ASP.NET MVC : Web API

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com

Plan

Web API

Service web (WS pour Web Service en anglais), c'est quoi ?

- Un programme (ensemble de fonctionnalités) exposé en temps réel et sans intervention humaine
- Accessible via internet, ou intranet
- Indépendant de tout système d'exploitation
- Indépendant de tout langage de programmation
- Utilisant un système standard d'échange (XML ou JSON), ces messages sont généralement transportés par des protocoles internet connus HTTP (ou autres comme FTP, SMTP...)
- Pouvant communiquer avec d'autres WS

Web API

Les WS peuvent utiliser les technologies web suivantes :

- `HTTP` (Hypertext Transfer Protocol) : le protocole, connu, utilisé par le World Wide Web et inventé par Roy Fiedling.
- `REST` (Representational State Transfer) : une architecture de services Web, créée aussi par Roy Fielding en 2000 dans sa thèse de doctorat.
- `SOAP` (Simple object Access Protocol) : un protocole, défini par Microsoft et IBM ensuite standarisé par W3C, permettant la transmission de messages entre objets distants (physiquement distribués).

Web API

RESTful, REST et HTTP, c'est quoi le lien ?

- Une API (Application Programming Interface ou interface de programmation) : un ensemble de services offert par un programme pour être utilisé par d'autres programmes.
- Le Web et l'API REST sont basés sur le protocole `HTTP` (architecture client/serveur)
- RESTful : un adjectif désignant une API REST.
- L'API REST utilise des méthodes comme `get`, `post`, `delete`... pour l'échange de données entre client et serveur (Un client envoie une requête, et le serveur retourne une réponse)

Web API

WS (Web API) avec ASP NET MVC

- Le contrôleur : un des composants indispensable d'une application ASP NET MVC
- Il reçoit une requête HTTP (de la part d'un utilisateur), communique avec le modèle et la vue puis il retourne une réponse
- Le contrôleur peut aussi recevoir une requête HTTP de la part d'une autre application Back-end ou Front-end, communiquer avec le modèle et puis retourner une réponse HTTP sans construire de vue (sous format XML, JSON...)

Web API

WS (Web API) avec ASP NET MVC

- Le contrôleur : un des composants indispensable d'une application ASP NET MVC
- Il reçoit une requête HTTP (de la part d'un utilisateur), communique avec le modèle et la vue puis il retourne une réponse
- Le contrôleur peut aussi recevoir une requête HTTP de la part d'une autre application Back-end ou Front-end, communiquer avec le modèle et puis retourner une réponse HTTP sans construire de vue (sous format XML, JSON...)

Ceci est l'objet de ce chapitre.

Web API

Étapes

- Créer un nouveau projet `Fichier > Nouveau > Projet`
- Cliquer sur `Installé` et choisir `C#`
- Étendre la rubrique `Web` et sélectionner `Application web ASP.NET (.NET Framework)`
- Remplir le champs `Nom` par `FirstWebApi`
- Valider puis sélectionner un modèle `Vide`, cocher la case `web API` et décocher toutes les autres cases
- Valider et attendre la fin de création du projet

Web API

Étapes

- Créer un nouveau projet `Fichier > Nouveau > Projet`
- Cliquer sur `Installé` et choisir `C#`
- Étendre la rubrique `Web` et sélectionner `Application web ASP.NET (.NET Framework)`
- Remplir le champs `Nom` par `FirstWebApi`
- Valider puis sélectionner un modèle `Vide`, cocher la case `web API` et décocher toutes les autres cases
- Valider et attendre la fin de création du projet

N'oublions pas de définir ce projet comme projet de démarrage

Web API

Constat

- Ce projet ne contient pas de répertoire `Views`

Web API

Étapes

- Préparer le Modèle
- Créer le contrôleur Web API
- Envoyer de requêtes à ce contrôleur Web API

Web API

Créons un premier POCO `Personne` dans `Models`

```
public class Personne
{
    public int Num { get; set; }
    public string Nom { get; set; }
    public string Prenom { get; set; }
    public int Age { get; set; }
}
```

Web API

Créons le contrôleur

- Faire clic droit sur le répertoire `Controllers`
- Aller dans `Ajouter > Contrôleur`
- Choisir `Web API 2 Controller - Empty`
- Cliquer sur `Add`
- Renommer en `PersonneController` puis Valider

Web API

Le code généré

```
namespace FirstWebApiStatique.Controllers
{
    public class PersonneController : ApiController
    {
    }
}
```

Web API

Le code généré

```
namespace FirstWebApiStatique.Controllers
{
    public class PersonneController : ApiController
    {
    }
}
```

Remarques

- Ce contrôleur Web API étend la classe `ApiController` (pas `Controller`)
- Toute action de ce contrôleur doit avoir comme préfixe la méthode HTTP ciblée (`Get`, `Post`, `Put`, `Delete`...)

Web API

Commençons par simuler l'existence d'une base de données en créant une liste `personnes` dans le contrôleur `PersonneController`

```
public class PersonneController : ApiController
{
  List<Personne> personnes = new List<Personne>()
  {
    new Personne { Num = 1, Nom = "Morena", Prenom = "Andreas", Age =
       42 },
    new Personne { Num = 2, Nom = "Benamar", Prenom = "Karim", Age = 37
        },
    new Personne { Num = 3, Nom = "Paul", Prenom = "Jean", Age = 51 }
  };
}
```

Web API

Définissons maintenant une méthode qui nous permet de retourner tout le contenu de la liste `personnes`

```
public class PersonneController : ApiController
{
  List<Personne> personnes = new List<Personne>()
  {
    new Personne { Num = 1, Nom = "Morena", Prenom = "Andreas", Age =
       42 },
    new Personne { Num = 2, Nom = "Benamar", Prenom = "Karim", Age = 37
        },
    new Personne { Num = 3, Nom = "Paul", Prenom = "Jean", Age = 51 }
  };

  public IEnumerable<Personne> GetAllPersonnes()
  {
    return personnes;
  }
}
```

Web API

Quelle URL faut-il saisir pour tester ? Allons voir `WebApiConfig` dans `App_Start`

```
namespace FirstWebApi
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            // Configuration et services API Web

            // Itinéraires de l'API Web
            config.MapHttpAttributeRoutes();

            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
        }
    }
}
```

Web API

Quelle URL faut-il saisir pour tester ? Allons voir `WebApiConfig` dans `App_Start`

```
namespace FirstWebApi
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            // Configuration et services API Web

            // Itinéraires de l'API Web
            config.MapHttpAttributeRoutes();

            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
        }
    }
}
```

L'URL à saisir pour tester : `localhost:<port>/api/personne`

Web API

Le résultat est :

```
<Personne>
  <Age>42</Age>
  <Nom>Morena</Nom>
  <Num>1</Num>
  <Prenom>Andreas</Prenom>
</Personne>
<Personne>
  <Age>37</Age>
  <Nom>Benamar</Nom>
  <Num>2</Num>
  <Prenom>Karim</Prenom>
</Personne>
<Personne>
  <Age>51</Age>
  <Nom>Paul</Nom>
  <Num>3</Num>
  <Prenom>Jean</Prenom>
</Personne>
```

Web API

Le résultat est :

```
<Personne>
  <Age>42</Age>
  <Nom>Morena</Nom>
  <Num>1</Num>
  <Prenom>Andreas</Prenom>
</Personne>
<Personne>
  <Age>37</Age>
  <Nom>Benamar</Nom>
  <Num>2</Num>
  <Prenom>Karim</Prenom>
</Personne>
<Personne>
  <Age>51</Age>
  <Nom>Paul</Nom>
  <Num>3</Num>
  <Prenom>Jean</Prenom>
</Personne>
```

Par défaut le résultat retourné est sous format `XML`

Web API

Pour choisir le format de la réponse ⇒ Utiliser `Postman` (simulateur de client REST)

- Aller sur le navigateur `Chrome`
- Chercher et installer l'application `Postman`
- Démarrer l'application sans s'authentifier

Web API

Pour choisir le format de la réponse ⇒ Utiliser `Postman` (simulateur de client REST)

- Aller sur le navigateur `Chrome`
- Chercher et installer l'application `Postman`
- Démarrer l'application sans s'authentifier

Il existe plusieurs autres tel que ARC (pour `Advanced REST Client`)...

Web API

Dans **Postman**

- Choisir `Get` dans la liste déroulante
- Saisir l'URL : `localhost:<port>/api/personne`
- Dans `Headers`, saisir `Content-Type` comme `Key` et `application/json` comme `Value`
- Ensuite cliquer sur `Send`

Le résultat est :


```
[
    {
        "Num": 1,
        "Nom": "Morena",
        "Prenom": "Andreas",
        "Age": 42
    },
    {
        "Num": 2,
        "Nom": "Benamar",
        "Prenom": "Karim",
        "Age": 37
    },
    {
        "Num": 3,
        "Nom": "Paul",
        "Prenom": "Jean",
        "Age": 51
    }
]
```

Le résultat est :

```
[
    {
        "Num": 1,
        "Nom": "Morena",
        "Prenom": "Andreas",
        "Age": 42
    },
    {
        "Num": 2,
        "Nom": "Benamar",
        "Prenom": "Karim",
        "Age": 37
    },
    {
        "Num": 3,
        "Nom": "Paul",
        "Prenom": "Jean",
        "Age": 51
    }
]
```

On peut mettre `Application/xml` **comme valeur pour les propriétés** `Content-Type` **et** `Accpet` **pour avoir une réponse sous format** `XML`

Web API

Pour récupérer un seul objet de type `Personne` selon son identifiant (`Num`)

```
public class PersonneController : ApiController
{
  List<Personne> personnes = new List<Personne>()
  {
    new Personne { Num = 1, Nom = "Morena", Prenom = "Andreas", Age =
       42 },
    new Personne { Num = 2, Nom = "Benamar", Prenom = "Karim", Age = 37
        },
    new Personne { Num = 3, Nom = "Paul", Prenom = "Jean", Age = 51 }
  };

  public IEnumerable<Personne> GetAllPersonnes()
  {
    return personnes;
  }

  public Personne GetPersonne(int id)
  {
    return personnes.FirstOrDefault((p) => p.Num == id);
  }
}
```

Web API

En allant sur l'URL `localhost:<port>/api/personne`**, le résultat est :**

```
<Personne xmlns:i="http://www.w3.org/2001/XMLSchema-
   instance" xmlns="http://schemas.datacontract.org
   /2004/07/FirstWebApi.Models">
  <Age>42</Age>
  <Nom>Morena</Nom>
  <Num>1</Num>
  <Prenom>Andreas</Prenom>
</Personne>
```

Web API

On peut modifier la méthode `GetPersonne(int id)` pour qu'elle retourne une réponse HTTP contenant la personne demandée ou une erreur 404 si la personne n'existe pas

```
[ResponseType(typeof(Personne))]
public IHttpActionResult GetPersonne(int id)
{
  var personne = personnes.FirstOrDefault((p) => p.Num == id);
  if (personne == null)
  {
    return NotFound();
  }
  return Ok(personne);
}
```

Web API

On peut modifier la méthode `GetPersonne(int id)` pour qu'elle retourne une réponse HTTP contenant la personne demandée ou une erreur 404 si la personne n'existe pas

```
[ResponseType(typeof(Personne))]
public IHttpActionResult GetPersonne(int id)
{
  var personne = personnes.FirstOrDefault((p) => p.Num == id);
  if (personne == null)
  {
    return NotFound();
  }
  return Ok(personne);
}
```

Il faut utiliser l'espace de noms `System.Web.Http.Description` pour le décorateur `ResponseType`

Web API

On peut modifier la méthode `GetPersonne(int id)` pour qu'elle retourne une réponse HTTP contenant la personne demandée ou une erreur 404 si la personne n'existe pas

```
[ResponseType(typeof(Personne))]
public IHttpActionResult GetPersonne(int id)
{
  var personne = personnes.FirstOrDefault((p) => p.Num == id);
  if (personne == null)
  {
    return NotFound();
  }
  return Ok(personne);
}
```

Il faut utiliser l'espace de noms `System.Web.Http.Description` pour le décorateur `ResponseType`

En testant, on aura le même résultat sauf si la personne n'existe pas.

Web API

Pour ajouter une nouvelle personne dans la liste `personnes`

```
[ResponseType(typeof(Personne))]
public IHttpActionResult PostPersonne(Personne
   personne)
{
  personnes.Add(personne);
  return Ok(personne);
}
```

Web API

Pour tester, aller dans **Postman** et

- Choisir `Post` dans la liste déroulante
- Saisir l'URL : `localhost:<port>/api/personne`
- Dans `Body`, cocher la case `raw` et remplacer `Text` par `JSON(Application/json)`
- Dans la zone texte, saisir

```
{
    "Num": 10,
    "Nom": "Mitro",
    "Prenom": "glou",
    "Age": 51
}
```

- Ensuite cliquer sur `Send`

Web API

Pour supprimer une personne de la liste `personnes`

```
[ResponseType(typeof(Personne))]
public IHttpActionResult DeletePersonne(int id)
{
  Personne personne = personnes.Find(elt => elt.Num
     == id);
  if (personne != null)
  {
    personne.Num = 0; // juste pour vérifier que c'
       est bien cette méthode qui a été exécutée et
       pas GetPersonne(int id)
    personnes.Remove(personne);
    return Ok(personne);
  }
  return NotFound();
}
```

Web API

Pour tester, aller dans **Postman** et

- Choisir `Delete` dans la liste déroulante
- Saisir l'URL : `localhost:<port>/api/personne/1`
- Ensuite cliquer sur `Send`

Web API

Pour tester, aller dans **Postman** et

- Choisir `Delete` dans la liste déroulante
- Saisir l'URL : `localhost:<port>/api/personne/1`
- Ensuite cliquer sur `Send`

Le résultat :

```
{
    "Num": 0,
    "Nom": "Morena",
    "Prenom": "Andreas",
    "Age": 42
}
```

Web API

Pour modifier une personne de la liste `personnes`

```
[ResponseType(typeof(Personne))]
public IHttpActionResult PutPersonne(int id, Personne
   personne)
{
  if (id != personne.Num)
  {
    return BadRequest();
  }
  Personne personneUpd = personnes.Find(elt => elt.Num ==
      id);
  if (personneUpd != null)
  {
     personneUpd = personne;
     return Ok(personneUpd);
  }
  return NotFound();
}
```

Web API

Pour tester, aller dans **Postman** et

- Choisir `Put` dans la liste déroulante
- Saisir l'URL : `localhost:<port>/api/personne/1`
- Dans `Body`, cocher la case `raw` et remplacer `Text` par `JSON(Application/json)`
- Dans la zone texte, saisir

```
{
    "Num": 1,
    "Nom": "Iniesta",
    "Prenom": "Andreas",
    "Age": 42
}
```

- Ensuite cliquer sur `Send`

Web API

Le résultat :

```
{
    "Num": 1,
    "Nom": "Iniesta",
    "Prenom": "Andreas",
    "Age": 42
}
```

Web API

Étapes

- Installer EF (s'il n'est pas déjà installé)
- Préparer les entités
- Générer le contexte
- Créer le contrôleur REST

Web API

Créons une première entité `Personne` dans le `Models`

```
public class Personne
{
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public int Num { get; set; }
    public string Nom { get; set; }
    public string Prenom { get; set; }
    public int Age { get; set; }
}
```

Web API

Créons une première entité `Personne` dans le `Models`

```
public class Personne
{
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public int Num { get; set; }
    public string Nom { get; set; }
    public string Prenom { get; set; }
    public int Age { get; set; }
}
```


Pour les décorateurs, il faut utiliser les espaces de noms suivants :

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
```

Web API

N'oublions pas de

Préparer le contexte (clic droit sur le projet et aller `Ajouter > Nouvel élément > Données > ADO.NET Entity Data Model > Modèle vide Code First`)

Web API

Mettons à jour le contexte (la classe `Model1`)

```
public class Model1 : DbContext
{
    public Model1()
        : base("name=Model1")
    {
    }
    public virtual DbSet<Personne> Personnes { get;
       set; }
}
```

Web API

Mettons à jour le contexte (la classe `Model1`)

```
public class Model1 : DbContext
{
    public Model1()
        : base("name=Model1")
    {
    }
    public virtual DbSet<Personne> Personnes { get;
       set; }
}
```

N'oublions pas

```
using FirstWebApi.Models;
```

Web API

Exercice

Générer puis écrire le code d'un contrôleur Web API 2 - Empty `PersonneController` qui permet aux clients REST de manipuler les données de la table `personne` (CRUD)

Web API

Remarque

Il est possible de générer le code d'un contrôleur Web API 2

Web API

Remarque

Il est possible de générer le code d'un contrôleur Web API 2

Pour cela, il faut

- Faire clic droit sur le nom du projet dans l'`Explorateur de solutions` et choisir `Générer`
- Faire clic droit sur le répertoire `Controllers` et aller dans `Ajouter > Contrôleur`
- Choisir `Web API 2 Controller with actions using Entity Framework`
- Dans `Model class :` sélectionner le modèle ensuite choisir le contexte (`Model1`)
- Saisir un nom pour le contrôleur puis valider

Web API

Pour le CORS, allez dans `Web.config` **et vérifiez le contenu de la section** `handlers` **de** `system.webServer`

```
<system.webServer>
  <handlers>
    <remove name="ExtensionlessUrlHandler-Integrated-4.0" />
    <remove name="OPTIONSVerbHandler" />
    <remove name="TRACEVerbHandler" />
    <add name="ExtensionlessUrlHandler-Integrated-4.0" path="*." verb="
      *" type="System.Web.Handlers.TransferRequestHandler"
      preCondition="integratedMode,runtimeVersionv4.0" />
  </handlers>
</system.webServer>
```

Dans la section `system.webServer` **de** `Web.config`**, ajoutez la balise** `httpProtocol` **après** `handlers`

```
<system.webServer>
  <handlers>
    <remove name="ExtensionlessUrlHandler-Integrated-4.0" />
    <remove name="OPTIONSVerbHandler" />
    <remove name="TRACEVerbHandler" />
    <add name="ExtensionlessUrlHandler-Integrated-4.0" path="*." verb="
       *" type="System.Web.Handlers.TransferRequestHandler"
       preCondition="integratedMode,runtimeVersionv4.0" />
  </handlers>
  <httpProtocol>
    <customHeaders>
      <add name="Access-Control-Allow-Origin" value="*" />
      <add name="Access-Control-Allow-Methods" value="GET,PUT,POST,
         DELETE,HEAD,OPTIONS" />
      <add name="Access-Control-Allow-Credentials" value="true"/>
      <add name="Access-Control-Allow-Headers" value="X-Requested-With,
          origin, content-type, accept" />
    </customHeaders>
  </httpProtocol>
</system.webServer>
```

Web API

Dans `Global.asax`**, ajouter la méthode** `Application_BeginRequest()`

```
public class WebApiApplication : System.Web.HttpApplication
{
    protected void Application_Start()
    {
        GlobalConfiguration.Configure(WebApiConfig.Register);
    }

    protected void Application_BeginRequest()
    {
        if (Request.Headers.AllKeys.Contains("Origin") &&
           Request.HttpMethod == "OPTIONS")
        {
            Response.Flush();
        }
    }
}
```

Web API

Pour tester la méthode `PostPersonne()` avec Angular, remplacez la dernière instruction de cette méthode)

```
return CreatedAtRoute("DefaultApi", new { id =
   personne.Num }, personne);
```


par

```
return Ok(personne);
```