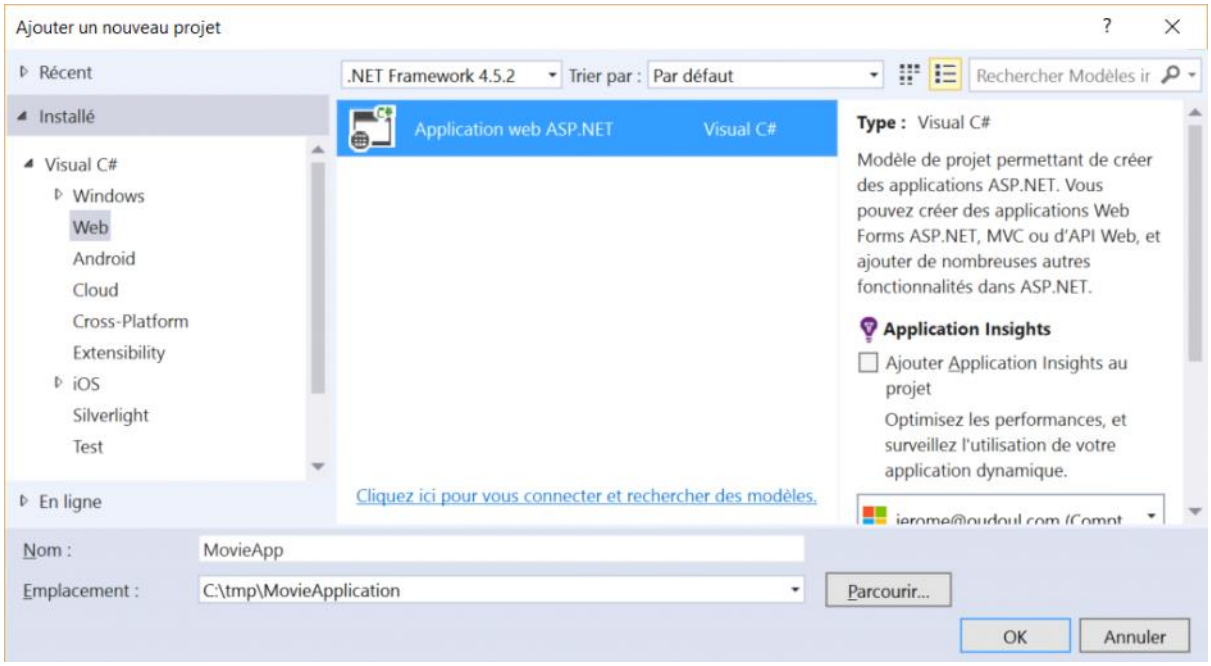
TP1

mardi 26 avril 2016 22:08

TP1

Création d'un projet ASP.NET MVC

Créer nouveau projet web ASP.NET MovieApplication



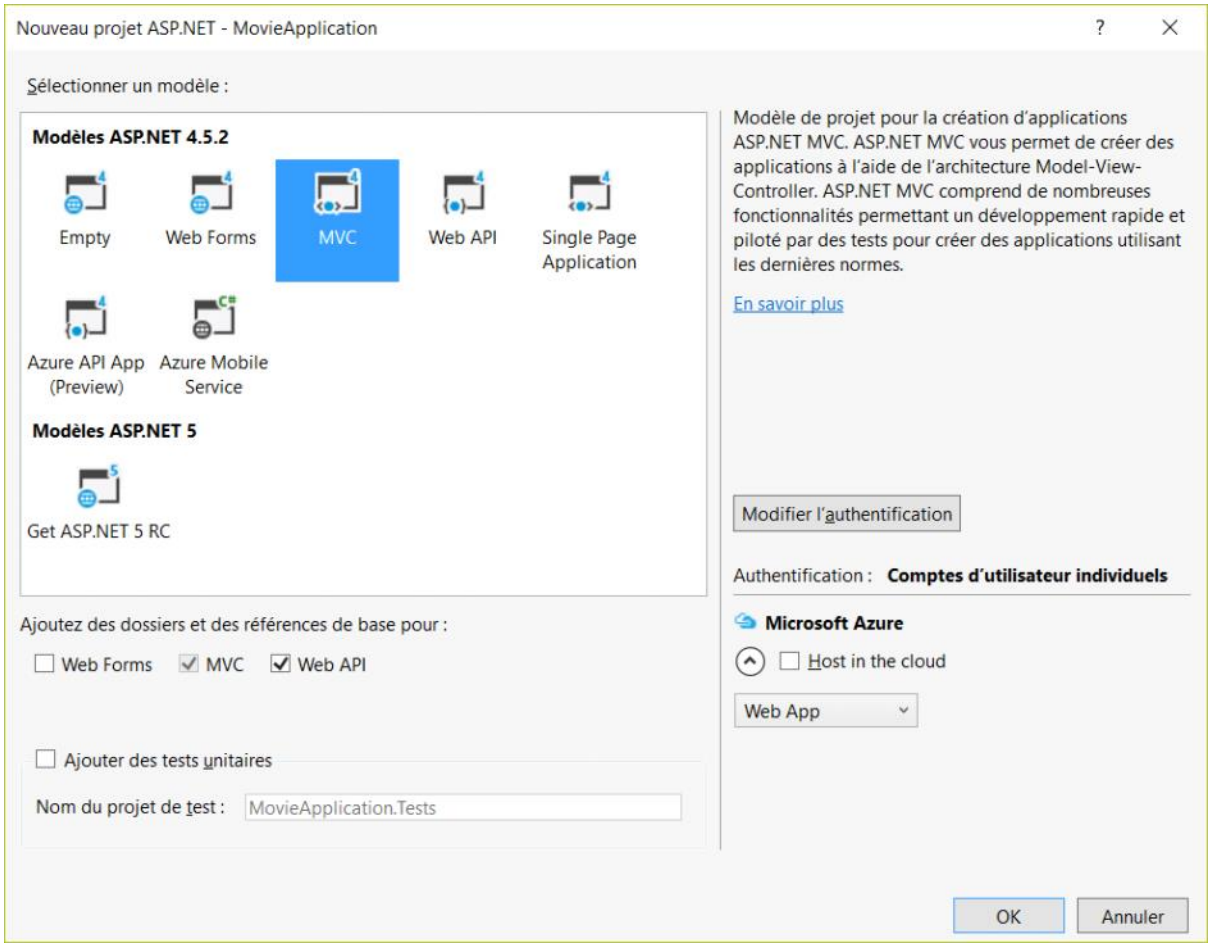
ASP.NET 4.5.2 **MVC**, Authentification : Comptes d'utilisateur individuels

(cliquez Modifier l'authentification si besoin)

Cocher Web API (sera utile dans le TP2)

Laisser décoché le projet Tests Unitaires

Décocher Host in the cloud



Dans VS, choisir Google Chrome par exemple puis lancer l'appli via F5 pour la visualiser.

Changer le nom de l'application ("Nom de l'application") dans _Layout.cshtml en Moviez

Modifier le ViewBag.Title de la view Home/Contact.cshtml en "Your private contact page"

Ajouter l'annotation [Authorize] sur l'action method *Contact* du controller HomeController.cs, build la solution puis recharger la view Contact

Grâce au lien *S'inscrire* dans le header, créer un compte jerome@softeam.fr sans mot de passe

Renseigner un mot de passe trop court, toto par exemple => message d'erreur de validation en rouge

Aller voir la validation dans le Model AccountViewModels.cs

Montrer la construction de l'attribut [StringLength]

Changer le mdp en **Softeam123!** pour répondre aux critères de sécurité

Dans l'explorateur de solution, cliquer sur "Afficher tous les fichiers" puis dans App_Data, refresh et clique droit "Ouvrir" sur le .mdf

Cela devrait ouvrir l'explorateur de serveurs sinon :

View > SQL Server Object Explorer pour explorer LocalDB

Table AspNetUsers => Afficher les données

Montrer le mot de passe du user créé

Retourner sur la page Contact qui s'affiche cette fois, maintenant qu'on est loggué.

Dans _Layout.cshtml, montrer l'appel à la partialView _LoginPartial

Dans _LoginPartial.cshtml , montrer le @Html.ActionLink("Bonjour...

et le détailler :

Controller/Action, paramètre null de la route (car optionnel)

Ajouter le HtmlAttribute @class = "glyphicon glyphicon-user" (Bootstrap)

Resizer la fenêtre pour montrer l'apparition du menu hamburger en haut à droite

Model

Ajoutons un Model : Add class Movie.cs

Ajouter le code de la classe Movie et MovieDBContext

```
public class Movie
{
     public int ID { get; set; }
     [Required]
     public string Title { get; set; }
     public DateTime ReleaseDate { get; set; }
     public string Genre { get; set; }
     public decimal Price { get; set; }
}

    public class MovieDBContext : DbContext
    {
        public DbSet<Movie> Movies { get; set; }
    }
```

On utilise l'ORM Entity Framework en Code First 'est-à-dire qu'on commence par écrire le code .NET et déclenche la création de la BDD depuis VS basé sur ce code.

On va activer **Code First Migrations** (https://msdn.microsoft.com/en-us/data/jj591621)

Lancer Console du gestionnaire de package et activer Code First Migrations dans le projet comme suit :

```
PM > Enable-Migrations -ContextTypeName
MovieApplication.Models.MovieDBContext
```

(en cas d'échec, supprimer le mdf, modifier le nom du context dans la classe Movie.cs et idem dans le web.config et réessayer la commande ci-dessus)

Ouvrir le dossier Migrations > Configuration.cs

Dans la méthode Seed, renseigner quelques films :

```
context.Movies.AddOrUpdate(
              m => m.Title,
              new Movie { ID = 1, Title = "Frozen", ReleaseDate = new
```

```
DateTime(2013, 1, 1), Price = 15 },
              new Movie { ID = 2, Title = "Star Wars VII", ReleaseDate =
new DateTime(2015, 1, 1), Price = 35 },
              new Movie { ID = 3, Title = "Batman", ReleaseDate = new
DateTime(1989, 1, 1), Price = 10 },
              new Movie { ID = 4, Title = "Toy Story", ReleaseDate = new
DateTime(2013, 1, 1), Price = 16 },
              new Movie { ID = 5, Title = "Home Alone", ReleaseDate = new
DateTime(2015, 1, 1), Price = 28 },
              new Movie { ID = 6, Title = "Avatar", ReleaseDate = new
DateTime(1989, 1, 1), Price = 19 },
              new Movie { ID = 7, Title = "Aliens", ReleaseDate = new
DateTime(2013, 1, 1), Price = 12 },
              new Movie { ID = 8, Title = "Mars Attacks", ReleaseDate =
new DateTime(2015, 1, 1), Price = 8 },
              new Movie { ID = 9, Title = "Zorro", ReleaseDate = new
DateTime(1989, 1, 1), Price = 9 },
              new Movie { ID = 10, Title = "Candyman", ReleaseDate = new
DateTime(2013, 1, 1), Price = 5 },
              new Movie { ID = 11, Title = "Lost in translation",
ReleaseDate = new DateTime(2015, 1, 1), Price = 10 },
              new Movie { ID = 12, Title = "Contact", ReleaseDate = new
DateTime(1992, 1, 1), Price = 11 }
            );
```

Passer le AutomaticMigrationsEnabled à true

Répercuter dans la BDD ces ajouts en attente via la commande suivante :

`PM > Update-Database -ConnectionStringName DefaultConnection -Verbose`

Rafraichir l'explorateur de serveurs pour voir apparaître la table Movies

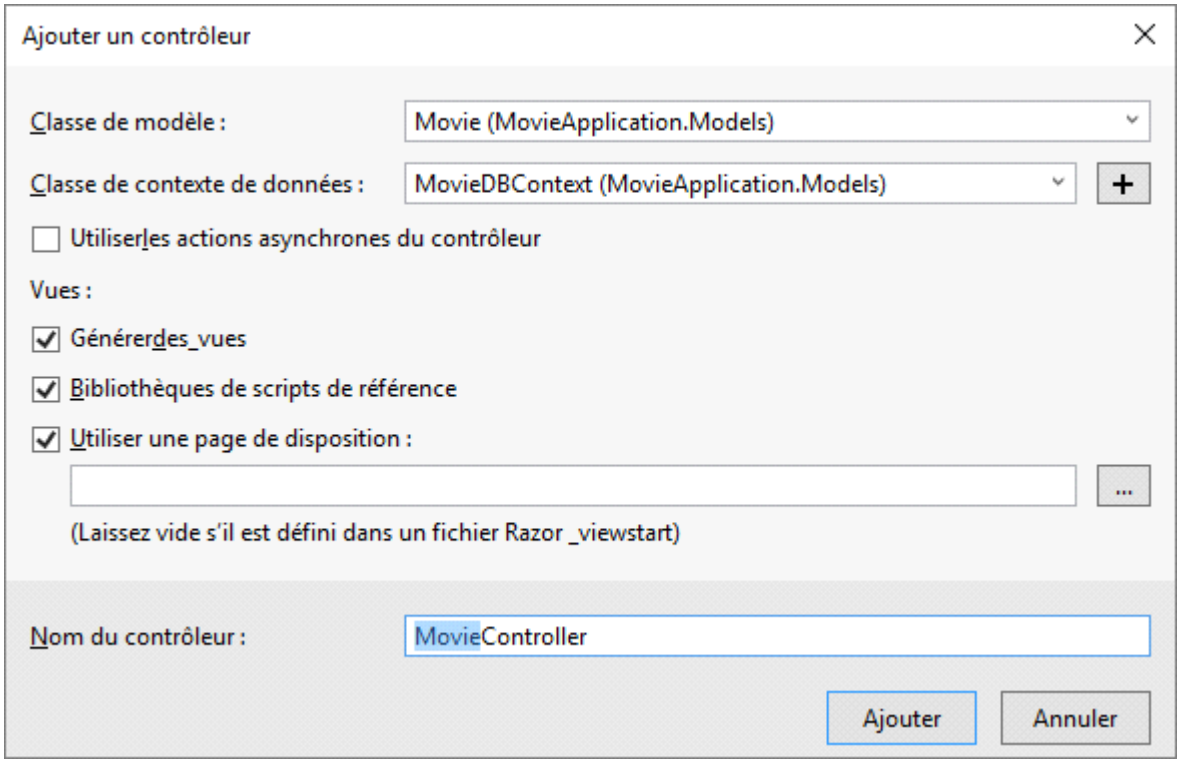
NB : ouvrir un Command Prompt (cmd), taper `SqlLocalDb info` pour lister les noms de serveurs.

`Dans l'Explorateur de serveurs de VS, se connecter au serveur (localdb)\MSSQLLocalDB`

Controller

Controllers > Add Controller

Controller MVC5 avec vues, utilisant Entity Framework



Le scaffolding crée toutes les action methods et toutes les views

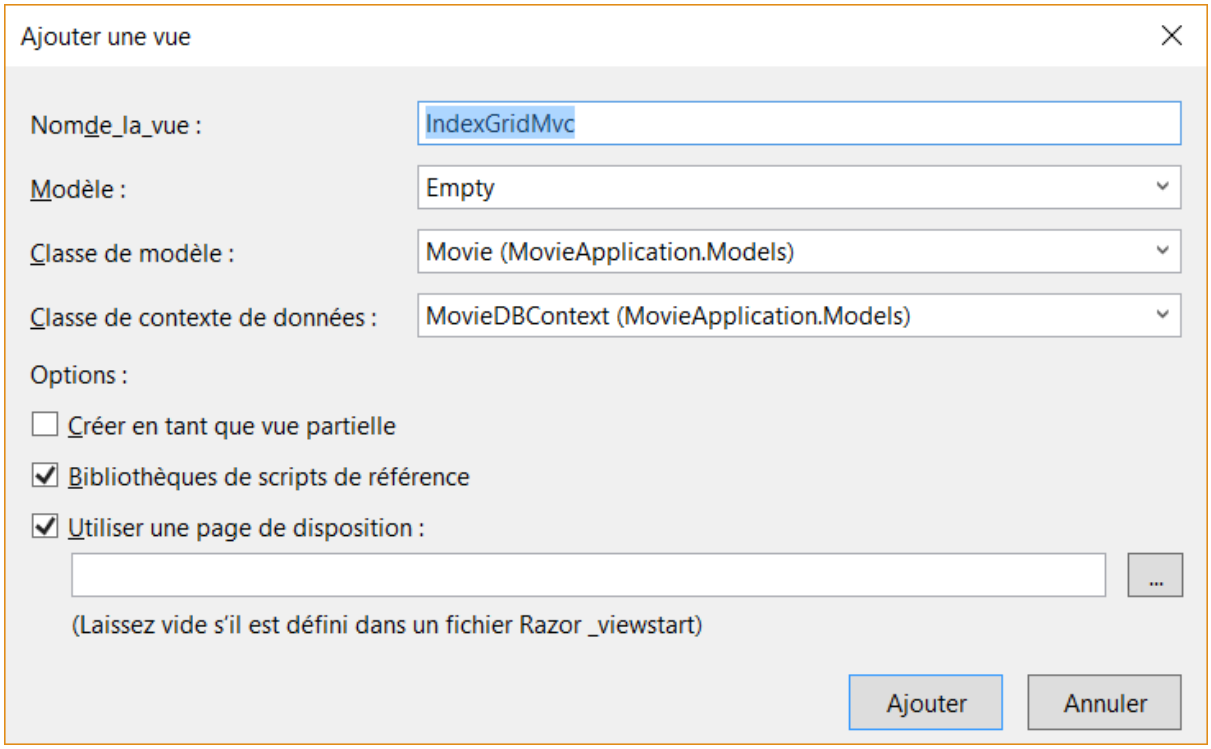
Accéder à /Movie dans le browser

Installer Grid.MVC via NuGet (dernière prerelease)

https://gridmvc.codeplex.com/wikipage?title=Quick%20start

Consulter *Grid.mvc.readme*

Créer une nouvelle View :



Ajouter @using GridMvc.Html en haut de cette nouvelle View

Et passer le Model en :

*@model **IEnumerable**<MovieApplication.Models.Movie>*

D'abord ajouter la ligne

@Html.Grid(Model).AutoGenerateColumns()

Modifier l'action method Index de MovieController.cs :

return View("IndexGridMvc", db.Movies.ToList());

Et afficher la page http://localhost:53338/Movie

Ajouter @using GridMvc.Sorting

Puis remplacer la ligne précédente par le bloc de *columns.Add*

```
@Html.Grid(Model).Named("MoviesGrid").Columns(columns =>
{
    columns.Add(m => m.Title).Titled("Movie
title").SetWidth(110).SortInitialDirection(GridSortDirection.Ascending);
    columns.Add(m => m.Genre);
    columns.Add(m => m.ReleaseDate).Titled("Year").Format("{0:yyyy}");
    columns.Add(m => m.Price);
}).WithPaging(5).Sortable().Filterable()
```

Dans BundleConfig.cs , ajouter après les bundles jquery:

http://www.asp.net/mvc/overview/performance/bundling-and-minification

```
bundles.Add(new ScriptBundle("~/bundles/gridmvc").Include(
                        "~/Scripts/gridmvc*"));
```

Dans _Layout.cshtml , enrichir et remonter sous <body> le bloc comme suit :

http://stackoverflow.com/questions/15458523/how-to-insert-jquery-code-uncaught-referenceerror-is-not-defined-in-view-raz

```
@Scripts.Render("~/bundles/jquery")
@Scripts.Render("~/bundles/gridmvc")
@Scripts.Render("~/bundles/bootstrap")
@RenderSection("scripts", required: false)
```

Ajouter colonnes Edit et Delete dans la view IndexGridMvc :

```
columns.Add()
               .Encoded(false)
               .Sanitized(false)
               .SetWidth(100)
               .RenderValueAs(m =>
```

```
            @<b>
                @Html.ActionLink("Edit movie", "Edit", new { id = m.ID },
new { @class = "modal_link" })
            </b>);
    columns.Add()
               .Encoded(false)
               .Sanitized(false)
               .SetWidth(80)
               .RenderValueAs(m =>
            @<b>
                @Html.ActionLink("Delete", "Delete", new { id = m.ID },
new { @class = "modal_link" })
            </b>);
```

Ajouter un lien Create :

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