Module 12: Création d'une application Web résiliente 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/20486C12.html#P1) |
| **Leçon 1:** | [Développement de sites sécurisés](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C12.html#P2) |
| **Leçon 2:** | [Gestion de l'état](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C12.html#P3) |
| **Laboratoire:** | [Création d'une application Web ASP.NET MVC 5 résiliente](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C12.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/20486C12.html#P5) |

# **Aperçu du module**

La sécurité est toujours la priorité absolue des applications Web, car les applications Web accessibles au public sont généralement la cible de différents types d'attaques Web. Les pirates utilisent des attaques Web pour accéder à des informations sensibles. Pour éviter ces attaques, vous devez savoir comment utiliser AntiXSS et demander la validation de votre application Web. À l'aide de divers mécanismes de sécurité, vous pouvez créer une application Web sécurisée. Un autre problème auquel les développeurs sont généralement confrontés est la nécessité de conserver les informations parmi plusieurs publications. Vous devez savoir comment utiliser les techniques de gestion d'état pour conserver les informations pour plusieurs requêtes HTTP. Grâce à la gestion des états, vous pouvez réduire la nécessité pour les utilisateurs de ressaisir des informations chaque fois qu'ils ont besoin de faire une demande.

### **Objectifs**

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

|  |  |
| --- | --- |
| • | Développez des sites sécurisés. |
| • | Mettre en œuvre la gestion de l'état. |

# Leçon 1: Développement de sites sécurisés

Les applications Web font souvent l'objet d'attaques de sécurité. Ces attaques empêchent les applications de fonctionner correctement et les attaquants tentent d'accéder aux informations sensibles stockées dans le magasin de données sous-jacent. ASP.NET fournit des mécanismes de protection intégrés qui aident à prévenir de telles attaques. Cependant, ces mécanismes ont également tendance à affecter le fonctionnement de vos applications. Vous devez savoir quand activer ou désactiver les mécanismes de protection pour éviter tout impact sur les fonctionnalités de votre application. Vous devez également savoir comment utiliser Secure Sockets Layer (SSL) pour empêcher tout accès non autorisé aux informations lors de la transmission des informations.

## **Objectifs de la leçon**

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

|  |  |
| --- | --- |
| • | Décrivez comment éviter les attaques malveillantes qui utilisent des scripts intersites. |
| • | Décrivez les techniques d'attaque courantes utilisées par les utilisateurs malveillants. |
| • | Décrivez comment désactiver la protection contre les attaques dans votre application MVC. |
| • | Décidez si vous souhaitez désactiver la protection contre les attaques dans divers scénarios. |
| • | Décrivez le rôle de Secure Sockets Layer dans la sécurisation des applications Web. |

## **Script intersite**

Les scripts intersites impliquent l'insertion malveillante de scripts dans une session utilisateur. Les scripts intersites présentent des informations à d'autres sites Web, en utilisant les informations d'authentification des utilisateurs, à leur insu. Par exemple, considérons le code inséré dans une application Web pour publier de manière malveillante des messages sur des sites de réseaux sociaux, à l'insu des utilisateurs. Lorsqu'un script est inséré dans une application Web, le script a un accès complet au modèle d'objet de document (DOM) du HTML. Cet accès permet à l'attaquant de créer de fausses boîtes de saisie dans l'application, de créer de faux utilisateurs et de publier de fausses informations sur l'application Web.

L'attaque de script intersite prend généralement une entrée à partir d'une sortie échappée de manière incorrecte. Ces attaques de script ont généralement un impact sur les chaînes de requête. Par exemple, considérez l'URL suivante:

http: // localhost / Default1 /? msg = Bonjour

**Accès à un paramètre de chaîne de requête**

Index ActionResult public (chaîne msg)

{

ViewBag.Msg = msg;

retour Vue ();

}

**Affichage du paramètre de chaîne de requête**

<div class = "messages"> @ ViewBag.Msg <div>

Après avoir exécuté les exemples de code précédents, l'application doit afficher le mot résultant, Hello. À présent, envisagez un scénario dans lequel le paramètre de chaîne de requête msg est remplacé par une valeur moins bénigne, ce qui donne l'URL suivante:

http: // localhost / Default1 /? msg = <script> alert ('pwnd') </ script>

Par conséquent, le bloc de script inclus dans la chaîne de requête est affiché aux utilisateurs. Dans de tels cas, les attaquants peuvent injecter du code malveillant dans votre application en utilisant la valeur d'un paramètre de chaîne de requête.

ASP.NET inclut la validation des demandes, pour aider à protéger les valeurs d'entrée qui sont sujettes à des attaques de script intersite. Cependant, les attaquants peuvent contourner ce mécanisme en utilisant le codage pour contourner les filtres de script intersite communs. Par exemple, voici la même chaîne de requête, cette fois codée:

http: // localhost / Default1 /? msg = Jon \ x3cscript \ x3e% 20alert (\ x27pwnd \ x27)% 20 \ x3c / script \ x3e

**Utilisation de la fonction JavaScriptStringEncode**

<div class = "messages"> @ Ajax.JavaScriptStringEncode (ViewBag.Msg) <div>

Vous pouvez également importer la bibliothèque AntiXSS pour vérifier le contenu de la chaîne de requête pour d'éventuelles attaques. La bibliothèque AntiXSS fait partie de la bibliothèque de protection Web, qui a été développée par Microsoft pour détecter des attaques Web plus complexes que celles que la validation de demande d'ASP.NET peut détecter.

**Lecture supplémentaire:**Pour afficher plus d'informations sur la bibliothèque AntiXSS, accédez à<http://go.microsoft.com/fwlink/?LinkID=293690&clcid=0x409>

Après avoir importé la bibliothèque AntiXSS dans votre application MVC, vous pouvez utiliser la bibliothèque pour encoder tout contenu de sortie en HTML.

**Utilisation de la bibliothèque AntiXSS**

@utilisation de Microsoft.Security.Application

<div class = "messages"> @ Encoder.JavaScriptEncode (ViewBag.Msg) <div>

Le code de l'exemple précédent illustre comment coder des valeurs d'entrée à l'aide de la méthode JavaScriptEncode de la bibliothèque AntiXSS, lors de l'affichage de la sortie au format HTML. Cette pratique garantit que les valeurs d'entrée peuvent être affichées en toute sécurité.

**Question**: Quelles sont les causes des attaques de script intersite?

## **Autres techniques d'attaque**

En plus des attaques de scripts intersites, les pirates peuvent utiliser d'autres types d'attaques, y compris la falsification de requêtes intersites et les attaques par injection SQL pour subvertir les applications Web.

**Falsification de requêtes intersites**

La falsification de requête intersite (CSRF) est une attaque qui se produit lorsque vous ouvrez une URL dans un navigateur Web, en utilisant votre contexte utilisateur, sans savoir que vous autorisez les attaquants à apporter des modifications à votre système. Par exemple, considérez que votre application utilise des chaînes de requête pour transmettre des informations à d'autres applications. Vous recevez un e-mail avec un lien tel que le suivant:

<a href="http://localhost/Default1/?id=100"> Cliquez sur moi </a>

Lorsque vous cliquez sur le lien, l'action associée à la vue s'exécute sur votre navigateur Web. Parce que vous êtes un utilisateur authentifié dans l'application, l'attaquant peut désormais accéder à votre système.

Vous pouvez empêcher CSRF en utilisant les règles suivantes:

|  |  |
| --- | --- |
| 1. | Assurez-vous qu'une requête GET n'est pas rejouée en cliquant sur un lien. Les spécifications HTTP pour les requêtes GET impliquent que les requêtes GET ne doivent être utilisées que pour la récupération et non pour les modifications d'état. |
| 2. | Assurez-vous qu'une requête n'est pas rejouée si un attaquant utilise JavaScript pour simuler une requête POST de formulaire. |
| 3. | Empêchez toute modification de données utilisant la requête GET. Ces modifications devraient nécessiter une certaine interaction de l'utilisateur. Cette pratique consistant à introduire une interaction utilisateur n'aide pas à empêcher les attaques basées sur les formulaires. Cependant, l'interaction de l'utilisateur limite plusieurs types d'attaques plus simples, telles que les liens malveillants intégrés dans des sites compromis par XSS. |

La fonction @ Html.AntiForgeryToken () permet de protéger votre système contre CSRF en utilisant des jetons uniques qui sont transmis à l'application avec les requêtes. La fonction @ Html.AntiForgeryToken () utilise non seulement un champ de formulaire masqué mais également une valeur de cookie, ce qui rend plus difficile la falsification d'une requête.

**Utilisation d'un jeton anti-falsification**

@utilisation (Html.BeginForm ())

{

@ Html.AntiForgeryToken ();

@ Html.EditorForModel ();

<input type = "submit" value = "Submit" />

}

**Forcer la vérification des jetons anti-falsification**

[HttpPost]

[ValidateAntiForgeryToken ()]

public ActionResult Index (utilisateur utilisateur)

{

Vue de retour ();

}

Notez l'attribut ValidateAntiForgeryToken () dans l'exemple de code précédent. Cet attribut permet au contrôleur de vérifier si l'entrée utilisateur à partir du formulaire HTML inclut le jeton généré par @ Html.AntiForgeryToken (), avant d'accepter une requête.

**Attaque par injection SQL**

Les attaques par injection SQL sont similaires aux attaques de scripts intersites. Cependant, la différence est que l'entrée utilisateur est utilisée pour créer du SQL dynamique au lieu du HTML. Observez la ligne de code suivante.

cmd.CommandText = "sélectionnez userID dans Users où userID = '" + Request.QueryString ["user"] + "'";

Considérez un scénario dans lequel un attaquant modifie la chaîne de requête comme affiché dans la ligne de code suivante.

user = 1 'AND 1 = 1 GO SELECT \* from Users WHERE NOT 0 ='

La ligne de code suivante indique le SQL résultant.

sélectionnez userID dans Users où userID = '1' AND 1 = 1 GO SELECT \* from Users WHERE NOT 0 = ''

Dans ce cas, SQL renvoie tout de la table Users.

Pour éviter les attaques par injection SQL, vous devez:

|  |  |
| --- | --- |
| 1. | Validez l'entrée utilisateur. |
| 2. | Évitez d'utiliser des concaténations de chaînes pour créer du SQL dynamique. |
| 3. | Utilisez des commandes paramétrées avec SQL dynamique. |
| 4. | Stockez toutes les informations sensibles et confidentielles dans des formats cryptés. |
| 5. | Assurez-vous que l'application n'utilise pas ou n'accède pas à la base de données avec des privilèges d'administrateur. |

**Question**: Quelle est la meilleure façon d'empêcher les attaques par injection SQL si vous ne pouvez pas éviter le SQL dynamique?

## **Désactivation de la protection contre les attaques**

La validation des demandes est une fonctionnalité ASP.NET qui examine une demande HTTP et détermine si elle contient un contenu potentiellement dangereux. Le contenu potentiellement dangereux peut inclure n'importe quel balisage HTML ou code JavaScript dans le corps, l'en-tête, la chaîne de requête ou les cookies de la demande.

Cependant, la validation de la demande peut également entraîner un dysfonctionnement de l'application en empêchant certaines entrées d'entrer dans la page de l'application pour traitement. Considérez une situation dans laquelle votre application utilise un éditeur HTML pour générer du code HTML pour l'entrée utilisateur, avant d'enregistrer l'entrée dans la base de données. Dans ce cas, vous souhaiterez peut-être désactiver la validation de la demande pour permettre à l'éditeur HTML de fonctionner correctement.

**Utilisation de l'attribut ValidateInput**

[HttpPost]

[ValidateInput (false)]

public ActionResult Edit (commentaire de chaîne)

{

return View (commentaire);

}

**Utilisation de l'attribut AllowHtml**

[AllowHtml]

chaîne publique Prop1 {get; ensemble; }

Vous devriez envisager d'utiliser des techniques de protection contre les attaques qui:

|  |  |
| --- | --- |
| • | Avoir un impact minimum sur l'application. |
| • | Impliquez des champs minimums pour accepter les éléments HTML dans les validations de demande. |

**Question**: Décrivez un scénario dans lequel vous souhaitez désactiver la validation des demandes?

## **Couche de sockets sécurisés**

Secure Sockets Layer (SSL) est un protocole de couche application qui permet:

|  |  |
| --- | --- |
| • | Chiffrez le contenu à l'aide des clés d'infrastructure à clé publique (PKI). |
| • | Protégez le contenu transmis entre le serveur et le client. |
| • | Empêchez l'accès non autorisé au contenu pendant la transmission. |
| • | Rassurez les utilisateurs qu'un site est authentique et certifié. |

Vous pouvez utiliser SSL sur les vues qui acceptent l'entrée de l'utilisateur si l'entrée comprend des informations sensibles, telles que des informations de carte de crédit et des mots de passe. L'utilisation de SSL sur des vues aussi cruciales garantit la confidentialité du contenu et l'authenticité de l'expéditeur. Cependant, vous ne pourrez peut-être pas analyser votre code et détecter facilement si un utilisateur accède à l'application Web à l'aide de SSL.

ASP.NET MVC 5 inclut l'attribut RequireHttps qui vous permet d'utiliser SSL sur les vues qui impliquent des informations sensibles. L'attribut RequireHttps redirige les utilisateurs vers le lien SSL, s'ils demandent une vue à l'aide du protocole HTTP normal.

**Utilisation de l'attribut RequireHttps**

[RequireHttps]

public class Controller1

{

[RequireHttps]

public ActionResult Edit ()

{

}

}

Vous pouvez utiliser l'attribut RequireHttps au niveau du contrôleur ou au niveau de l'action. Cette flexibilité vous permet de choisir SSL lorsque cela est nécessaire dans votre application Web.

Notez que les serveurs Web exigent que vous configuriez le certificat PKI afin que le serveur accepte les connexions SSL. Les certificats SSL doivent être achetés auprès d'une autorité de certification (CA). Pendant le développement de l'application, vous pouvez utiliser le certificat d'auto-signature pour simplifier le processus de configuration.

**Question**: Quelle action doit être effectuée sur le serveur Web, avant d'implémenter SSL?

# Leçon 2: Gestion des états

Lors du développement d'applications, vous souhaiterez peut-être créer des fonctions qui nécessitent que les informations soient conservées entre les demandes. Par exemple, considérons une application dans laquelle vous devez d'abord sélectionner un client, puis travailler sur la commande pertinente pour le client. HTTP est un protocole sans état. Par conséquent, ASP.NET inclut différentes techniques de gestion d'état pour aider à stocker des informations pour plusieurs requêtes HTTP. Vous devez savoir comment configurer et mettre à l'échelle les mécanismes de stockage d'état pour prendre en charge les batteries de serveurs Web.

## **Objectifs de la leçon**

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

|  |  |
| --- | --- |
| • | Décrivez les avantages du stockage des informations d'état. |
| • | Répertoriez les options de stockage des informations d'état. |
| • | Configurez le stockage d'état. |
| • | Décrivez l'évolutivité des mécanismes de stockage d'état. |
| • | Stockez et récupérez les informations d'état dans une application Web MVC 5. |

## **Pourquoi stocker des informations d'état?**

HTTP est un protocole sans état qui ne lie pas les demandes soumises les unes aux autres. HTTP traite chaque requête comme une nouvelle requête. Cependant, cette indépendance entre les demandes affecte les applications métier qui ont des dépendances d'informations entre les fonctions. Par exemple, considérez que vous ne pouvez accéder à la fonction Mettre à jour l'adresse qu'après avoir fourni les informations de connexion de l'utilisateur. La page qui gère la fonction d'ouverture de session utilisateur est différente de la page qui gère la fonction de mise à jour d'adresse. Par conséquent, ces fonctions nécessitent deux requêtes HTTP distinctes. Avec l'implémentation HTTP par défaut, le serveur peut perdre le résultat de la demande d'ouverture de session lorsqu'un utilisateur accède à la page Mettre à jour l'adresse. ASP.NET implémente la technologie de gestion de session pour conserver les informations dépendantes pour plusieurs requêtes HTTP.

La gestion de session permet aux applications Web de stocker des valeurs pour plusieurs requêtes HTTP, que la requête soit dirigée vers la même page ou vers plusieurs pages. Le processus de gestion de session implique deux techniques - techniques côté serveur et techniques côté client, pour conserver les informations utilisateur pertinentes pour plusieurs requêtes HTTP.

**Gestion de session côté client**

Les technologies de gestion de session côté client comprennent:

|  |  |
| --- | --- |
| • | Champs cachés |
| • | Biscuits |
| • | Chaînes de requête |

Les technologies de gestion de session côté client permettent de conserver les informations dépendantes lorsqu'un utilisateur déclenche l'action HTTP POST en cliquant sur un bouton sur une page Web. Ensuite, lorsqu'une page est rendue, le serveur génère les informations dépendantes.

**Noter:**Si vous êtes familiarisé avec les formulaires Web ASP.NET, vous avez peut-être utilisé l'état d'affichage et l'état du contrôle comme stockage d'état côté client. Ces techniques ne sont pas disponibles dans MVC car elles reposent sur des contrôles serveur Web Forms.

**Les serveurs**Gestion de session ide

Les options côté serveur offrent une sécurité plus élevée pour le stockage des informations de page que les options côté client, car les informations conservées à l'aide des options côté client ne sont pas visibles pour le système client. Cependant, les options côté serveur nécessitent plus de ressources de serveur Web telles que la mémoire pour stocker les données. Cette utilisation supplémentaire des ressources peut entraîner des problèmes d'évolutivité si la taille des informations à stocker est importante.

ASP.NET fournit les options de gestion de session côté serveur suivantes:

|  |  |
| --- | --- |
| • | État de l'application |
| • | État de la session |
| • | TempData |
| • | Propriétés du profil |
| • | Prise en charge de la base de données |

**Lecture supplémentaire:**Pour plus d'informations sur les recommandations de gestion de l'état ASP.NET, accédez à<http://go.microsoft.com/fwlink/?LinkID=288986&clcid=0x415>

**Question**: Pourquoi avez-vous besoin de mettre en œuvre la gestion de session?

## **Options de stockage d'état**

Vous pouvez stocker des valeurs pour les demandes et les demandes suivantes liées aux demandes précédentes, à l'aide de l'objet TempData. Par défaut, l'objet TempData utilise la variable Session pour stocker les informations relatives aux demandes. Vous pouvez utiliser l'objet TempData pour accéder aux informations de session, au lieu d'utiliser la variable Session. L'objet TempData inclut une logique pour gérer la redirection des informations vers différentes vues.

Vous pouvez utiliser l'élément sessionState dans le fichier Web.config pour définir le référentiel qui doit conserver les informations sur l'état de la session. Vous pouvez également utiliser l'élément sessionState pour désactiver la prise en charge de l'état de session. La liste suivante décrit certaines des options de stockage d'état de session les plus couramment utilisées:

|  |  |
| --- | --- |
| • | Le mode InProc. Stocke la session dans le processus ASP.NET. |
| • | Le mode StateServer. Stocke les informations de session dans des services dédiés s'exécutant sur le serveur Web ou sur un serveur dédié. Il ne stocke pas d'informations sur les services Internet (IIS). |
| • | Le mode SQLServer. Stocke les informations de session dans une base de données SQL Server. |
| • | Le mode Off. Désactive la prise en charge de l'état de session dans votre application. |

**Question**: Considérez que vous disposez d'un serveur Web contenant plusieurs serveurs Web. Vous souhaitez autoriser les utilisateurs à accéder aux mêmes informations d'état pour tous les serveurs Web auxquels ils accèdent. Dans ce cas, quel mode de stockage d'état devez-vous utiliser?

## **Configuration du stockage d'état**

Vous pouvez configurer le stockage d'état pour utiliser les modes StateServer et SQLServer.

**Le mode StateServer**

Vous devez exécuter le service d'état ASP.NET sur le serveur utilisé pour stocker les informations de session, avant d'utiliser le mode StateServer. Le service d'état ASP.NET est installé en tant que service lorsque ASP.NET et le .NET Framework sont installés dans le cadre de Microsoft .NET Framework. Le service ASP.NET est installé à l'emplacement suivant: systemroot \ Microsoft.NET \ Framework \ versionNumber \ aspnet\_state.exe

Pour configurer une application ASP.NET pour utiliser le mode StateServer, vous devez effectuer les étapes suivantes: To configure an ASP.NET application for using the StateServer mode, you should perform the following steps:

|  |  |
| --- | --- |
| • | Dans le fichier Web.config de l'application, définissez l'attribut mode de l'élément sessionState sur StateServer. |
| • | Dans le fichier Web.config de l'application, définissez l'attribut stateConnectionString sur tcpip = <serverName>: 42424. |

Pour configurer la prise en charge de StateServer dans votre application, vous devez effectuer les étapes suivantes:

|  |  |
| --- | --- |
| 1. | Exécutez le fichier services.msc. |
| 2. | Définissez l'attribut Type de démarrage du service d'état ASP.NET sur Automatique et cliquez avec le bouton droit sur le service. |
| 3. | Ouvrez le fichier Web.config, identifiez l'élément sessionState, puis mettez à jour le code. |

**Configuration du mode serveur d'état**

<configuration>

<system.web>

<sessionState mode = "StateServer"

stateConnectionString = "tcpip = SampleStateServer: 42424"

cookieless = "faux"

timeout = "20" />

</system.web>

</configuration>

**Noter:**Les objets stockés dans l'état de session doivent être sérialisables, si le mode est défini sur StateServer ou SQLServer.

**Le mode SQLServer**

Avant d'utiliser le mode SQLServer, vous devez installer la base de données d'état de session ASP.NET sur SQL Server. Vous pouvez installer la base de données d'état de session ASP.NET à l'aide de l'outil aspnet\_regsql.exe. Pour configurer une application ASP.NET pour utiliser le mode SQLServer, vous devez effectuer les étapes suivantes:

|  |  |
| --- | --- |
| • | Dans le fichier Web.config de l'application, définissez l'attribut mode de l'élément sessionState sur SQLServer. |
| • | Dans le fichier Web.config de l'application, définissez l'attribut sqlConnectionString sur une chaîne de connexion pour votre base de données SQL Server. |

Pour créer la base de données d'état de session sur SQL Server, vous devez effectuer les étapes suivantes:

|  |  |
| --- | --- |
| 1. | Ouvrez l'invite de commande avec des privilèges administratifs. |
| 2. | À partir du dossier <systemroot \ Microsoft.NET \ Framework \ versionNumber> sur votre serveur Web, exécutez l'outil Aspnet\_regsql.exe. |
| 3. | Ouvrez le fichier Web.config, identifiez l'élément sessionState, puis mettez à jour le code. |

**Configuration du stockage de l'état de session SQL Server**

<configuration>

<system.web>

<sessionState mode = "SQLServer"

sqlConnectionString = "Sécurité intégrée = SSPI; données

source = SampleSqlServer; "/>

</system.web>

</configuration>

Vous pouvez personnaliser le comportement de l'outil aspnet\_regsql.exe à l'aide de commutateurs tels que -E, -S et -d. Vous pouvez utiliser ces commutateurs pour modifier le nom de la base de données ASPState par défaut.

**Lecture supplémentaire:**Pour plus d'informations sur l'outil aspnet\_regsql.exe, accédez à:<http://go.microsoft.com/fwlink/?LinkID=293693&clcid=0x409>

**Question**: Quel outil devez-vous installer avant d'utiliser le mode SQLServer?

## **Mise à l'échelle des mécanismes de stockage d'état**

Les modes StateServer et SQLServer permettent d'accéder au stockage d'état sur un serveur Web. Ces modes permettent à plusieurs serveurs d'accéder aux informations de stockage d'état. Ce processus répartit la charge de travail Web entre plusieurs serveurs Web.

Vous pouvez implémenter le partitionnement de l'état de session pour permettre à plusieurs serveurs d'état ou à plusieurs bases de données SQL Server de gérer les informations d'état de session. Vous pouvez appliquer la logique de partitionnement en implémentant l'interface IPartitionResolver dans votre application. Vous devez configurer le fichier Web.config pour contrôler le moteur de gestion de session. La configuration du moteur de gestion de session vous aide à utiliser la logique pour identifier le serveur Web approprié sur lequel vous pouvez stocker des informations.

**Utilisation de l'interface IPartitionResolver**

<configuration>

<system.web>

<sessionState

mode = "SQLServer"

partitionResolverType =

"PartitionResolverClass" />

</system.web>

</configuration>

Vous ne pouvez utiliser l'interface IPartitionResolver que lorsque l'état de session utilise les modes SQLServer ou StateServer. Dans l'exemple de code précédent, notez qu'aucune chaîne de connexion n'est spécifiée à l'aide des attributs sqlConnectionString ou stateConnectionString.

**Question**: Quand devez-vous implémenter l'interface IPartitionResolver?

## **Démonstration: comment stocker et récupérer des informations d'état**

Dans cette démonstration, vous verrez comment stocker et récupérer les préférences utilisateur à partir de l'état de session dans les actions du contrôleur.

### **Étapes de démonstration**

|  |  |
| --- | --- |
| • | Vous trouverez les étapes dans la section «Leçon 2: Gestion des états» sur la page suivante:<https://github.com/MicrosoftLearning/20486-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486C/20486C_MOD12_DEMO.md>. |

# **Atelier: Création d'une application Web résiliente ASP.NET MVC 5**

### **Scénario**

Le développeur principal vous a demandé d'implémenter la fonctionnalité suivante dans votre application Web de partage de photos.

|  |  |
| --- | --- |
| • | Tout visiteur de l'application, y compris les utilisateurs anonymes, doit pouvoir marquer une photo comme favori. |
| • | Si un utilisateur a marqué un favori, un lien doit être disponible pour afficher la photo préférée. |
| • | Les photos favorites doivent être affichées dans la vue diaporama. |

### **Objectifs**

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

|  |  |
| --- | --- |
| • | Stockez un paramètre pour un utilisateur anonyme ou authentifié dans l'état de session. |
| • | Vérifiez une préférence utilisateur lors du rendu d'un lien d'action. |
| • | Rendez une page Web en vérifiant les valeurs d'état dans l'application. |

##### **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_MOD12_LAB_MANUAL.md>.

Vous trouverez les étapes détaillées sur la page suivante:<https://github.com/MicrosoftLearning/20486-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486C/20486C_MOD12_LAK.md>.

### **Exercice 1: Création d'actions de contrôleur de favoris**

##### **Scénario**

Il vous a été demandé de créer une fonctionnalité qui stocke les photos préférées des visiteurs dans l'état de session de l'application Web. Une fois que les utilisateurs ont ajouté des photos à leurs favoris, ils pourront afficher un diaporama de toutes les photos qu'ils ont sélectionnées comme favoris.

Dans cet exercice, vous allez:

|  |  |
| --- | --- |
| • | Créez l'action Diaporama Favoris. |
| • | Créez l'action Ajouter un favori. |

### **Exercice 2: Implémentation de favoris dans les vues**

##### **Scénario**

Vous avez créé les actions de contrôleur nécessaires pour implémenter les photos préférées. Maintenant, vous devez implémenter les composants de l'interface utilisateur pour afficher un contrôle pour ajouter un favori. Si un utilisateur a des favoris, vous devez afficher un lien vers l'action FavoritesSlideShow.

Dans cet exercice, vous allez:

|  |  |
| --- | --- |
| • | Ajoutez un lien d'action AJAX dans la vue Affichage des photos. |
| • | Ajoutez un lien et mettez à jour le plan du site. |

### **Question (s) de révision**

**Vérifiez vos connaissances**

**Découverte**

**Dans cet atelier, vous avez stocké la liste des photos favorites dans l'état de session. Lors du test, votre responsable remarque que les utilisateurs authentifiés perdent leur liste de photos préférées à chaque fois qu'ils ferment leur navigateur. Où stockeriez-vous une liste de favoris pour chaque utilisateur authentifié afin que la liste soit préservée chaque fois qu'un utilisateur se connecte à l'application Web?**

Afficher la solution Réinitialiser

**Vérifiez vos connaissances**

**Découverte**

**Comment créeriez-vous une vue des photos préférées avec la présentation de type carte que les utilisateurs voient sur leToutes les photospage?**

Afficher la solution Réinitialiser

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

Les applications Web sont généralement soumises à différents types d'attaques. Ces attaques permettent aux attaquants d'accéder aux informations sensibles stockées dans la base de données et d'effectuer des actions malveillantes. Vous pouvez utiliser AntiXSS et demander une validation, pour protéger les applications contre les attaques Web. Vous pouvez également utiliser des techniques de gestion d'état pour stocker des informations pour plusieurs requêtes HTTP afin d'éviter que les utilisateurs ne saisissent les mêmes informations plusieurs fois.

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

Lors de la mise en œuvre d'applications Web, vous souhaiterez peut-être utiliser un éditeur d'entrée au format riche pour permettre aux utilisateurs de mettre en forme l'entrée dans des zones de texte. Par conséquent, vous devrez peut-être désactiver la validation des demandes pour permettre à ASP.NET de capturer et de traiter les entrées utilisateur.

Les fonctions métier complexes impliquent généralement plusieurs vues. Ces fonctions peuvent poser des problèmes car les informations doivent être partagées entre plusieurs vues. La gestion de l'état de session permet de résoudre ces problèmes, car elle permet de conserver les informations pertinentes pour plusieurs vues.

### **Question (s) de révision**