Module 7: Structuration des 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/20486C07.html#P1) |
| **Leçon 1:** | [Analyse de l'architecture de l'information](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C07.html#P2) |
| **Leçon 2:** | [Configurer les itinéraires](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C07.html#P3) |
| **Lecon 3:** | [Créer une structure de navigation](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C07.html#P4) |
| **Laboratoire:** | [Structuration des 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/20486C07.html#P5) |
|  | [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/20486C07.html#P6) |

# **Aperçu du module**

Une application Web MVC peut devenir volumineuse et complexe et l'application peut contenir de nombreux types d'informations qui se rapportent les unes aux autres de manière complexe. Un défi majeur pour ces applications Web est de s'assurer que les utilisateurs peuvent facilement localiser les informations qui les intéressent. Bien qu'un outil de recherche soit utile, vous devez concevoir votre application Web de manière à ce que les utilisateurs puissent accéder à la page dont ils ont besoin via une courte séquence de liens. À chaque niveau, les utilisateurs doivent savoir sur quel lien cliquer ensuite. Les URL que les utilisateurs voient dans la barre d'adresses doivent être lisibles et compréhensibles. Vous devez comprendre comment présenter les classes de modèle que vous avez conçues de manière logique pour l'utilisateur. Vous devez également vous assurer que les moteurs de recherche Internet peuvent explorer et analyser la structure de votre application Web, afin que le contenu de votre application apparaisse près du haut des résultats des moteurs de recherche. Vous pouvez atteindre ces objectifs en découvrant le routage et la navigation ASP.NET.

### **Objectifs**

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

|  |  |
| --- | --- |
| • | Planifier une architecture d'information appropriée pour un ensemble donné d'exigences fonctionnelles. |
| • | Ajoutez des itinéraires au moteur de routage ASP.NET et assurez-vous que les URL sont conviviales dans toute une application Web MVC. |
| • | Ajoutez des contrôles de navigation à une application pour permettre aux utilisateurs de localiser rapidement les données. |

# Leçon 1: Analyse de l'architecture de l'information

Lorsque vous analysez des cas d'utilisation et des user stories pour planifier un modèle pour votre application Web, vous devez déterminer les types d'objets que votre application Web gérera. Par exemple, si votre application Web fournit une documentation technique pour un produit, vous pouvez planifier des classes de modèles telles que des guides d'installation, des guides de l'utilisateur, des questions fréquemment posées (FAQ), des réponses aux FAQ, des catégories, des commentaires et d'autres classes de modèles. Le modèle impose une structure logique d'objets sur l'application Web. Cependant, vous devez également tenir compte de la manière dont les utilisateurs s'attendent à ce que les informations soient structurées. Les utilisateurs peuvent s'attendre à une structure hiérarchique, dans laquelle les FAQ, les réponses aux FAQ et les commentaires sont présentés sous le produit auquel ils se rapportent. Lorsque vous créez une application Web MVC complexe, vous devez savoir comment présenter les classes de modèle de manière logique, afin que votre application Web soit conviviale.

## **Objectifs de la leçon**

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

|  |  |
| --- | --- |
| • | Décrivez l'architecture de l'information. |
| • | Décrivez l'optimisation des moteurs de recherche et son importance pour les développeurs Web. |

## **Qu'est-ce que l'architecture de l'information?**

Lorsque les informations gérées par votre application Web sont complexes et multiformes, il est facile de présenter des objets de manière déroutante. À moins que vous ne réfléchissiez attentivement à la manière dont les utilisateurs s'attendent à ce que les informations soient structurées et à la manière dont ils s'attendent à accéder à un contenu utile, vous pouvez créer involontairement une application Web inutilisable. Pendant le développement, lorsque vous travaillez avec une quantité limitée de données, cette confusion peut ne pas devenir apparente. Ensuite, lorsque des données réelles sont ajoutées à votre base de données au moment du déploiement de la production, il devient clair que l'application Web prête à confusion. Vous pouvez éviter cette éventualité en planifiant l'architecture des informations.

L'architecture de l'information est une structure logique pour les objets gérés par votre application Web. Vous devez concevoir une telle architecture de manière à ce que les utilisateurs puissent trouver rapidement du contenu sans comprendre les aspects techniques de votre application Web. Par exemple, les utilisateurs devraient être en mesure de trouver une réponse technique sur leur produit sans comprendre la structure de la base de données ou les classes qui constituent le modèle.

**Exemple de scénario: un site Web de documentation technique**

Pour comprendre la nécessité d'une architecture de l'information, envisagez une application Web qui présente des informations techniques sur les produits d'une entreprise pour les clients et les ingénieurs. Dans cet exemple, l'entreprise fabrique des fours de chauffage domestique, qui sont installés dans les maisons par des chauffagistes qualifiés. L'application Web est conçue pour permettre aux clients de trouver des instructions, des conseils et des astuces. Cette application Web est également destinée à permettre aux ingénieurs d'obtenir la documentation technique sur l'installation et l'entretien des fours.

L'équipe de développement a identifié les user stories simples suivantes:

|  |  |
| --- | --- |
| • | Les clients ont un certain problème avec leurs chaudières. Ils veulent trouver une réponse FAQ spécifique qui résout le problème. Ils connaissent le nom du produit de la chaudière et le combustible, mais pas le numéro de produit. Ils visitent l'application Web et accèdent au nom de la chaudière. Ils cliquent sur la FAQ pour le nom de leur chaudière et trouvent la réponse dont ils ont besoin. |
| • | Les ingénieurs ont besoin du dernier manuel d'installation d'une chaudière. Ils connaissent le numéro de produit de la chaudière, le nom du produit et le type de combustible. Ils visitent le site et accèdent au nom de la chaudière. Ils cliquent sur le lien Manuels et trouvent le manuel d'installation. |
| • | Les ingénieurs ont des applications Web, qu'ils veulent lier à un nom de chaudière spécifique. Vous voulez vous assurer que l'URL est simple et lisible pour les clients et les ingénieurs. |

Vous avez déjà planifié les classes de modèles suivantes:

|  |  |
| --- | --- |
| • | fourneau. Chaque objet de four est un produit fabriqué et vendu par l'entreprise. |
| • | Catégorie. Les catégories organisent les fours par type. Par exemple, vous pouvez avoir des catégories telles que les combustibles au mazout, au gaz et solides. |
| • | FAQ. Chaque FAQ concerne un seul four. Chaque four peut avoir de nombreuses questions. La classe comprend à la fois les propriétés Question et Réponse. |
| • | Installation Manuel. Chaque four a un seul manuel d'installation sous la forme d'un document PDF. |
| • | Manuel de l'Utilisateur. Chaque four dispose d'un seul manuel d'utilisation sous la forme d'un document PDF. |

**Planification d'une hiérarchie logique**

Vous pouvez voir dans les user stories que la FAQ et les manuels sont tous deux accessibles en accédant d'abord au produit concerné. Vous pouvez également voir que l'entreprise propose différents produits et que les clients et les ingénieurs connaissent le type de combustible pour un four particulier. Par conséquent, vous pouvez organiser les fours en catégories par type de combustible. La liste suivante présente une hiérarchie logique d'objets, qui aide à la fois les clients et les ingénieurs à trouver les informations dont ils ont besoin en cliquant sur chaque niveau:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| • | Catégorie   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | o | fourneau   |  |  | | --- | --- | | ▪ | FAQ | | ▪ | Manuel de l'Utilisateur | | ▪ | Installation Manuel | | |

**Présentation d'une hiérarchie dans les contrôles de navigation**

L'architecture des informations que vous concevez doit être présentée sur les pages Web sous la forme de commandes de navigation. Les approches courantes de ces contrôles comprennent:

|  |  |
| --- | --- |
| • | Menus du site. La plupart des sites Web ont un menu principal qui présente les principaux domaines de contenu. Pour les applications Web simples, le menu principal peut inclure un petit nombre de liens statiques. Pour les applications Web plus volumineuses, lorsque les utilisateurs cliquent sur un lien de menu de site, un sous-menu apparaît. |
| • | Vues arborescentes. Une arborescence est un menu qui montre plusieurs niveaux de hiérarchie des informations. Habituellement, les utilisateurs peuvent développer ou réduire des objets à chaque niveau, pour localiser le contenu dont ils ont besoin. Les arborescences sont utiles pour présenter des hiérarchies complexes dans des structures navigables. |
| • | Sentiers d'Ariane. Un fil d'Ariane est un contrôle de navigation qui montre à l'utilisateur où il se trouve dans l'application Web. Habituellement, un fil d'Ariane affiche les pages actuelles et toutes les pages parentes de la hiérarchie, avec la page d'accueil comme niveau supérieur. Les fil d'Ariane vous permettent de comprendre comment une page s'intègre dans l'architecture des informations affichée dans les menus et les arborescences. |

Les types de contrôles de navigation que vous créez dans votre application Web dépendent de la manière dont vous attendez des utilisateurs qu'ils trouvent des informations.

**Présentation d'une hiérarchie dans les URL**

Vous pouvez augmenter la convivialité de votre application Web en reflétant l'architecture des informations dans les URL, que les utilisateurs voient dans la barre d'adresses du navigateur Web. Dans de nombreuses applications Web, les URL incluent souvent des informations longues et insondables telles que des identificateurs uniques globaux (GUID) et de longues chaînes de requête avec de nombreux paramètres. Ces URL empêchent les utilisateurs de formuler manuellement l'adresse d'un élément, et ces URL sont difficiles à lier à une page de votre application Web. Au lieu de cela, les URL peuvent être simples et compréhensibles, pour aider les utilisateurs à parcourir votre contenu.

Dans les applications Web MVC, la configuration par défaut est simple, mais elle est basée sur des contrôleurs, des actions et des paramètres. Voici quelques exemples d'URL qui suivent ce modèle par défaut:

|  |  |
| --- | --- |
| • | http: // site / Fournaise / Détails / 23:Cette URL renvoie au contrôleur de la fournaise et à l'action Détails, et elle affiche la fournaise avec l'ID 23. |
| • | http: // site / FAQQuestion / Détails / 234:Cette URL renvoie au contrôleur FAQQuestion et à l'action Détails, et affiche la FAQ avec l'ID 234. |
| • | http: // site / InstallationManual / Details / 3654:Cette URL renvoie au contrôleur InstallationManual et à l'action Détails, et affiche le manuel avec l'ID 3654. |

Comme vous pouvez le voir, l'utilisateur de l'application Web doit comprendre le fonctionnement des contrôleurs, des actions et des paramètres d'action, pour formuler eux-mêmes les URL. Au lieu de cela, les utilisateurs peuvent utiliser des URL plus faciles à comprendre, telles que les suivantes, car elles reflètent la hiérarchie des informations:

|  |  |
| --- | --- |
| • | http: // site / OilFired / HotBurner2000:Cette URL renvoie à une fournaise en spécifiant la catégorie de combustible et le nom du produit. Les clients et les ingénieurs peuvent comprendre ces valeurs. |
| • | http: // site / OilFired / HotBurner2000 / HowToRestartMyFurnace:Cette URL renvoie à une FAQ en spécifiant le nom du four auquel la question se rapporte. |
| • | http: // site / OilFired / HotBurner2000 /:Cette URL renvoie au manuel d'installation en spécifiant le nom de la fournaise auquel le manuel se rapporte. |

Comme vous pouvez le voir, ces URL sont faciles à comprendre pour les clients et les ingénieurs, car les URL sont basées sur une hiérarchie logique et les informations dont disposent déjà les utilisateurs. Vous pouvez contrôler les URL utilisées par votre application Web ASP.NET en configurant le moteur de routage ASP.NET.

**Question:**Pourquoi peut-il être difficile pour les utilisateurs de comprendre les URL en fonction des contrôleurs, des actions et des paramètres?

## **Qu'est-ce que l'optimisation des moteurs de recherche?**

La plupart des utilisateurs trouvent des applications Web à l'aide de moteurs de recherche. Les utilisateurs ont tendance à visiter les liens qui apparaissent en haut des résultats des moteurs de recherche plus fréquemment que ceux situés en bas et ceux de la deuxième page de résultats. Pour cette raison, les administrateurs et les développeurs de sites Web essaient de s'assurer que leur application Web apparaît en bonne place dans les résultats des moteurs de recherche, en utilisant un processus appelé Search Engine Optimization (SEO). Le référencement garantit que plus de personnes visitent votre application Web.

Les moteurs de recherche examinent le contenu de votre application Web, en l'explorant avec un programme appelé robot Web. Si vous comprenez les priorités que les robots Web et les index des moteurs de recherche utilisent pour classer les résultats de la recherche, vous pouvez créer une application Web conforme à ces priorités et qui apparaît ainsi en bonne place dans les résultats des moteurs de recherche.

**Meilleures pratiques SEO**

Différents moteurs de recherche ont différents robots Web avec différents algorithmes pour hiérarchiser les résultats. Les détails complets de ces algorithmes ne sont généralement pas publiés. Cependant, si vous adoptez les bonnes pratiques suivantes, votre site a de bonnes chances de figurer en bonne place dans les résultats de recherche:

|  |  |
| --- | --- |
| • | Assurez-vous que chaque page Web de votre application Web comporte un élément <title> significatif dans la section <head> du HTML. |
| • | Assurez-vous d'inclure une balise <meta name = "keywords"> dans l'élément <head> de chaque page. L'attribut content de cet élément doit inclure des mots clés qui décrivent le contenu de la page avec précision. |
| • | Assurez-vous d'inclure une balise <meta name = "description"> dans l'élément <head> de chaque page. L'attribut content de cet élément doit inclure une phrase qui décrit le contenu de la page avec précision. |
| • | Assurez-vous que l'élément <title> et les éléments <meta> sont différents pour chaque page de votre application Web. |
| • | Choisissez un nom de domaine qui comprend un ou plusieurs de vos mots clés les plus importants. |
| • | Assurez-vous que les mots clés apparaissent dans les éléments <h1>, <h2> ou <h3> de votre page Web. |
| • | Assurez-vous que les contrôles de navigation permettent aux robots Web d'explorer l'ensemble de votre application Web. Par exemple, votre site peut contenir du contenu que les utilisateurs ne peuvent trouver qu’avec l’outil de recherche, et non en cliquant sur des liens. Les robots Web ne pouvant pas utiliser les outils de recherche, ce contenu ne sera pas indexé. |
| • | Assurez-vous que les URL n'incluent pas de GUID ou de texte de requête long. |

**SEO et structure de l'application Web**

L'architecture de l'information est un sujet étroitement lié au référencement. En effet, l'architecture de l'information et le référencement sont tous deux pertinents pour la structure, la hiérarchie et l'accessibilité des objets de votre application Web. Les utilisateurs cliquent sur les liens des menus pour accéder aux pages qui les intéressent. Les robots Web utilisent les mêmes liens pour naviguer dans l'application Web et analyser son contenu. Les utilisateurs préfèrent les URL sans GUID et le texte de requête long car ils sont significatifs. Les robots Web ignorent souvent les liens contenant des GUID et un long texte de requête. De plus, lorsque des mots clés apparaissent dans les URL, les robots Web donnent la priorité à une page Web dans les résultats de recherche.

En tant que développeur ASP.NET MVC, vous devez comprendre les principes de référencement et les utiliser chaque fois que vous écrivez du code, pour vous assurer de ne pas endommager le positionnement de votre application Web sur les moteurs de recherche. Les vues sont essentielles au référencement car elles affichent les balises <meta> et les éléments <title>. Les itinéraires et la configuration du moteur de routage sont également essentiels, car, en utilisant des itinéraires, vous pouvez contrôler les URL générées par votre application Web.

**Question:**Un développeur crée une vue partielle contenant des balises <meta> et des balises <title>. Le développeur utilise cette vue partielle sur chaque page de l'application Web. Ces actions sont-elles conformes aux meilleures pratiques SEO?

# Leçon 2: Configuration des itinéraires

ASP.NET permet aux développeurs de contrôler les URL utilisées par une application Web, de lier les URL et le contenu en configurant des itinéraires. Une route est un objet qui analyse une URL demandée et détermine le contrôleur et l'action vers lesquels la demande doit être transmise. Ces routes sont appelées routes entrantes. Les assistants HTML utilisent également des routes lorsqu'ils formulent des liens vers des contrôleurs et des actions. Ces routes sont appelées routes sortantes. Vous devez savoir comment écrire du code qui ajoute une nouvelle route à votre application. Vous devez également savoir comment le moteur de routage interprète une route afin que les requêtes soient envoyées aux contrôleurs appropriés et que les utilisateurs voient des URL significatives.

## **Objectifs de la leçon**

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

|  |  |
| --- | --- |
| • | Décrivez les fonctionnalités du moteur de routage ASP.NET. |
| • | Décrivez les avantages de l'ajout de routes à une application Web ASP.NET MVC. |
| • | Expliquez comment ajouter et configurer des itinéraires. |
| • | Expliquez comment utiliser les routes pour passer des paramètres. |
| • | Ajoutez des itinéraires pour gérer la formulation d'URL. |

## **Le moteur de routage ASP.NET**

Le routage régit la manière dont les URL sont formulées et comment elles correspondent aux contrôleurs et aux actions. Le routage ne fonctionne pas sur le protocole, le serveur, le domaine ou le numéro de port d'une URL, mais uniquement sur les répertoires et le nom de fichier dans l'URL relative. Par exemple, dans l'URL, http://www.advworks.com/photo/display/1, le routage opère sur le chemin relatif / photo / display / 1.

Dans ASP.NET, les routes sont utilisées à deux fins:

|  |  |
| --- | --- |
| • | Pour analyser les URL demandées par les navigateurs. Cette analyse garantit que les demandes sont transmises aux bons contrôleurs et actions. Celles-ci sont appelées URL entrantes. |
| • | Pour formuler des URL dans des liens de pages Web et d'autres éléments. Lorsque vous utilisez des helpers tels que Html.ActionLink () et Url.Action () dans les vues MVC, les helpers construisent des URL en fonction des itinéraires dans la table de routage. Celles-ci sont appelées URL sortantes. |

Lorsque vous créez une application Web MVC dans Visual Studio à l'aide d'un modèle de projet, l'application dispose d'un itinéraire par défaut. Vous devez comprendre cette route par défaut avant d'envisager d'ajouter des routes supplémentaires pour manipuler les URL dans l'application.

**Noter:**Vous pouvez utiliser les itinéraires et le moteur de routage pour gérer les URL relatives dans n'importe quelle application ASP.NET et pas seulement dans celles qui utilisent le modèle de programmation MVC.

**La route par défaut**

L'itinéraire par défaut est simple mais logique et fonctionne bien dans de nombreuses applications Web. L'itinéraire par défaut examine les trois premiers segments de l'URL. Chaque segment est délimité par une barre oblique:

|  |  |
| --- | --- |
| • | Le premier segment est interprété comme le nom du contrôleur. Le moteur de routage transmet la demande à ce contrôleur. Si un premier segment n'est pas spécifié, la route par défaut transmet la demande à un contrôleur appelé Home. |
| • | Le deuxième segment est interprété comme le nom de l'action. Le moteur de routage transmet la demande à cette action. Si un deuxième segment n'est pas spécifié, la route par défaut transmet la demande à un contrôleur appelé Index. |
| • | Le troisième segment est interprété comme une valeur d'ID, qui est transmise à l'action en tant que paramètre. Le paramètre est facultatif, donc si un troisième segment n'est pas spécifié, aucune valeur par défaut n'est transmise à l'action. |

Vous pouvez voir que lorsque l'utilisateur demande l'URL, http://www.advworks.com/photo/display/1, l'itinéraire par défaut transmet le paramètre ID 1 à l'action Afficher du contrôleur Photo.

**La route par défaut**

routes.MapRoute (

nom: "Par défaut",

url: "{controller} / {action} / {id}",

par défaut: nouveau {

controller = "Accueil",

action = "Index",

id = UrlParameter.Optional

}

);

**Itinéraires personnalisés**

Les développeurs ajoutent leurs propres routes personnalisées à une application Web pour deux raisons principales:

|  |  |
| --- | --- |
| • | Pour faciliter la compréhension des URL par les visiteurs du site. Une URL telle que http://www.advworks.com/photo/display/1 est logique pour le visiteur, mais elle nécessite quelques connaissances pour formuler de telles URL. Dans ce cas, pour saisir la bonne URL dans la barre d'adresse du navigateur Web, l'utilisateur doit connaître certaines informations. Ces informations incluent le nom du contrôleur, la photo, le nom de l'action, l'affichage et l'ID de la photo d'intérêt. Si vous utilisez des identificateurs uniques globaux (GUID) dans la base de données, le segment ID de l'URL peut être long et difficile à mémoriser. Idéalement, vous devez tenir compte de ce que les utilisateurs savent et créer des itinéraires qui acceptent ces informations. Dans cet exemple, les utilisateurs peuvent connaître le titre d'une photo qu'ils avaient vue précédemment dans votre application. Vous devez créer un itinéraire capable d'interpréter une URL telle que http://www.advworks.com/photo/title/My%20Photo%20Title, et transférez l'URL vers une action appropriée pour afficher la bonne photo. Bien que les utilisateurs cliquent généralement sur des liens pour faire des demandes, des URL conviviales comme celles-ci facilitent l'utilisation d'un site et permettent de créer des liens depuis d'autres sites. |
| • | Pour améliorer le classement des moteurs de recherche. Les moteurs de recherche ne donnent pas la priorité aux pages Web qui ont des GUID ou un long texte de requête dans l'URL. Certains robots Web n'explorent même pas ces liens. De plus, certains moteurs de recherche améliorent le classement d'une page lorsqu'un ou plusieurs de ses mots clés apparaissent dans l'URL. Les URL conviviales sont donc un outil essentiel dans le référencement. |

**Usines et itinéraires de contrôleur**

MVC utilise des fabriques de contrôleurs qui aident à créer une instance d'une classe de contrôleur pour gérer une demande. MVC utilise des invocateurs d'action pour appeler la bonne action et transmettre des paramètres à cette méthode d'action. Les usines de contrôleurs et les invocateurs d'action se réfèrent à la table de routage pour terminer leurs tâches. Voici les étapes que MVC exécute lorsqu'une demande est reçue d'un navigateur Web:

|  |  |
| --- | --- |
| 1. | Un objet MvcHandler crée une fabrique de contrôleurs. La fabrique de contrôleurs est l'objet qui instancie un contrôleur pour répondre à la demande. |
| 2. | L'usine de contrôleur consulte la table de routage pour déterminer la bonne classe de contrôleur à utiliser. |
| 3. | La fabrique de contrôleurs crée un objet Controller et le MvcHandler appelle la méthode Execute dans ce contrôleur. |
| 4. | ControllerActionInvoker examine l'URL de la demande et consulte la table de routage pour déterminer l'action dans l'objet Controller à appeler. |
| 5. | ControllerActionInvoker utilise un classeur de modèle pour déterminer les valeurs qui doivent être transmises à l'action en tant que paramètres. Le classeur de modèles consulte la table de routage pour déterminer si des segments de l'URL doivent être transmis en tant que paramètres. Le classeur de modèles peut également transmettre des paramètres à partir d'un formulaire publié, du texte de la requête d'URL ou des fichiers téléchargés. |
| 6. | ControllerActionInvoker exécute l'action. Souvent, l'action crée une nouvelle instance d'une classe de modèle, peut-être en interrogeant la base de données avec les paramètres que l'invocateur lui a transmis. Cet objet de modèle est passé à une vue, pour afficher les résultats à l'utilisateur. |

Comme vous pouvez le voir, vous pouvez utiliser des routes pour gérer le comportement des fabriques de contrôleurs, des invocateurs d'action et des classeurs de modèles, car tous ces objets font référence à la table de routage. MVC est hautement extensible; par conséquent, les développeurs peuvent créer des implémentations personnalisées des fabriques de contrôleurs, des invocateurs d'action et des classeurs de modèles. Cependant, en utilisant des itinéraires, vous pouvez généralement implémenter la fonctionnalité d'URL dont vous avez besoin avec les implémentations par défaut de ces classes. Vous devez vous assurer que les routes ne peuvent pas implémenter les fonctionnalités requises avant de planifier de personnaliser les fabriques de contrôleurs, les invocateurs d'action ou les classeurs de modèles.

**Question:**Un utilisateur souhaite modifier un commentaire qu'il a créé dans votre application MVC. Vous n'avez créé aucun itinéraire personnalisé. Quelle URL pensez-vous que l'utilisateur doit demander, pour voir la vue d'édition avec le bon commentaire?

## **Ajout et configuration d'itinéraires**

Chaque application Web MVC a un objet RouteTable dans lequel les routes sont stockées, dans les propriétés Routes. Vous pouvez ajouter des itinéraires à la propriété Routes en appelant la méthode MapRoute ().

Dans les modèles de projet Visual Studio pour MVC 5, un fichier de code RouteConfig.cs dédié existe dans le dossier App\_Start. Ce fichier inclut la méthode statique RouteConfig.RegisterRoutes () où la route par défaut est ajoutée à l'objet RouteTable. Vous pouvez ajouter des itinéraires personnalisés dans cette méthode. Le fichier Global.asax.cs inclut un appel à RouteConfig.RegisterRoutes () dans la méthode Application\_Start (), ce qui signifie que les routes sont ajoutées à la table de routage chaque fois que l'application MVC démarre.

**Propriétés d'une route**

Avant de pouvoir ajouter un itinéraire, vous devez comprendre les propriétés d'un itinéraire. Cela permet de garantir que ces propriétés correspondent correctement aux segments d'URL et de transmettre les demandes et les paramètres au bon emplacement. Les propriétés d'un itinéraire sont les suivantes:

|  |  |
| --- | --- |
| • | Nom. Cette propriété de chaîne attribue un nom à une route. Il n'est pas impliqué dans la mise en correspondance ou le transfert de demande. |
| • | URL. Cette propriété de chaîne est un modèle d'URL comparé à une demande, pour déterminer si l'itinéraire doit être utilisé. Vous pouvez utiliser des variables de segment pour faire correspondre une partie de l'URL. Vous pouvez spécifier une variable de segment en utilisant des accolades. Par exemple, si vous spécifiez l'URL "photo / {title}", la route correspond à toute demande où l'URL relative commence par la chaîne "photo /" et comprend un autre segment. La variable de segment est «titre» et peut être utilisée ailleurs dans l'itinéraire. |
| • | Contraintes. Parfois, vous devez placer des contraintes supplémentaires sur l'itinéraire pour vous assurer qu'il correspond uniquement aux demandes appropriées. Par exemple, si vous souhaitez que des URL relatives au format "photo / 34" spécifient une photo avec l'ID "34", vous devez utiliser une propriété d'URL telle que "photo / {id}". Cependant, observez que ce modèle d'URL correspond également à l'URL relative, «photo / créer», car il comporte un segment supplémentaire. Pour les ID, vous devez contraindre l'URL à ne correspondre qu'aux segments comprenant des chiffres. Vous pouvez le faire en ajoutant une contrainte à l'itinéraire. La propriété Contraintes vous permet de spécifier une expression régulière pour chaque variable de segment. L'itinéraire correspondra à une demande uniquement si toutes les variables de segment correspondent aux expressions régulières que vous spécifiez. |
| • | Valeurs par défaut. Cette propriété de chaîne peut attribuer des valeurs par défaut aux variables de segment dans le modèle d'URL. Les valeurs par défaut sont utilisées pour les variables de segment lorsque la demande ne les spécifie pas. |

**Le code d'itinéraire par défaut**

La route par défaut spécifie les propriétés Name, URL et Defaults pour obtenir les noms de contrôleur et d'action, ainsi que les valeurs d'ID, à partir de l'URL relative demandée. En examinant ces propriétés, vous pouvez comprendre comment créer vos propres itinéraires.

**La route par défaut dans RouteConfig.cs**

routes.MapRoute (

nom: "Par défaut",

url: "{controller} / {action} / {id}",

par défaut: new {controller = "Home", action = "Index", id = UrlParameter.Optional}

);

Observez que le modèle d'URL spécifie trois segments: {controller}, {action} et {id}. Cela signifie qu'une URL relative sous la forme «photo / affichage / 23» sera envoyée à l'action Afficher de la classe PhotoController. Le troisième segment sera passé en paramètre à la méthode d'action. Par conséquent, si l'action Afficher accepte un paramètre nommé id, la valeur 23 sera transmise par MVC.

Dans la propriété par défaut, le segment {id} est marqué comme facultatif. Cela signifie que l'itinéraire correspond toujours à une URL relative, même si aucun troisième segment n'est spécifié. Par exemple, l'itinéraire correspond à l'URL relative, «photo / créer», et transmet l'URL à l'action Créer de la classe PhotoController.

La propriété defaults spécifie une valeur par défaut «Index» pour le segment {action}. Cela signifie que l'itinéraire correspond toujours à une URL relative, même si aucun deuxième segment n'est spécifié. Par exemple, l'itinéraire correspond à l'URL relative, «photo», et transmet l'URL à l'action Index de la classe PhotoController.

Enfin, la propriété defaults spécifie une valeur par défaut «Home» pour le segment {controller}. Cela signifie que l'itinéraire correspond toujours à une URL relative, même si aucun segment n'est spécifié, c'est-à-dire s'il n'y a pas d'URL relative. Par exemple, l'URL absolue «http://www.advworks.com» correspond à cette route et l'URL est transmise à l'action Index de la classe HomeController.

Comme vous pouvez le voir, avec la route par défaut en place, les développeurs créent la page d'accueil de l'application Web en créant un contrôleur nommé, HomeController, avec une action nommée Index. Cette action renvoie généralement une vue appelée Index à partir du dossier Views / Home.

**Créer des itinéraires personnalisés**

Vous pouvez ajouter des itinéraires personnalisés en appelant la méthode routes.MapHttpRoute (), tout comme l'itinéraire par défaut.

**Un itinéraire personnalisé**

routes.MapRoute (

nom: "PhotoRoute",

url: "photo / {id}",

par défaut: new {controller = "Photo", action = "Details"},

contraintes: nouveau {id = "[0-9] +"}

);

Observez que le modèle d'URL de l'itinéraire correspond à toute URL relative commençant par «photo /» et comportant un ou plusieurs segments. Cependant, dans la propriété contraintes, la route spécifie une expression régulière pour la variable de segment {id}. Cette expression régulière ne correspond que si le segment est composé de chiffres compris entre 1 et 9. Si l'utilisateur demande l'URL relative, «photo / 234», alors cette route correspond et le moteur de routage transmet la demande à l'action Détails de la classe PhotoController. Si l'utilisateur demande l'URL relative, «photo / créer», alors cette route ne correspond pas et une route suivante doit être utilisée pour transférer la demande.

**Préséance des itinéraires**

Les itinéraires sont évalués par le moteur de routage dans l'ordre dans lequel ils sont ajoutés à la collection RouteTable.Routes. Si une route correspond, le moteur de routage l'utilise et n'évalue aucune route ultérieure dans la collection. Si un itinéraire ne correspond pas, le moteur de routage l'ignore et évalue l'itinéraire suivant dans la collection. Pour cette raison, vous devez ajouter des routes à la collection RouteTable.Routes dans l'ordre approprié. Vous pouvez d'abord ajouter les itinéraires les plus spécifiques, tels que les itinéraires qui n'ont pas de variables de segment et aucune valeur par défaut. Les itinéraires avec contraintes doivent être ajoutés avant les itinéraires sans contraintes.

La route nommée "Par défaut" correspond à toutes les demandes, y compris celles sans URL relative. Cette route par défaut doit être la dernière que vous ajoutez à la collection RouteTable.Routes. Si le moteur de routage évalue cette route, la route est toujours utilisée. Les itinéraires ajoutés après l'itinéraire «par défaut» ne sont jamais utilisés.

**Question:**Un développeur a supprimé tout le code de la méthode Application\_Start () dans Global.asax.cs. Lorsque le développeur exécute l'application, il reçoit des erreurs 404 pour toute demande, quelle que soit l'URL relative. Pourquoi cela se produit-il?

## **Utilisation d'itinéraires pour passer des paramètres**

Le moteur de routage sépare l'URL relative d'une requête en un ou plusieurs segments. Chaque barre oblique délimite un segment du suivant. Si vous souhaitez que l'un des segments spécifie le nom du contrôleur, vous pouvez utiliser la variable de segment {controller}. L'usine de contrôleur interprète toujours cette variable comme le contrôleur à instancier. Pour corriger une route vers un seul contrôleur, vous pouvez également spécifier une valeur pour la variable de contrôleur dans la propriété defaults. De la même manière, si vous souhaitez que l'un des segments spécifie la méthode d'action, vous pouvez utiliser la variable de segment {action}. L'invocateur d'action interprète toujours cette variable comme l'action à appeler. Sinon, pour corriger une route vers une seule action, vous pouvez spécifier une valeur pour la variable d'action dans la propriété defaults.

Les variables de segment ou les valeurs par défaut avec d'autres noms n'ont aucune signification particulière pour MVC et sont transmises aux actions. Vous pouvez accéder aux valeurs de ces variables à l'aide de l'une des deux méthodes suivantes: la collection RouteData.Values ​​ou à l'aide de la liaison de modèle pour transmettre des valeurs aux paramètres d'action.

**La collection RouteData.Values**

Dans la méthode d'action, vous pouvez accéder aux valeurs de n'importe quelle variable de segment à l'aide de la collection RouteData.Values. Prenons l'exemple suivant:

|  |  |
| --- | --- |
| • | Dans RouteConfig.cs, vous avez défini une route avec le modèle d'URL «{controller} / {action} / {title}». |
| • | L'utilisateur demande l'URL relative, "photo / edit / my% 20photo". |

La collection RouteData.Values ​​contient les valeurs suivantes, que vous pouvez utiliser dans la méthode d'action:

|  |  |
| --- | --- |
| • | **RouteData.Value ["contrôleur"]**: Cette valeur a la valeur «photo». |
| • | **RouteData.Value ["action"]**: Cette valeur a la valeur «edit». |
| • | **RouteData.Value ["titre"]**: Cette valeur a la valeur «my% 20photo». |

**Utilisation de la liaison de modèle pour obtenir des paramètres**

Considérez la méthode d'action suivante.

public void ActionMethod Display (int PhotoID)

{

vue de retour (PhotoID);

}

L'invocateur d'action MVC par défaut est une classe nommée ControllerActionInvoker. Cet appelant utilise une classe nommée, DefaultModelBinder, pour transmettre les paramètres appropriés aux actions. Ce classeur de modèles utilise la logique suivante pour transmettre les paramètres:

|  |  |
| --- | --- |
| 1. | Le classeur examine la définition de l'action à laquelle il doit passer des paramètres. Dans l'exemple, le classeur détermine que l'action nécessite un paramètre entier appelé PhotoID. |
| 2. | Le classeur recherche des valeurs dans la requête, qui peuvent être transmises en tant que paramètres par nom. Dans l'exemple, le classeur recherche des entiers car l'action nécessite des entiers. Le classeur recherche les valeurs aux emplacements suivants dans un ordre séquentiel:   |  |  | | --- | --- | | une. | Valeurs de formulaire. Si l'utilisateur remplit un formulaire et clique sur le bouton Soumettre, le classeur peut trouver des paramètres dans la collection Request.Form. | | b. | Valeurs d'itinéraire. Si l'itinéraire correspondant inclut la variable de segment {photoid}, le classeur peut trouver la valeur du paramètre PhotoID dans la collection RouteData.Values. Cette correspondance est insensible à la casse. | | c. | Chaînes de requête. Si la demande utilisateur inclut des paramètres nommés après un point d'interrogation, le classeur peut trouver ces paramètres dans la collection Request.QueryString. | | ré. | Des dossiers:Si la demande utilisateur inclut des fichiers téléchargés, ces fichiers peuvent être utilisés comme paramètres dans une action. Les paramètres de fichier dans les actions doivent utiliser le type HttpPostedFileBase. | |

Par conséquent, vous pouvez utiliser le classeur de modèle par défaut pour transmettre des paramètres aux chaînes d'action. Si le nom d'un paramètre de méthode d'action correspond au nom d'une variable de segment d'itinéraire, le classeur de modèle transmet le paramètre automatiquement.

**Paramètres facultatifs**

Parfois, vous souhaitez que les demandes correspondent à un itinéraire, que les valeurs de paramètre soient ou non fournies. Par exemple, vous souhaiterez peut-être permettre aux utilisateurs de spécifier une couleur pour un produit.

**Une route avec un paramètre facultatif**

routes.MapRoute (

nom: "ProductRoute",

url: "product / {id} / {color}",

par défaut: new {color = UrlParameter.Optional}

);

Les valeurs par défaut spécifient que {color} est facultatif. Si l'URL relative comporte trois segments, l'itinéraire correspond et le troisième segment est transmis à la valeur RouteData.Values ​​["color"]. Si l'URL relative ne comporte que deux segments, l'itinéraire correspond, car la {couleur} est facultative. Dans ce cas, la valeur RouteData.Values ​​["color"] n'existe pas. Si l'URL relative n'a qu'un seul segment, l'itinéraire ne correspond pas.

**Question:**Un développeur a remplacé le classeur de modèles par défaut par un classeur de modèles personnalisé. Désormais, plusieurs méthodes d'action lancent des exceptions sur les lignes qui utilisent des paramètres d'action. Comment pouvez-vous corriger ce bug?

## **Démonstration: comment ajouter des itinéraires**

Dans cette démonstration, vous verrez comment créer des URL en ajoutant des routes à l'application Web MVC. Vous verrez également comment permettre aux visiteurs de l'application Web Opéras de parcourir les opéras par titre.

### **Étapes de démonstration**

|  |  |
| --- | --- |
| • | Vous trouverez les étapes dans la section «Leçon 2: Configuration des itinéraires» sur la page suivante:<https://github.com/MicrosoftLearning/20486-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486C/20486C_MOD07_DEMO.md>. |

## **Tests unitaires et itinéraires**

Les tests unitaires sont de petites méthodes isolées qui vérifient un aspect spécifique de votre code. Au fur et à mesure que votre projet devient de plus en plus complexe, vous pouvez réexécuter tous les tests unitaires pour vous assurer que la fonctionnalité vérifiée reste en état de fonctionnement. Si votre équipe utilise le développement piloté par les tests (TDD), vous pouvez définir des tests pour spécifier une exigence fonctionnelle, puis écrire du code qui réussit votre test. De telles approches augmentent considérablement la qualité de votre code, en mettant en évidence les bogues dès qu'ils surviennent.

Vous pouvez utiliser des tests unitaires pour vous assurer que les routes de votre application Web MVC se comportent comme prévu. Si vous effectuez de tels tests, vous pouvez empêcher de nombreuses erreurs HTTP 404 et erreurs de serveur internes d'atteindre les utilisateurs d'applications Web. Les utilisateurs ont généralement une faible tolérance pour de telles erreurs, et si des erreurs surviennent trop souvent, ils navigueront vers les sites Web de vos concurrents.

**Création d'un double de test de contexte HTTP**

Dans une demande de page Web, ASP.NET stocke les détails de la demande dans un objet de contexte HTTP. Ces détails incluent l'URL demandée, les détails du navigateur et d'autres informations. Pour écrire un test unitaire pour les routes, vous devez simuler l'objet Contexte HTTP en créant une double classe de test.

**Un double test de contexte HTTP**

classe publique FakeHttpContextForRouting: HttpContextBase

{

FakeHttpRequestForRouting \_request;

FakeHttpResponseForRouting \_response;

// Le constructeur vous permet de spécifier une URL de requête et un chemin d'application dans le test

public FakeHttpContextForRouting (chaîne appPath = "/", chaîne requestUrl = "~ /")

{

// Créer de faux objets de requête et de réponse

\_request = new FakeHttpRequestForRouting (appPath, requestUrl);

\_response = new FakeHttpResponseForRouting ();

}

// Cette propriété renvoie la fausse requête

public override HttpRequestBase Request

{

get {return \_request; }

}

// Cette propriété renvoie la fausse réponse

public override HttpResponseBase Response

{

get {return \_response; }

}

}

// Ceci est la fausse classe de requête

classe publique FakeHttpRequestForRouting: HttpRequestBase

{

string \_appPath;

string \_requestUrl;

public FakeHttpRequestForRouting (chaîne appPath, chaîne requestUrl)

{

\_appPath = appPath;

\_requestUrl = requestUrl;

}

chaîne de substitution publique ApplicationPath

{

get {return \_appPath; }

}

chaîne de substitution publique AppRelativeCurrentExecutionFilePath

{

get {return \_requestUrl; }

}

chaîne de substitution publique PathInfo

{

get {return ""; }

}

public override NameValueCollection ServerVariables

{

get {return new NameValueCollection (); }

}

}

// Ceci est la fausse classe de réponse

classe publique FakeHttpResponseForRouting: HttpResponseBase

{

chaîne de substitution publique ApplyAppPathModifier (chaîne virtualPath)

{

return virtualPath;

}

}

**Ecrire des tests de routage**

Après avoir créé un double de test de contexte HTTP, vous pouvez écrire des tests unitaires pour chaque route dans la table de routage. Ces tests unitaires adoptent les phases générales suivantes:

|  |  |
| --- | --- |
| • | Organiser. Dans la phase d'arrangement du test, vous pouvez créer un nouveau contexte HTTP à partir de votre double classe de test. Vous pouvez définir l'URL de demande de cet objet comme l'URL que vous souhaitez tester. Vous pouvez ensuite créer une nouvelle collection de routes et appeler la méthode RouteConfig.RegisterRoutes () dans votre application Web. |
| • | Acte. Dans la phase Act, vous pouvez tester les routes en appelant la méthode GetRouteData () de la collection de routes. Vous pouvez ensuite transmettre le faux contexte HTTP à cette méthode. |
| • | Affirmer. Dans la phase Assert, vous pouvez utiliser la collection RouteData.Values ​​pour vérifier que le contrôleur, l'action et les autres valeurs sont correctement attribués. |

**Un test d'unité de routage**

[Méthode d'essai]

public void Test\_Default\_Route\_ControllerOnly ()

{

// Ce test vérifie la route par défaut lorsque seul le contrôleur est spécifié

//Organiser

var context = new FakeHttpContextForRouting (requestUrl: "~ / ControllerName");

var routes = new RouteCollection ();

MyMVCApplication.RouteConfig.RegisterRoutes (routes);

// Agir

RouteData routeData = routes.GetRouteData (contexte);

// Affirmer

Assert.AreEqual ("ControllerName", routeData.Values ​​["controller"]);

Assert.AreEqual ("Index", routeData.Values ​​["action"]);

Assert.AreEqual (UrlParameter.Optional, routeData.Values ​​["id"]);

}

**Question:**Vous écrivez un test unitaire pour vérifier la table de routage de votre application Web. Vous voulez vous assurer que la demande est transmise à une méthode d'action appelée «Modifier». Quelle ligne de code utiliseriez-vous dans la phase Assert du test pour vérifier ce fait?

# Leçon 3: Création d'une structure de navigation

Une structure de navigation est une hiérarchie logique de pages Web, qui permet aux utilisateurs de parcourir les informations qui les intéressent. Dans les applications Web simples, la conception de la structure de navigation est également généralement simple. Au fur et à mesure que votre application Web se développe, il devient difficile de présenter des menus et des commandes de navigation qui montrent clairement aux utilisateurs ce que devrait être leur prochain clic, à chaque étape. Vous devez comprendre comment créer des menus, qui aident les utilisateurs à comprendre où ils se trouvent sur votre site et où aller ensuite. Vous pouvez le faire en créant des menus et des commandes de fil d'Ariane.

## **Objectifs de la leçon**

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

|  |  |
| --- | --- |
| • | Décrivez l'importance d'une navigation bien conçue. |
| • | Créez des menus clairs dans une application Web. |
| • | Ajoutez des contrôles de fil d'Ariane à une application Web. |
| • | Construisez la navigation de site dans une application Web. |

## **L'importance d'une navigation bien conçue**

L'architecture d'informations que vous concevez pour votre application Web est une hiérarchie logique d'objets. Cette hiérarchie doit être claire et compréhensible pour le public de votre application Web. Si votre application Web est conçue pour des personnes ayant des connaissances spécialisées, vous pouvez mettre en œuvre une architecture de l'information qui nécessite des connaissances spécialisées. Par exemple, dans une application Web d'ingénierie, vous pouvez baser votre architecture d'informations sur des technologies et des solutions spécialisées que les ingénieurs comprennent. Cependant, une telle architecture de l'information ne serait pas utile au grand public.

Après avoir conçu l'architecture des informations, vous devez la présenter dans votre application Web. Vous pouvez ajouter des itinéraires pour vous assurer que les URL reflètent votre architecture d'informations. De même, vous devez ajouter des contrôles de navigation pour présenter l'architecture des informations dans le corps de chaque page Web. Les contrôles de navigation que vous créez sont les principaux moyens par lesquels les utilisateurs parcourent votre application Web et localisent le contenu. Vous devez vous assurer de deux choses avec ces contrôles de navigation:

|  |  |
| --- | --- |
| • | Une prochaine étape évidente. Vous devez vous assurer que les utilisateurs peuvent facilement décider sur quel lien cliquer sur chaque page. Pour atteindre cet objectif, vous pouvez envisager des cas d'utilisation. Pour chaque cas d'utilisation, vous pouvez répertorier ce que l'utilisateur sait lorsqu'il arrive sur votre application Web. L'utilisateur doit pouvoir choisir le bon lien sans aucune connaissance spécialisée ou technique. Par exemple, évitez de baser vos liens sur tout type de jargon technique, à moins que vous ne sachiez que votre public le comprend. |
| • | Un contexte clair. Dans une application Web simple avec seulement un ou deux niveaux dans l'architecture de l'information, les utilisateurs peuvent facilement suivre où ils se trouvent. Dans les applications Web plus volumineuses comportant de nombreux niveaux, vous devez fournir des contrôles de navigation, qui indiquent aux utilisateurs où ils se trouvent et comment remonter la hiérarchie. Les arborescences et les commandes de fil d'Ariane remplissent bien cette fonction. |

**Meilleur entrainement:**Basez vos contrôles de navigation sur la hiérarchie des informations, plutôt que sur le modèle et les classes de modèle, les tables de base de données ou d'autres structures techniques de votre application Web. Les utilisateurs ne comprendront probablement pas les classes de modèles, les contrôleurs, les vues et les mécanismes de stockage de données qui sous-tendent votre application Web.

**Types courants de commandes de navigation**

N'oubliez pas que la plupart des visiteurs des applications Web ont parcouru des milliers d'applications Internet avant de visiter votre page d'accueil. Les visiteurs sont généralement familiarisés avec les types de menus courants utilisés dans de nombreuses applications Web, et vous pouvez utiliser ces connaissances en utilisant les mêmes types de menus dans votre application Web. Les types de menu courants sont les suivants:

|  |  |
| --- | --- |
| • | Menus principaux. Souvent placés en haut de la page ou immédiatement sous la marque de l'application, les menus du haut renvoient aux principales zones de votre application Web. Dans certaines applications, chaque élément du menu supérieur comporte un sous-menu avec des liens vers des éléments et des pages de niveau inférieur. |
| • | Vues arborescentes. Une arborescence est un menu hiérarchique qui peut afficher de nombreux éléments à plusieurs niveaux dans votre application Web. Par défaut, une arborescence n'affiche souvent qu'un ou deux niveaux de l'architecture des informations. Les utilisateurs peuvent étendre les niveaux inférieurs pour trouver le contenu dont ils ont besoin. Les arborescences permettent aux utilisateurs d'explorer plusieurs niveaux sur une seule page. Les arborescences peuvent également montrer clairement la position de la page actuelle, dans la hiérarchie. |
| • | Sentiers d'Ariane. Souvent présenté comme une séquence horizontale de liens en haut d'une page Web, un fil d'Ariane montre la page actuelle et ses parents dans tous les niveaux supérieurs de la hiérarchie. Les fil d'Ariane indiquent la position actuelle de l'utilisateur, dans le contexte de l'application Web, et permettent aux utilisateurs de naviguer vers des pages plus élevées et moins spécifiques. |
| • | Menus de pied de page. Le pied de page n'est souvent pas visible lorsqu'un utilisateur arrive sur une page. L'utilisateur doit faire défiler jusqu'à la fin de la page pour voir le pied de page. Le pied de page est souvent utilisé comme emplacement pour des liens importants, mais pas critiques pour l'entreprise. Par exemple, des liens vers des droits d'auteur et d'autres informations juridiques, des liens vers des sociétés mères et des liens vers des plans de site sont souvent placés dans le menu du pied de page. |

**Le fournisseur de plan de site MVC**

Dans ASP.NET, une façon de créer rapidement une architecture d'informations et des contrôles de navigation pour présenter cette architecture consiste à utiliser un fournisseur de plan de site. Un fournisseur de plan de site est un composant ASP.NET qui stocke la hiérarchie logique de votre application Web. Les commandes de navigation telles que les menus supérieurs, les vues arborescentes et les chemins de navigation peuvent prendre leurs liens à partir de cette hiérarchie unique.

Le fournisseur de plan de site MVC est un fournisseur conçu pour fonctionner avec les contrôleurs et les actions MVC. Il présente l'architecture des informations que vous pouvez configurer dans un simple fichier XML. Cette hiérarchie est totalement indépendante du modèle, des contrôleurs et des actions de vos applications, bien que vous puissiez créer des liens vers ces derniers. Le fournisseur comprend également des assistants HTML, que vous pouvez utiliser dans les vues pour rendre les menus, les vues arborescentes et les pistes de navigation.

**Noter:**Le fournisseur de plan de site MVC est un composant de Maarten Balliauw qui n'est pas inclus par défaut dans les applications Web MVC. Vous pouvez le trouver dans le gestionnaire de packages NuGet en recherchant «MvcSiteMapProvider.MVC5».

**Question:**L'analyse des journaux Web a montré que les visiteurs de votre application Web peuvent accéder rapidement et facilement aux pages de bas niveau de votre architecture d'informations. Cependant, ils trouvent par la suite d'autres pages en retournant à la page d'accueil et en naviguant à nouveau dans toute la hiérarchie à partir du haut, ou bien ils utilisent l'outil de recherche. Comment pouvez-vous permettre aux utilisateurs de naviguer vers des niveaux supérieurs sans recommencer à partir de la page d'accueil?

## **Configuration du fournisseur de plan de site MVC**

Après avoir installé le fournisseur de plan de site MVC à partir du gestionnaire de packages NuGet, vous devez configurer le fournisseur et configurer la hiérarchie logique qu'il doit présenter, avant de pouvoir afficher des menus.

**Configuration du fournisseur de plan de site MVC dans Web.Config**

Vous devez configurer le fournisseur de plan de site MVC en ajoutant des éléments au fichier Web.config dans le dossier racine de votre application Web MVC. Lorsque vous installez le fournisseur de plan de site, le package NuGet ajoute un élément <siteMap> à la section <system.web> de ce fichier.

**L'élément <siteMap> dans Web.config**

<siteMap defaultProvider = "MvcSiteMapProvider" enabled = "true">

<fournisseurs>

<clear />

<add name = "MvcSiteMapProvider"

type = "MvcSiteMapProvider.DefaultSiteMapProvider, MvcSiteMapProvider"

siteMapFile = "~ / Mvc.Sitemap"

securityTrimmingEnabled = "faux"

cacheDuration = "5"

enableLocalization = "false"

scanAssembliesForSiteMapNodes = "false"

includeAssembliesForScan = ""

excludeAssembliesForScan = ""

attributesToIgnore = ""

nodeKeyGenerator = "MvcSiteMapProvider.DefaultNodeKeyGenerator, MvcSiteMapProvider"

controllerTypeResolver = "MvcSiteMapProvider.DefaultControllerTypeResolver, MvcSiteMapProvider"

actionMethodParameterResolver = "MvcSiteMapProvider.DefaultActionMethodParameterResolver,

MvcSiteMapProvider "

aclModule = "MvcSiteMapProvider.DefaultAclModule, MvcSiteMapProvider"

siteMapNodeUrlResolver = "MvcSiteMapProvider.DefaultSiteMapNodeUrlResolver,

MvcSiteMapProvider "

siteMapNodeVisibilityProvider = "MvcSiteMapProvider.DefaultSiteMapNodeVisibilityProvider,

MvcSiteMapProvider "

siteMapProviderEventHandler = "MvcSiteMapProvider.DefaultSiteMapProviderEventHandler,

MvcSiteMapProvider "

/>

</providers>

</siteMap>

Notez que vous pouvez configurer le nom du fichier XML dans lequel le plan du site est stocké, activer le découpage de sécurité et configurer la mise en cache.

**Lecture supplémentaire:**Pour obtenir des détails complets sur les valeurs de configuration que vous pouvez utiliser avec le fournisseur de plan de site MVC, ainsi que d'autres documents, accédez à<http://go.microsoft.com/fwlink/?LinkID=288963&clcid=0x409>

**Création du fichier de plan du site**

Le fournisseur de plan de site MVC est le seul fournisseur de plan de site qui vous permet de spécifier un contrôleur et une action pour chaque élément du plan de site. Pour ce faire, remplissez un fichier de plan de site. Par défaut, le fichier de plan de site est nommé Mvc.sitemap et est stocké au niveau le plus élevé de votre application Web. La hiérarchie se compose d'éléments <mvcSiteMapNode> imbriqués. Il s'agit de la hiérarchie que les assistants afficheront dans les contrôles de navigation.

**Un fichier de plan de site MVC**

<? xml version = "1.0" encoding = "utf-8"?>

<mvcSiteMap xmlns: xsi = "http://www.w3.org/2001/XMLSchema-instance"

xmlns = "http://mvcsitemap.codeplex.com/schemas/MvcSiteMap-File-4.0"

xsi: schemaLocation =

"http://mvcsitemap.codeplex.com/schemas/MvcSiteMap-File-4.0 MvcSiteMapSchema.xsd">

<mvcSiteMapNode title = "Home" controller = "Home" action = "Index">

<mvcSiteMapNode title = "Products" controller = "Product" action = "Index" key = "AllProducts">

<mvcSiteMapNode title = "Bikes" controller = "Category" action = "Display">

</mvcSiteMapNode>

</mvcSiteMapNode>

<mvcSiteMapNode title = "Dernières nouvelles" controller = "Article" action = "DisplayLatest">

<mvcSiteMapNode title = "About Us" controller = "Home" action = "About" />

</mvcSiteMapNode>

</mvcSiteMap>

**Question:**Vous utilisez le fournisseur de plan de site MVC. Votre patron remarque qu'un lien incorrect apparaît dans le menu le plus élevé et dans l'arborescence de chaque page de votre application Web. Votre patron vous demande de corriger chaque page de l'application Web. Quelles mesures devez-vous prendre?

## **Ajout de commandes de menu**

Après avoir configuré le fournisseur de plan de site et créé un fichier de plan de site, vous pouvez ajouter des menus, des vues arborescentes et d'autres contrôles de navigation à vos vues. Le fournisseur de plan de site MVC inclut des aides HTML pour rendre cela facile.

**Rendu d'un contrôle de menu**

@ Html.MvcSiteMap (). Menu (faux, faux, vrai)

Dans cet exemple, le premier paramètre spécifie que le menu ne démarre pas à partir du nœud actuel. Le menu démarre à partir du nœud de niveau supérieur, qui est généralement la page d'accueil de l'application Web. Le deuxième paramètre spécifie que le nœud de départ ne s'affiche pas en tant que nœud enfant au même niveau que les nœuds de second niveau. Le troisième paramètre spécifie que le nœud de départ s'affiche dans le menu.

**Rendu d'un fil d'Ariane**

@ Html.MvcSiteMap (). SiteMapPath ()

L'assistant Html.MvcSiteMap () comprend également des fonctions que vous pouvez utiliser pour restituer les titres des nœuds et compléter les plans de site.

**Lecture supplémentaire:**Pour une documentation de référence complète qui détaille l'assistant Html.MvcSiteMap () et ses fonctions, accédez à<http://go.microsoft.com/fwlink/?LinkID=288964&clcid=0x409>

**Question**: Quels sont certains des avantages de l'utilisation de l'assistant Html.MvcSiteMap ()?

## **Démonstration: comment créer une navigation sur le site**

Dans cette démonstration, vous verrez comment créer des menus et des pistes de navigation à l'aide du fournisseur de plan de site MVC et de ses assistants HTML.

### **Étapes de démonstration**

|  |  |
| --- | --- |
| • | Vous trouverez les étapes dans la section «Leçon 3: Création d'une structure de navigation» sur la page suivante:<https://github.com/MicrosoftLearning/20486-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486C/20486C_MOD07_DEMO.md>. |

# **Laboratoire: Structuration des applications Web ASP.NET MVC 5**

### **Scénario**

Une priorité de conception importante pour l'application de partage de photos est que les visiteurs doivent être en mesure de localiser facilement et logiquement les photographies. De plus, les galeries de photos et les photos doivent apparaître en bonne place dans les résultats des moteurs de recherche. Pour mettre en œuvre ces priorités, il vous a été demandé de configurer des itinéraires qui permettent la saisie d'URL conviviales pour accéder aux photos.

On vous a demandé de vous assurer que les URL des formulaires suivants fonctionnent pour afficher une photo:

|  |  |
| --- | --- |
| • | ~ / photo / affichage / PhotoId. Dans cette forme d'URL, PhotoId est l'ID de base de données de l'objet photo. Cette forme d'URL fonctionne déjà car elle correspond à l'itinéraire par défaut. |
| • | ~ / photo / PhotoId. Dans cette forme d'URL, PhotoId est l'ID de base de données de l'objet photo. Il s'agit de l'URL logique à saisir lorsque vous connaissez l'ID de la photo à laquelle vous souhaitez accéder. |
| • | ~ / photo / title / PhotoTitle. Dans cette forme d'URL, PhotoTitle est le titre de l'objet photo. Il s'agit de l'URL logique à saisir lorsque vous connaissez le titre de la photo à laquelle vous souhaitez accéder. |

Il vous a également été demandé d'implémenter les commandes de navigation suivantes dans l'application Partage de photos:

|  |  |
| --- | --- |
| • | Un menu avec des liens vers les principales zones du site |
| • | Un contrôle du fil d'Ariane |

Ces contrôles de navigation seront ajoutés au menu après l'achèvement des zones principales du site.

### **Objectifs**

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

|  |  |
| --- | --- |
| • | Ajoutez des itinéraires au moteur de routage ASP.NET dans une application ASP.NET MVC. |
| • | Créez des contrôles de navigation dans les vues ASP.NET. |

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

Durée estimée: 35 minutes

Vous trouverez les étapes de haut niveau sur la page suivante:<https://github.com/MicrosoftLearning/20486-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486C/20486C_MOD07_LAB_MANUAL.md>.

Vous trouverez les étapes détaillées sur la page suivante:<https://github.com/MicrosoftLearning/20486-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486C/20486C_MOD07_LAK.md>.

### **Exercice 1: Utilisation du moteur de routage**

##### **Scénario**

Dans cet exercice, vous allez:

|  |  |
| --- | --- |
| • | Créez des tests unitaires pour les itinéraires que vous souhaitez créer. |
| • | Ajoutez des itinéraires à l'application qui satisfont vos tests. |
| • | Essayez les itinéraires en tapant des URL dans la barre d'adresses d'Internet Explorer. |

Cette approche est conforme aux principes du Test Driven Development (TDD).

### **Exercice 2: Facultatif - Construction de commandes de navigation**

##### **Scénario**

Dans cet exercice, vous allez:

|  |  |
| --- | --- |
| • | Ajoutez le fournisseur de plan de site MVC à votre application de partage de photos. |
| • | Utilisez le fournisseur de plan de site MVC pour créer un menu et un contrôle de fil d'Ariane. |

À ce stade de développement, la plupart des principaux domaines de l'application de partage de photos ne sont pas encore créés; par conséquent, le menu affichera uniquement la page d'accueil et la galerie Toutes les photos. Votre équipe ajoutera de nouveaux nœuds au plan du site au fur et à mesure que les zones du site seront terminées.

Effectuez cet exercice si le temps le permet.

### **Question (s) de révision**

**Vérifiez vos connaissances**

**Découverte**

**Dans l'exercice 1, lorsque vous avez exécuté les tests pour la première fois, pourquoiTest\_Default\_Route\_Controller\_Onlypasser quandTest\_Photo\_Route\_With\_PhotoIDetTest\_Photo\_Title\_Routeéchouer?**

Afficher la solution Réinitialiser

**Vérifiez vos connaissances**

**Découverte**

**Pourquoi la contrainte est-elle nécessaire dans lePhotoRouteroute?**

Afficher la solution Réinitialiser

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

Pour créer une application Web convaincante et facile à naviguer, vous devez examiner attentivement l'architecture des informations que vous présentez. Vous devez essayer de vous assurer que la hiérarchie des objets que vous présentez dans les URL et les contrôles de navigation correspond aux attentes des utilisateurs. Vous devez également vous assurer que les utilisateurs peuvent comprendre cette hiérarchie d'objets sans connaissances techniques spécialisées. En utilisant des itinéraires et des fournisseurs de plan de site, vous pouvez présenter l'architecture des informations logiques aux visiteurs de l'application.

**Meilleur entrainement:**La route par défaut est logique. Cependant, les utilisateurs doivent connaître les contrôleurs et les actions pour comprendre les URL. Vous pouvez envisager de créer des itinéraires personnalisés qui peuvent être compris avec des informations dont les utilisateurs disposent déjà.

**Meilleur entrainement:**Vous pouvez utiliser des pistes de navigation et des commandes de navigation dans l'arborescence pour présenter l'emplacement actuel d'un utilisateur, dans le contexte de la hiérarchie logique de l'application Web.

**Meilleur entrainement:**Vous pouvez utiliser des tests unitaires pour vérifier les itinéraires dans la table de routage, de la même manière que vous utilisez les tests pour vérifier les contrôleurs, les actions et les classes de modèle. Il est facile pour une petite erreur de changer complètement la façon dont les URL sont gérées dans votre application Web. En effet, les nouvelles routes, si elles sont mal codées, peuvent remplacer d'autres routes. Les tests unitaires mettent en évidence ces bogues dès qu'ils surviennent.

**Problèmes courants et conseils de dépannage**

| **Problème commun** | **Astuce de dépannage** |
| --- | --- |
| Un itinéraire ne prend jamais effet. | Veuillez consulter le contenu du compagnon étudiant pour ce cours. |

### **Question (s) de révision**