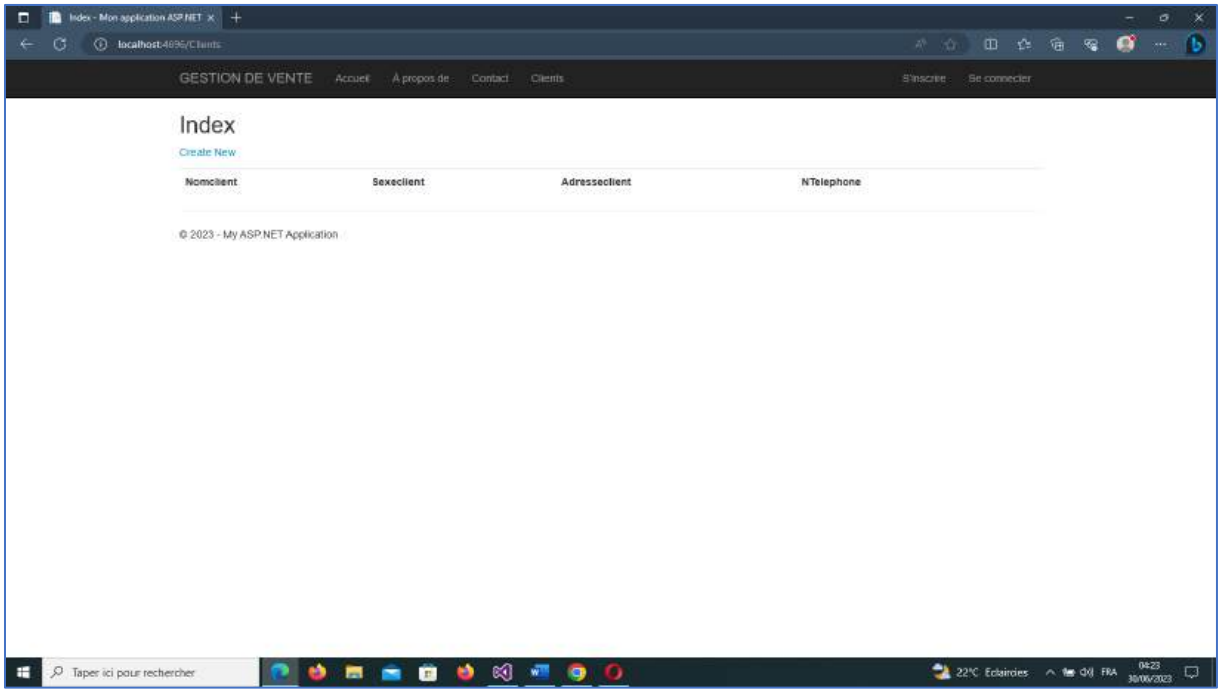
Exemple d'un ; ClientControllers.cs , dans ce contrôleur se trouve une méthode index de l'actionResultat

public ActionResult Index()

```
    {
        return View(db.Client.ToList());
    }
```

Méthode d'appel d'un lien home

@Html.ActionLink("Nom d'application", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })




```
<div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse"
data-target=".navbar-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button><img src=""/>
                @Html.ActionLink("GESTION DE VENTE", "Index", "Home", new { area = ""
}, new { @class = "navbar-brand" })
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li>@Html.ActionLink("Accueil", "Index", "Home")</li>
                    <li>@Html.ActionLink("À propos de", "About", "Home")</li>
                    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                    <li> @Html.ActionLink("Clients", "Index", "Clients")</li>
                     <li> @Html.ActionLink("Categories","Index", "Catégories")</li>
                </ul>
                @Html.Partial("_LoginPartial")
            </div>
        </div>
    </div>
```

Explication de notre lien

 @Html.ActionLink("Clients", "Index", "Clients")

@Html.ActionLink : syntaxe pour générer une action de lien

- Segment 1 : Etiquette à ajouter dans le menu
- Segment 2 : Méthodes de notre contrôleur
- Segment 3 : Nom de la vue.

b. Mise en forme de la page

Nous allons utiliser Bootstrap pour embellir ou améliorer nos pages, pour plusieurs de connaissance, consulter le site office de Bootstrap (documentations versions 3 ou 4). Pour notre cas nous allons prendre client précisément la page index. Cshtml. Transformer un lien sous forme d'un bouton. Par défaut vous aurez ceci :

```
<p>
    @Html.ActionLink("Create New", "Create")
</p>
```

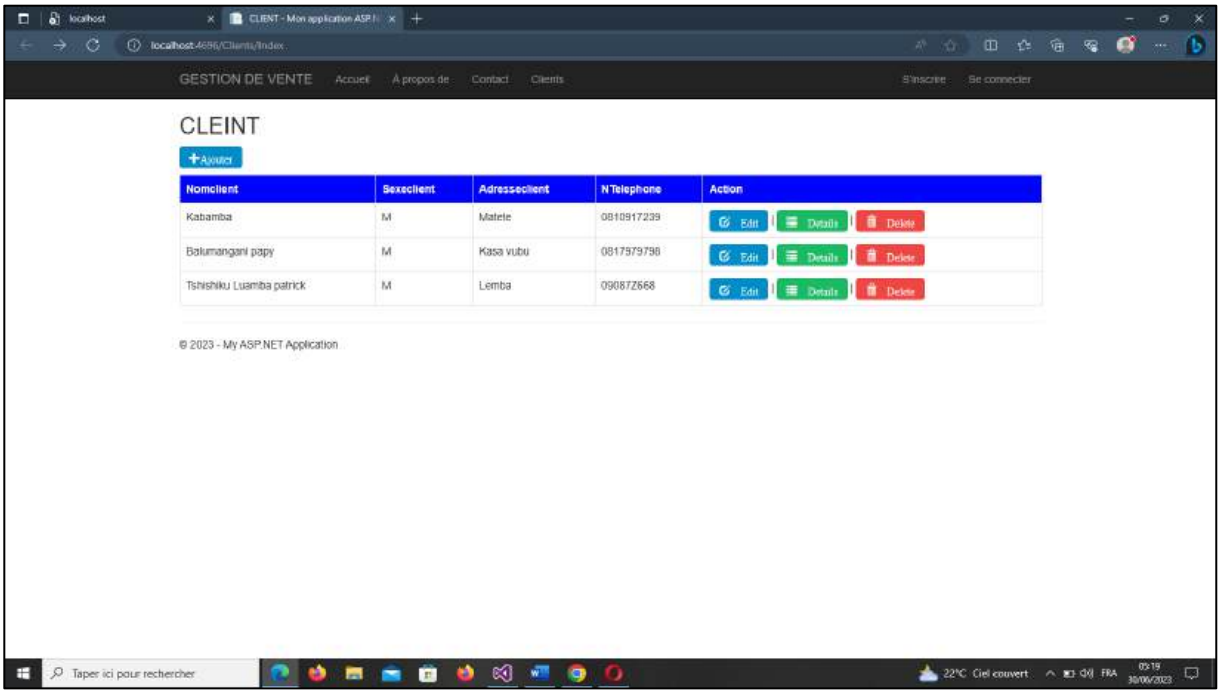
Après transformation vous auriez ceci

```
<p>
    @Html.ActionLink("Ajouter client", "Create", new{aria="", new{@class="btn btn-
primary"})
</p>
```

Mise en forme du table des données

```
@model IEnumerable<L2Balustique.Models.Client>

@{
    ViewBag.Title = "CLIENT";
}

<h2>CLEINT </h2>

<p>
    @Html.ActionLink("Ajouter client", "Create", new { aria = ""}, new { @class = "btn
btn-primary glyphicon glyphicon-plus"})
</p>
<table class="table table-bordered">
    <tr  style="background-color:blue; color:#ffffff">
        <th>
            @Html.DisplayNameFor(model => model.Nomclient)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Sexeclient)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Adresseclient)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.NTelephone)
        </th>
```

```
            <th></th>
        </tr>

@foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Nomclient)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Sexeclient)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Adresseclient)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.NTelephone)
            </td>
            <td>
                @Html.ActionLink(" Edit ", "Edit", new { id=item.Numclient }, new { @class
= "btn btn-primary glyphicon glyphicon-edit" }) |
                @Html.ActionLink(" Details ", "Details", new { id=item.Numclient },new {
@class = "btn btn-success glyphicon glyphicon-list" }) |
                @Html.ActionLink(" Delete ", "Delete", new { id=item.Numclient },new { @class
= "btn btn-danger glyphicon glyphicon-trash"})
            </td>
        </tr>
}

</table>
```



Conclusion