# TP2

lundi 30 mai 2016 00:06

# TP2

**WebAPI 2 + Postman**

Remarques : dans la route :

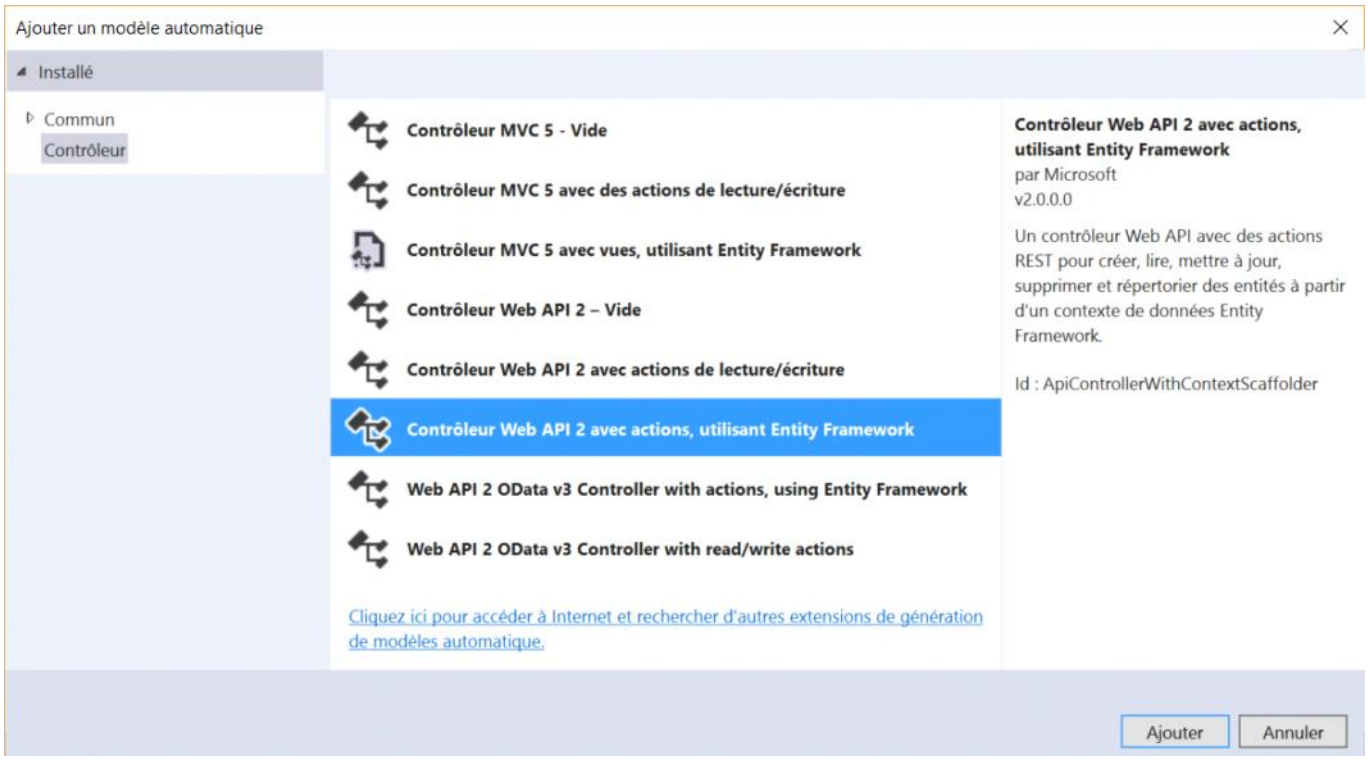
+ MapHttpRoute

+ pas d'action

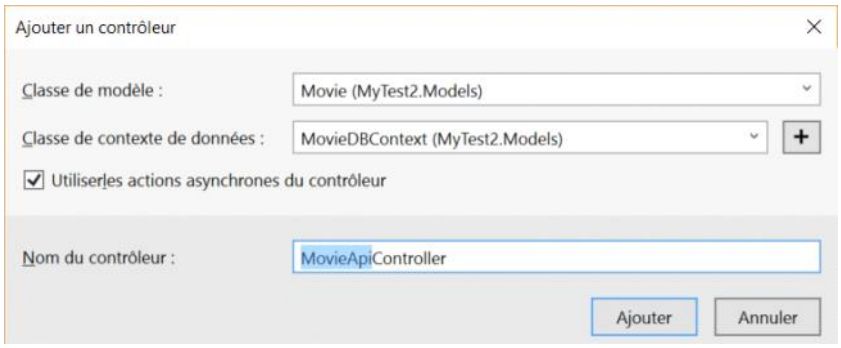
+ path démarre par api/

Possibilité d'avoir 2 méthodes Get aux signatures différentes.

Ajouter un Controller :




Cocher "Utiliser les actions asynchrones"




Ouvrir

http://localhost:55559/api/MovieApi

dans Chrome et Postman

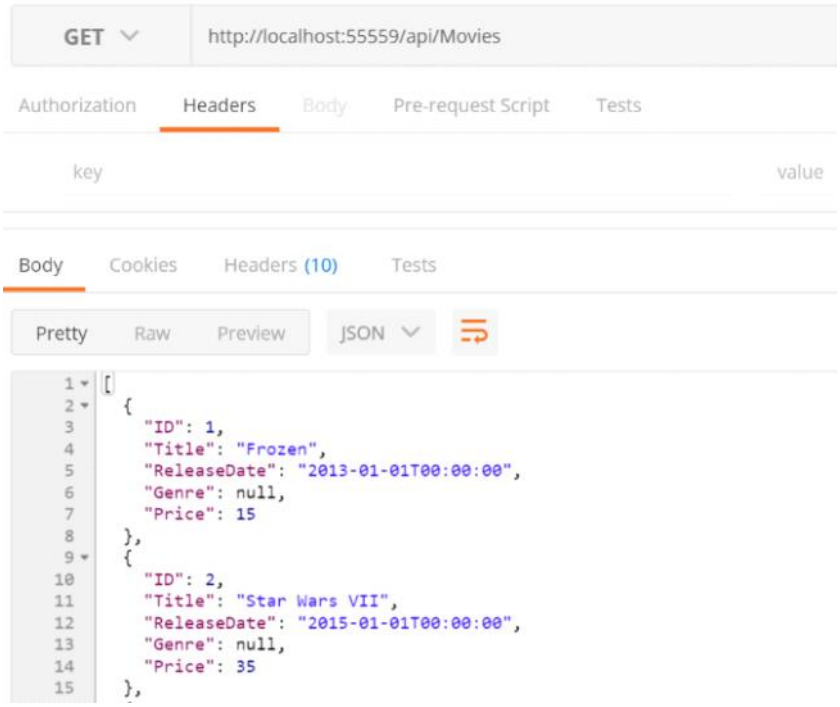
Rendons les urls un peu plus user-friendly grâce à l'attribute routing

Dans MovieApiController.cs, ajouter

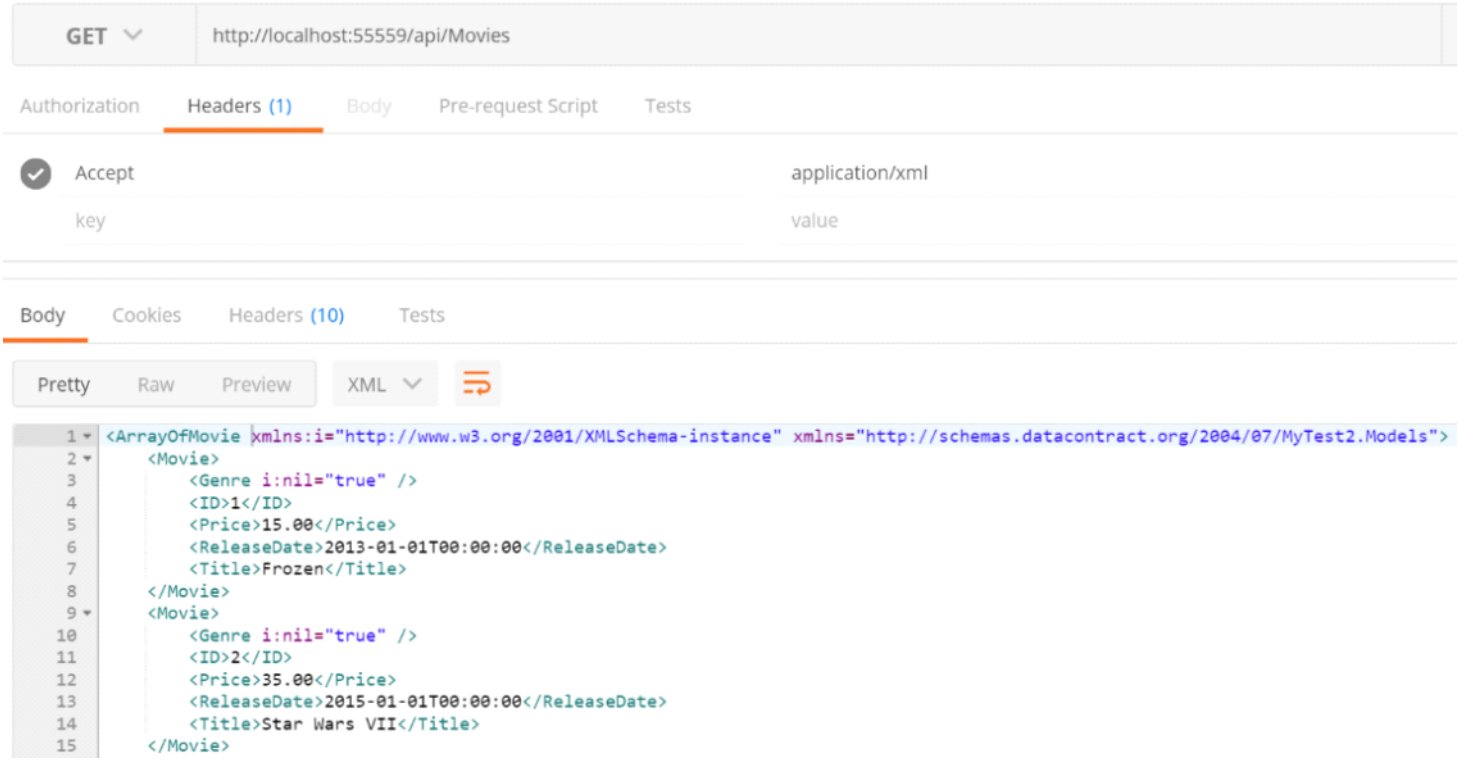
[Route("api/Movies")]
devant GetMovies()
et
[Route("api/Movies/{id}")]
devant GetMovie(int id)

Retester urls :
http://localhost:55559/api/Movies
http://localhost:55559/api/Movies/1

Content Negotiation : par défaut retourne json
Dans **Postman**, faire un GET de :
http://localhost:55559/api/Movies




Puis ajouter Header :
Accept : application/xml
Et relancer le GET




Tester l'ajout d'un Movie via un POST :

http://localhost:55559/api/MovieApi

Choisir POST

Onglet Body, raw et JSON (application/json)

Puis coller :

{

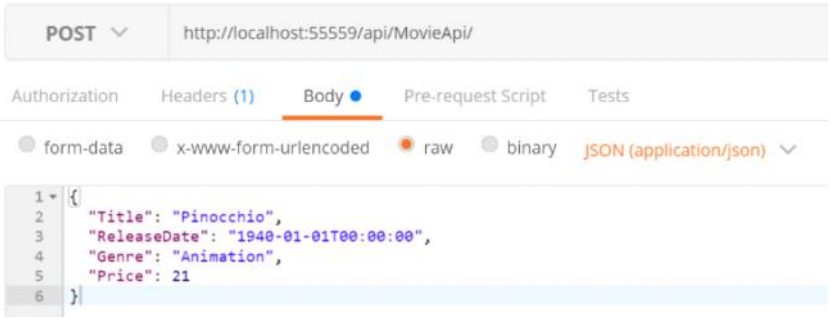
"Title": "Pinocchio",

"ReleaseDate": "1940-01-01T00:00:00",

"Genre": "Animation",

"Price": 21

}

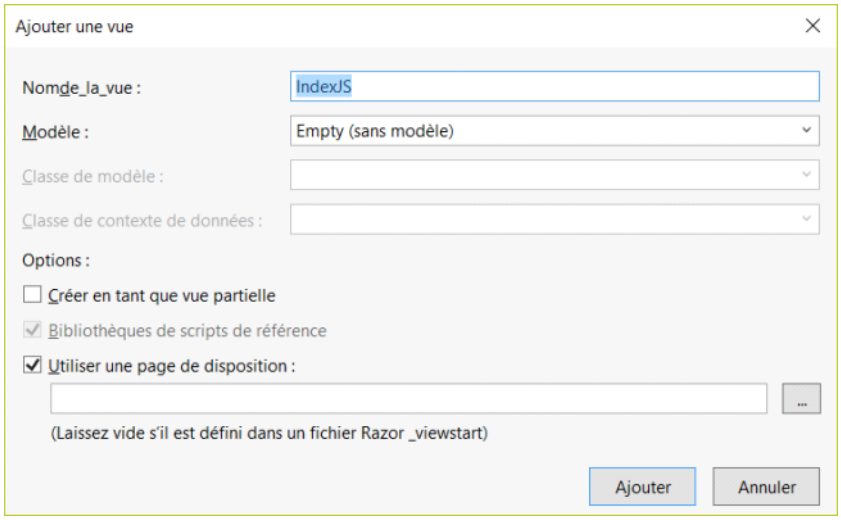



Maintenant qu'on a vérifié le bon fonctionnement de l'API, essayons de l'appeler depuis une page web en jQuery.

Dans HomeController.cs, créer une action method IndexJS :

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

Faire un click droit sur le nom de l'action method et choisir "Ajouter une vue"




La view IndexJS.cshtml s'ouvre. Y ajouter le code suivant :

```
<div>
    <h2>All Movies</h2>
    <ul id="movies" />
</div>
<div>
    <h2>Search by ID</h2>
    <input type="text" id="movieId" size="5" />
    <input type="button" value="Search" onclick="find();" />
    <p id="movie" />
</div>


<script>
    var uri = '/api/Movies';


    $(document).ready(function () {
        // Send an AJAX request
        $.getJSON(uri)
            .done(function (data) {
                // On success, 'data' contains a list of movies.
                $.each(data, function (key, item) {
                    // Add a list item for the movie.
                    $('<li>', { text: formatItem(item) }).appendTo($('#movies'));
                });
            });
    });


    function formatItem(item) {
        return item.Title + ': €' + item.Price;
    }


    function find() {
        var id = $('#movieId').val();
        $.getJSON(uri + '/' + id)
            .done(function (data) {
                $('#movie').text(formatItem(data));
            })
            .fail(function (jqXHR, textStatus, err) {
                $('#movie').text('Error: ' + err);
```

```
        });
    }
</script>
```

Installer FontAwesome via NuGet
Référencez-la dans BundleConfig.cs dans la section StyleBundle("~/Content/css") comme suit
"~/Content/font-awesome.css"

Dans MovieApiController.cs ajouter un
Thread.Sleep(2000);
au début des méthodes GetMovies() et GetMovie(int id)

Dans IndexJS.cshtml, ajouter un gros spinner après le bouton Search

```
<div class="container" id="spinner">
<i class="fa fa-spinner fa-spin fa-3x"></i>
</div>
```

Ajouter un event handler *always* à la fin de la *ready* function

```
.always(function () {
                $('#spinner').hide();
            });
```

Puis dans la function find(), ajouter au début

```
$('#spinner').show();
```

Et à la fin

```
.always(function () {
            $('#spinner').hide();
        });
```