Module 13: Implémentation des API Web dans les applications Web ASP.NET MVC 5

# **Contenu:**

|  |  |
| --- | --- |
|  | [Aperçu du module](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C13.html#P1) |
| **Leçon 1:** | [Développer une API Web](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C13.html#P2) |
| **Leçon 2:** | [Appeler une API Web](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C13.html#P3) |
| **Laboratoire:** | [Implémentation d'API dans les applications Web ASP.NET MVC 5](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C13.html#P4) |
|  | [Examen du module et points à retenir](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C13.html#P5) |

# **Aperçu du module**

La plupart des applications Web nécessitent une intégration avec des systèmes externes tels que les applications mobiles. Vous devez savoir comment utiliser les API Web pour promouvoir l'interaction des applications avec les systèmes externes. Vous pouvez utiliser l'API Web pour implémenter les services REST (Representational State Transfer) dans votre application. Les services REST permettent de réduire la surcharge des applications et de limiter les données transmises entre les systèmes client et serveur. Vous devez savoir comment appeler les services d'API Web à l'aide du code côté serveur, du code jQuery et de la bibliothèque JSON.NET pour implémenter efficacement les API Web de style REST dans votre application.

### **Objectifs**

Après avoir terminé ce module, vous serez en mesure de:

|  |  |
| --- | --- |
| • | Développez une API Web. |
| • | Appelez une API Web. |

# Leçon 1: Développement d'une API Web

Vous devez savoir comment développer l'API Web pour les applications, car l'API Web facilite la création d'API pour les applications mobiles, les applications de bureau, les services Web, les applications Web et d'autres applications. En créant une API Web, vous rendez les informations de votre application Web disponibles pour que d'autres développeurs les utilisent dans leurs systèmes. Chaque application Web a une méthodologie fonctionnelle différente; cette différence peut entraîner des problèmes d'interopérabilité dans les applications. Les services REST ont une conception légère et l'API Web aide à implémenter les services REST pour résoudre les problèmes d'interopérabilité. Vous devez savoir comment utiliser les différentes méthodes de routage fournies par ASP.NET pour implémenter les services REST.

## **Objectifs de la leçon**

Après avoir terminé cette leçon, vous serez en mesure de:

|  |  |
| --- | --- |
| • | Décrivez les API Web. |
| • | Créez une API Web. |
| • | Décrivez les services REST. |
| • | Décrivez les formats de retour de données. |
| • | Expliquez comment utiliser les routes et les contrôleurs pour implémenter REST dans les API Web. |
| • | Explorez une API Web en utilisant Internet Explorer en tant que client. |

## **Qu'est-ce qu'une API Web?**

L'API Web est une infrastructure qui vous permet de créer des API compatibles REST (Representational State Transfer). Les API compatibles REST aident les systèmes externes à utiliser la logique métier implémentée dans votre application pour augmenter la réutilisabilité de la logique applicative. L'API Web facilite la communication bidirectionnelle entre le système client et le serveur via des tâches telles que:

|  |  |
| --- | --- |
| • | Demander à une application d'effectuer une tâche spécifique |
| • | Lecture des valeurs de données |
| • | Mise à jour des valeurs de données |

L'API Web permet aux développeurs d'obtenir des informations commerciales à l'aide de REST, sans créer de requêtes XML complexes telles que le protocole SOAP (Simple Object Access Protocol). Les API Web utilisent des URL dans les requêtes, éliminant ainsi le besoin de requêtes compliquées. Par exemple, l'URL suivante obtient des informations pour une entité client avec l'ID 1: http://api.contoso.com/api/customers/1

**Une réponse JSON de l'API Web**

[{"Id": 1, "Name": "Tomato soup", "Category": "Epiceries", "Price": 1.0}, {"Id": 2, "Name":

"Yo-yo", "Catégorie": "Jouets", "Prix": 3.75}, {"Id": 3, "Nom": "Marteau", "Catégorie":

"Matériel", "Prix": 16,99}]

REST et Web API permettent à toutes sortes d'applications différentes, y compris les applications pour appareils mobiles, d'interagir avec les services. En particulier, REST et Web API offrent les avantages suivants pour les applications mobiles:

|  |  |
| --- | --- |
| • | Ils réduisent la puissance de traitement nécessaire pour créer des messages de demande complexes pour la récupération de données. |
| • | Ils améliorent les performances de l'application en réduisant la quantité d'échange de données entre le client et le serveur. |

**Question**: Quel est le principal avantage de l'utilisation de REST avec les API Web?

## **Routage dans l'API Web**

Lorsque vous créez un nouveau projet à l'aide du modèle d'API Web dans Visual Studio 2017, il inclut une règle de routage par défaut. Cette règle de routage permet de mapper les requêtes HTTP aux contrôleurs et actions d'API Web à l'aide des verbes HTTP et de l'URL de la requête. Vous pouvez utiliser une convention de dénomination pour mapper les demandes aux actions, ou vous pouvez contrôler le comportement du mappage en utilisant des annotations sur les méthodes d'action.

**La route API par défaut**

À l'instar des applications Web MVC, les applications d'API Web utilisent des itinéraires pour mapper les demandes au bon contrôleur et à l'action d'API. Dans les modèles de projet Visual Studio, la route API par défaut est définie dans le fichier WebApiConfig.cs du dossier App\_Start.

**La route API par défaut**

routes.MapHttpRoute (

nom: "API par défaut",

routeTemplate: "api / {controller} / {id}",

par défaut: new {id = RouteParameter.Optional}

);

Dans l'exemple de code précédent, observez que l'itinéraire par défaut inclut l'API du segment de chemin littéral. Ce segment garantit que les demandes d'API Web sont clairement distinctes des routes de contrôleur MVC, car les demandes d'API Web doivent commencer par l'API.

La première variable d'espace réservé, {controller}, permet d'identifier le contrôleur d'API vers lequel transférer la requête. Comme pour les contrôleurs MVC, l'API Web ajoute le mot, Contrôleur, à cette valeur pour localiser la bonne classe de contrôleur d'API. Par exemple, l'API Web achemine une demande vers l'URI, api / products, vers le contrôleur appelé ProductsController. Comme pour les contrôleurs MVC, la variable d'espace réservé facultative, {id}, est envoyée à l'action en tant que paramètre.

Vous pouvez également définir vos propres routes API de la même manière que vous le faites pour les routes MVC. Notez, cependant, que les routes API Web peuvent gérer les demandes de nombreux types de systèmes clients, y compris les applications de périphérique mobile, les applications de bureau, les applications Web et les services Web. Les routes MVC ne gèrent que les requêtes du navigateur Web.

**Utilisation de la convention de dénomination des actions**

La route API Web par défaut n'inclut pas de variable d'espace réservé pour le nom de l'action. En effet, l'API Web utilise le verbe HTTP et une convention de dénomination pour acheminer les demandes vers la bonne action dans un contrôleur donné.

Les clients peuvent effectuer des requêtes HTTP avec l'un des quatre verbes standard: GET, POST, PUT et DELETE. D'autres verbes sont possibles. L'API Web recherche une action dont le nom commence par le verbe HTTP demandé. Par exemple, si le client envoie une demande DELETE à l'URI api / products / 23, l'API Web recherche un contrôleur appelé ProductsController. Dans ce contrôleur, il localise une action dont le nom commence par Supprimer. Selon la route par défaut, le segment 23 est le paramètre {id}. S'il existe plusieurs actions dont le nom commence par Supprimer, l'API Web choisit l'action qui accepte un paramètre appelé id.

**Les attributs HttpGet, HttpPut, HttpPost et HttpDelete**

Vous pouvez utiliser les attributs HttpGet, HttpPut, HttpPost ou HttpDelete dans votre action de contrôleur pour remplacer la convention de dénomination d'action. Vous pouvez également utiliser ces verbes pour spécifier qu'une fonction est mappée à un verbe HTTP spécifique. Le tableau suivant décrit comment les attributs HTTP sont mappés aux verbes HTTP.

| **Attribut** | **Verbe HTTP** |
| --- | --- |
| HttpGet | AVOIR |
| HttpPut | METTRE |
| HttpPost | PUBLIER |
| HttpDelete | EFFACER |

**Spécification du verbe HTTP**

Public class ProductsController: ApiController

{

[HttpGet]

public Product FindProduct (id) {}

}

Dans l'exemple de code précédent, observez que les attributs HTTP autorisent uniquement le mappage d'un verbe HTTP à une action dans le contrôleur.

**L'attribut AcceptVerbs**

L'utilisation de l'attribut AcceptVerbs vous permet de spécifier des verbes HTTP personnalisés ou plusieurs verbes HTTP pour les mêmes actions dans le contrôleur.

**L'attribut AcceptVerbs**

Public class ProductsController: ApiController

{

[AcceptVerbs ("GET", "HEAD")]

public Product FindProduct (id) {}

[AcceptVerbs ("MKCOL")]

public void MakeCollection () {}

}

**L'attribut ActionName**

Par défaut, le nom de l'action est le nom de l'action que vous spécifiez dans le contrôleur. Vous pouvez utiliser l'attribut ActionName pour spécifier le nom de l'action à utiliser dans le routage.

**Cartographie d'une action**

Public class ProductsController: ApiController

{

[HttpGet]

[ActionName ("Thumbnail")]

public HttpResponseMessage GetThumbnailImage (int id);

[HttpPost]

[ActionName ("Thumbnail")]

public void AddThumbnailImage (int id);

}

**Question**: Quel est le but de l'utilisation des attributs HTTP?

## **Créer une API Web**

Visual Studio 2017 fournit un modèle de projet d'API Web qui permet d'implémenter l'API Web dans un projet.

L'image suivante montre la liste des modèles de projet disponibles dans Microsoft Visual Studio.

**FIGURE 13.1: LA NOUVELLE BOÎTE DE DIALOGUE PROJET**

Pour implémenter un modèle d'API Web dans votre projet, vous devez effectuer les étapes suivantes:

|  |  |
| --- | --- |
| 1. | Dans la boîte de dialogue Nouveau projet, cliquez sur Application Web ASP.NET (.NET Framework). |
| 2. | Dans la boîte de dialogue Nouvelle application Web ASP.NET, cliquez sur API Web. |

L'image suivante montre la boîte de dialogue Nouvelle application Web ASP.NET

**FIGURE 13.2: LA NOUVELLE BOÎTE DE DIALOGUE APPLICATION WEB ASP.NET**

Après avoir sélectionné le modèle d'API Web, vous devez ajouter une nouvelle classe de contrôleur d'API qui dérive d'ApiController. La classe de contrôleur d'API héberge le code d'application pour gérer les demandes d'API Web. Le moteur ASP.NET mappe l'URL avec le verbe HTTP et le contrôleur ou la fonction d'action d'un contrôleur, au format suivant.

<verbe http> http: // <nom d'hôte> / api / <entité nom> / <paramètres>

Le verbe HTTP communique à l'API Web sur les opérations à effectuer; tandis que le reste consiste à communiquer sur quelle entité et quelles opérations effectuer. Par conséquent, HTTP joue un rôle important. Par exemple, considérez la classe de contrôleur API suivante.

Public class ProductsController: ApiController

{

public IEnumerable <Product> GetAllProducts ()

{

}

Public Product GetProductById (ID int)

{

}

}

Dans l'exemple de code précédent, notez les classes de contrôleur GetAllProducts et GetProductById. La classe de contrôleur GetAllProducts permet d'obtenir une liste complète des produits de la base de données. Vous pouvez mapper l'URL et le verbe HTTP suivants avec la classe de contrôleur GetAllProducts.

GET / api / produits

La classe de contrôleur GetProductById permet d'obtenir un produit spécifique à l'aide du détail de l'ID. Vous pouvez mapper l'URL suivante avec la classe de contrôleur GetProductById.

GET / api / produits /identifiant

**Question**: Quelle est la syntaxe utilisée par le moteur ASP.NET pour mapper les contrôleurs d'API et les fonctions d'action?

## **Services RESTful**

REST utilise des URL et des verbes HTTP pour identifier de manière unique l'entité sur laquelle il opère et l'action qu'il effectue. REST permet de récupérer les informations commerciales du serveur. Cependant, en plus de l'extraction de données, les applications métier exécutent davantage de tâches telles que la création, la mise à jour et la suppression d'informations sur la base de données. L'API Web et REST facilitent la gestion de ces tâches supplémentaires. Ils utilisent la méthode HTTP pour identifier l'opération que l'application doit effectuer.

Le tableau suivant fournit des informations sur certaines méthodes HTTP utilisées par l'API Web et REST.

| **Verbe HTTP** | **La description** |
| --- | --- |
| **AVOIR** | Utilisez cette méthode avec l'URL suivante pour obtenir une liste de tous les clients.  / api / clients |
| **AVOIR** | Utilisez cette méthode avec l'URL suivante pour obtenir un client en utilisant le détail de l'ID.  / api / clients / id |
| **AVOIR** | Utilisez cette méthode avec l'URL suivante pour obtenir des clients à l'aide du détail de la catégorie.  / api / clients? country = country |
| **PUBLIER** | Utilisez cette méthode avec l'URL suivante pour créer un enregistrement client.  / api / clients |
| **METTRE** | Utilisez cette méthode avec l'URL suivante pour mettre à jour un enregistrement client.  / api / clients / id |
| **EFFACER** | Utilisez cette méthode avec l'URL suivante pour supprimer un enregistrement client.  / api / clients / id |

**Opérations CRUD**

public HttpResponseMessage PostCustomer (élément client)

{

}

public void PutCustomer (identifiant int, élément client)

{

}

public void DeleteProduct (identifiant int)

{

}

**Question**: Quelle est l'exigence obligatoire de création et de mise à jour des demandes?

## **Formats de retour de données**

Lorsqu'un client fait une demande à un contrôleur d'API Web, l'action du contrôleur renvoie souvent des données. Pour les demandes GET, par exemple, ces données peuvent être toutes les propriétés d'un produit spécifique ou toutes les propriétés de tous les produits. L'API Web peut renvoyer ces données dans l'un des deux formats suivants: JavaScript Object Notation (JSON) ou XML.

**Formats de données JSON et XML**

JSON et XML sont des formats de texte qui représentent des informations sous forme de chaînes. Vous pouvez également utiliser JSON en dehors du code JavaScript.

**Une réponse JSON**

{"Nom": "Albert", "Âge": 29, "Taille": 145, "Compétences": ["Programmation", "Rédaction technique"]}

**Une réponse XML**

<Nom de l'employé = "Albert" Age = "29" Hauteur = "145">

<Compétences>

<Nom de la compétence = "Programmation" />

<Nom de la compétence = "Rédaction technique" />

</Skills>

</Employee>

Lorsqu'un client effectue une demande, le client peut spécifier le format de données pour la réponse. Si le format de données n'est pas spécifié, l'API Web formate les données au format JSON par défaut.

**Formateurs de médias**

L'API Web utilise un formateur multimédia pour formater ou sérialiser les informations renvoyées par un service REST d'API Web. Les applications Web utilisent généralement le format JSON pour formater les données renvoyées par les fonctions. Cependant, vous pouvez également utiliser le formateur de média XML ou ajouter un formateur de média personnalisé pour contrôler les données renvoyées. Les formateurs de médias mettent en forme le contenu que le serveur rend aux systèmes clients.

Par exemple, considérez que vous souhaitez que l'API Web renvoie des fichiers CSV. Dans ce cas, vous devez créer un formateur multimédia personnalisé, pour créer en tant que fichiers de données de sortie au format CSV, au lieu de fichiers XML au format JSON.

**Un formateur multimédia personnalisé**

classe publique CsvFormatter: BufferedMediaTypeFormatter

{

}

Les classes de formateur de média héritent des classes suivantes:

|  |  |
| --- | --- |
| • | MediaTypeFormatter. C'est la classe fondamentale pour toutes les classes de formateur. |
| • | BufferedMediaTypeFormatter. Il s'agit d'une classe étendue de classes de formateur, qui prend en charge la mise en mémoire tampon et les opérations asynchrones. |

**Lecture supplémentaire:**Pour plus d'informations sur la création de formateurs de médias, accédez à:<http://go.microsoft.com/fwlink/?LinkID=288992&clcid=0x421>

**Question**: Pourquoi devriez-vous utiliser un formateur multimédia pour les services REST de l'API Web?

## **Utilisation d'itinéraires et de contrôleurs dans les API Web**

ASP.NET utilise une table de routage pour mapper une URL et un contrôleur d'API. Lorsque vous créez un projet, ASP.NET ajoute un itinéraire par défaut à l'aide du modèle d'API Web. Cette route par défaut permet de prendre en charge les opérations des API Web de style REST.

**La route API par défaut**

routes.MapHttpRoute (

nom: "API par défaut",

routeTemplate: "api / {controller} / {id}",

par défaut: new {id = RouteParameter.Optional}

);

Considérez que vous souhaitez inclure plusieurs actions, telles que la création de clients avec XML et JSON, dans la même méthode HTTP. Dans ce cas, vous ne pouvez pas utiliser la route par défaut car elle nécessite une nouvelle requête pour chaque combinaison de méthode HTTP et d'URL. Par conséquent, vous devez mettre à jour le routage en modifiant la carte d'itinéraire dans la classe WebApiConfig.

**Prise en charge de plusieurs opérations**

routes.MapHttpRoute (

nom: "ActionApi",

routeTemplate: "api / {controller} / {action} / {id}",

par défaut: new {id = RouteParameter.Optional}

);

Vous pouvez utiliser la classe WebApiConfig pour permettre à plusieurs versions d'API de coexister dans le même projet. Par exemple, vous pouvez inclure / api / v1 / {controller} en tant que version un de votre API et inclure / api / v2 / {controller} en tant que nouvelle version ou version deux de l'API.

Vous souhaiterez peut-être inclure des fonctions de prise en charge dans la classe de contrôleur et masquer les fonctions de prise en charge des fonctions REST. Vous pouvez éliminer l'exposition de la fonction à l'interface REST en ajoutant l'attribut NonAction à la fonction d'action.

**Utilisation de NonAction**

[NonAction]

chaîne publique GetPrivateData ()

{

}

Par défaut, l'API Web expose toutes les méthodes publiques en tant que services REST. Vous pouvez empêcher cela en rendant la fonction privée, mais cette action empêche le code d'application du même projet d'accéder à la fonction. Par conséquent, vous pouvez utiliser l'attribut NonAction pour les fonctions qui doivent être publiques, mais qui n'ont pas besoin d'être exposées dans REST.

**Question**: Quel est le principal avantage de l'utilisation de la carte de routage?

## **Démonstration: Comment explorer une API Web à l'aide d'Internet Explorer**

Dans cette démonstration, vous verrez comment:

|  |  |
| --- | --- |
| • | Créez une API Web simple pour une application Web ASP.NET existante. |
| • | Téléchargez et examinez les fichiers JSON générés par l'API Web à l'aide d'Internet Explorer. |

### **Étapes de démonstration**

|  |  |
| --- | --- |
| • | Vous trouverez les étapes dans la section «Leçon 1: Développement d'une API Web» sur la page suivante:<https://github.com/MicrosoftLearning/20486-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486C/20486C_MOD13_DEMO.md>. |

# Leçon 2: Appel d'une API Web

Une fois le développement des services API Web terminé, vous devez créer les applications clientes pour appeler ces services. L'appel de services d'API Web est différent de l'appel de services WCF. Cependant, les méthodes que vous devez utiliser pour appeler ces services sont similaires, quelle que soit la plate-forme. Vous devez savoir comment appeler les services d'API Web à l'aide du code côté serveur, du code jQuery et de la bibliothèque JSON.NET, pour implémenter efficacement les services d'API Web dans la plupart des plates-formes d'application.

## **Objectifs de la leçon**

Après avoir terminé cette leçon, vous serez en mesure de:

|  |  |
| --- | --- |
| • | Appelez les API Web à l'aide du code côté serveur. |
| • | Appelez les API Web à l'aide de jQuery. |

## **Appel d'API Web à l'aide du code côté serveur**

Vous pouvez appeler des services de style REST à l'aide du code côté serveur ASP.NET. Vous pouvez utiliser la classe HttpWebRequest pour créer une requête HTTP manuelle aux services REST. ASP.NET fournit une bibliothèque .NET que vous pouvez utiliser dans les applications Web pour appeler les services d'API Web compatibles REST à partir du serveur .NET. Pour utiliser la bibliothèque .NET, vous devez installer le package NuGet Microsoft.AspNet.WebApi.Client. Ce package NuGet permet d'accéder à la classe HttpClient. La classe HttpClient simplifie l'interaction avec les API Web, car elle réduit les efforts de codage.

**Initialisation de la classe HttpClient**

Client HttpClient = nouveau HttpClient ();

client.BaseAddress = new Uri ("http: // localhost /");

client.DefaultRequestHeaders.Accept.Add (new MediaTypeWithQualityHeaderValue ("application / json"));

La dernière ligne de code de l'exemple de code précédent informe le système client du type de média que le système client doit utiliser. Le type de support par défaut utilisé par les applications est application / json. Cependant, les applications peuvent utiliser tout autre type de support, en fonction du type de support pris en charge par les services de style REST.

**Appel de l'API à partir du code côté serveur**

HttpResponseMessage response = client.GetAsync ("api / clients"). Result;

if (response.IsSuccessStatusCode)

{

var products = response.Content.ReadAsAsync <IEnumerable <Customer>> (). Result;

}

autre

{

Console.WriteLine ("{0} ({1})", (int) response.StatusCode, response.ReasonPhrase);

}

Après avoir exécuté le code dans l'exemple de code précédent, vous devez définir un modèle de données qui s'aligne sur celui utilisé par le service API Web pour permettre à la bibliothèque .NET de:

|  |  |
| --- | --- |
| • | Traitez les résultats du code côté serveur. |
| • | Renvoie les résultats sous forme d'objets .NET à utiliser par l'application. |

Ensuite, vous pouvez utiliser les méthodes GetAsync et ReadAsAsync pour:

|  |  |
| --- | --- |
| • | Créez des demandes aux services REST de l'API Web. |
| • | Analysez le contenu en objets .NET. |

La fonction PostAsJsonAsync utilise la méthode HTTP POST pour appeler les services d'API Web qui prennent en charge la méthode POST.

**Question**: Quel est l'avantage d'utiliser le package NuGet Microsoft.AspNet.WebApi.Client?

## **Appel d'API Web à l'aide du code jQuery**

Vous pouvez appeler les services d'API Web de la même manière que vous appelez d'autres services qui utilisent des technologies telles que WCF. Vous pouvez également appeler les services de l'API Web à l'aide de la fonction jQuery ajax.

**Utilisation de la fonction jQuery ajax**

$ .ajax ({

url: 'http: // localhost / api / clients /,

tapez: 'GET',

dataType: 'json',

succès: fonction (données) {

},

erreur: function (e) {

}

});

Dans l'exemple de code précédent, observez le paramètre dataType de la fonction ajax. Vous devez définir ce paramètre sur json ou sur un autre type de données pris en charge par le service API Web. La plupart des applications utilisent JSON car il est léger. La fonction ajax a des fonctionnalités intégrées qui analysent les résultats JSON pour la facilité des développeurs.

**Utilisation de la fonction stringify**

var client = {

ID: '1',

CustName: 'client 1'

};

$ .ajax ({

url: 'http: // localhost / api / client',

tapez: 'POST',

données: JSON.stringify (client),

contentType: "application / json; charset = utf-8",

succès: fonction (données) {

},

erreur: fonction (x) {

}

});

**Question**: Quel est l'avantage d'utiliser JSON.stringify () dans la fonction ajax?

# **Atelier: Implémentation d'API dans les applications Web ASP.NET MVC 5**

### **Scénario**

Votre responsable souhaite s'assurer que les photos et les informations stockées dans l'application de partage de photos peuvent être intégrées à d'autres données dans des mash-ups Web, des applications mobiles et d'autres emplacements. Pour réutiliser ces données, tout en maintenant la sécurité, vous devez implémenter une API Web RESTful pour l'application. Vous utiliserez cette API Web pour afficher les emplacements des photos sur une page Bing Maps.

### **Objectifs**

Après avoir terminé cet atelier, vous serez en mesure de:

|  |  |
| --- | --- |
| • | Créez une API Web à l'aide des nouvelles fonctionnalités d'ASP.NET MVC 5. |
| • | Ajoutez des routes et des contrôleurs à une application pour gérer les demandes REST. |
| • | Appelez une API Web REST à partir du code côté client jQuery. |

##### **Configuration du laboratoire**

Durée estimée: 60 minutes

Vous trouverez les étapes de haut niveau sur la page suivante:<https://github.com/MicrosoftLearning/20486-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486C/20486C_MOD13_LAB_MANUAL.md>.

Vous trouverez les étapes détaillées sur la page suivante:<https://github.com/MicrosoftLearning/20486-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486C/20486C_MOD13_LAK.md>.

### **Exercice 1: Ajout d'une API Web à l'application de partage de photos**

##### **Scénario**

Il vous a été demandé de mettre en œuvre une API Web pour l'application de partage de photos afin de garantir que les photos peuvent être utilisées sur des sites Web tiers, des applications pour appareils mobiles et d'autres applications.

Dans cet exercice, vous allez:

|  |  |
| --- | --- |
| • | Ajoutez un contrôleur d'API Web pour la classe de modèle Photo. |
| • | Configurez les formateurs et les itinéraires pour prendre en charge l'API Web. |
| • | Testez l'API à l'aide d'Internet Explorer. |

### **Exercice 2: Utilisation de l'API Web pour un affichage Bing Maps**

##### **Scénario**

Vous devez utiliser la nouvelle API Web pour obtenir les photos dans le code jQuery côté client. Vous utiliserez les propriétés de latitude et de longitude pour afficher ces photos sous forme d'épingles sur une carte API Bing.

Pour créer l'affichage de la carte dans l'application Partage de photos, vous devez ajouter une nouvelle vue et une nouvelle action pour le contrôleur photo. Vous devez également ajouter une nouvelle vue de modèle car le contrôle Bing Maps AJAX nécessite une directive <! DOCTYPE> différente de celle utilisée ailleurs dans l'application de partage de photos. Vous importerez un fichier JavaScript contenant du code Bing Maps de base. À ce fichier JavaScript, vous ajouterez du code pour appeler l'API Web, obtenir les détails de la photo et les afficher sur la carte.

Dans cet exercice, vous allez:

|  |  |
| --- | --- |
| • | Créez une nouvelle vue de modèle. |
| • | Créez une action de carte, une vue et un fichier de script. |
| • | Obtenez et affichez des photos. |
| • | Testez le contrôle Bing Maps. |

### **Question (s) de révision**

**Vérifiez vos connaissances**

**Découverte**

**Comment les actions d'API que vous avez ajoutées auPhotoApiControllercontrôleur de l'exercice 1 diffèrent-ils des autres actions des contrôleurs MVC?**

Afficher la solution Réinitialiser

# **Examen du module et points à retenir**

Vous pouvez utiliser la structure de l'API Web pour faciliter la création d'appels d'API Web de style REST dans les applications. L'API Web de style REST est recommandée pour les applications mobiles en raison de la conception légère des services REST.

Les services REST utilisent des méthodes HTTP telles que GET, POST et PUT pour notifier l'API de l'action qu'elle doit effectuer. Les API Web utilisent le formateur multimédia et la bibliothèque JSON.NET pour sérialiser et désérialiser les informations, respectivement. Vous pouvez appeler les services d'API Web à l'aide du code côté serveur, du code jQuery et de la bibliothèque JSON.NET.

### **Problèmes et scénarios du monde réel**

Considérez que vous développez une application mobile à l'aide d'API Web et que l'application doit utiliser des services de taux de change. Pour cette application, vous ne pouvez pas utiliser WCF, car WCF peut entraver les performances de l'application en utilisant XML pour les échanges de données. Par conséquent, vous devez utiliser REST et JSON dans l'application pour réduire les données transmises entre le système client et le serveur.

### **Question (s) de révision**