

TP2

lundi 30 mai 2016 00:06

TP2

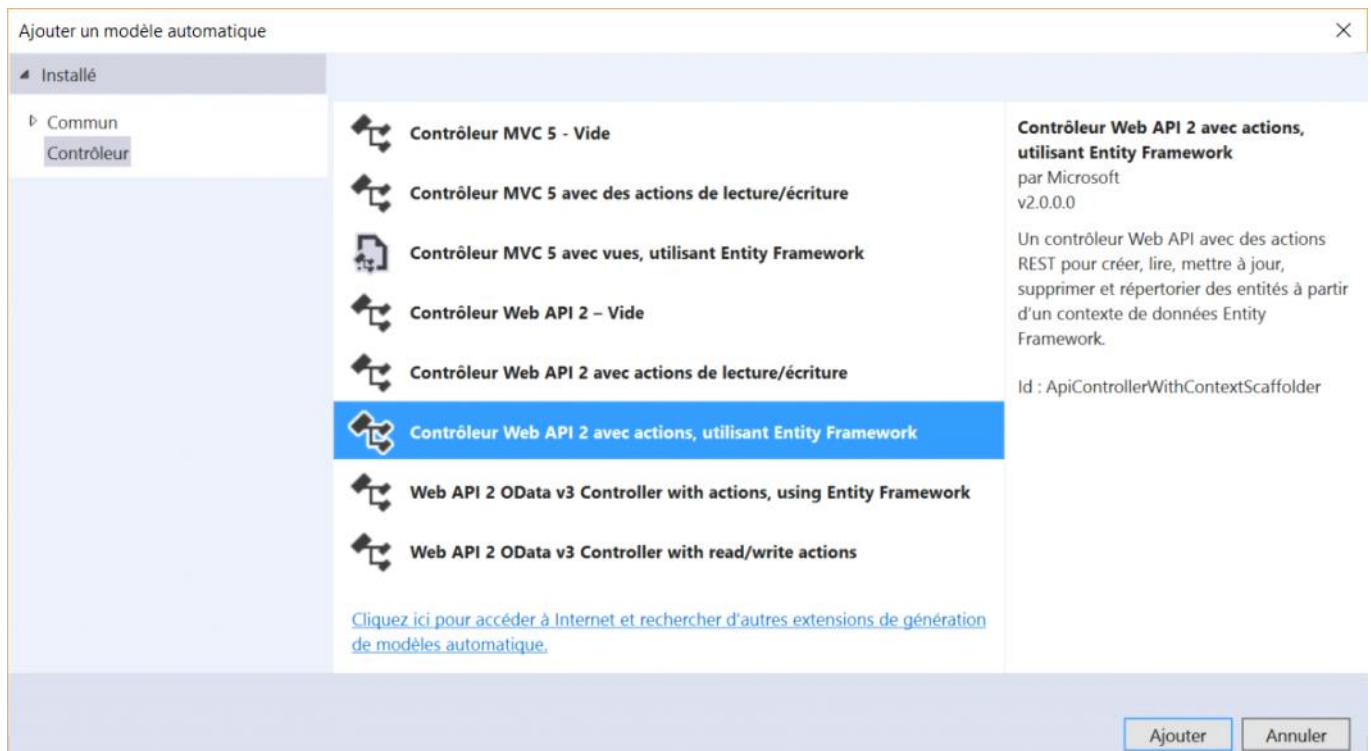
WebAPI 2 + Postman

Remarques : dans la route :

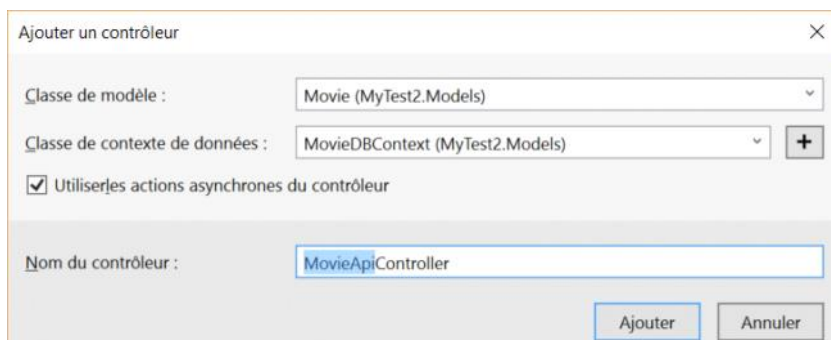
- + MapHttpRequest
- + pas d'action
- + path démarre par api/

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)
```

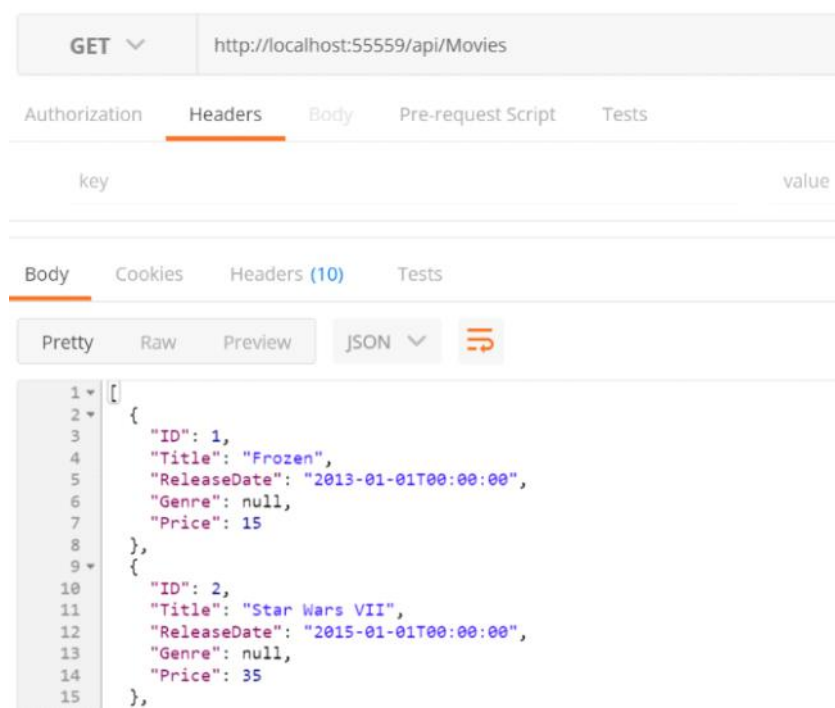
Retester urls :

<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

GET ▼ http://localhost:55559/api/Movies

Authorization Headers (1) Body Pre-request Script Tests

Accept application/xml

key value

Body Cookies Headers (10) Tests

Pretty Raw Preview XML ≡

```

1 <ArrayOfMovie xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.datacontract.org/2004/07/MyTest2.Models">
2   <Movie>
3     <Genre i:nil="true" />
4     <ID>1</ID>
5     <Price>15.00</Price>
6     <ReleaseDate>2013-01-01T00:00:00</ReleaseDate>
7     <Title>Frozen</Title>
8   </Movie>
9   <Movie>
10    <Genre i:nil="true" />
11    <ID>2</ID>
12    <Price>35.00</Price>
13    <ReleaseDate>2015-01-01T00:00:00</ReleaseDate>
14    <Title>Star Wars VII</Title>
15  </Movie>

```

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
}

```

POST ▼ http://localhost:55559/api/MovieApi/

Authorization Headers (1) Body Pre-request Script Tests

☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary JSON (application/json) ▼

```

1 {
2   "Title": "Pinocchio",
3   "ReleaseDate": "1940-01-01T00:00:00",
4   "Genre": "Animation",
5   "Price": 21
6 }

```

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"

Ajouter une vue

Nom de la vue :

IndexJS

Modèle :

Empty (sans modèle)

Classe de modèle :

Classe de contexte de données :

Options :

☐ Créer en tant que vue partielle

☒ Bibliothèques de scripts de référence

☒ Utiliser une page de disposition :

...

(Laissez vide s'il est défini dans un fichier Razor _viewstart)

Ajouter
Annuler

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

Référez-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 fonction find(), ajouter au début

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

Et à la fin

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