

 eBook Gratuit

APPRENEZ sharepoint

eBook gratuit non affilié créé à partir des
contributeurs de Stack Overflow.

#sharepoint

Table des matières

À propos.....	1
Chapitre 1: Démarrer avec sharepoint.....	2
Remarques.....	2
Versions.....	2
Exemples.....	3
Installation de SharePoint 2016 pour une batterie de serveurs unique.....	3
introduction.....	3
Exigences.....	3
Installation.....	3
Configuration.....	4
Configuration de la ferme.....	4
Construire un composant WebPart avec SharePoint Framework.....	5
Journaux et journalisation ULS SharePoint.....	6
Outillage.....	6
Identifiant de corrélation.....	6
Ajout de SPMonitoredScope à mon code.....	6
Chapitre 2: App SharePoint.....	7
Introduction.....	7
Remarques.....	7
Exemples.....	7
SharePoint 2013: accéder aux données de service de profil utilisateur à l'aide de JSOM dan.....	7
Chapitre 3: Création d'une application hébergée par un fournisseur.....	9
Exemples.....	9
Définition de l'environnement de développement.....	9
Préparation pour le site développeur.....	10
Créer une application dans Visual studio.....	11
Commençons à coder.....	15
Création d'une page d'article complète.....	18
Chapitre 4: Rejets majeurs.....	21

Exemples.....	21
SharePoint 2016.....	21
SharePoint 2013.....	21
Chapitre 5: Rendu côté client SharePoint 2013.....	23
Introduction.....	23
Exemples.....	23
Modifier le lien hypertexte des champs / colonnes dans la vue liste à l'aide de la CSR.....	23
Masquer la colonne à partir de la vue de liste SharePoint à l'aide de CSR.....	24
Appliquer des validations sur Nouveau / Modifier un formulaire d'élément à l'aide de CSR.....	24
Modifier le nom d'affichage de la colonne dans la vue liste à l'aide de CSR.....	29
Chapitre 6: Services REST.....	31
Remarques.....	31
URL du noeud final du service REST.....	31
Envoi de demandes REST.....	31
Syntaxe XMLHttpRequest.....	31
jQuery AJAX Syntaxe.....	32
Exemples.....	32
Travailler avec des listes.....	32
Obtention d'éléments de liste avec des colonnes de recherche.....	33
Tableau de liste d'animaux.....	33
Tableau des types d'animaux.....	33
Exemple de code.....	33
Ajout de sélections à un champ de recherche à valeurs multiples.....	35
Éléments de liste de pagination renvoyés par une requête.....	36
Récupérer un identifiant d'élément nouvellement créé dans la liste SharePoint.....	37
Comment effectuer des opérations CRUD à l'aide de l'interface REST SharePoint 2010.....	38
Chapitre 7: Travailler avec des boîtes de dialogue modales avec JavaScript.....	42
Syntaxe.....	42
Paramètres.....	42
Remarques.....	43
Exemples.....	43

Effectuer une action lorsqu'une boîte de dialogue est fermée.....	43
Afficher une page existante dans une boîte de dialogue.....	43
Afficher une boîte de dialogue personnalisée.....	44
Chapitre 8: Utilisation du modèle d'objet client JavaScript (JSOM).....	45
Remarques.....	45
Exemples.....	46
Obtenir des types de contenu de bibliothèque à l'aide du nom de la bibliothèque.....	46
Supprimer un élément dans une liste.....	46
Création d'éléments ou de dossiers.....	46
Création d'éléments de liste.....	46
Création de dossiers.....	47
Obtenir les informations d'utilisateur actuelles.....	48
Obtenir un élément de liste par ID.....	48
Obtenir des éléments de liste par la requête CAML.....	49
Exemple de base.....	49
Recherche des résultats d'une requête CAML.....	49
Chapitre 9: Utilisation du modèle d'objet côté client géré (CSOM).....	51
Remarques.....	51
Exemples.....	51
Bonjour tout le monde (obtenir le titre du site).....	51
Web Récupération des propriétés d'un site Web.....	52
Web Récupérer uniquement les propriétés spécifiées d'un site Web.....	52
Web Mise à jour du titre et de la description d'un site Web.....	52
Web Création d'un site Web.....	52
Liste. Récupération de toutes les propriétés de toutes les listes d'un site Web.....	53
Liste. Récupérer uniquement les propriétés spécifiées des listes.....	53
Liste. Stockage des listes récupérées dans une collection.....	53
Liste. Récupération des champs de liste d'un site Web.....	54
Liste. Créer et mettre à jour une liste.....	54
Liste. Ajouter un champ à une liste.....	55
Liste. Supprimer une liste.....	55
Article. Récupérer des éléments d'une liste.....	55

Article. Récupération d'éléments (à l'aide de la méthode Include).....	55
Article. Récupération de champs spécifiques à partir d'un nombre spécifié d'éléments.....	56
Article. Récupération d'éléments de toutes les listes d'un site Web.....	56
Article. Récupération d'éléments à l'aide de la position de collecte d'éléments de liste.....	57
Article. Créer un élément de liste.....	58
Article. Mise à jour d'un élément de liste.....	58
Article. Supprimer un élément de la liste.....	58
Groupes. Récupération de tous les utilisateurs d'un groupe SharePoint.....	58
Groupes. Récupération des propriétés spécifiques des utilisateurs.....	59
Groupes. Récupération de tous les utilisateurs dans tous les groupes d'une collection de s.....	59
Groupes. Ajout d'un utilisateur à un groupe SharePoint.....	60
Les rôles. Créer une définition de rôle.....	60
Les rôles. Affectation d'un utilisateur à un rôle sur un site Web.....	60
Les rôles. Création d'un groupe SharePoint et ajout du groupe à un rôle.....	61
Autorisations Briser l'héritage de sécurité d'une liste.....	61
Autorisations Briser l'héritage de sécurité d'un document et ajouter un utilisateur en tan.....	61
Autorisations Briser l'héritage de sécurité d'un document et modifier les autorisations d'.....	62
Action personnalisée Ajout d'une action personnalisée pour les éléments de la liste.....	62
Action personnalisée Modification d'une action personnalisée de l'utilisateur.....	63
Action personnalisée Ajout d'une action personnalisée d'un utilisateur aux actions de site.....	63
Web part. Mise à jour du titre d'un composant WebPart.....	64
Web part. Ajout d'un composant WebPart à une page.....	64
Web part. Suppression d'un composant WebPart d'une page.....	65
Le contexte. Utilisation d'un cache d'informations d'identification pour une exécution éle.....	65
Chapitre 10: Utilisation du modèle d'objet côté serveur géré (approbation totale).....	67
Remarques.....	67
Hiérarchie Conceptuelle.....	67
Mises en garde côté serveur.....	67
Exemples.....	67
Hello World (obtenir le titre du site).....	67
Faire le tour de toute la batterie de serveurs SharePoint.....	68
Récupérer des éléments de la liste.....	68
Récupérer des éléments à l'aide de la pagination.....	68

Obtenir la liste par URL.....	69
Créer un élément de liste.....	69
Crédits.....	70

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [sharepoint](#)

It is an unofficial and free sharepoint ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official sharepoint.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec sharepoint

Remarques

SharePoint peut faire référence à un ou plusieurs produits de la famille Microsoft SharePoint.

- **SharePoint Foundation** : il s'agissait de la technologie sous-jacente pour tous les sites SharePoint et n'est plus disponible pour SharePoint 2016.
- **SharePoint Server** : il s'agit de la version locale de SharePoint. Vous pouvez déployer un ou plusieurs serveurs SharePoint. Il offre des fonctionnalités supplémentaires par rapport à SharePoint Foundation, telles que les fonctionnalités de BI, la gestion du contenu d'entreprise et bien plus encore.
- **SharePoint Online** : version Cloud de SharePoint. Le client n'a pas besoin de se soucier de l'infrastructure du serveur ou de son évolutivité.

Office 365 est une offre Microsoft distincte qui inclut le service SharePoint Online, bien que tous les plans ne prennent pas en charge toutes les fonctionnalités SharePoint.

Les liens suivants fournissent des comparaisons de fonctionnalités complètes entre les versions SharePoint disponibles:

- SharePoint 2013 sur site vs SharePoint 2016 sur site: [disponibilité des fonctionnalités sur les plans SharePoint sur site](#)
- Fonctionnalités SharePoint dans Office 365: [disponibilité des fonctionnalités dans les plans SharePoint](#)
- Fonctionnalités SharePoint dans SharePoint Online (sans Office 365): [disponibilité des fonctionnalités dans les plans autonomes SharePoint](#)
- Comparaison des fonctionnalités fusionnées entre SharePoint 2013 et SharePoint Online: <http://www.buckleyplanet.com/2014/06/sharepoint-online-vs-onprem-feature-comparison.html>

Versions

Version	Nom officiel	Date de sortie
Avant 2003	SharePoint Portal Server	2002-07-09
2003	SharePoint Portal Server 2003	2003-11-23
2007	SharePoint Server 2007	2007-01-27
2010	Microsoft SharePoint Server 2010	2010-07-15
2013	Microsoft SharePoint Server 2013	2013-01-09
2016	Microsoft SharePoint Server 2016	2016-05-01

Exemples

Installation de SharePoint 2016 pour une batterie de serveurs unique

introduction

SharePoint 2016 est la version 16 de la famille de produits SharePoint. Il a été publié le 4 mai 2016. Cet exemple couvre l'installation de SharePoint 2016 à l'aide de la configuration de la batterie de serveurs unique. Cette configuration couvre les bases de la configuration d'une batterie SharePoint sans avoir besoin de plusieurs serveurs. Notez que les scénarios couverts par une seule batterie de serveurs sont généralement limités à des scénarios de développement et de production de très petite taille.

Exigences

Avant d'installer SharePoint, l'environnement de base doit être configuré. SharePoint stocke des documents ainsi que des métadonnées, des journaux, des applications personnalisées, des personnalisations et bien plus encore. Assurez-vous d'avoir suffisamment d'espace disque et de RAM disponibles au-dessus des exigences de base.

- 4 cœurs sur des processeurs compatibles 64 bits
- 12 à 24 Go de RAM (en fonction du test ou du déploiement de prod)
- 80 Go de disque dur pour le système
- 100 Go de disque dur comme deuxième disque dur
- Serveur avec Windows Server 2012 R2 64 bits ou Aperçu technique «Seuil»
- SQL Server 2014 ou SQL Server 2016
- .NET Framework 4.5.2 ou .NET Framework 4.6
- Domaine connecté à l'ordinateur et aux comptes de service de batterie de serveurs délégués

Toutes les autres conditions préalables peuvent être installées manuellement ou à l'aide du programme d'installation de SharePoint Prerequisite fourni avec l'installation de SharePoint.

Installation

- Exécutez le programme d'installation des prérequis. il peut demander à redémarrer le serveur avant de continuer
- Exécuter Setup.exe à partir de l'installation SharePoint
- Entrez la clé de licence
- Accepter le contrat de licence
- Sélectionnez "Complete" sur l'onglet Type de serveur
- Le programme d'installation doit se terminer avec succès
- Sur la page complète, laissez la case à cocher à côté de l'Assistant Exécuter la configuration

du produit et cliquez sur Fermer.

Configuration

Si vous continuez à partir de l'étape précédente, l'Assistant Configuration du produit SharePoint 2016 devrait s'ouvrir automatiquement. Si la boîte ne s'affiche pas ou si vous exécutez la configuration ultérieurement, ouvrez l'assistant de configuration en cliquant sur Démarrer -> Produits SharePoint 2016 -> Assistant Configuration du produit SharePoint 2016.

- Cliquez sur suivant sur la page d'accueil
- Une boîte de dialogue modale apparaîtra indiquant que certains services peuvent être redémarrés pendant la configuration. rien n'a encore été installé, alors cliquez sur oui
- Ajouter le serveur de base de données pour la batterie
 - Entrez le nom de la machine exécutant SQL Server. dans ce cas, c'est la machine locale
 - Entrez le nom de la base de données de configuration ou conservez le nom par défaut SharePoint_Config
 - Entrez le nom d'utilisateur de l'utilisateur du service de domaine qui accédera à la base de données (sous la forme DOMAIN \ user) * Entrez le mot de passe de l'utilisateur du domaine
 - Cliquez sur Suivant lorsque vous avez terminé
- Entrez le mot de passe de la batterie cela sera utilisé lors de la connexion de serveurs supplémentaires à la nouvelle batterie de serveurs
- Sélectionnez le rôle de batterie de serveurs unique
- Configurez l'application Web d'administration centrale (à partir de laquelle SharePoint sera géré par les administrateurs de la batterie de serveurs), sélectionnez le numéro de port et sélectionnez le type de fédération d'authentification (NTLM ou Négociation (Kerberos)).
- Passez en revue les paramètres sur les dernières pages et apportez les modifications nécessaires.
- Lorsque vous êtes prêt, lancez la configuration qui peut prendre quelques minutes
- À la fin, vous allez ouvrir l'assistant vous permettra d'ouvrir le site Central Admin
- En cas d'échec, vous pouvez rechercher les journaux dans le dossier%
COMMONPROGRAMFILES% \ Microsoft Shared \ Extensions serveur Web \ 16 \ LOG.

Configuration de la ferme

Une fois que l'application Web centrale, la base de données de configuration et l'administrateur central sont configurés, vous êtes prêt à configurer la batterie pour une utilisation par les utilisateurs ou le développement. Vous pouvez marquer l'emplacement du site Central Admin ou y accéder via un raccourci situé au même emplacement que l'assistant de configuration du produit.

- Si vous lancez la configuration ultérieurement, cliquez sur Lancement rapide -> Assistants de configuration -> Assistant Configuration de batterie
- Si vous démarrez l'Assistant à partir de l'étape d'installation, cliquez sur Démarrer l'assistant.
- Choisissez si vous souhaitez faire partie du programme d'amélioration de la clientèle en

cliquant sur Oui ou Non.

- Sur la page de configuration de la batterie, sélectionnez le compte de domaine qui exécutera les services d'arrière-plan sur la batterie de serveurs.
 - Bien que ce compte puisse être le même que le compte de base de données, il peut également être différent pour la séparation des rôles et des privilèges.
 - Entrez le compte en tant que DOMAIN \ user
- Valider les services que vous souhaitez mettre à disposition sur la batterie de serveurs sur la page Services
- Créez la première collection de sites sur la batterie (cette étape peut être ignorée et effectuée ultérieurement).
 - Entrez le titre, la description, l'adresse Web de la collection de sites (généralement le premier site est à la racine du serveur) et le modèle
 - La plupart des choses peuvent être modifiées (titre, description) peuvent être facilement modifiées, mais d'autres, comme l'URL Web, peuvent nécessiter beaucoup plus de travail. le modèle peut également ne pas être facilement annulé, mais SharePoint permet une grande quantité de personnalisations qui vous permettent de prendre n'importe quel modèle de base et de convertir le style et la présentation du site.
- Lorsque vous avez terminé la configuration, cliquez sur Terminer

La batterie et la première collection de sites sont maintenant configurées pour être utilisées.

Construire un composant WebPart avec SharePoint Framework

dev.office.com/sharepoint est un endroit idéal pour commencer avec SharePoint Framework.

SharePoint Framework est une approche client moderne de développement SharePoint initialement ciblée sur SharePoint Online dans Office 365. Les composants WebPart créés avec SharePoint Framework constituent un nouveau type de composant WebPart et peuvent être ajoutés aux pages SharePoint existantes. nouvelles pages SharePoint.

Il existe un excellent exemple de bonjour pour ce processus hébergé à la [création de votre premier composant WebPart côté client SharePoint \(partie 1 de Hello World\)](#) . Tous les exemples sur dev.office.com sont disponibles pour les contributions de la communauté via github.

Les étapes de base de Hello World dans SharePoint Framework sont les suivantes:

1. Générez le squelette du projet avec [le générateur SharePoint Yeoman](#) .

```
yo @ Microsoft / SharePoint
```

2. Modifiez le code généré dans l'éditeur de votre choix. La prise en charge de [Visual Studio Code](#) est forte sur toutes les plates-formes.

3. Prévisualiser le composant WebPart à l'aide de gulp et de SharePoint Workbench local

```
gulp servir
```

4. Aperçu dans votre environnement SharePoint Online

Accédez à l'URL suivante: ' https://votre-sharepoint-site/_layouts/workbench.aspx '

Journaux et journalisation ULS SharePoint

Le service ULS (Unified Logging Service) de SharePoint fournit des fonctionnalités de support et de débogage pour les opérations et les développeurs. Comprendre comment lire les journaux est une première étape importante pour résoudre les problèmes.

Outillage

Microsoft fournit le [visualiseur ULS](#) pour permettre la lecture des anciens journaux et journaux en cours d'écriture pendant l'exécution de la batterie. Il peut également filtrer et appliquer le formatage aux journaux pour aider à réduire un problème.

Identifiant de corrélation

Pour isoler un problème, il est utile de ne regarder qu'un identifiant de corrélation particulier. Chaque identifiant de corrélation est associé à une requête ou à une action de bout en bout du système (tel qu'un job time). En cas de problème lié au rendu d'une page Web, la localisation de la demande dans les journaux ULS et son isolation par rapport à l'ID de corrélation spécifique supprime tout le bruit des autres journaux, ce qui aide à identifier le problème.

Ajout de SPMonitoredScope à mon code

Une façon de créer un journal et de surveiller les performances consiste à ajouter SPMonitoredScope à votre code.

```
using (new SPMonitoredScope("Feature Monitor"))
{
    // My code here
}
```

Ce code enregistre le début et la fin de vos demandes ainsi que certaines données de performances. La création de votre propre moniteur personnalisé qui implémente ISPScopedPerformanceMonitor vous permet de définir le niveau de trace ou la durée d'exécution maximale pour un ensemble de codes.

[Lire Démarrer avec sharepoint en ligne: https://riptutorial.com/fr/sharepoint/topic/950/demarrer-avec-sharepoint](https://riptutorial.com/fr/sharepoint/topic/950/demarrer-avec-sharepoint)

Chapitre 2: App SharePoint

Introduction

Application hébergée SharePoint

Remarques

Référence requise du site: <http://www.letsharepoint.com/what-is-user-information-list-in-sharepoint-2013/>

Exemples

SharePoint 2013: accéder aux données de service de profil utilisateur à l'aide de JSOM dans SharePoint 2013

SharePoint 2013: accéder aux données de service de profil utilisateur à l'aide de JSOM dans SharePoint 2013

Dans cet article, nous allons apprendre à gérer ou à accéder à l'application UPS (User Profile Service) en utilisant JSOM (Javascript Object Model) et à créer une application de base. Avant de commencer, passons d'abord par la terminologie de base de l'onduleur.

Profil d'utilisateur - Il contient toutes les informations des personnes dans une organisation de manière organisée. Il affiche toutes les propriétés telles que AccountName, FirstName, LastName, WorkEmail, etc. liées à un utilisateur.

Application de service de profil utilisateur - Il est considéré comme un emplacement centralisé pour stocker tous les profils utilisateur et permet aux administrateurs de configurer ou de gérer des profils, la synchronisation des profils, Mon site, les balises sociales, etc.

Mon site - Site personnalisé permettant à un utilisateur individuel de gérer ses informations et de stocker des documents, des liens, etc. Mon site est accessible en cliquant sur Nom d'utilisateur dans le coin supérieur droit de la page SharePoint.

Gérer et accéder aux données de profil utilisateur

Comme nous allons travailler avec JSOM, nous ne pouvons effectuer que des opérations de «lecture», à l'exception de la possibilité de modifier l'image de profil à l'aide de JSOM (ou CSOM ou REST).

* Server Side Code permet de lire / écrire les deux opérations.

Récupérer les propriétés du profil utilisateur à l'aide de JSOM

Permet de créer une application hébergée SharePoint et de récupérer les informations utilisateur

dans cette application -

Lancez Visual Studio 2013 et sélectionnez "App for SharePoint 2013" dans Nouveau projet. Après avoir sélectionné le type de projet ci-dessus, une fenêtre vous permet de vous connecter au site SharePoint et de sélectionner le type d'application à déployer (voir la capture d'écran ci-dessous). Cliquez sur Terminer.

3.) Une fois le projet créé, vous verrez un ensemble de dossiers / fichiers ajoutés dans l'Explorateur de solutions ajouté par défaut au projet.

4.) Si vous ouvrez la page "Default.aspx", vous trouverez certaines bibliothèques JavaScript déjà ajoutées à la page.

Ici, nous devons ajouter une bibliothèque supplémentaire pour commencer à travailler avec les profils utilisateur

Lire App SharePoint en ligne: <https://riptutorial.com/fr/sharepoint/topic/9876/app-sharepoint>

Chapitre 3: Création d'une application hébergée par un fournisseur

Exemples

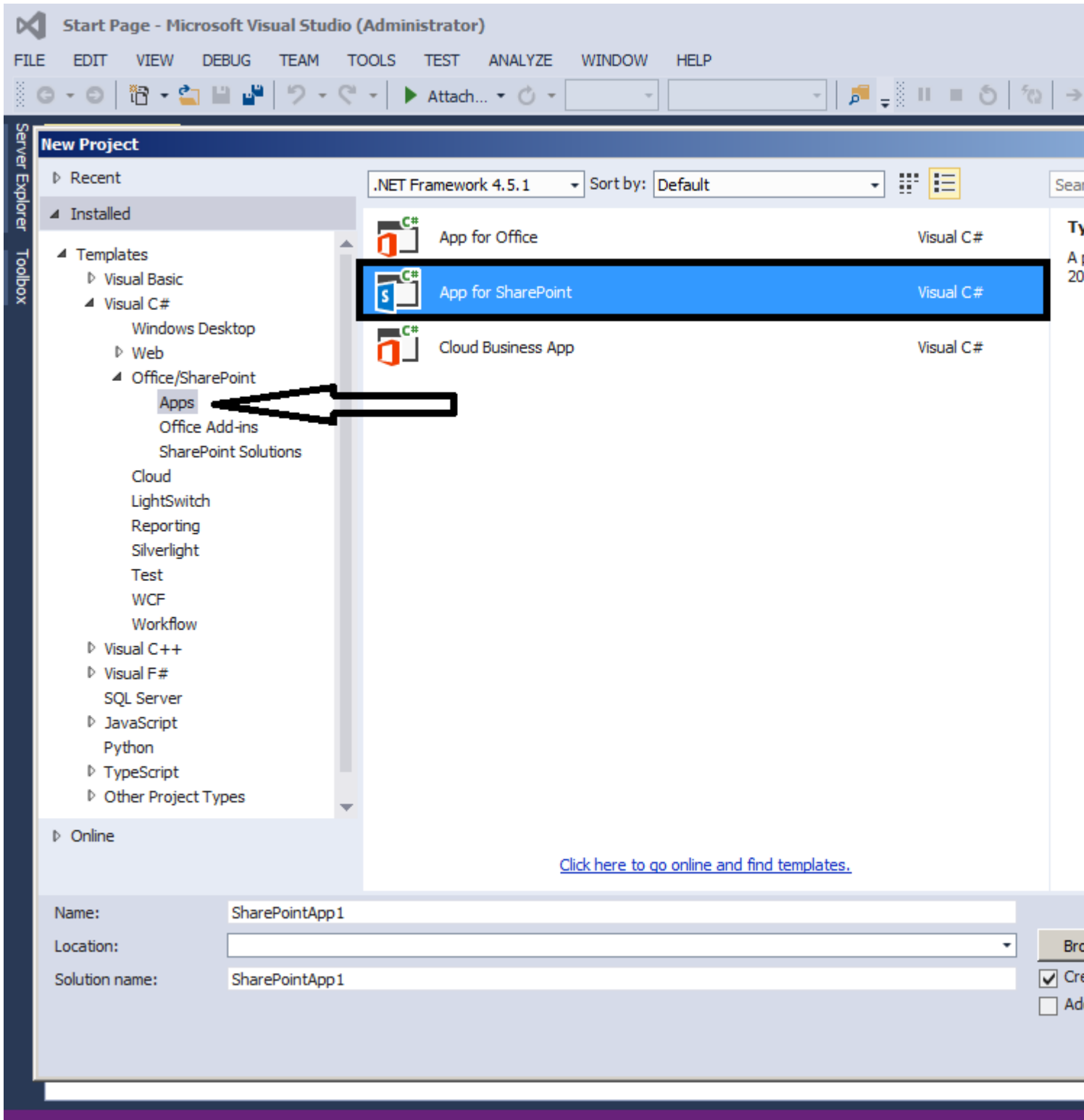
Définition de l'environnement de développement

Pour commencer avec le développement d'applications, nous avons besoin de Visual Studio 2013 ou d'une version ultérieure. Téléchargez la dernière édition de la communauté ou de l'expression à partir d'ici> <https://www.visualstudio.com/products/free-developer-offers-vs>

Une fois téléchargé et installé

Ouvrir et **cliquer créer un nouveau projet**

Développez la section Office / SharePoint, vous devriez voir une option pour App comme indiqué ci-dessous.



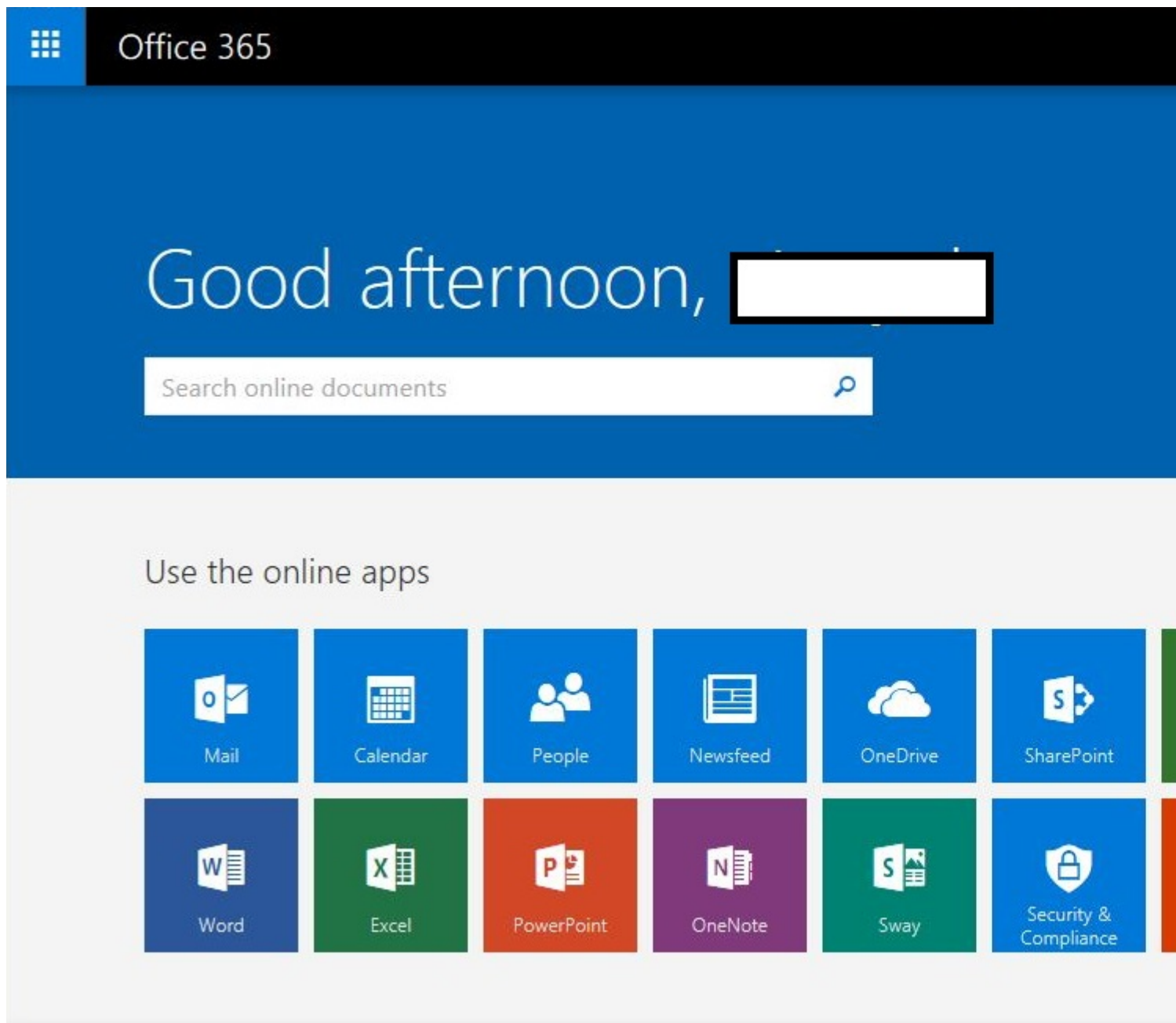
Si l'option de l'application n'est pas disponible Fermez le VS, téléchargez et installez les **outils de développement Microsoft Office** <https://www.visualstudio.com/en-us/features/office-tools-vs.aspx>

Préparation pour le site développeur

Une fois que nous avons Visual Studio, nous avons besoin d'un site de développeur pour déployer des applications sur SharePoint. Manière la plus simple est d'obtenir est> Inscrivez - vous gratuitement à un an compte Office 365 développeur

<https://profile.microsoft.com/RegSysProfileCenter/wizardnp.aspx?wizid=14b845d0-938c-45af-b061-f798fbb4d170&lcid=1033>

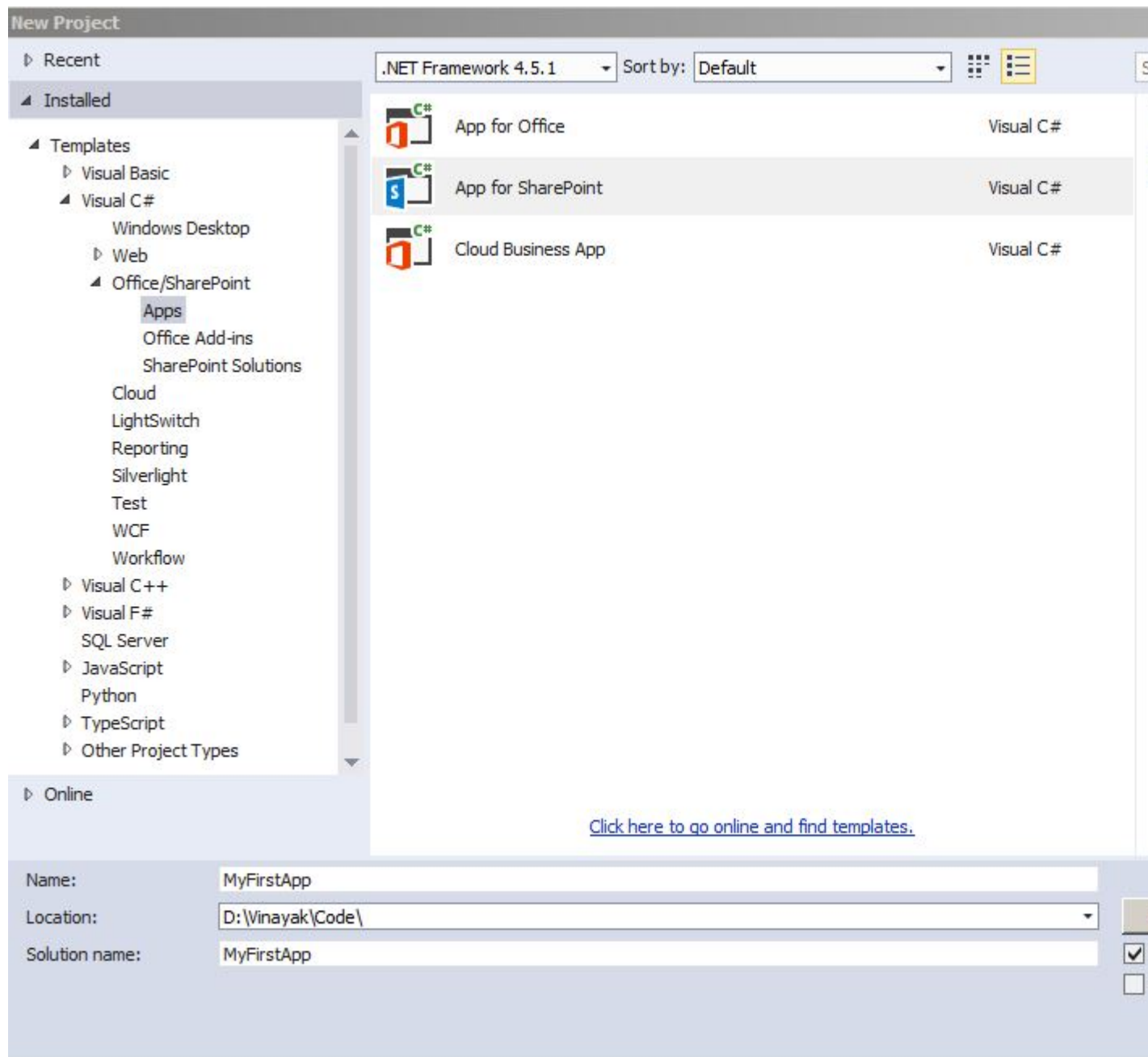
Une fois le processus d'inscription terminé <https://www.office.com/> URL du centre de toutes vos applications



Créer une application dans Visual studio


Commençons par créer notre première application

1. Ouvrir studio visuel et> créer un nouveau projet
2. Entrez le nom et l'emplacement



3. Entrez l'URL de votre site de développeur créée à l'étape précédente et sélectionnez Hébergement hébergé par le fournisseur.

New app for SharePoint ? ×



Specify the app for SharePoint settings

What SharePoint site do you want to use for debugging your app?

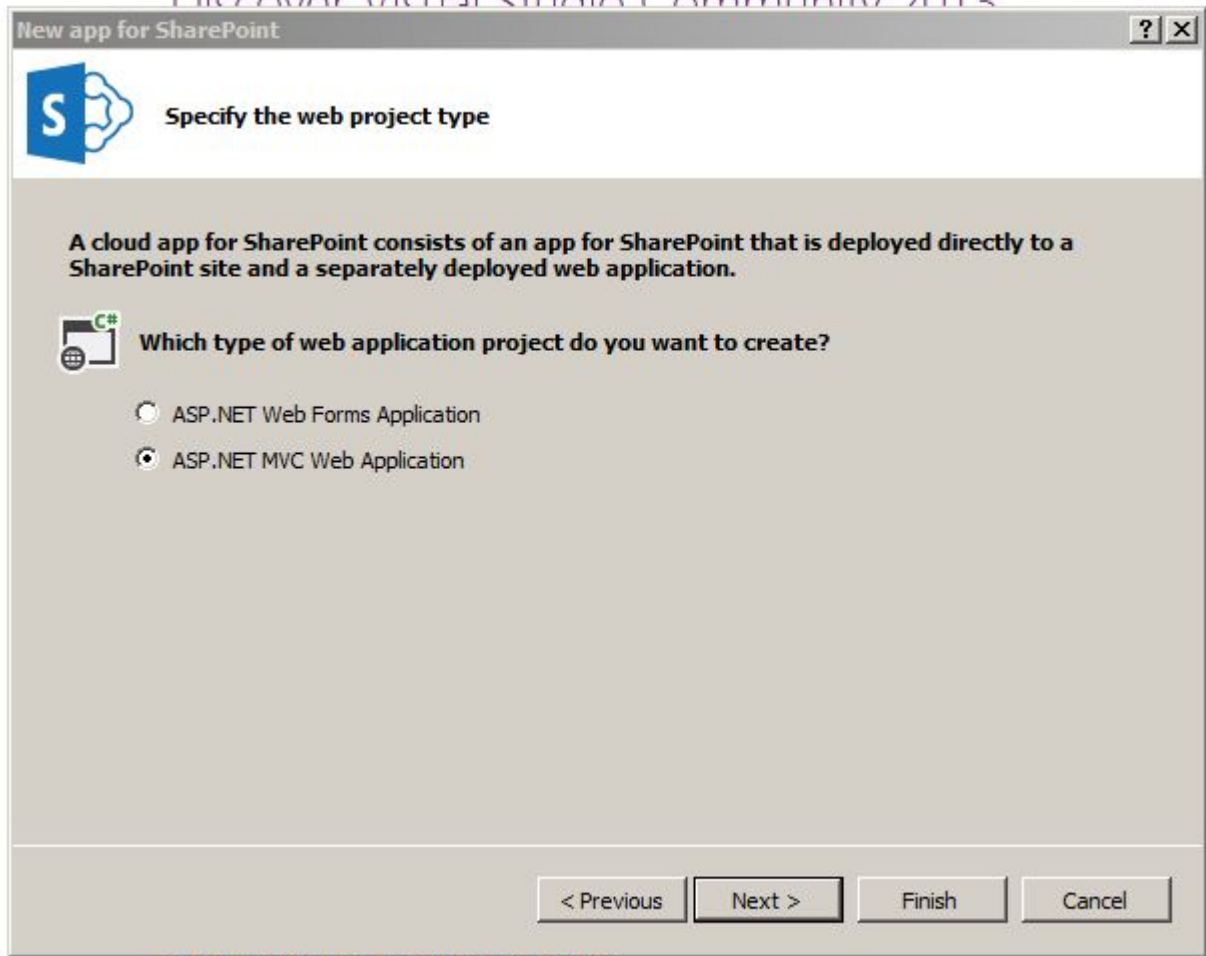
Don't have a developer site?
[Sign up for an Office 365 Developer site to develop, test and deploy apps for Office and SharePoint](#)

How do you want to host your app for SharePoint?

Provider-hosted
 SharePoint-hosted
[Learn more about this choice](#)

4. Popup ouvrira qui sera comme pour la connexion
5. La prochaine étape consistera, pour le type d'application, à sélectionner MVC ou Webform.
Je sélectionne MCV ici

3

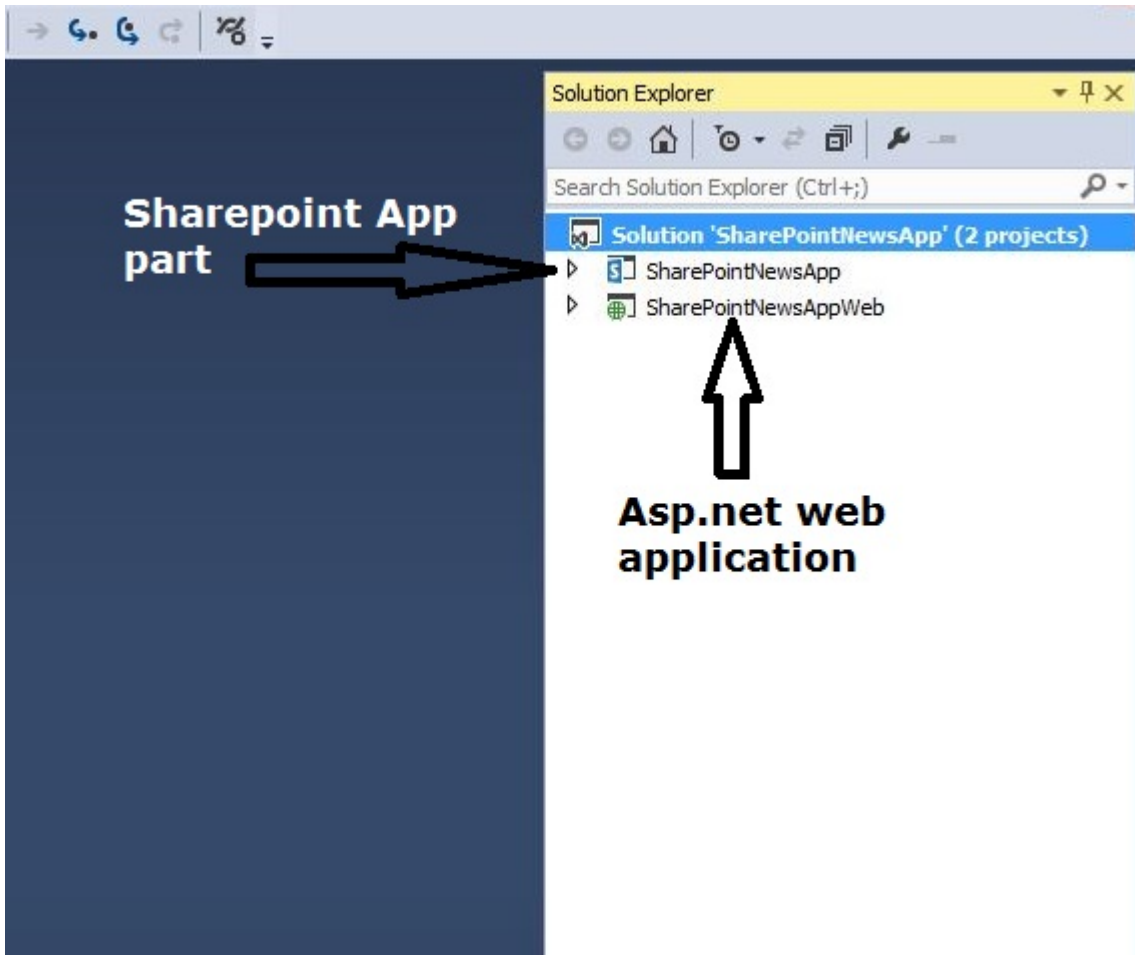


[Exploring dotnet new with .NET Core](#)

Wednesday, July 27, 2016

I'm very enjoying the "dotnet" command line. Mostly I do "dotnet new" and then add to the default Hello World app with the Visual Studio Code editor. Recently, though, I realized that the -t "type" and -l "lang" options are there

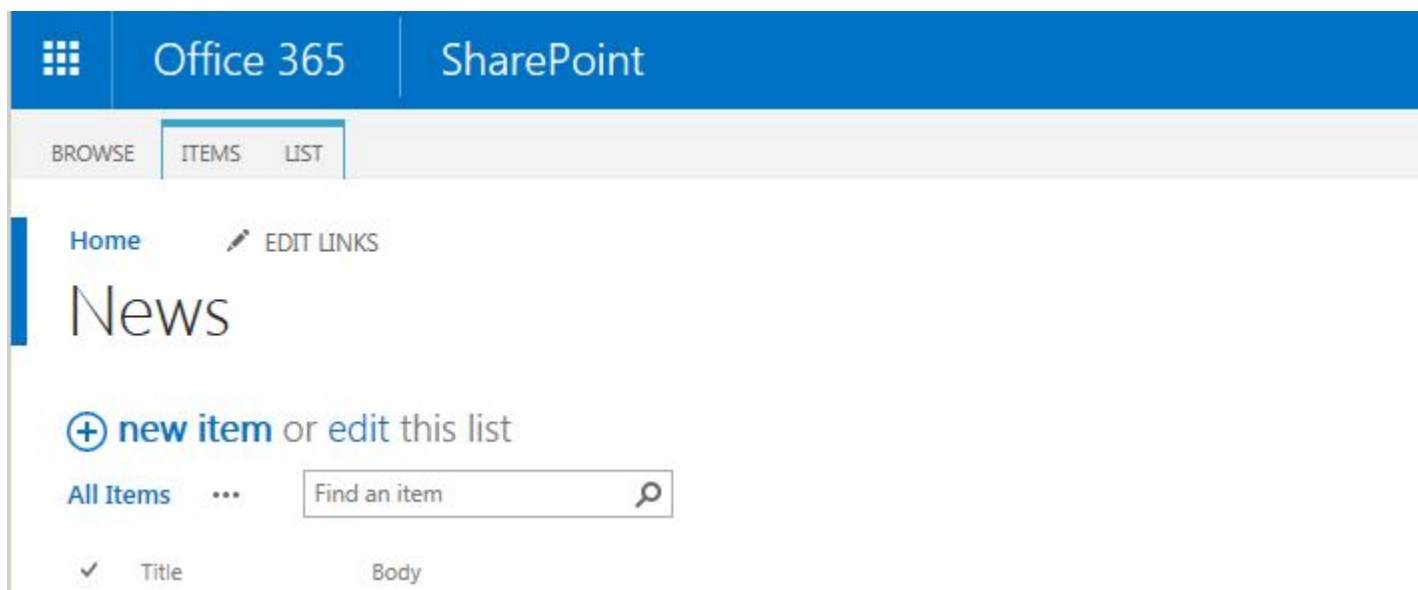
6. Sous Comment voulez-vous que votre complément s'authentifie?, Sélectionnez Utiliser le service de contrôle d'accès Windows Azure et cliquez sur Terminer.
7. Dans l'explorateur de solutions, nous pouvons voir que 2 projets ont été créés. L'une est l'application SharePoint et une autre l'application Web asp.net



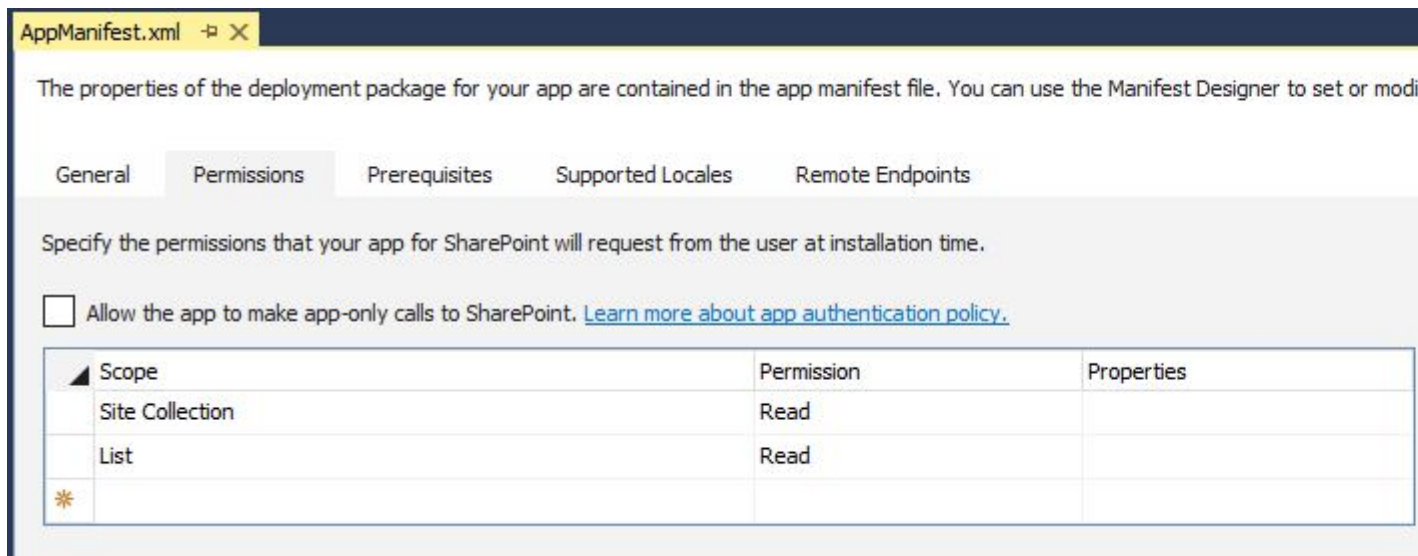
Commençons à coder

Ici, je prends l'exemple d'une application de base

1. Ouvrez le site de développeur SharePoint et créez une liste pour stocker nos articles d'actualité
2. Créez une liste personnalisée et ajoutez 3 colonnes supplémentaires Body, Summery, ThumbnailImageUrl



3. Revenez à notre application SharePoint, ouvrez le fichier AppManifest.xml, cliquez sur l'onglet d'autorisation et accordez l'autorisation de lecture à la collection de sites et enregistrez-la.



4. Ouvrez HomeController depuis une application Web, dans mon cas, c'est une application MVC. Si vous créez une application Webform, votre code doit être dans la page default.aspx.cs
5. Voici l'extrait de code pour obtenir les dernières nouvelles de la liste. Voici comment notre page d'index devrait ressembler.

```
[SharePointContextFilter]
public ActionResult Index()
{
    User spUser = null;

    var spContext = SharePointContextProvider.Current.GetSharePointContext(HttpContext);
    List<NewsList> newsList = new List<NewsList>();
    using (var clientContext = spContext.CreateUserClientContextForSPHost())
    {
        if (clientContext != null)
        {
            spUser = clientContext.Web.CurrentUser;

            clientContext.Load(spUser, user => user.Title);

            clientContext.ExecuteQuery();

            ViewBag.UserName = spUser.Title;

            List lst = clientContext.Web.Lists.GetByTitle("News");
            CamlQuery queryNews = CamlQuery.CreateAllItemsQuery(10);
            ListItemCollection newsItems = lst.GetItems(queryNews);
            clientContext.Load(newsItems, includes => includes.Include(i => i.Id, i =>
i.DisplayName, i => i["ThumbnailImageUrl"], i => i["Summery"]));

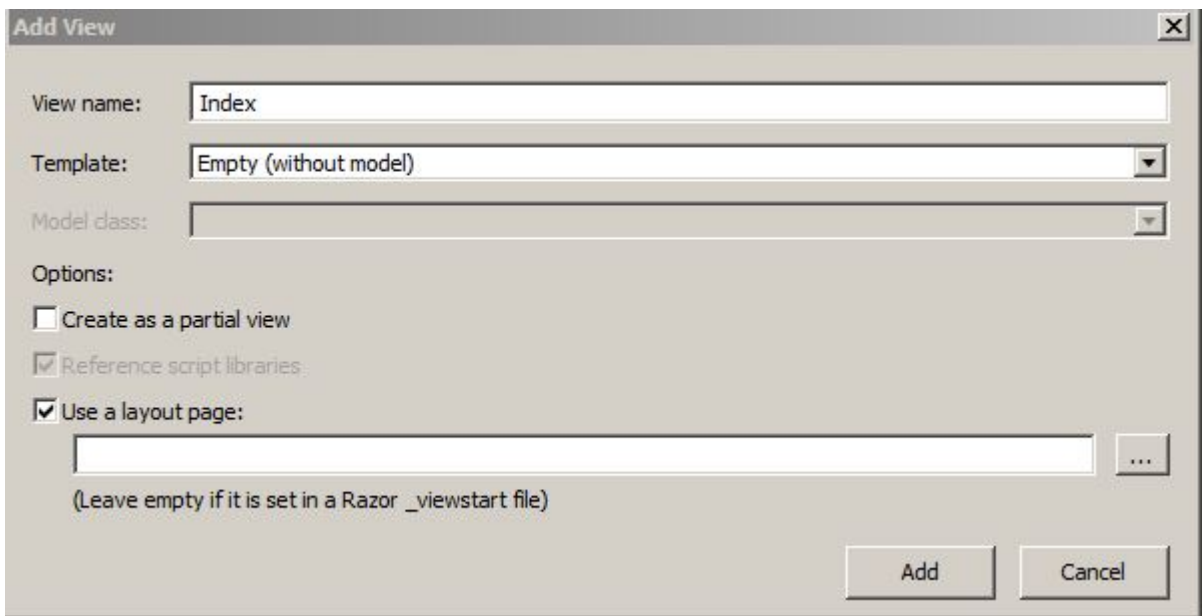
            clientContext.ExecuteQuery();

            if (newsItems != null)
            {
                foreach (var lstProductItem in newsItems)
            
```

```
        {
            newsList.Add(
                new NewsList
                {
                    Id = Convert.ToInt32(lstProductItem.Id.ToString()),
                    Title = lstProductItem.DisplayName.ToString(),
                    Summery = lstProductItem["Summery"].ToString(),
                    Thumbnail = lstProductItem["ThumbnailImageUrl"].ToString()
                });
        }
    }
}

return View(newsList);
}
```

6. Maintenant, cliquez avec le bouton droit sur **Index** et cliquez sur **Ajouter une vue**. Cliquez ensuite sur **Ajouter**



7. Maintenant, ouvrez le fichier **Index.cshtml** du répertoire **Views> Home**

8. Voici l'extrait de code pour le fichier **index.cshtml**

```
@model List<SharePointNewsAppWeb.Models.NewsList>
@{
    ViewBag.Title = "My News - browse latest news";
}
<br />
@foreach (var item in Model)
{
    <div class="row panel panel-default">
        <div class="col-xs-3">
            <a href="/home/aticle?ArticleId=@item.Id">
                
            </a>
        </div>
    </div>
}
```

```

<div class="col-xs-9 panel-default">
  <div class="panel-heading">
    <h4><a href="/home/article?ArticleId=@item.Id">@item.Title.ToUpper()</a></h4>
  </div>
  <div class="panel-body">
    <p>@item.Summary</p>
  </div>
</div>

```

9. Cliquez avec le bouton droit sur le dossier Model dans votre solution et ajoutez un fichier de classe CS. Ajouter les classes de modèle ci-dessous

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace SharePointNewsAppWeb.Models
{
    public class NewsApp
    {
    }
    public class NewsList
    {

    public int Id { get; set; }

    public string Title { get; set; }

    public string Summary { get; set; }

    public string Thumbnail { get; set; }
    }
    public class FullArticle
    {

    public int Id { get; set; }

    public string Title { get; set; }

    public string Body { get; set; }

    }
}

```

10. Utilisez la touche F5 pour déployer et exécuter votre complément. Si une fenêtre d'alerte de sécurité vous invite à faire confiance au certificat Localhost auto-signé, choisissez Oui.

Et maintenant, la première application est prête

Création d'une page d'article complète

Nous avons déjà créé la première page qui montrera tous les articles d'actualité. Cette page affichera l'article complet.

1. Ajouter une méthode d'action supplémentaire à HomeController

```
[SharePointContextFilter]
public ActionResult Aticle(int ArticleId)
{
    User spUser = null;

    var spContext = SharePointContextProvider.Current.GetSharePointContext(HttpContext);
    FullArticle article = new FullArticle();
    using (var clientContext = spContext.CreateUserClientContextForSPHost())
    {
        if (clientContext != null)
        {
            spUser = clientContext.Web.CurrentUser;

            clientContext.Load(spUser, user => user.Title);

            clientContext.ExecuteQuery();

            ViewBag.UserName = spUser.Title;

            List lst = clientContext.Web.Lists.GetByTitle("News");
            CamlQuery queryNews = new CamlQuery();
            queryNews.ViewXml = @"<View><Query><Where><Eq><FieldRef Name='ID' />" +
                "<Value Type='Number'>" + ArticleId + "</Value></Eq></Where></Query>" +
                "<ViewFields><FieldRef Name='ID' /><FieldRef Name='Title' /><FieldRef" +
                "Name='Body' /></ViewFields></View>"; //
            ListItemCollection newsItems = lst.GetItems(queryNews);
            clientContext.Load(newsItems, includes => includes.Include(i => i.Id, i =>
                i.DisplayName, i => i["Body"]));

            clientContext.ExecuteQuery();

            if (newsItems != null)
            {
                foreach (var lstProductItem in newsItems)
                {
                    article.Id = Convert.ToInt32(lstProductItem.Id.ToString());
                    article.Title = lstProductItem.DisplayName.ToString();
                    article.Body = lstProductItem["Body"].ToString();
                }
            }
        }
    }
    return View(article);
}
```

2. Encore une fois Cliquez avec le bouton droit sur Action et créez une vue avec le même nom Nom de la méthode d'action. Dans mon cas, la vue s'appellera **Aticle**

```
@model SharePointNewsAppWeb.Models.FullArticle

@{
    ViewBag.Title = "Aticle";
}

<br />
<div class="panel panel-default">
    <div class="panel-heading"><a style="font-size:20px;" href="/"><i class="glyphicon
```

```
glyphicon-chevron-left"></i> <i class="glyphicon glyphicon-home"></i> </a></div>
<div class="panel-heading"><h1 class="h2">@Model.Title.ToUpper () </h1></div>
<div class="panel-body">@Html.Raw (@Model.Body) </div>
</div>
```

Ceci est le code pour la page complète de l'article qui montre le corps de l'article de nouvelles

Lire [Création d'une application hébergée par un fournisseur en ligne](https://riptutorial.com/fr/sharepoint/topic/6301/creation-d-une-application-hebergee-par-un-fournisseur):

<https://riptutorial.com/fr/sharepoint/topic/6301/creation-d-une-application-hebergee-par-un-fournisseur>

Chapitre 4: Rejets majeurs

Exemples

SharePoint 2016

Numéro de construction	La description	Produit
16.0.4366.1000	Mise à jour cumulative avril 2016	SharePoint Server 2016
16.0.4336.1000	RTM	SharePoint Server 2016
16.0.4327.1000	Release Candidate	SharePoint Server 2016
16.0.4266.1001	16.0.4306.1002 Bêta 2	SharePoint Server 2016

SharePoint 2013

Numéro de construction	La description
15.0.4623.1001	Juin 2014
15.0.4631.1001	Juillet 2014
15.0.4641.1001	Août 2014
15.0.4649.1001	Septembre 2014
15.0.4659.1001	Octobre 2014
15.0.4667.1000	Novembre 2014
15.0.4675.1000	Décembre 2014
15.0.4693.1001	Février 2015
15.0.4701.1001	Mars 2015
15.0.4711.1000	Avril 2015 (SP1 REQ)
15.0.4719.1002	Mai 2015
15.0.4727.1001	Juin 2015
15.0.4737.1000	Juillet 2015
15.0.4745.1000	Août 2015

Numéro de construction	La description
15.0.4753.1003	Septembre 2015
15.0.4763.1002	Octobre 2015
15.0.4771.1000	Novembre 2015
15.0.4779.1000	Décembre 2015
15.0.4787.1000	MS16-004
15.0.4787.1000	Janvier 2016
15.0.4797.1001	Février 2016
15.0.4805.1000	Mars 2016
15.0.4815.1000	Avril 2016
15.0.4823.1003	Mai 2016
15.0.4833.1000	Juin 2016

Source: [SharePoint 2013 Build Numbers et CU's](#)

Lire Rejets majeurs en ligne: <https://riptutorial.com/fr/sharepoint/topic/2737/rejets-majeurs>

Chapitre 5: Rendu côté client SharePoint 2013

Introduction

Le rendu côté client (CSR) est un nouveau concept introduit dans SharePoint 2013. Il vous fournit un mécanisme vous permettant d'utiliser votre propre rendu de sortie pour un ensemble de contrôles hébergés dans une page SharePoint (affichages de liste, formulaires de liste). et résultats de recherche). Le rendu du site client est simple lorsque les données sont transformées à l'aide du client plutôt que du serveur. Cela signifie qu'il faut utiliser des technologies côté client, telles que HTML et JavaScript, plutôt que d'écrire XSLT.

Exemples

Modifier le lien hypertexte des champs / colonnes dans la vue liste à l'aide de la CSR

L'exemple ci-dessous montre comment modifier le lien hypertexte pour les champs " ID " et " Titre (LinkTitle) " dans la vue de liste à l'aide de la CSR.

Etape 1: Créez un fichier JS et collez ci-dessous le code

```
(function () {

    function registerRenderer() {
        var ctxForm = {};
        ctxForm.Templates = {};

        ctxForm.Templates = {
            Fields : {
                'LinkTitle': { //----- Change Hyperlink of LinkTitle
                    View : function (ctx) {
                        var url = String.format('{0}?ID={1}',
                            "/sites/Lists/testlist/EditItem.aspx", ctx.CurrentItem.ID);
                        return String.format('<a href="{0}" onclick="EditItem2(event, \'{0}\');return false;">{1}</a>', url, ctx.CurrentItem.Title);
                    }
                },
                'ID' : { //----- Change Hyperlink from ID field
                    View : function (ctx) {
                        var url = String.format('{0}?ID={1}',
                            "/IssueTracker/Lists/testlist/DisplayItem.aspx", ctx.CurrentItem.ID);
                        return String.format('<a href="{0}" onclick="EditItem2(event, \'{0}\');return false;">{1}</a>', url, ctx.CurrentItem.ID);
                    }
                },
            },
        };
        SPClientTemplates.TemplateManager.RegisterTemplateOverrides(ctxForm);
    }
})();
```

```

}
ExecuteOrDelayUntilScriptLoaded(registerRenderer, 'clienttemplates.js');
})();

```

Étape 2: GoTo propriétés de composant WebPart de View List et ajouter une référence JS Link à ce fichier js nouvellement créé (par exemple, ~ sitecollection / SiteAssets / CSRCodeFile.js)

(Remarque: reportez-vous à votre JSlink dans ce format uniquement. "~ Sitecollection / YourJSFilePath".)

Étape 3: Appy et fait

Masquer la colonne à partir de la vue de liste SharePoint à l'aide de CSR.

Cet exemple montre comment masquer un champ "Date" à partir de la vue de liste SharePoint à l'aide de CSR.

```

(function () {

    function RemoveFields(ctx) {
        var fieldName = "Date"; // here Date is field or column name to be hide
        var header = document.querySelectorAll("[displayname=" + fieldName +
        "]" )[0].parentNode;
        var index = [].slice.call(header.parentNode.children).indexOf(header) + 1;
        header.style.display = "none";
        for (var i = 0, cells = document.querySelectorAll("td:nth-child(" + index + ")"); i <
        cells.length; i++) {
            cells[i].style.display = "none";
        }
    }

    function registerRenderer() {
        var ctxForm = {};
        ctxForm.Templates = {};
        ctxForm.OnPostRender = RemoveFields;
        SPClientTemplates.TemplateManager.RegisterTemplateOverrides(ctxForm);
    }
    ExecuteOrDelayUntilScriptLoaded(registerRenderer, 'clienttemplates.js');
})();

```

Appliquer des validations sur Nouveau / Modifier un formulaire d'élément à l'aide de CSR

Supposons que nous ayons une liste SharePoint et que celle-ci comporte quatre champs. Titre, Nom complet, Email, Numéro de mobile, etc. Maintenant, si vous souhaitez appliquer une validation personnalisée dans le formulaire Nouveau / Modifier un élément, vous pouvez facilement le faire avec le code RSE. Le ci-dessous mentionné peut valider les conditions suivantes dans des formulaires:

- Valeurs vides dans les champs

- Vérification du format d'ID de messagerie avec une expression régulière
- Format de numéro de mobile Vérifiez avec une expression régulière
- Le champ Nom complet ne doit pas contenir de valeurs numériques

Etape: 1 Créez un fichier JS, par exemple `CSRValidations.js` et copiez le code suivant dans le fichier JS

```
(function () {

    // Create object that have the context information about the field that we want to
    change it's output render
    var fieldContext = {};
    fieldContext.Templates = {};
    fieldContext.Templates.Fields = {
        // Apply the new rendering for Email field on New and Edit Forms
        "Title": {
            "NewForm": titleFieldTemplate,
            "EditForm": titleFieldTemplate
        },
        "Full_x0020_Name": {
            "NewForm": fullNameFieldTemplate,
            "EditForm": fullNameFieldTemplate
        },
        "Email": {
            "NewForm": emailFieldTemplate,
            "EditForm": emailFieldTemplate
        },
        "Mobile_x0020_Phone": {
            "NewForm": mobilePhoneFieldTemplate,
            "EditForm": mobilePhoneFieldTemplate
        }
    }
};

SPClientTemplates.TemplateManager.RegisterTemplateOverrides(fieldContext);

})();

// This function provides the rendering logic
function emailFieldTemplate(ctx) {

    var formCtx = SPClientTemplates.Utility.GetFormContextForCurrentField(ctx);

    // Register a callback just before submit.
    formCtx.registerGetValueCallback(formCtx.fieldName, function () {
        return document.getElementById('inpEmail').value;
    });

    //Create container for various validations
    var validators = new SPClientForms.ClientValidation.ValidatorSet();
    validators.RegisterValidator(new emailValidator());

    // Validation failure handler.
    formCtx.registerValidationErrorCallback(formCtx.fieldName, emailOnError);

    formCtx.registerClientValidator(formCtx.fieldName, validators);

    return "<span dir='none'><input type='text' value='" + formCtx.fieldValue + "'
maxlength='255' id='inpEmail' class='ms-long'> \ <br><span id='spnEmailError' class='ms-
formvalidation ms-csrformvalidation'></span></span>";
};
```



```

    var isError = false;
    var errorMessage = "";

    if (value.trim() == "") {
        isError = true;
        errorMessage = "You must specify a value for this required field.";
    }

    //Send error message to error callback function (titleOnError)
    return new SPClientForms.ClientValidation.ValidationResult(isError, errorMessage);

};

};

// Add error message to spnError element under the input field element
function titleOnError(error) {
    document.getElementById("spnTitleError").innerHTML = "<span role='alert'" +
error.errorMessage + "</span>";
}

// This function provides the rendering logic
function mobilePhoneFieldTemplate(ctx) {

    var formCtx = SPClientTemplates.Utility.GetFormContextForCurrentField(ctx);

    // Register a callback just before submit.
    formCtx.registerGetValueCallback(formCtx.fieldName, function () {
        return document.getElementById('inpMobilePhone').value;
    });

    //Create container for various validations
    var validators = new SPClientForms.ClientValidation.ValidatorSet();
    validators.RegisterValidator(new mobilePhoneValidator());

    // Validation failure handler.
    formCtx.registerValidationErrorCallback(formCtx.fieldName, mobilePhoneOnError);

    formCtx.registerClientValidator(formCtx.fieldName, validators);

    return "<span dir='none'" + "<input type='text' value='" + formCtx.fieldValue + "'
maxlength='255' id='inpMobilePhone' class='ms-long'" + " \ <br><span id='spnMobilePhoneError'
class='ms-formvalidation ms-csrformvalidation'" + "</span></span>";
}

// Custom validation object to validate mobilePhone format
mobilePhoneValidator = function () {
    mobilePhoneValidator.prototype.Validate = function (value) {
        var isError = false;
        var errorMessage = "";

        //MobilePhone format Regex expression
        //var mobilePhoneRejex = /\S+@\S+\.\S+\/;
        var mobilePhoneRejex = /^[0-9]+$/;

        if (value.trim() == "") {
            isError = true;
            errorMessage = "You must specify a value for this required field.";
        } else if (!mobilePhoneRejex.test(value) && value.trim()) {
            isError = true;
            errorMessage = "Please enter valid mobile phone number";
        }
    }
}

```

```

        //Send error message to error callback function (mobilePhoneOnError)
        return new SPClientForms.ClientValidation.ValidationResult(isError, errorMessage);

    };

};

// Add error message to spnError element under the input field element
function mobilePhoneOnError(error) {
    document.getElementById("spnMobilePhoneError").innerHTML = "<span role='alert'>" +
error.errorMessage + "</span>";
}

// This function provides the rendering logic
function fullNameFieldTemplate(ctx) {

    var formCtx = SPClientTemplates.Utility.GetFormContextForCurrentField(ctx);

    // Register a callback just before submit.
    formCtx.registerGetValueCallback(formCtx.fieldName, function () {
        return document.getElementById('inpFullName').value;
    });

    //Create container for various validations
    var validators = new SPClientForms.ClientValidation.ValidatorSet();
    validators.RegisterValidator(new fullNameValidator());

    // Validation failure handler.
    formCtx.registerValidationErrorCallback(formCtx.fieldName, fullNameOnError);

    formCtx.registerClientValidator(formCtx.fieldName, validators);

    return "<span dir='none'><input type='text' value='" + formCtx.fieldValue + "'
maxlength='255' id='inpFullName' class='ms-long'> \ <br><span id='spnFullNameError' class='ms-
formvalidation ms-csrformvalidation'></span></span>";
}

// Custom validation object to validate fullName format
fullNameValidator = function () {
    fullNameValidator.prototype.Validate = function (value) {
        var isError = false;
        var errorMessage = "";

        //FullName format Regex expression
        var fullNameRejex = /^[a-z ,.'-]+$&#x2F;i;

        if (value.trim() == "") {
            isError = true;
            errorMessage = "You must specify a value for this required field.";
        }else if (!fullNameRejex.test(value) && value.trim()) {
            isError = true;
            errorMessage = "Please enter valid name";
        }

        //Send error message to error callback function (fullNameOnError)
        return new SPClientForms.ClientValidation.ValidationResult(isError, errorMessage);

    };

};

// Add error message to spnError element under the input field element

```

```
function fullNameOnError(error) {
    document.getElementById("spnFullNameError").innerHTML = "<span role='alert'>" +
error.errorMessage + "</span>";
}
```

Etape: 2 Ouvrez Nouveau formulaire d'élément dans le navigateur. Modifier la page et modifier le composant WebPart.

Etape: 3 Dans les propriétés du composant WebPart, allez dans Divers -> JS link -> collez le chemin de votre fichier js (par exemple, ~ sitecollection / SiteAssets / CSRValidations.js)

Etape: 4 Enregistrez les propriétés et la page du composant WebPart.

Modifier le nom d'affichage de la colonne dans la vue liste à l'aide de CSR

Il y a des cas où vous devez changer le nom d'affichage de la colonne dans une vue de liste

Par exemple, le nom de colonne affiché dans la vue est "IsApprovalNeeded" et vous souhaitez apparaître comme "Est-ce qu'une approbation est nécessaire?".

Vous pouvez bien sûr modifier le nom d'affichage d'une colonne en modifiant le titre de la colonne dans les paramètres de liste, mais si vous souhaitez le conserver tel quel dans les paramètres de la liste et le modifier uniquement dans l'aperçu de la page, vous pouvez le faire en utilisant RSE (Rendu côté client).

Voici le code ...

```
(function () {

    function preTaskFormRenderer(renderCtx) {
        modifyColumns(renderCtx);
    }

    function modifyColumns(renderCtx)
    {
        var arrayLength= renderCtx.ListSchema.Field.length;
        for (var i=0; i < arrayLength;i++)
        {
            if(renderCtx.ListSchema.Field[i].DisplayName == 'IsApprovalNeeded')
            {
                var newTitle= "Is Approval Needed?";
                var linkTitleField = renderCtx.ListSchema.Field[i];
                linkTitleField.DisplayName = newTitle;
            }
        }
    }

    function registerRenderer()
    {
        var ctxForm = {};
        ctxForm.Templates = {};
        ctxForm.OnPreRender = preTaskFormRenderer;
        SPClientTemplates.TemplateManager.RegisterTemplateOverrides(ctxForm);
    }
}
```

```
ExecuteOrDelayUntilScriptLoaded(registerRenderer, 'clienttemplates.js');  
})();
```

Lire Rendu côté client SharePoint 2013 en ligne:

<https://riptutorial.com/fr/sharepoint/topic/8317/rendu-cote-client-sharepoint-2013>

Chapitre 6: Services REST

Remarques

URL du noeud final du service REST

L'API d'accès au client REST a été introduite pour la première fois dans SharePoint 2010, mais elle a été considérablement étendue dans SharePoint 2013. L'API REST dans [SharePoint 2010](#) est accessible via le service Web ListData à l'URL `/_vti_bin/ListData.svc`. [SharePoint 2013](#) a introduit les URL de point de terminaison `/_api/lists/` et `/_api/web`, qui se comportent légèrement différemment.

Les URL de noeud final ci-dessus doivent être précédées de `http://server/site` où `server` représente le nom du serveur et `site` représente le nom ou le chemin d'accès du site spécifique.

Exemple d'URL pour ...	SharePoint 2010	SharePoint 2013
Récupérer une liste:	<code>/_vti_bin/ListData.svc/ListItemName</code>	<code>/_api/lists('ListGuid')</code>
Récupérer un article:	<code>/_vti_bin/ListData.svc/ListItemName(1)</code>	<code>/_api/lists('ListGuid')/items(1)</code>
Récupérer un site Web:	(pas d'équivalent)	<code>/_api/web</code>

Malgré les différences d'accès aux listes et aux éléments de liste, l'utilisation de ces résultats est très similaire dans les deux versions.

Notez que le service `ListData.svc` est toujours disponible dans SharePoint 2013 pour la compatibilité ascendante.

Envoi de demandes REST

Une demande REST peut être soumise via un XMLHttpRequest JavaScript natif ou via la construction d'encapsulation jQuery AJAX.

Syntaxe XMLHttpRequest

```
var xhr = new XMLHttpRequest();
xhr.open(verb, url, true);
xhr.setRequestHeader("Content-Type", "application/json");
xhr.send(data);
```

jQuery AJAX Syntaxe

```
$.ajax({
  method: verb,
  url: url,
  headers: { "Content-Type":"application/json" },
  data: data
});
```

Pour plus de détails sur l'envoi de requêtes via AJAX, consultez [la documentation JavaScript AJAX](#).

Exemples

Travailler avec des listes

Obtention d'éléments de la liste

Cet exemple montre comment récupérer tous les éléments de la liste et les parcourir. Vous pouvez utiliser le paramètre `top` pour demander un certain nombre de résultats. Vous pouvez également utiliser le paramètre `select` pour sélectionner certains champs (`$select=id, Title, uri`).

JavaScript

```
function GetListItems(){
  $.ajax({
    url: "../_api/web/lists/getbytitle('List Title')/items?$top=50"
    contentType: "application/json;odata=verbose",
    method: "GET",
    headers: { "accept": "application/json;odata=verbose" },
    success: function (data) {
      $.each(data.d.results, function(index,item){
        //use item to access the individual list item
        console.log(item.Id);
      });
    },
    error: function(error){
      console.log(error);
    }
  });
}
```

Obtenir un élément de liste individuel

JavaScript

```
function GetListItem(){
  $.ajax({
    url: "../_api/web/lists/getbytitle('List Title')/items(1)",
    contentType: "application/json;odata=verbose",
    method: "GET",
```

```

headers: { "accept": "application/json;odata=verbose" },
success: function (data) {
    console.log(data.d.Id);
},
error: function(error){
    console.log(error);
}
});
}

```

Obtention d'éléments de liste avec des colonnes de recherche

Parfois, vous pouvez avoir une structure de liste qui ressemble à ceci:

Tableau de liste d'animaux

prénom	Type	La description
Titre	Chaîne (texte)	Nom de l'animal
Âge	Nombre	Quel âge a l'animal
Valeur	Devise	Valeur de l'animal
Type	<i>Recherche (tableau des types d'animaux)</i>	Champ de recherche (Donne la liste déroulante des choix de la Table des types d'animaux)

Tableau des types d'animaux

prénom	Type	La description
Titre	Chaîne (texte)	Nom de l'espèce / du type d'animal (ex. Porc)
NumLegs	Nombre	Nombre de pattes sur l'animal

Probablement pas l'exemple le plus grave mais le problème est toujours valable. Lorsque vous utilisez la requête habituelle pour extraire des valeurs de la liste SharePoint, pour le `Type` de l'animal, vous ne récupérez qu'un champ appelé `TypeId` dans la réponse JSON. Afin d'élargir ces éléments en un seul appel AJAX, un balisage supplémentaire est requis dans les paramètres de l'URL.

Cet exemple s'applique à plus que des colonnes de recherche. Lorsque vous utilisez des colonnes de `People/Groups`, ce ne sont que des recherches, vous pouvez donc facilement extraire des éléments tels que `Title`, `Email` et autres.

Exemple de code

Remarque importante : Lorsque vous définissez les champs que vous souhaitez récupérer à partir des colonnes de recherche, vous devez préfixer le nom du champ avec le nom du champ de recherche dans la table d'origine. Par exemple, si vous souhaitez récupérer l'attribut `NumLegs` dans la colonne de recherche, vous devez taper `Type/NumLegs` .

JavaScript

```
// baseUrl: The url of the site (ex. https://www.contoso.com/sites/animals)
// listTitle: The name of the list you want to query
// selectFields: the specific fields you want to get back
// expandFields: the name of the fields that need to be pulled from lookup tables
// callback: the name of the callback function on success
function getItem(webUrl, listTitle, selectFields, expandFields, callback) {
    var endpointUrl = webUrl + "/_api/web/lists/getbytitle('" + listTitle + "')/items";
    endpointUrl += '?$select=' + selectFields.join(",");
    endpointUrl += '&$expand=' + expandFields.join(",");
    return executeRequest(endpointUrl, 'GET', callback);
}

function executeRequest(url, method, callback, headers, payload)
{
    if (typeof headers == 'undefined') {
        headers = {};
    }
    headers["Accept"] = "application/json;odata=verbose";
    if (method == "POST") {
        headers["X-RequestDigest"] = $("#__REQUESTDIGEST").val();
    }

    var ajaxOptions =
    {
        url: url,
        type: method,
        contentType: "application/json;odata=verbose",
        headers: headers,
        success: function (data) { callback(data) }
    };
    if (method == "POST") {
        ajaxOptions.data = JSON.stringify(payload);
    }

    return $.ajax(ajaxOptions);
}

// Setup the ajax request by setting all of the arguments to the getItem function
function getAnimals() {
    var url = "https://www.contoso.com/sites/animals";
    var listTitle = "AnimalListing";

    var selectFields = [
        "Title",
        "Age",
        "Value",
        "Type/Title",
        "Type/NumLegs"
    ]
}
```



```

];

var expandFields = [
    "Type/Title",
    "Type/NumLegs"
];

getItems(url, listTitle, selectFields, expandFields, processAnimals);
}

// Callback function
// data: returns the data given by SharePoint
function processAnimals(data) {
    console.log(data);
    // Process data here
}

// Start the entire process
getAnimals();

```

Ajout de sélections à un champ de recherche à valeurs multiples

Cet exemple suppose que votre colonne de recherche s'appelle `MultiLookupColumnName` et que vous souhaitez définir votre champ de recherche multiple pour rechercher les éléments dont les ID 1 et 2.

Utiliser jQuery AJAX

2010

```

var listName = "YourListName";
var lookupList = "LookupListName";
var idOfItemToUpdate = 1;
var url = "/server/site/_vti_bin/ListData.svc/"+listName+"("+idOfItemToUpdate+)";
var data = JSON.stringify({
    MultiLookupColumnName:[
        {__metadata:{uri:"http://yoursiteurl/_vti_bin/ListData.svc/"+lookupList+"(1)"}}},
        {__metadata:{uri:"http://yoursiteurl/_vti_bin/ListData.svc/"+lookupList+"(2)"}}
    ]
});
$.ajax({
    method: 'POST',
    url: url,
    contentType: 'application/json',
    headers: {
        "X-HTTP-Method" : "MERGE",
        "If-Match" : "*"
    },
    data: data
});

```

2013

```

var listGuid = "id-of-list-to-update"; // use list GUID here
var lookupGuid = "id-of-lookup-list"; // use lookup list GUID here
var idOfItemToUpdate = 1;
var url = "/server/site/_api/lists('"+ listGuid + "')/items("+ idOfItemToUpdate + ")";

```

```

var data = JSON.stringify({
  MultiLookupColumnName:[
    {__metadata:{uri:"http://yoursiteurl/_api/lists('" + lookupGuid + "')/items(1)"}}},
    {__metadata:{uri:"http://yoursiteurl/_api/lists('" + lookupGuid + "')/items(2)"}}
  ]
});
$.ajax({
  method: 'POST',
  url: url,
  contentType: 'application/json',
  headers: {
    "X-HTTP-Method" : "MERGE",
    "If-Match" : "*"
  },
  data: data
});

```

Utiliser XMLHttpRequest

2010

```

var listName = "YourListName";
var lookupList = "LookupListName";
var idOfItemToUpdate = 1;
var url = "/server/site/_vti_bin/ListData.svc/YourListName("+idOfItemToUpdate+)";
var data = JSON.stringify({
  MultiLookupColumnName:[
    {__metadata:{uri:"http://yoursiteurl/_vti_bin/ListData.svc/"+lookupList+"(1)"}}},
    {__metadata:{uri:"http://yoursiteurl/_vti_bin/ListData.svc/"+lookupList+"(2)"}}
  ]
});
var xhr = new XMLHttpRequest();
xhr.open("POST",url,true);
xhr.setRequestHeader("X-HTTP-Method", "MERGE");
xhr.setRequestHeader("If-Match", "*");
xhr.setRequestHeader("Content-Type","application/json");
xhr.send(data);

```

2013

```

var listGuid = "id-of-list-to-update";
var lookupGuid = "id-of-lookup-list";
var idOfItemToUpdate = 1;
var url = "/server/site/_api/lists('" + listGuid + "')/items("+ idOfItemToUpdate + ")";
var data = JSON.stringify({
  MultiLookupColumnName:[
    {__metadata:{uri:"http://yoursiteurl/_api/lists('" + lookupGuid + "')/items(1)"}}},
    {__metadata:{uri:"http://yoursiteurl/_api/lists('" + lookupGuid + "')/items(2)"}}
  ]
});
var xhr = new XMLHttpRequest();
xhr.open("POST",url,true);
xhr.setRequestHeader("X-HTTP-Method", "MERGE");
xhr.setRequestHeader("If-Match", "*");
xhr.setRequestHeader("Content-Type","application/json");
xhr.send(data);

```

Éléments de liste de pagination renvoyés par une requête

Pour simuler la pagination à l'aide de REST, vous pouvez effectuer les opérations suivantes:

1. Utilisez le paramètre `$skip=n` pour ignorer les `n` premières entrées en fonction du paramètre `$orderby`
2. Utilisez le paramètre `$top=n` pour renvoyer les `n` premières entrées en fonction des `$orderby` et `$skip`.

```
var endpointUrl = "/_api/lists('guid')/items"; // SP2010: "/_vti_bin/ListData.svc/ListName";
$.getJSON(
    endpointUrl + "?$orderby=Id&$top=1000",
    function(data) {
        processData(data); // you can do something with the results here
        var count = data.d.results.length;
        getNextBatch(count, processData, onComplete); // fetch next page
    }
);

function getNextBatch(totalSoFar, processResults, onCompleteCallback) {
    $.getJSON(
        endpointUrl + "?$orderby=Id&$skip="+totalSoFar+"&$top=1000",
        function(data) {
            var count = data.d.results.length;
            if(count > 0) {
                processResults(data); // do something with results
                getNextBatch(totalSoFar+count, callback); // fetch next page
            } else {
                onCompleteCallback();
            }
        }
    );
}
```

Récupérer un identifiant d'élément nouvellement créé dans la liste SharePoint

Cet exemple montre comment extraire un ID d'un élément nouvellement créé à l'aide de l'API REST SharePoint.

Remarque :

listName - Cette variable contient le nom de votre liste.

newItemBody - Ce sera votre corps de requête pour ajouter un nouvel élément dans la liste.

par exemple `var newItemBody = {__metadata: {'type': 'SP.Data.MyListNameItem'}, Titre: 'Some title value'};`

```
function CreateListItemWithDetails(listName, newItemBody) {

    var item = newItemBody;
    return $.ajax({
        url: _spPageContextInfo.siteAbsoluteUrl + "/_api/web/lists/getbytitle('" + listName +
        "')/items",
        type: "POST",
        contentType: "application/json;odata=verbose",
        data: JSON.stringify(item),
```

```

    headers: {
      "Accept": "application/json;odata=verbose",
      "X-RequestDigest": $("#__REQUESTDIGEST").val(),
      "content-Type": "application/json;odata=verbose"
    }
  });
}

CreateListItemWithDetails(listName, newItemBody)
  .then(function(data) {
    //success callback
    var NewlyCreatedItemId = data.d.ID;
  }, function(data) {
    //failure callback
  });

```

Comment effectuer des opérations CRUD à l'aide de l'interface REST SharePoint 2010

Créer

Pour effectuer une opération de création via REST, vous devez effectuer les actions suivantes:

Créez une requête HTTP en utilisant le verbe `POST`. Utilisez l'URL de service de la liste à laquelle vous souhaitez ajouter une entité en tant que cible pour le POST. Définissez le type de contenu sur `application/json`. Sérialisez les objets JSON qui représentent vos nouveaux éléments de liste en tant que chaîne et ajoutez cette valeur à l'exemple JavaScript du corps de la requête:

```

function createListItem(webUrl, listName, itemProperties, success, failure) {

  $.ajax({
    url: webUrl + "/_vti_bin/listdata.svc/" + listName,
    type: "POST",
    processData: false,
    contentType: "application/json;odata=verbose",
    data: JSON.stringify(itemProperties),
    headers: {
      "Accept": "application/json;odata=verbose"
    },
    success: function (data) {
      success(data.d);
    },
    error: function (data) {
      failure(data.responseJSON.error);
    }
  });
}

```

Usage

```

var taskProperties = {
  'TaskName': 'Order Approval',
  'AssignedToId': 12
};

```

```

createListItem('https://contoso.sharepoint.com/project/', 'Tasks', taskProperties, function(task) {

    console.log('Task' + task.TaskName + ' has been created');
},
function(error) {
    console.log(JSON.stringify(error));
}
);

```

Lis

Pour effectuer une opération de lecture via REST, vous devez effectuer les actions suivantes:

Créez une requête HTTP en utilisant le verbe `GET`. Utilisez l'URL de service de l'élément de liste auquel vous souhaitez ajouter une entité en tant que cible pour le GET. Définissez le type de contenu sur `application/json`. Exemple JavaScript:

```

function getListItemById(webUrl, listName, itemId, success, failure) {
    var url = webUrl + "/_vti_bin/listdata.svc/" + listName + "(" + itemId + ")";
    $.ajax({
        url: url,
        method: "GET",
        headers: { "Accept": "application/json; odata=verbose" },
        success: function (data) {
            success(data.d);
        },
        error: function (data) {
            failure(data.responseJSON.error);
        }
    });
}

```

Usage

```

getItemById('https://contoso.sharepoint.com/project/', 'Tasks', 2, function(taskItem) {
    console.log(taskItem.TaskName);
},
function(error) {
    console.log(JSON.stringify(error));
}
);

```

Mettre à jour

Pour mettre à jour une entité existante, vous devez effectuer les actions suivantes:

Créez une requête HTTP en utilisant le verbe `POST`. Ajoutez un `X-HTTP-Method` tête `X-HTTP-Method` avec une valeur de `MERGE`. Utilisez l'URL de service de l'élément de liste que vous souhaitez mettre à jour en tant que cible pour le POST Ajoutez un en `If-Match` tête `If-Match` avec une valeur de l'ETag d'origine de l'entité, ou `*`. Exemple JavaScript:

```

function updateListItem(webUrl, listName, itemId, itemProperties, success, failure)
{
    getItemById(webUrl, listName, itemId, function (item) {

```

```

$.ajax({
  type: 'POST',
  url: item.__metadata.uri,
  contentType: 'application/json',
  processData: false,
  headers: {
    "Accept": "application/json;odata=verbose",
    "X-HTTP-Method": "MERGE",
    "If-Match": item.__metadata.etag
  },
  data: Sys.Serialization.JavaScriptSerializer.serialize(itemProperties),
  success: function (data) {
    success(data);
  },
  error: function (data) {
    failure(data);
  }
});

},
function(error){
  failure(error);
});
}

```

Usage

```

var taskProperties = {
  'TaskName': 'Approval',
  'AssignedToId': 12
};

updateListItem('https://contoso.sharepoint.com/project/', 'Tasks', 2, taskProperties, function (item) {

  console.log('Task has been updated');
},
function(error){
  console.log(JSON.stringify(error));
}
);

```

Effacer

Pour supprimer une entité, vous devez effectuer les actions suivantes:

Créez une requête HTTP en utilisant le verbe `POST`. Ajoutez un `X-HTTP-Method` tête `X-HTTP-Method` avec une valeur `DELETE`. Utilisez l'URL de service de l'élément de liste que vous souhaitez mettre à jour en tant que cible pour le `POST`. Ajoutez un en `If-Match` tête `If-Match` avec une valeur de l'ETag d'origine de l'entité. Exemple JavaScript:

```

function deleteListItem(webUrl, listName, itemId, success, failure) {
  getListItemById(webUrl, listName, itemId, function (item) {
    $.ajax({
      url: item.__metadata.uri,

```

```

    type: "POST",
    headers: {
      "Accept": "application/json;odata=verbose",
      "X-Http-Method": "DELETE",
      "If-Match": item.__metadata.etag
    },
    success: function (data) {
      success();
    },
    error: function (data) {
      failure(data.responseJSON.error);
    }
  });
},
function (error) {
  failure(error);
});
}

```

Usage

```

deleteListItem('https://contoso.sharepoint.com/project/', 'Tasks', 3, function() {
  console.log('Task has been deleted');
},
function(error) {
  console.log(JSON.stringify(error));
}
);

```

Lire Services REST en ligne: <https://riptutorial.com/fr/sharepoint/topic/3045/services-rest>

Chapitre 7: Travailler avec des boîtes de dialogue modales avec JavaScript

Syntaxe

- `var options = SP.UI.$ create_DialogOptions ();`
- `var modalDialog = SP.UI.ModalDialog.showModalDialog (options);`

Paramètres

options Propriété	La description
Titre	Une chaîne contenant le titre de la boîte de dialogue
URL	Chaîne contenant l'URL de la page qui apparaît dans la boîte de dialogue. L' URL ou le HTML doit être spécifié. L'URL a priorité sur HTML .
html	Un élément HTML à afficher dans la boîte de dialogue.
X	Le décalage en x de la boîte de dialogue sous la forme d'un nombre entier.
y	Le décalage en y du dialogue sous la forme d'un nombre entier.
largeur	La largeur de la boîte de dialogue sous la forme d'un nombre entier. Si non spécifié et autosize est faux , la largeur est définie sur 768px
la taille	La hauteur de la boîte de dialogue sous la forme d'un nombre entier. Si non spécifié et que la taille automatique est fausse , la hauteur est définie sur 576px
permettreMaximiser	Valeur booléenne indiquant si le bouton Maximiser doit être affiché.
showMaximized	Valeur booléenne spécifiant si la boîte de dialogue s'ouvre agrandie.
montrerFermer	Valeur booléenne spécifiant si le bouton Fermer apparaît dans la boîte de dialogue.
autoSize	Valeur booléenne qui spécifie si la plateforme de dialogue gère automatiquement le dimensionnement des dialogues.

options Propriété	La description
dialogReturnValueCallback	Un pointeur de fonction qui spécifie la fonction de rappel de retour. Fonction prend deux paramètres: un <i>dialogResult</i> de type SP.UI.DialogResult Enumeration et un objet <i>returnValue</i> qui contient les données renvoyées par la boîte de dialogue.
args	Objet contenant des données transmises à la boîte de dialogue.

Remarques

L' `SP.UI.ModalDialog` noms `SP.UI.ModalDialog` été introduit dans le [modèle d'objet JavaScript](#) avec SharePoint 2010 et est disponible dans les versions ultérieures de SharePoint 2013, Office365 et 2016.

Matériaux de référence supplémentaires:

- [Référence MSDN pour SP.UI.ModalDialog.showModalDialog \(options\)](#)
- [Référence MSDN pour l'énumération SP.UI.DialogResult](#)

Exemples

Effectuer une action lorsqu'une boîte de dialogue est fermée

```
SP.SOD.executeOrDelayUntilScriptLoaded(showDialog, "sp.js");

function showDialog() {
    var options = SP.UI.$create_DialogOptions();
    options.url = "/mySite/lists/myList/NewForm.aspx";
    options.dialogReturnValueCallback = myCallBackFunction;
    SP.UI.ModalDialog.showModalDialog(options);
    function myCallBackFunction(result, data) {
        switch(result) {
            case SP.UI.DialogResult.invalid:
                alert("The dialog result was invalid");
                break;
            case SP.UI.DialogResult.cancel:
                alert("You clicked cancel or close");
                break;
            case SP.UI.DialogResult.OK:
                alert("You clicked OK, creating an item in the list.");
                break;
        }
    }
}
```

Afficher une page existante dans une boîte de dialogue

```
SP.SOD.executeOrDelayUntilScriptLoaded(showDialog, "sp.js");

function showDialog() {
```

```
SP.UI.ModalDialog.showModalDialog(  
    { url: "/org/it/web/wik/Lists/ExampleCode/DispForm.aspx?ID=6" }  
);  
}
```

Afficher une boîte de dialogue personnalisée

```
SP.SOD.executeOrDelayUntilScriptLoaded(showDialog, "sp.js");  
  
function showDialog(){  
    var dialogOptions = SP.UI.$create_DialogOptions();  
    dialogOptions.title = "Your Title Here!";  
    var dummyElement = document.createElement("div");  
    dummyElement.style.textAlign = "center";  
    dummyElement.appendChild(document.createElement("br"));  
    dummyElement.appendChild(document.createTextNode("Some beautifully crafted text."));  
    dummyElement.appendChild(document.createElement("br"));  
    dialogOptions.html = dummyElement;  
    SP.UI.ModalDialog.showModalDialog(dialogOptions);  
}
```

Lire [Travailler avec des boîtes de dialogue modales avec JavaScript en ligne](https://riptutorial.com/fr/sharepoint/topic/6868/travailler-avec-des-boites-de-dialogue-modales-avec-javascript):

<https://riptutorial.com/fr/sharepoint/topic/6868/travailler-avec-des-boites-de-dialogue-modales-avec-javascript>

Chapitre 8: Utilisation du modèle d'objet client JavaScript (JSOM)

Remarques

Contexte

Le modèle d'objet JavaScript a été introduit dans SharePoint 2010. Il expose côté client de nombreux objets auparavant uniquement accessibles via du code côté serveur ou des services Web dédiés.

Intégration de JavaScript dans les pages SharePoint

Dans SharePoint 2013, vous pouvez placer votre code JavaScript dans un composant WebPart Éditeur de scripts.

Dans SharePoint 2010, vous pouvez utiliser la propriété "lien de contenu" d'un composant WebPart Éditeur de contenu pour créer un lien vers un fichier HTML contenant votre script incorporé.

Référence d'objet

Les constructeurs, méthodes et propriétés de tous les objets trouvés dans l'espace de noms `SP` sont documentés dans la référence du modèle d'objet client SharePoint 2013 [ici](#).

La référence du modèle d'objet client SharePoint 2010 JavaScript est disponible [ici](#).

Modèle de programmation asynchrone de JSOM

Lorsque vous utilisez le modèle d'objet client JavaScript, le code utilise généralement le modèle suivant:

1. Obtenir un objet `ClientContext`.
2. Utilisez l'objet `ClientContext` pour récupérer des objets représentant des entités dans le modèle d'objet SharePoint, telles que des listes, des dossiers, des vues.
3. Mettre en file d'attente les instructions à effectuer sur les objets. Ces instructions ne sont pas encore transmises au serveur.
4. Utilisez la fonction de `load` pour indiquer à `ClientContext` les informations que vous souhaitez recevoir du serveur.
5. Appelez la fonction `executeQueryAsync` l'objet `ClientContext` pour envoyer les instructions en file d'attente au serveur, en transmettant deux fonctions de rappel pour `ClientContext` s'exécutent en cas de succès ou d'échec.
6. Dans la fonction de rappel, travaillez avec les résultats renvoyés par le serveur.

Des alternatives

Les alternatives côté client au JSOM incluent les services Web SharePoint, les [noeuds finaux REST](#) et le [modèle d'objet client .NET](#) .

Exemples

Obtenir des types de contenu de bibliothèque à l'aide du nom de la bibliothèque

```
function getContentTypes(site_url,name_of_the_library){
    var ctx = new SP.ClientContext(site_url);
    var web = ctx.get_web();
    list = web.get_lists().getByTitle(name_of_the_library);

    // You can include any property of the SP.ContentType object (sp.js), for this example we
    are just getting the name
    ctx.load(list, 'ContentTypes.Include(Name)');
    ctx.executeQueryAsync(onQuerySucceeded, onQueryFailed);
}

function onQuerySucceeded(sender, args) {
    // var list is the one that we used in function "getContentTypes"
    var contentTypesEnumerator = (list.get_contentTypes()).getEnumerator();

    while (contentTypesEnumerator.moveNext()) {
        var contentType = contentTypesEnumerator.get_current();
        alert(contentType.get_name());
    }
}

function onQueryFailed(sender, args) {
    alert('Request failed. ' + args.get_message() + '\n' + args.get_stackTrace());
}
```

Supprimer un élément dans une liste

```
SP.SOD.executeOrDelayUntilScriptLoaded( function(){ deleteItem(1); }, "sp.js");

function deleteItem(id){
    var clientContext = new SP.ClientContext();
    var list = clientContext.get_web().get_lists().getByTitle("List Title");
    var item = list.getItemById(id);
    item.deleteObject();
    clientContext.executeQueryAsync(function(){
        alert("Item #"+id+" deleted successfully!");
    },function(sender,args){alert(args.get_message());});
}
```

Création d'éléments ou de dossiers

Création d'éléments de liste

```
SP.SOD.executeOrDelayUntilScriptLoaded(createItem,"sp.js");
```

```

function createItem(){
    var clientContext = new SP.ClientContext();
    var list = clientContext.get_web().get_lists().getByTitle("List Title");
    var newItem = list.addItem();
    newItem.set_item("Title", "Example Title");
    newItem.update();
    clientContext.load(newItem); // only needed to retrieve info from newly created item
    clientContext.executeQueryAsync(function(){
        var itemId = newItem.get_item("ID");
        alert("Item #"+itemId+" Created Successfully!");
    },function(sender, args){
        alert(args.get_message());
    });
}

```

L'exemple ci-dessus montre qu'un élément de liste est créé en procédant comme suit:

1. Appelez la méthode `addItem` d'un objet liste pour obtenir un objet élément
2. Appelez la méthode `set_item` sur l'objet d'élément de liste résultant pour définir chaque valeur de champ comme vous le souhaitez
3. Appelez la méthode `update` sur l'objet d'élément de liste pour indiquer que les modifications doivent être validées
4. Appelez la méthode `executeQueryAsync` de l'objet de contexte client pour exécuter les instructions en file d'attente

Notez que vous n'avez **pas** besoin de transmettre le nouvel objet d'élément à la méthode de `load` du contexte client pour créer l'élément. Cette étape n'est nécessaire que si vous souhaitez récupérer l'une des valeurs de champ de l'élément à partir du serveur.

Création de dossiers

La création d'un dossier est similaire à l'ajout d'un élément à une liste. La différence est qu'il faut d'abord créer un `ListItemCreationInformation` objet et définir sa `underlyingObjectType` propriété à `SP.FileSystemObjectType.folder`, et sa `leafName` propriété au nom de votre choix du nouveau dossier.

L'objet est ensuite transmis en tant que paramètre dans la méthode `addItem` de la bibliothèque pour créer le dossier.

```

// ...
var itemCreateInfo = new SP.ListItemCreationInformation();
itemCreateInfo.set_underlyingObjectType(SP.FileSystemObjectType.folder);
itemCreateInfo.set_leafName(folderName);
var newItem = list.addItem(itemCreateInfo);
// ...

```

Pour valider la modification, `ClientContext` la méthode `executeQueryAsync` de l'objet `ClientContext` via lequel la bibliothèque a été accédée.

L'exemple complet ci-dessous crée un dossier avec un nom basé sur l'horodatage actuel, puis

ouvre ce dossier dans une boîte de dialogue modale.

```
SP.SOD.executeOrDelayUntilScriptLoaded(createFolder, "sp.js");

function createFolder() {
    var now = new Date();
    var timeStamp = now.getYear() + "-" + (now.getMonth()+1) + "-" + now.getDate()
        + "T" + now.getHours()+"_"+now.getMinutes()+"
"+now.getSeconds()+"_"+now.getMilliseconds();
    var clientContext = new SP.ClientContext();
    var list = clientContext.get_web().get_lists().getByTitle("Library Title");
    var itemCreateInfo = new SP.ListItemCreationInformation();
    itemCreateInfo.set_underlyingObjectType(SP.FileSystemObjectType.folder);
    itemCreateInfo.set_leafName(timeStamp);
    var newItem = list.addItem(itemCreateInfo);
    newItem.update();
    clientContext.load(newItem);
    var rootFolder = list.get_rootFolder(); // Note: use a list's root folder to determine its
server relative URL
    clientContext.load(rootFolder);
    clientContext.executeQueryAsync(function() {
        var itemId = newItem.get_item("ID");
        var name = newItem.get_item("FileLeafRef");
        SP.UI.ModalDialog.showModalDialog(
            {
                title: "Folder \""+name+"\" (#"+itemId+") Created Successfully!",
                url: rootFolder.get_serverRelativeUrl() + "/" + name
            }
        );
    }, function(sender, args) {alert(args.get_message());});
}
```

Obtenir les informations d'utilisateur actuelles

```
SP.SOD.executeOrDelayUntilScriptLoaded(showUserInfo, "sp.js");

function showUserInfo() {
    var clientContext = new SP.ClientContext();
    var user = clientContext.get_web().get_currentUser();
    clientContext.load(user);
    clientContext.executeQueryAsync(function() {
        var details = "ID: "+user.get_id()+"\n"+
            "Title: "+user.get_title()+"\n"+
            "Login: "+user.get_loginName()+"\n"+
            "Email: "+user.get_email();
        alert(details);
    }, function(sender, args) {alert(args.get_message());});
}
```

Obtenir un élément de liste par ID

```
SP.SOD.executeOrDelayUntilScriptLoaded(myFunction, "sp.js");

function myFunction() {
    var clientContext = new SP.ClientContext();
    var list = clientContext.get_web().get_lists().getByTitle("List Title");
    var item = list.getItemById(1); // get item with ID == 1
}
```

```

clientContext.load(item);
clientContext.executeQueryAsync(
    function(){ // onSuccess
        var title = item.get_item("Title");
        alert(title);
    },
    function(sender,args){ // onError
        alert(args.get_message());
    }
);
}

```

Obtenir des éléments de liste par la requête CAML

Exemple de base

Utilisez la méthode `set_viewXml` de l'objet `SP.CamlQuery` pour spécifier une requête CAML permettant de récupérer des éléments.

```

SP.SOD.executeOrDelayUntilScriptLoaded(showListItems, "core.js");

function showListItems(){
    var clientContext = new SP.ClientContext();
    var list = clientContext.get_web().get_lists().getByTitle("List Title");
    var camlQuery = new SP.CamlQuery();
    camlQuery.set_viewXml(
        "<View><Query>" +
        "  <Where>" +
        "    <Eq><FieldRef Name=\"Title\"/><Value Type=\"Text\">Value</Value></Eq>" +
        "  </Where>" +
        "  <OrderBy><FieldRef Name=\"Modified\" Ascending=\"FALSE\"/></OrderBy>" +
        "</Query>" +
        // "<RowLimit>5000</RowLimit>" +
        "</View>");
    var items = list.getItems(camlQuery);
    clientContext.load(items);
    clientContext.executeQueryAsync(function(){
        var itemArray = [];
        var itemEnumerator = items.getEnumerator();
        while(itemEnumerator.moveNext()){
            var item = itemEnumerator.get_current();
            var id = item.get_item("ID");
            var title = item.get_item("Title");
            itemArray.push(id + ": " + title);
        }
        alert("ID: Title\n"+itemArray.join("\n"));
    },function(sender,args){alert(args.get_message());});
}

```

Recherche des résultats d'une requête CAML

Vous pouvez tirer parti de l'élément `RowLimit` dans une requête CAML pour extraire uniquement un sous-ensemble de résultats avec chaque requête.

Utilisez la méthode `get_listItemCollectionPosition` d'une collection d'éléments de liste pour extraire la position actuelle, puis utilisez cette valeur comme paramètre dans la méthode `set_listItemCollectionPosition` un objet `set_listItemCollectionPosition` pour extraire le lot de résultats suivant.

```
SP.SOD.executeOrDelayUntilScriptLoaded(showListItems, "sp.js");

function showListItems(){
    var itemArray = [];
    var clientContext = new SP.ClientContext();
    var list = clientContext.get_web().get_lists().getByTitle("List Title");
    var viewXml =
        "<View><Query>" +
            "<OrderBy><FieldRef Name=\"Modified\" Ascending=\"FALSE\"/></OrderBy>" +
        "</Query>" +
            "<RowLimit>1</RowLimit>" +
        "</View>";
    var camlQuery = new SP.CamlQuery();
    camlQuery.set_viewXml(viewXml);
    var items = list.getItems(camlQuery);
    clientContext.load(items);
    clientContext.executeQueryAsync(loadResults, showError);

    function loadResults(){
        var resultsFound = false;
        var itemEnumerator = items.getEnumerator();
        while(itemEnumerator.moveNext()){
            var item = itemEnumerator.get_current();
            var id = item.get_item("ID");
            var title = item.get_item("Title");
            itemArray.push(id + ": " + title);
        }
        var pos = items.get_listItemCollectionPosition();// <- get position
        if(pos !== null){ // <-- position is null when no more results are returned
            if(confirm("Results so far: \nID: Title\n"+itemArray.join("\n"))){
                camlQuery = new SP.CamlQuery();
                camlQuery.set_listItemCollectionPosition(pos);// <- set position for next
                batch
                    camlQuery.set_viewXml(viewXml);
                    items = list.getItems(camlQuery);
                    clientContext.load(items);
                    clientContext.executeQueryAsync(loadResults, showError);
            }
        }else{
            alert("Total Results: \nID: Title\n"+itemArray.join("\n")); // <- display when no
            more results
        }
    }
    function showError(sender, args){
        alert(args.get_message());
    }
}
```

Lire Utilisation du modèle d'objet client JavaScript (JSOM) en ligne:

<https://riptutorial.com/fr/sharepoint/topic/1316/utilisation-du-modele-d-objet-client-javascript--jsom->

Chapitre 9: Utilisation du modèle d'objet côté client géré (CSOM)

Remarques

- La plupart des exemples proviennent de [MSDN](#) .
- Pour créer une application client gérée .NET qui utilise le modèle objet client, vous devez définir des références à deux DLL de bibliothèque client: Microsoft.SharePoint.Client.dll et Microsoft.SharePoint.Client.Runtime.dll. Vous le trouverez dans le dossier % ProgramFiles% \ Fichiers communs \ Microsoft Shared \ extensions de serveur Web \ 16 \ ISAPI ou votre serveur SharePoint.
- ou Installez le package Microsoft.SharePointOnline.CSOM NuGet, qui fonctionnera "sur prem" ainsi que dans SP O365.
- La plupart des propriétés sont des propriétés de valeur et, avant d'y accéder, vous devez appeler explicitement clientContext.Load () et clientContext.ExecuteQuery (). Plus d'informations ici: [Call Load et ExecuteQuery avant d'accéder aux propriétés de valeur](#)

Exemples

Bonjour tout le monde (obtenir le titre du site)

Toutes les versions de SharePoint sont basées sur les sites (SPSite (SSOM) ou site (CSOM)) et les sites Web (SPWeb (SSOM) ou Web (CSOM)). Un site n'est pas rendu dans l'interface utilisateur bien qu'il contienne des métadonnées et des fonctionnalités appliquées à ses enfants. Un site Web est le composant de base qui rend une interface utilisateur à l'utilisateur accédant au site. Tous les sites ont un site Web racine qui contient des informations et / ou des métadonnées comme les bibliothèques de documents. Cet exemple montre un appel de base pour récupérer le site Web situé sur le serveur `MyServer` sous les `sites` chemin virtuel.

```
using System;
using Microsoft.SharePoint.Client;

namespace Microsoft.SDK.SharePointServices.Samples
{
    class RetrieveWebsite
    {
        static void Main()
        {
            // This is the URL of the target web we are interested in.
            string siteUrl = "http://MyServer/sites/MySiteCollection";
            // The client context is allows us to queue up requests for the server
            // Note that the context can only ask questions about the site it is created for
            using (ClientContext clientContext = new ClientContext(siteUrl))
            {
                // To make it easier to read the code, pull the target web
                // context off of the client context and store in a variable
                Web oWebsite = clientContext.Web;
            }
        }
    }
}
```

```

        // Tell the client context we want to request information about the
        // Web from the server
        clientContext.Load(oWebsite);
        // After we are done creating the batch of information we need from the sever,
        // request the data from SharePoint
        clientContext.ExecuteQuery();
        // Print the results of the query
        Console.WriteLine("Title: {0} Description: {1}", oWebsite.Title,
oWebsite.Description);
    }
}
}
}

```

Web Récupération des propriétés d'un site Web

```

ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;
clientContext.Load(oWebsite);
clientContext.ExecuteQuery();
Console.WriteLine("Title: {0} Description: {1}", oWebsite.Title, oWebsite.Description);

```

Web Récupérer uniquement les propriétés spécifiées d'un site Web

```

ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;
clientContext.Load(
    oWebsite,
    website => website.Title,
    website => website.Created);
clientContext.ExecuteQuery();
Console.WriteLine("Title: {0} Created: {1}", oWebsite.Title, oWebsite.Created);

```

Web Mise à jour du titre et de la description d'un site Web

```

ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = context.Web;
oWebsite.Title = "Updated Web Site";
oWebsite.Description = "This is an updated Web site.";
oWebsite.Update();
clientContext.ExecuteQuery();

```

Web Création d'un site Web

```

string siteUrl = "http://MyServer/sites/MySiteCollection";
string blogDescription = "A new blog Web site.";
int blogLanguage = 1033;
string blogTitle = "Blog Web Site";
string blogUrl = "blogwebsite";
bool blogPermissions = false;
string webTemplate = "BLOG#0";

ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;

```

```

WebCreationInformation webCreateInfo = new WebCreationInformation();
webCreateInfo.Description = blogDescription;
webCreateInfo.Language = blogLanguage;
webCreateInfo.Title = blogTitle;
webCreateInfo.Url = blogUrl;
webCreateInfo.UseSamePermissionsAsParentSite = blogPermissions;
webCreateInfo.WebTemplate = webTemplate;

Web oNewWebsite = oWebsite.Webs.Add(webCreateInfo);

clientContext.Load(
    oNewWebsite,
    website => website.ServerRelativeUrl,
    website => website.Created);

clientContext.ExecuteQuery();

Console.WriteLine("Server-relative Url: {0} Created: {1}", oNewWebsite.ServerRelativeUrl,
oNewWebsite.Created);

```

Liste. Récupération de toutes les propriétés de toutes les listes d'un site Web

```

ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;
ListCollection collList = oWebsite.Lists;

clientContext.Load(collList);

clientContext.ExecuteQuery();

foreach (List oList in collList)
{
    Console.WriteLine("Title: {0} Created: {1}", oList.Title, oList.Created.ToString());
}

```

Liste. Récupérer uniquement les propriétés spécifiées des listes

```

ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;
ListCollection collList = oWebsite.Lists;

clientContext.Load(
    collList,
    lists => lists.Include(
        list => list.Title,
        list => list.Id));

clientContext.ExecuteQuery();

foreach (List oList in collList)
{
    Console.WriteLine("Title: {0} ID: {1}", oList.Title, oList.Id.ToString("D"));
}

```

Liste. Stockage des listes récupérées dans une collection

```

ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;
ListCollection collList = oWebsite.Lists;

IEnumerable<List> resultCollection = clientContext.LoadQuery(
    collList.Include(
        list=>list.Title,
        list=>list.Id));

clientContext.ExecuteQuery();

foreach (List oList in resultCollection)
{
    Console.WriteLine("Title: {0} ID: {1}", oList.Title, oList.Id.ToString("D"));
}

```

Liste. Récupération des champs de liste d'un site Web

```

ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;
ListCollection collList = oWebsite.Lists;

IEnumerable<SP.List> listInfo = clientContext.LoadQuery(
    collList.Include(
        list => list.Title,
        list => list.Fields.Include(
            field => field.Title,
            field => field.InternalName)));

clientContext.ExecuteQuery();

foreach (SP.List oList in listInfo)
{
    FieldCollection collField = oList.Fields;

    foreach (SP.Field oField in collField)
    {
        Regex regEx = new Regex("name", RegexOptions.IgnoreCase);

        if (regEx.IsMatch(oField.InternalName))
        {
            Console.WriteLine("List: {0} \n\t Field Title: {1} \n\t Field Internal Name: {2}",
                oList.Title, oField.Title, oField.InternalName);
        }
    }
}

```

Liste. Créer et mettre à jour une liste

```

ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;

ListCreationInformation listCreationInfo = new ListCreationInformation();
listCreationInfo.Title = "My Announcements List";
listCreationInfo.TemplateType = (int)ListTemplateType.Announcements;

```

```
List oList = oWebsite.Lists.Add(listCreationInfo);  
  
clientContext.ExecuteQuery();
```

Liste. Ajouter un champ à une liste

```
ClientContext clientContext = new ClientContext(siteUrl);  
  
SP.List oList = clientContext.Web.Lists.GetByTitle("Announcements");  
  
SP.Field oField = oList.Fields.AddFieldAsXml("<Field DisplayName='MyField' Type='Number' />",  
    true, AddFieldOptions.DefaultValue);  
  
SP.FieldNumber fieldNumber = clientContext.CastTo<FieldNumber>(oField);  
fieldNumber.MaximumValue = 100;  
fieldNumber.MinimumValue = 35;  
  
fieldNumber.Update();  
  
clientContext.ExecuteQuery();
```

Liste. Supprimer une liste

```
ClientContext clientContext = new ClientContext(siteUrl);  
Web oWebsite = clientContext.Web;  
  
List oList = oWebsite.Lists.GetByTitle("My Announcements List");  
  
oList.DeleteObject();  
  
clientContext.ExecuteQuery();
```

Article. Récupérer des éléments d'une liste

```
ClientContext clientContext = new ClientContext(siteUrl);  
SP.List oList = clientContext.Web.Lists.GetByTitle("Announcements");  
  
CamlQuery camlQuery = new CamlQuery();  
camlQuery.ViewXml = "<View><Query><Where><Geq><FieldRef Name='ID' />" +  
    "<Value Type='Number'>10</Value></Geq></Where></Query><RowLimit>100</RowLimit></View>";  
ListItemCollection collListItem = oList.GetItems(camlQuery);  
  
clientContext.Load(collListItem);  
  
clientContext.ExecuteQuery();  
  
foreach (ListItem oListItem in collListItem)  
{  
    Console.WriteLine("ID: {0} \nTitle: {1} \nBody: {2}", oListItem.Id, oListItem["Title"],  
        oListItem["Body"]);  
}
```

Article. Récupération d'éléments (à l'aide de la méthode Include)

Cet exemple montre comment extraire des éléments du serveur et obtenir des propriétés plus profondes pour chaque élément de la liste. Par défaut, le serveur ne renvoie que la quantité minimale de données pour représenter l'objet. Il appartient à l'appelant de demander des informations supplémentaires au serveur.

```
ClientContext clientContext = new ClientContext(siteUrl);
List oList = clientContext.Web.Lists.GetByTitle("Announcements");

CamlQuery camlQuery = new CamlQuery();
camlQuery.ViewXml = "<View><RowLimit>100</RowLimit></View>";

ListItemCollection collListItem = oList.GetItems(camlQuery);

// The first line of this request indicates the list item collection to load from the server
// The second line uses a lambda to request that from the server
// also include additional properties in the response
// The third though fifth lines are the properties being requested from the server
clientContext.Load(collListItem,
    items => items.Include(
        item => item.Id,
        item => item.DisplayName,
        item => item.HasUniqueRoleAssignments));

clientContext.ExecuteQuery();

foreach (ListItem oListItem in collListItem)
{
    Console.WriteLine("ID: {0} \nDisplay name: {1} \nUnique role assignments: {2}",
        oListItem.Id, oListItem.DisplayName, oListItem.HasUniqueRoleAssignments);
}
```

Article. Récupération de champs spécifiques à partir d'un nombre spécifié d'éléments

```
ClientContext clientContext = new ClientContext(siteUrl);
SP.List oList = clientContext.Web.Lists.GetByTitle("Announcements");

CamlQuery camlQuery = new CamlQuery();
ListItemCollection collListItem = oList.GetItems(camlQuery);

clientContext.Load(
    collListItem,
    items => items.Take(5).Include(
        item => item["Title"],
        item => item["Body"]));

clientContext.ExecuteQuery();

foreach (ListItem oListItem in collListItem)
{
    Console.WriteLine("Title: {0} \nBody: {1}\n", oListItem["Title"], oListItem["Body"]);
}
```

Article. Récupération d'éléments de toutes les listes d'un site Web

```

ClientContext clientContext = new ClientContext(siteUrl);
ListCollection collList = clientContext.Web.Lists;

clientContext.Load(
    collList,
    lists => lists.Where(
        list => list.Hidden == false).Include(
        list => list.Title,
        list => list.Items.Take(10)));

clientContext.ExecuteQuery();

foreach (SP.List oList in clientContext.Web.Lists)
{
    string listTitle = oList.Title;
    int itemCount = oList.Items.Count;

    Console.WriteLine("List {0} returned with {1} items", listTitle, itemCount);
}

```

Article. Récupération d'éléments à l'aide de la position de collecte d'éléments de liste

```

ClientContext clientContext = new ClientContext(siteUrl);
SP.List oList = clientContext.Web.Lists.GetByTitle("Announcements");

ListItemCollectionPosition itemPosition = null;

while (true)
{
    CamlQuery camlQuery = new CamlQuery();

    camlQuery.ListItemCollectionPosition = itemPosition;

    camlQuery.ViewXml = "<View><ViewFields><FieldRef Name='ID' />" +
        "<FieldRef Name='Title' /><FieldRef Name='Body' />" +
        "</ViewFields><RowLimit>5</RowLimit></View>";

    ListItemCollection collListItem = oList.GetItems(camlQuery);

    clientContext.Load(collListItem);

    clientContext.ExecuteQuery();

    itemPosition = collListItem.ListItemCollectionPosition;

    foreach (ListItem oListItem in collListItem)
    {
        Console.WriteLine("Title: {0}: \nBody: {1}", oListItem["Title"], oListItem["Body"]);
    }

    if (itemPosition == null)
    {
        break;
    }

    Console.WriteLine("\n" + itemPosition.PagingInfo + "\n");
}

```

Article. Créer un élément de liste

Lors de la création d'un nouvel élément de liste, ses champs peuvent être définis à l'aide d'une syntaxe similaire à celle des tableaux de chaînes. Notez que ces champs ne sont pas créés à la volée et sont définis par le schéma de la liste. Ces champs (ou colonnes) doivent exister sur le serveur, sinon la création échouera. Tous les éléments de la liste auront le champ Titre. Certaines listes peuvent comporter des champs obligatoires à remplir avant que l'article ne soit publié dans la liste.

Dans cet exemple, la liste utilise le modèle Annonces. En plus du champ titre, la liste inclut le champ Corps qui affichera le contenu de l'annonce sur la liste.

```
ClientContext clientContext = new ClientContext(siteUrl);
List oList = clientContext.Web.Lists.GetByTitle("Annoncements");

ListItemCreationInformation itemCreateInfo = new ListItemCreationInformation();
ListItem oListItem = oList.AddItem(itemCreateInfo);
oListItem["Title"] = "My New Item!";
oListItem["Body"] = "Hello World!";

oListItem.Update();

clientContext.ExecuteQuery();
```

Article. Mise à jour d'un élément de liste

```
ClientContext clientContext = new ClientContext(siteUrl);
SP.List oList = clientContext.Web.Lists.GetByTitle("Annoncements");
ListItem oListItem = oList.Items.GetById(3);

oListItem["Title"] = "My Updated Title.";

oListItem.Update();

clientContext.ExecuteQuery();
```

Article. Supprimer un élément de la liste

```
ClientContext clientContext = new ClientContext(siteUrl);
SP.List oList = clientContext.Web.Lists.GetByTitle("Annoncements");
ListItem oListItem = oList.GetItemById(2);

oListItem.DeleteObject();

clientContext.ExecuteQuery();
```

Groupes. Récupération de tous les utilisateurs d'un groupe SharePoint

```
ClientContext clientContext = new ClientContext("http://MyServer/sites/MySiteCollection");
GroupCollection collGroup = clientContext.Web.SiteGroups;
Group oGroup = collGroup.GetById(7);
UserCollection collUser = oGroup.Users;
```



```

clientContext.Load(collUser);

clientContext.ExecuteQuery();

foreach (User oUser in collUser)
{
    Console.WriteLine("User: {0} ID: {1} Email: {2} Login Name: {3}",
        oUser.Title, oUser.Id, oUser.Email, oUser.LoginName);
}

```

Groupes. Récupération des propriétés spécifiques des utilisateurs

```

ClientContext clientContext = new ClientContext("http://MyServer/sites/MySiteCollection");
GroupCollection collGroup = clientContext.Web.SiteGroups;
Group oGroup = collGroup.GetById(7);
UserCollection collUser = oGroup.Users;

clientContext.Load(collUser,
    users => users.Include(
        user => user.Title,
        user => user.LoginName,
        user => user.Email));

clientContext.ExecuteQuery();

foreach (User oUser in collUser)
{
    Console.WriteLine("User: {0} Login name: {1} Email: {2}",
        oUser.Title, oUser.LoginName, oUser.Email);
}

```

Groupes. Récupération de tous les utilisateurs dans tous les groupes d'une collection de sites

```

ClientContext clientContext = new ClientContext("http://MyServer/sites/MySiteCollection");
GroupCollection collGroup = clientContext.Web.SiteGroups;

clientContext.Load(collGroup);

clientContext.Load(collGroup,
    groups => groups.Include(
        group => group.Users));

clientContext.ExecuteQuery();

foreach (Group oGroup in collGroup)
{
    UserCollection collUser = oGroup.Users;

    foreach (User oUser in collUser)
    {
        Console.WriteLine("Group ID: {0} Group Title: {1} User: {2} Login Name: {3}",
            oGroup.Id, oGroup.Title, oUser.Title, oUser.LoginName);
    }
}

```

Groupes. Ajout d'un utilisateur à un groupe SharePoint

```
ClientContext clientContext = new ClientContext("http://MyServer/sites/MySiteCollection ");
GroupCollection collGroup = clientContext.Web.SiteGroups;
Group oGroup = collGroup.GetById(6);

UserCreationInformation userCreationInfo = new UserCreationInformation();
userCreationInfo.Email = "alias@somewhere.com";
userCreationInfo.LoginName = @"DOMAIN\alias";
userCreationInfo.Title = "John";

User oUser = oGroup.Users.Add(userCreationInfo);

clientContext.ExecuteQuery();
```

Les rôles. Créer une définition de rôle

```
ClientContext oClientContext = new ClientContext("http://MyServer/sites/MySiteCollection");

Web oWebsite = clientContext.Web;

BasePermissions permissions = new BasePermissions();
permissions.Set(PermissionKind.CreateAlerts);
permissions.Set(PermissionKind.ManageAlerts);

RoleDefinitionCreationInformation roleCreationInfo = new RoleDefinitionCreationInformation();

roleCreationInfo.BasePermissions = permissions;
roleCreationInfo.Description = "A new role with create and manage alerts permission";
roleCreationInfo.Name = "Create and Manage Alerts";
roleCreationInfo.Order = 4;

RoleDefinition oRoleDefinition = oWebsite.RoleDefinitions.Add(roleCreationInfo);

clientContext.ExecuteQuery();

Console.WriteLine("{0} role created.", oRoleDefinition.Name);
```

Les rôles. Affectation d'un utilisateur à un rôle sur un site Web

```
ClientContext oClientContext = new
ClientContext("http://MyServer/sites/MySiteCollection/MyWebSite");
Web oWebsite = clientContext.Web;

Principal oUser = oWebsite.SiteUsers.GetByLoginName(@"DOMAIN\alias");

RoleDefinition oRoleDefinition = oWebsite.RoleDefinitions.GetByName("Create and Manage
Alerts");
RoleDefinitionBindingCollection collRoleDefinitionBinding = new
RoleDefinitionBindingCollection(clientContext);
collRoleDefinitionBinding.Add(oRoleDefinition);

RoleAssignment oRoleAssignment = oWebsite.RoleAssignments.Add(oUser,
collRoleDefinitionBinding);

clientContext.Load(oUser,
```

```

        user => user.Title);

clientContext.Load(oRoleDefinition,
    role => role.Name);

clientContext.ExecuteQuery();

Console.WriteLine("{0} added with {1} role.", oUser.Title, oRoleDefinition.Name);

```

Les rôles. Création d'un groupe SharePoint et ajout du groupe à un rôle

```

ClientContext oClientContext = new
ClientContext("http://MyServer/sites/MySiteCollection/MyWebSite");
Web oWebsite = clientContext.Web;

GroupCreationInformation groupCreationInfo = new GroupCreationInformation();
groupCreationInfo.Title = "My New Group";
groupCreationInfo.Description = "Description of new group.";
Group oGroup = oWebsite.SiteGroups.Add(groupCreationInfo);

RoleDefinitionBindingCollection collRoleDefinitionBinding = new
RoleDefinitionBindingCollection(clientContext);

RoleDefinition oRoleDefinition = oWebsite.RoleDefinitions.GetByType(RoleType.Contributor);

collRoleDefinitionBinding.Add(oRoleDefinition);

oWebsite.RoleAssignments.Add(oGroup, collRoleDefinitionBinding);

clientContext.Load(oGroup,
    group => group.Title);

clientContext.Load(oRoleDefinition,
    role => role.Name);

clientContext.ExecuteQuery();

Console.WriteLine("{0} created and assigned {1} role.", oGroup.Title, oRoleDefinition.Name);
}

```

Autorisations Briser l'héritage de sécurité d'une liste

```

string siteUrl = "http://MyServer/sites/MySiteCollection";
ClientContext oContext = new ClientContext(siteUrl);
SP.List oList = oContext.Web.Lists.GetByTitle("Announcements");

oList.BreakRoleInheritance(true, false);

oContext.ExecuteQuery();

```

Autorisations Briser l'héritage de sécurité d'un document et ajouter un utilisateur en tant que lecteur

```

ClientContext clientContext = new ClientContext(siteUrl);
SP.List oList = clientContext.Web.Lists.GetByTitle("MyList");

```

```

int itemId = 3;
ListItem oListItem = oList.Items.GetById(itemId);

oListItem.BreakRoleInheritance(false);

User oUser = clientContext.Web.SiteUsers.GetByLoginName(@"DOMAIN\alias");

RoleDefinitionBindingCollection collRoleDefinitionBinding = new
RoleDefinitionBindingCollection(clientContext);

collRoleDefinitionBinding.Add(clientContext.Web.RoleDefinitions.GetByType(RoleType.Reader));

oListItem.RoleAssignments.Add(oUser, collRoleDefinitionBinding);

clientContext.ExecuteQuery();

```

Autorisations Briser l'héritage de sécurité d'un document et modifier les autorisations d'un utilisateur

```

ClientContext clientContext = new ClientContext(siteUrl);
SP.List oList = clientContext.Web.Lists.GetByTitle("MyList");

int itemId = 2;
ListItem oListItem = oList.Items.GetById(itemId);

oListItem.BreakRoleInheritance(true);

User oUser = clientContext.Web.SiteUsers.GetByLoginName(@"DOMAIN\alias");
oListItem.RoleAssignments.GetByPrincipal(oUser).DeleteObject();

RoleDefinitionBindingCollection collRollDefinitionBinding = new
RoleDefinitionBindingCollection(clientContext);

collRollDefinitionBinding.Add(clientContext.Web.RoleDefinitions.GetByType(RoleType.Reader));

oListItem.RoleAssignments.Add(oUser, collRollDefinitionBinding);

clientContext.ExecuteQuery();

```

Action personnalisée Ajout d'une action personnalisée pour les éléments de la liste

```

string urlWebsite = "http://MyServer/sites/MySiteCollection";
ClientContext clientContext = new ClientContext(urlWebsite);
Web oWebsite = clientContext.Web;

List oList = oWebsite.Lists.GetByTitle("My List");
UserCustomActionCollection collUserCustomAction = oList.UserCustomActions;

UserCustomAction oUserCustomAction = collUserCustomAction.Add();
oUserCustomAction.Location = "EditControlBlock";
oUserCustomAction.Sequence = 100;
oUserCustomAction.Title = "My First User Custom Action";
oUserCustomAction.Url = urlWebsite + @"/_layouts/MyPage.aspx";
oUserCustomAction.Update();

```

```

clientContext.Load(oList,
    list => list.UserCustomActions);

clientContext.ExecuteQuery();

```

Action personnalisée Modification d'une action personnalisée de l'utilisateur

```

string urlWebsite = "http://MyServer/sites/SiteCollection";
ClientContext clientContext = new ClientContext(urlWebsite);
Web oWebsite = clientContext.Web;

List oList = oWebsite.Lists.GetByTitle("My List");
UserCustomActionCollection collUserCustomAction = oList.UserCustomActions;

clientContext.Load(collUserCustomAction,
    userCustomActions => userCustomActions.Include(
        userCustomAction => userCustomAction.Title));

clientContext.ExecuteQuery();

foreach (UserCustomAction oUserCustomAction in collUserCustomAction)
{
    if (oUserCustomAction.Title == "My First User Custom Action")
    {
        oUserCustomAction.ImageUrl = "http://MyServer/_layouts/images/MyIcon.png";
        oUserCustomAction.Update();

        clientContext.ExecuteQuery();
    }
}

```

Action personnalisée Ajout d'une action personnalisée d'un utilisateur aux actions de site d'un site Web

```

string urlWebsite = "http://MyServer/sites/MySiteCollection";
ClientContext clientContext = new ClientContext(urlWebsite);

Web oWebsite = clientContext.Web;
UserCustomActionCollection collUserCustomAction = oWebsite.UserCustomActions;

UserCustomAction oUserCustomAction = collUserCustomAction.Add();

oUserCustomAction.Location = "Microsoft.SharePoint.StandardMenu";
oUserCustomAction.Group = "SiteActions";
oUserCustomAction.Sequence = 101;
oUserCustomAction.Title = "Website User Custom Action";
oUserCustomAction.Description = "This description appears on the Site Actions menu.";
oUserCustomAction.Url = urlWebsite + @"/_layouts/MyPage.aspx";

oUserCustomAction.Update();

clientContext.Load(oWebsite,
    webSite => webSite.UserCustomActions);

clientContext.ExecuteQuery();

```

Web part. Mise à jour du titre d'un composant WebPart

```
ClientContext oClientContext = new ClientContext("http://MyServer/sites/MySiteCollection");
File oFile = oClientContext.Web.GetFileByServerRelativeUrl("Default.aspx");
LimitedWebPartManager limitedWebPartManager =
oFile.GetLimitedWebPartManager(PersonalizationScope.Shared);

oClientContext.Load(limitedWebPartManager.WebParts,
    wps => wps.Include(
    wp => wp.WebPart.Title));

oClientContext.ExecuteQuery();

if (limitedWebPartManager.WebParts.Count == 0)
{
    throw new Exception("No Web Parts on this page.");
}

WebPartDefinition oWebPartDefinition = limitedWebPartManager.WebParts[1];
WebPart oWebPart = oWebPartDefinition.WebPart;
oWebPart.Title = "My New Web Part Title";

oWebPartDefinition.SaveWebPartChanges();

oClientContext.ExecuteQuery();
```

Web part. Ajout d'un composant WebPart à une page

```
ClientContext oClientContext = new ClientContext("http://MyServer/sites/MySiteCollection");
File oFile = oClientContext.Web.GetFileByServerRelativeUrl("Default.aspx");
LimitedWebPartManager limitedWebPartManager =
oFile.GetLimitedWebPartManager(PersonalizationScope.Shared);

string xmlWebPart = "<?xml version=\"1.0\" encoding=\"utf-8\"?>" +
    "<WebPart xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" " +
    " xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\" " +
    " xmlns=\"http://schemas.microsoft.com/WebPart/v2\">" +
    "<Title>My Web Part</Title><FrameType>Default</FrameType>" +
    "<Description>Use for formatted text, tables, and images.</Description>" +
    "<IsIncluded>true</IsIncluded><ZoneID></ZoneID><PartOrder>0</PartOrder>" +
    "<FrameState>Normal</FrameState><Height /><Width /><AllowRemove>true</AllowRemove>" +
    "<AllowZoneChange>true</AllowZoneChange><AllowMinimize>true</AllowMinimize>" +
    "<AllowConnect>true</AllowConnect><AllowEdit>true</AllowEdit>" +
    "<AllowHide>true</AllowHide><IsVisible>true</IsVisible><DetailLink /><HelpLink />" +
    "<HelpMode>Modeless</HelpMode><Dir>Default</Dir><PartImageSmall />" +
    "<MissingAssembly>Cannot import this Web Part.</MissingAssembly>" +
    "<PartImageLarge>/_layouts/images/mscontl.gif</PartImageLarge><IsIncludedFilter />" +
    "<Assembly>Microsoft.SharePoint, Version=13.0.0.0, Culture=neutral, " +
    "PublicKeyToken=94de0004b6e3fcc5</Assembly>" +
    "<TypeName>Microsoft.SharePoint.WebPartPages.ContentEditorWebPart</TypeName>" +
    "<ContentLink xmlns=\"http://schemas.microsoft.com/WebPart/v2/ContentEditor\" />" +
    "<Content xmlns=\"http://schemas.microsoft.com/WebPart/v2/ContentEditor\">" +
    "<![CDATA[This is a first paragraph!<DIV>&nbsp;</DIV>And this is a second  
paragraph.]]></Content>" +
    "<PartStorage xmlns=\"http://schemas.microsoft.com/WebPart/v2/ContentEditor\"  
></WebPart>";

WebPartDefinition oWebPartDefinition = limitedWebPartManager.ImportWebPart(xmlWebPart);
```

```
limitedWebPartManager.AddWebPart(oWebPartDefinition.WebPart, "Left", 1);  
  
oClientContext.ExecuteQuery();
```

Web part. Suppression d'un composant WebPart d'une page

```
ClientContext oClientContext = new ClientContext("http://MyServer/sites/MySiteCollection");  
File oFile =  
oClientContext.Web.GetFileByServerRelativeUrl("/sites/MySiteCollection/SitePages/Home.aspx");  
LimitedWebPartManager limitedWebPartManager =  
oFile.GetLimitedWebPartManager(PersonalizationScope.Shared);  
  
oClientContext.Load(limitedWebPartManager.WebParts);  
  
oClientContext.ExecuteQuery();  
  
if (limitedWebPartManager.WebParts.Count == 0)  
{  
    throw new Exception("No Web Parts to delete.");  
}  
  
WebPartDefinition webPartDefinition = limitedWebPartManager.WebParts[0];  
  
webPartDefinition.DeleteWebPart();  
  
oClientContext.ExecuteQuery();
```

Le contexte. Utilisation d'un cache d'informations d'identification pour une exécution élevée du code

Bien que le code côté serveur puisse s'exécuter avec des privilèges élevés, il n'existe pas de méthode équivalente pour élever les privilèges dans le code côté client (pour des raisons de sécurité évidentes). Vous pouvez également spécifier les informations d'identification pour émuler l'accès d'un compte d'utilisateur ou de service spécifique.

Pour spécifier les informations d'identification, `ClientContext` et `ClientContext` un objet `CredentialCache`, puis affectez-le à la propriété `Credentials` de votre objet `ClientContext`.

L'exemple ci-dessous émule le compte du pool d'applications et suppose un environnement SharePoint 2013 sur site avec NTLM.

```
using System.Net;  
using Microsoft.SharePoint.Client;  
  
using (ClientContext ctx = new ClientContext("https://onpremises.local/sites/demo/"))  
{  
    // need the web object  
    ctx.Load(ctx.Web);  
    ctx.ExecuteQuery();  
  
    // here the default network credentials relate to the identity of the account  
    // running the App Pool of your web application.  
    CredentialCache credCache = new CredentialCache();
```

```
cc.Add(new Uri(ctx.Web.Url), "NTLM", CredentialCache.DefaultNetworkCredentials);

ctx.Credentials = credCache;
ctx.AuthenticationMode = ClientAuthentication.Default;
ctx.ExecuteQuery();

// do stuff as elevated app pool account
}
```

Notez que l'attribution de privilèges élevés au compte du pool d'applications dans SharePoint est contraire aux meilleures pratiques, mais que toutes les informations d'identification réseau pertinentes peuvent être utilisées à sa place.

Lire [Utilisation du modèle d'objet côté client géré \(CSOM\) en ligne](https://riptutorial.com/fr/sharepoint/topic/2679/utilisation-du-modele-d-objet-cote-client-gere--csom-):

<https://riptutorial.com/fr/sharepoint/topic/2679/utilisation-du-modele-d-objet-cote-client-gere--csom->

Chapitre 10: Utilisation du modèle d'objet côté serveur géré (approbation totale)

Remarques

Hiérarchie Conceptuelle

Dans la hiérarchie conceptuelle SharePoint, les **collections** de **sites** contiennent des **sites** qui, à leur tour, contiennent des **listes**. Une collection de sites (`SPSite`) ne possède pas d'interface utilisateur explicite mais contient toujours un site de niveau racine (accessible via la propriété `RootWeb`) et éventuellement des sous-sites supplémentaires sous ce site racine. Un site ou un site Web (`SPWeb`) possède une interface utilisateur et contient des bibliothèques de listes / documents (`SPList`), des pages avec des composants Web et des éléments / documents (`SPListItem`).

Mises en garde côté serveur

- Pour créer une application qui utilise le modèle d'objet côté serveur SharePoint, dans votre projet Visual Studio, vous devez ajouter une référence à l'assembly Microsoft.SharePoint répertorié sous Assemblies de structure.
- Les applications utilisant le modèle d'objet côté serveur (approbation totale) ne peuvent s'exécuter que sur un serveur Windows hébergeant SharePoint.
- Vous ne pouvez pas vous connecter à un serveur SharePoint autre que celui sur lequel l'application s'exécute.

Exemples

Hello World (obtenir le titre du site)

2013

Les versions SharePoint 2013 et ultérieures sont uniquement 64 bits et l'assemblage / programme doit également être conçu pour un processeur 64 bits.

Juste après la création de votre projet, il est nécessaire de basculer la **cible Platform** de **Any CPU** vers **x64**, sinon une erreur se produira.

```
using System;
using Microsoft.SharePoint;

namespace StackOverflow
{
    class Samples
    {
```

```

static void Main()
{
    using (SPSite site = new SPSite("http://server/sites/siteCollection"))
    using (SPWeb web = site.OpenWeb())
    {
        Console.WriteLine("Title: {0} Description: {1}", web.Title, web.Description);
    }
}
}

```

Faire le tour de toute la batterie de serveurs SharePoint

Utilisation de PowerShell à partir d'un serveur Web SharePoint:

```

$wacoll = get-spwebapplication
foreach($wa in $wacoll){
    if($wa.IsAdministrationWebApplication -eq $false){
        foreach($site in $wa.Sites){
            foreach($web in $site.AllWebs){
                # your code here
                $web.Dispose()
            }
            $site.Dispose()
        }
    }
}
}

```

Récupérer des éléments de la liste

```

using (SPSite site = new SPSite("http://server/sites/siteCollection"))
using (SPWeb web = site.OpenWeb())
{
    SPList list = web.Lists["Some list"];

    // It is always better and faster to query list items with GetItems method with
    // empty SPQuery object than to use Items property
    SPListItemCollection items = list.GetItems(new SPQuery());
    foreach (SPListItem item in items)
    {
        // Do some operation with item
    }
}

```

Récupérer des éléments à l'aide de la pagination

```

using (SPSite site = new SPSite("http://server/sites/siteCollection"))
using (SPWeb web = site.OpenWeb())
{
    SPList list = web.Lists["Some list"];
    SPQuery query = new SPQuery()
    {
        RowLimit = 100
    };
}

```

```

do
{
    SPListItemCollection items = list.GetItems(query);
    foreach (SPListItem item in items)
    {
        // Do some operation with item
    }

    // Assign current position to SPQuery object
    query.ListItemCollectionPosition = items.ListItemCollectionPosition;
} while (query.ListItemCollectionPosition != null);
}

```

Obtenir la liste par URL

```

using (SPSite site = new SPSite("http://server/sites/siteCollection"))
using (SPWeb web = site.OpenWeb())
{
    string listUrl = string.Format("{0}{1}", web.ServerRelativeUrl, "Lists/SomeList");
    SPList list = web.GetList(listUrl);
}

```

Créer un élément de liste

Lors de la création d'un nouvel élément de liste, ses champs peuvent être définis à l'aide d'une syntaxe similaire à celle des tableaux de chaînes. Notez que ces champs ne sont pas créés à la volée et sont définis par le schéma de la liste. Ces champs (ou colonnes) doivent exister sur le serveur, sinon la création échouera. Tous les éléments de la liste auront le champ Titre. Certaines listes peuvent comporter des champs obligatoires à remplir avant que l'article ne soit publié dans la liste.

Dans cet exemple, la liste utilise le modèle Annonces. En plus du champ titre, la liste inclut le champ Corps qui affichera le contenu de l'annonce sur la liste.

```

using (SPSite site = new SPSite("http://server/sites/siteCollection"))
using (SPWeb web = site.OpenWeb())
{
    SPList list = web.Lists["Annoncements"];

    SPListItem item = list.AddItem();
    item[SPBuiltInFieldId.Title] = "My new item";
    item[SPBuiltInFieldId.Body] = "Hello World!";
    item.Update();
}

```

Lire Utilisation du modèle d'objet côté serveur géré (approbation totale) en ligne:

<https://riptutorial.com/fr/sharepoint/topic/7543/utilisation-du-modele-d-objet-cote-serveur-gere--approbation-totale->

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec sharepoint	Community , Marco , Ryan Gregg , Thriggle , Tom Resing , Zach Koehne
2	App SharePoint	Sunil sahu
3	Création d'une application hébergée par un fournisseur	vinayak hegde
4	Rejets majeurs	jjr2527 , MikhailSP
5	Rendu côté client SharePoint 2013	Rohit Waghela , Yayati
6	Services REST	Aaron , Brock Davis , ocelotsloth , R4mbi , Rohit Waghela , Thriggle
7	Travailler avec des boîtes de dialogue modales avec JavaScript	Thriggle
8	Utilisation du modèle d'objet client JavaScript (JSOM)	Thriggle , yngrdyn
9	Utilisation du modèle d'objet côté client géré (CSOM)	InvoiceGuy , Lukáš Nešpor , MikhailSP , RamenChef , Thriggle , Zach Koehne
10	Utilisation du modèle d'objet côté serveur géré (approbation totale)	Lukáš Nešpor , Thriggle