TP1 ASP.Net MVC Framework

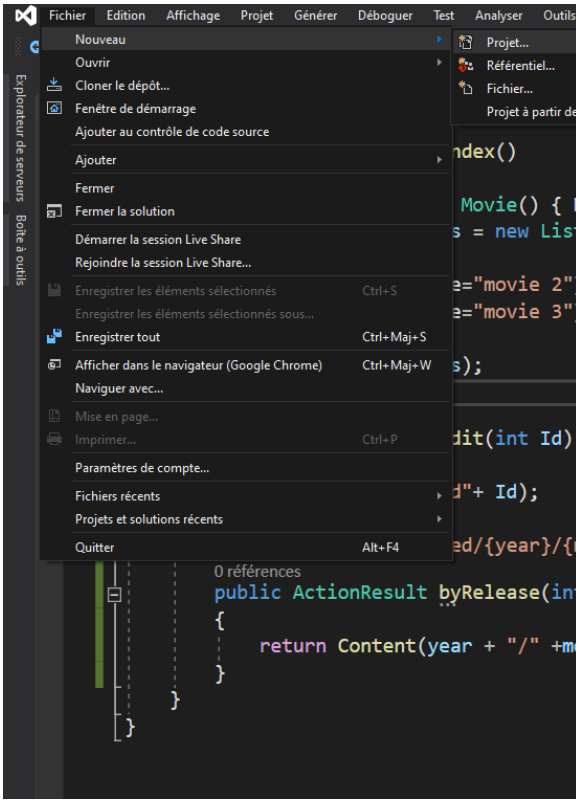
MVC est synonyme de *modèle-vue-contrôleur*. MVC est un pattern pour développer des applications qui sont bien architecturé, testables et facile à entretenir. MVC veut dire :

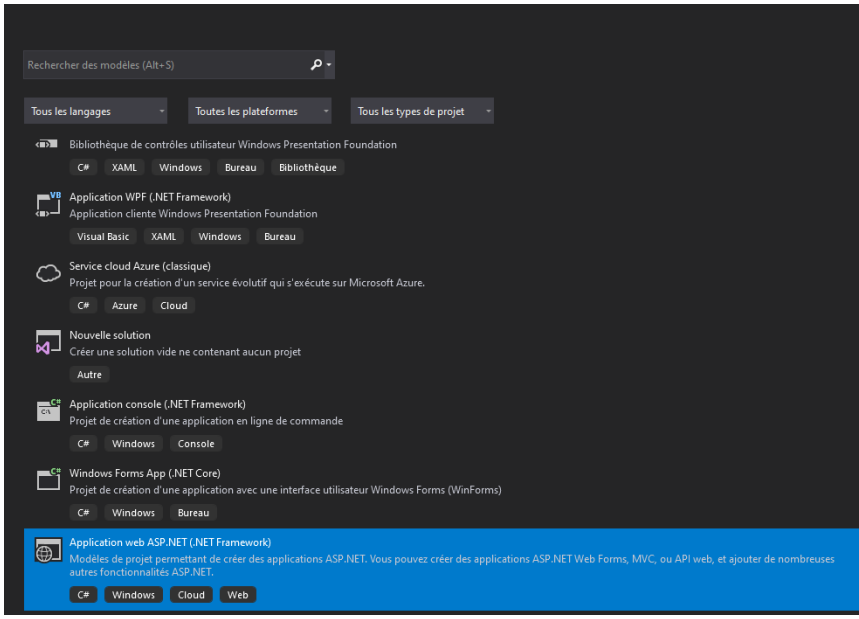
- **M**odel : Classes qui représentent les données de l'application et qui utilisent une logique de validation pour faire respecter les règles de métier.
- **V**iew : fichiers de modèle que votre application utilise pour générer dynamiquement les réponses HTML.
- **C**ontroller : Classes qui gèrent les demandes entrantes de navigateur, récupérer des données de modèle et ensuite spécifier des modèles de vue qui retournent une réponse au navigateur.

Nous couvrons tous ces concepts dans cette série de tutoriel et vous montrons comment s'en servir pour construire une application Web.

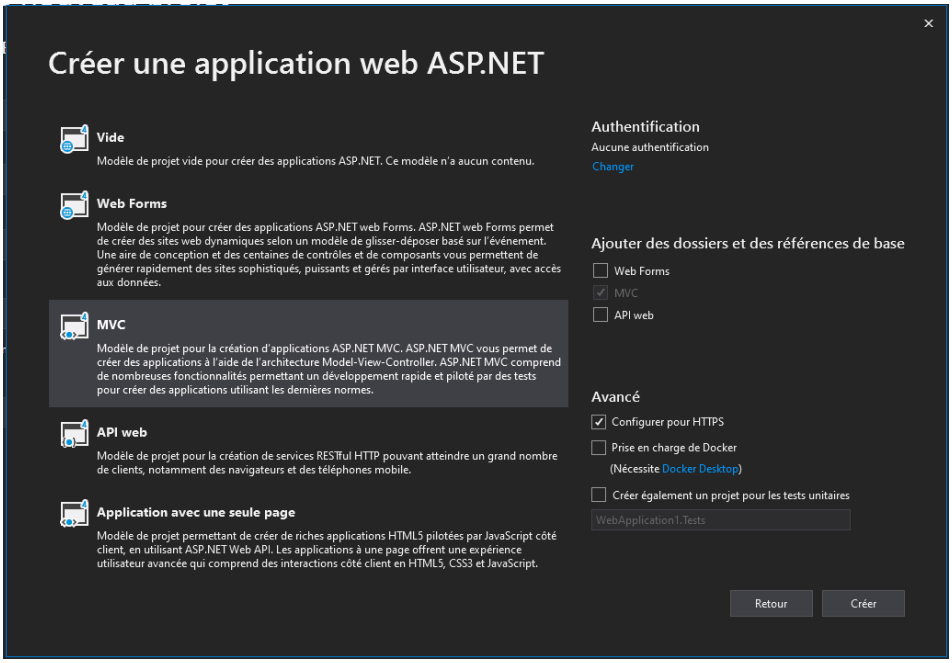
Création d'un projet ASP MVC

1. Cliquez sur **New Project**, puis sélectionnez Visual c# sur la gauche, puis **Web** et puis sélectionnez **ASP.NET Application Web**. Nommez votre projet «**AspMovieApp**» et puis cliquez sur **OK**.



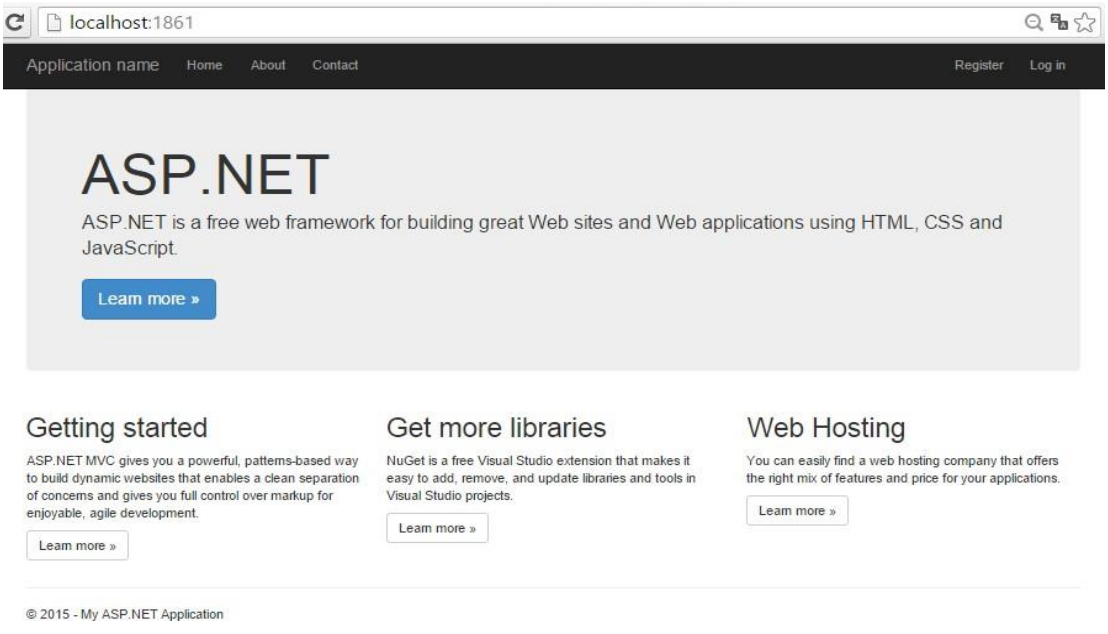


2. Dans la boîte de dialogue du nouveau projet, cliquez sur **MVC** et puis cliquez sur **OK**.
3. Changer l'authentification pour utiliser les comptes d'utilisateur individuels



4. Cliquez sur F5 pour démarrer le débogage. F5 oblige Visual Studio à lancer IIS Express et exécuter votre application web Visual Studio puis lance un navigateur et ouvre la page d'accueil de l'application.

Comme vous remarquez la barre d'adresse du navigateur contient : **localhost:port#** et non pas quelque chose comme **example.com**. C'est parce que **localhost** pointe toujours vers votre ordinateur local, qui dans ce cas exécute l'application que vous venez de créer. Lorsque Visual Studio exécute un projet web, un port aléatoire est utilisé pour le serveur web. Dans l'image ci-dessus, le numéro de port est 1861. Lorsque vous exécutez l'application, vous verrez un autre numéro de port.

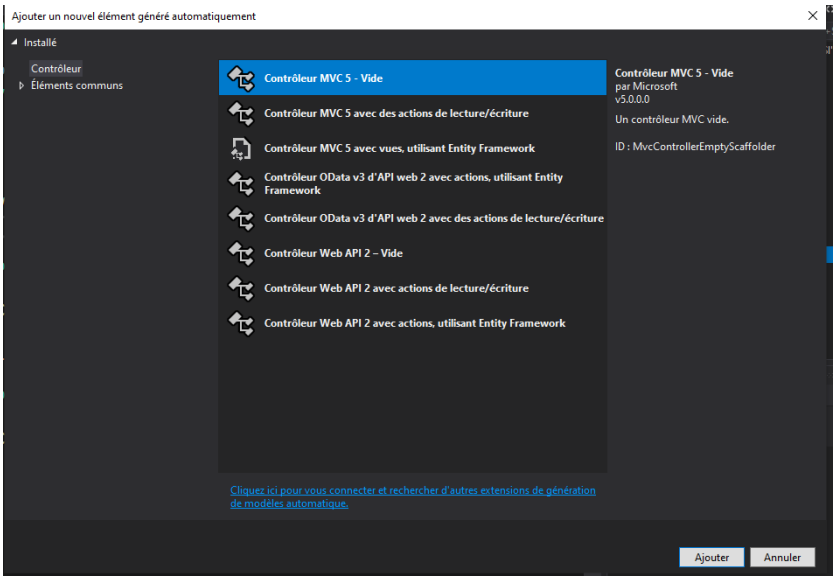


Création des classes du domaine (Model)

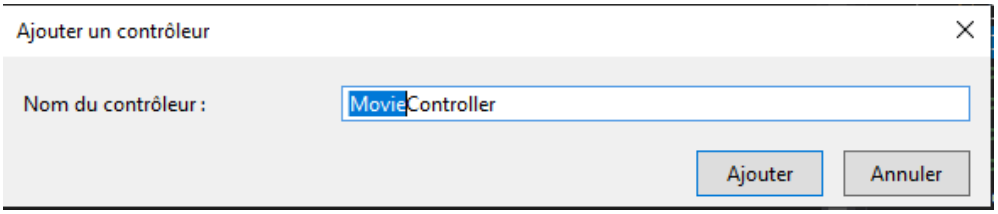
5. Créer une classe Movie (int Id, string Name)

Création d'un contrôleur

6. Nous allons commencer par créer une classe de contrôleur. Dans de **L'Explorateur de solutions**, faites un clic droit sur le dossier controllers, puis cliquez sur **Add**, puis **controller**.



7. Puis cliquer sur **MVC 5 Controller – Empty** ensuite sur **Add**



8. Nommez le nouveau contrôleur **MovieController**

```
namespace ApplicationASPFromScratchGLSI.Controllers
{
    0 références
    public class MovieController : Controller
    {
        // GET: Movie/Index
        0 références
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

Dans **L'Explorateur** un nouveau fichier a été créé nommé ***MovieController.cs*** et un nouveau dossier ***Views\Movie***. Le contrôleur est ouvert dans l'IDE.

9. Remplacez le contenu du fichier avec le code suivant.

```
public ActionResult Index()
    {
        Movie movie = new Movie() { Name = "movie 1" };
        List<Movie> movies = new List<Movie>()
        {
            new Movie{Name="movie 2"},
            new Movie{Name="movie 3"},
        };
        return View(movies);
    }
```

Le contrôleur est appelé **MovieController** et la première méthode ou **Action** est nommée **Index**. Nous allons l'appeler à partir d'un navigateur.

Configuration des routes

ASP.NET MVC invoque les différents contrôleurs et leurs différentes actions selon l'URL entrante.

La logique de routage URL utilisée par défaut dans ASP.NET MVC utilise un format comme celui-ci pour déterminer quel code invoquer : **/[Controller]/[ActionName]/[Parameters]**

Vous définissez le format de routage dans le fichier **App_Start/RouteConfig.cs.**

Lorsque vous exécutez l'application et vous ne fournissez pas des segments d'URL, il utilise par défaut le contrôleur « Home » et la méthode d'action « Index » spécifié dans la section valeurs par défaut du code ci- dessous.

```
public static void RegisterRoutes(RouteCollection routes)
{
    routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

    routes.MapRoute(
        name: "Default",
        url: "{controller}/{action}/{id}",
        defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional });
}
```

- La première partie de l'URL détermine le contrôleur qu'on veut exécuter. Si ***/Movie*** alors la classe **MovieController** sera appelée.
- La deuxième partie de l'URL détermine la méthode d'action à Exécuter. Si ***/Movie/Index*** alors la méthode Index de la classe MovieController sera exécutée.

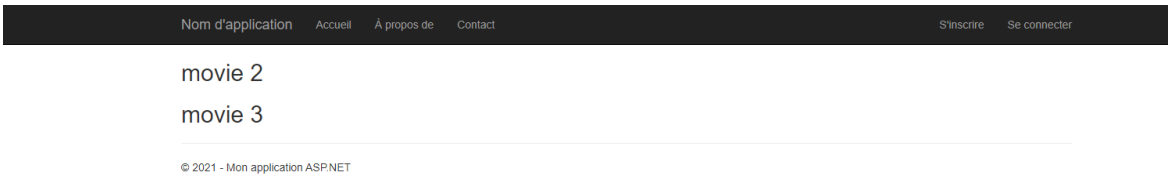
Une méthode nommée Index est la méthode par défaut et elle sera appelée sur un contrôleur si on n'a pas spécifié la méthode explicitement.

- La troisième partie du segment URL (Parameters) c'est pour acheminer les données.

Accédez à http://localhost:xxxx/Movie/Index.

La méthode d'action Index s'exécute et retourne les noms des films

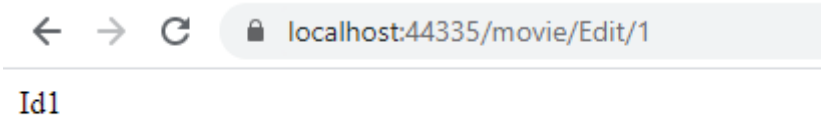
Le mappage de MVC par défaut est /[Controller]/[ActionName]/[Parameters]. Pour cette URL, le contrôleur est Movie et Index est l'action. Nous n'avons pas utilisé les paramètres [Parameters] dans l'URL.



Nous allons à présent ajouter un Action à notre exemple afin de passer des informations de paramètre de l'URL au contrôleur (par exemple, */Movie/Edit/1*.

10. Ajouter une méthode Edit pour inclure un paramètre « Id » indiqué ci-dessous.

```
public ActionResult Edit(int Id)
{
    return Content("Id"+ Id);
}
```



Dans l'exemple ci-dessus, le segment d'URL (Parameters) est utilisé et est transmis sous forme de paramètres de requête.

11. Modifiez le route pour inclure l'URL Movie/released/2020/03 pour une Action ByRelease ayant comme arguments (month, year).

```
public ActionResult byRelease(int year, int month)
        {
            return Content(year + "/" +month);
         }
```

```
public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
            routes.MapMvcAttributeRoutes();
            routes.MapRoute(
                "MoviesByDate",
                "movies/released/{year}/{month}",

                new { Controller = "Movie", Action = "byRelease" });
            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional }
            );
         }
```

12. Exécutez l'application et entrez l'adresse URL suivante : **http://localhost:xxxx/Movie/released/2020/03**

Dans les applications ASP.NET MVC, il est plus courant de passer des paramètres en tant que **paramètre de routes** (comme nous le faisions avec ID ci-dessus) que les transmettre sous forme de chaînes de requête.

Vous pouvez spécifier des routes à l'aide de l'attribut Route de cette façon

```
[Route("movies/released/{year}/{month}")]
        public ActionResult byRelease(int year, int month)
        {
            return Content(year + "/" +month);
         }
```

Mais il faut activer ce mode de routage dans la méthode RegisterRoute :

```
public class RouteConfig
{
  public static void RegisterRoutes(RouteCollection routes)
  {
    routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

    routes.MapMvcAttributeRoutes();
```

Ajout d'une vue

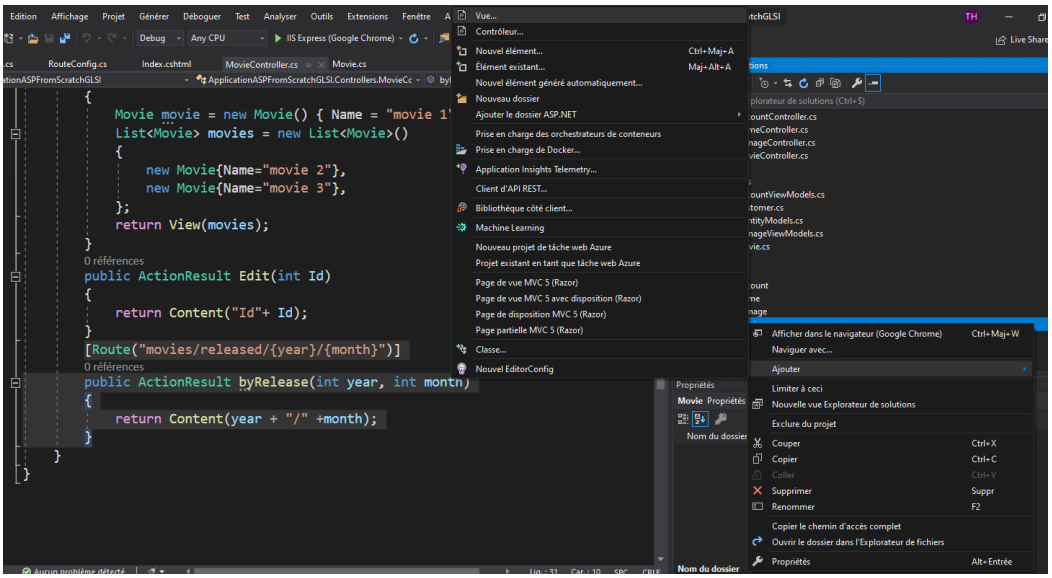
Dans cette section, nous allons voir comment retourner des vues afin d'encapsuler le processus d'envois d'Html aux clients. Ces vues utiliseront le moteur d'affichage Razor et auront une extension de fichier *.cshtml* et fourniront un moyen élégant pour générer le HTML de sortie à l'aide de c#.

1. Actuellement, la méthode Indexretourne une Vue, comme illustré dans le code suivant :

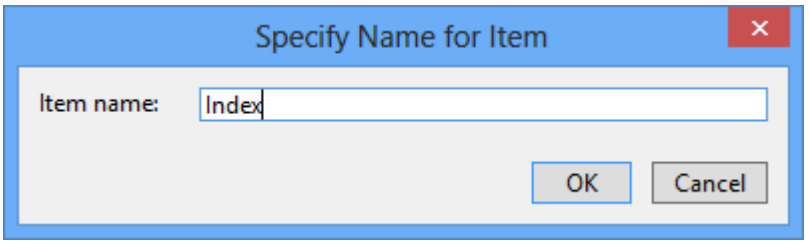
```
public ActionResult Index()
{
        return View();
}
```

La méthode de Index ci-dessus utilise un modèle d'affichage pour générer une réponse en HTML qui sera envoyer au navigateur du client. Les Méthodes des contrôleurs (également connues comme des action), telle que la méthode de Index ci-dessus, retournent généralement un ActionResult (ou une classe dérivée de ActionResult), et non pas des types primitifs comme les chaînes de caractères.

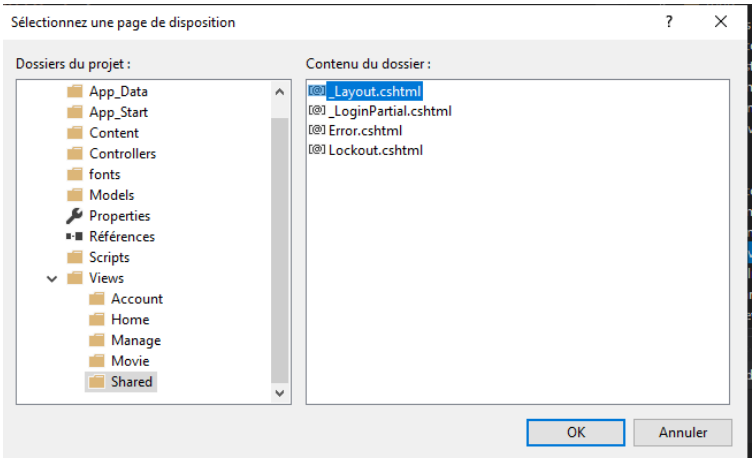
2. Clic droit sur le dossier *Views\Movie* et cliquez sur **Ajouter**, puis cliquez sur **MVC 5 Voir Page avec (layout Razor)**.



3. Dans la boîte de dialogue **Spécifier le nom d'élément,** entrez *Index* puis cliquez sur **OK**.



4. Dans la boîte de dialogue **Sélectionner une Page de disposition**, acceptez la valeur par défaut **_Layout.cshtml** , puis cliquez sur **OK**.



Dans la boîte de dialogue ci-dessus, le dossier *Views\Shared* est sélectionné dans le volet gauche. Si vous aviez un fichier de mise en page personnalisée dans un autre dossier, vous pouvez le sélectionner. Nous allons parler du fichier layout plus tard dans le tutoriel
Le fichier *Views\Movie\Index.cshtml* est créé.

5. Ajoutez le code mis en évidence au fichier index.cshtml.

```
@model IEnumerable<ApplicationASPFromScratchGLSI.Models.Movie>
@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

@foreach(var item in Model)
{
<h2>@item.Name</h2>
}
```

6. Exécutez le fichier sur le navigateur