EBook Gratuito

APPRENDIMENTO sharepoint

Free unaffiliated eBook created from
Stack Overflow contributors.

#sharepoint

Sommario

Di

Capitolo 1: Iniziare con sharepoint

Osservazioni

SharePoint può fare riferimento a uno o più prodotti della famiglia Microsoft SharePoint.

- **SharePoint Foundation** : questa era la tecnologia di base per tutti i siti di SharePoint e non è più disponibile per SharePoint 2016
- **SharePoint Server** : questa è la versione locale di SharePoint. È possibile distribuire uno o più server SharePoint. Offre funzionalità aggiuntive su SharePoint Foundation, come funzionalità di BI, Enterprise Content Management e altro
- **SharePoint Online** : versione basata su cloud di SharePoint. Il cliente non deve preoccuparsi dell'infrastruttura del server o della scalabilità.

Office 365 è un'offerta Microsoft separata che include il servizio SharePoint Online, sebbene non tutti i piani supportino tutte le funzionalità di SharePoint.

I seguenti collegamenti forniscono ampi confronti tra le versioni di SharePoint disponibili:

- SharePoint 2013 on-premises e SharePoint 2016 on-premises: disponibilità delle caratteristiche tra piani locali di SharePoint
- Funzionalità di SharePoint in Office 365: disponibilità delle funzionalità tra i piani di SharePoint
- Funzionalità di SharePoint in SharePoint Online (senza Office 365): disponibilità delle caratteristiche tra i piani autonomi di SharePoint
- Confronto unito delle funzionalità tra SharePoint 2013 e SharePoint Online: http://www.buckleyplanet.com/2014/06/sharepoint-online-vs-onprem-feature-comparison.html

Versioni

Versione	Nome ufficiale	Data di rilascio
Pre-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

Examples

Installazione di SharePoint 2016 per Single Server Farm

introduzione

SharePoint 2016 è la versione 16 della famiglia di prodotti SharePoint. È stato rilasciato il 4 maggio 2016. Questo esempio copre l'installazione di SharePoint 2016 utilizzando la configurazione di Farm Server singolo. Questa configurazione copre le basi della configurazione di una farm di SharePoint senza la necessità di avere più server. Si noti che gli scenari coperti da una singola server farm sono in genere limitati allo sviluppo e a scenari di produzione molto piccoli.

Requisiti

Prima di installare SharePoint, è necessario configurare l'ambiente di base. SharePoint archivia documenti e metadati, log, applicazioni personalizzate, personalizzazioni e molto altro. Assicurarsi di avere sufficiente spazio su disco e RAM disponibile sopra i requisiti della linea di base.

- 4 core su processori compatibili a 64 bit
- 12 - 24 GB di RAM (a seconda della distribuzione test o prod)
- Disco rigido da 80 GB per sistema
- Disco rigido da 100 GB come seconda unità
- Server con Windows Server 2012 R2 a 64 bit o Anteprima tecnica "Soglia"
- SQL Server 2014 o SQL Server 2016
- .NET Framework 4.5.2 o .NET Framework 4.6
- Account del computer unificato e account di servizio della farm delegati

Tutti gli altri prerequisiti possono essere installati manualmente o eseguiti utilizzando il programma di installazione prerequisito di SharePoint incluso con l'installazione di SharePoint.

Installazione

- Esegui il programma di installazione dei prerequisiti; potrebbe richiedere di riavviare il server prima di continuare
- Eseguire Setup.exe dall'installazione di SharePoint
- Inserisci la chiave di licenza
- Accetta il contratto di licenza
- Seleziona "Completa" nella scheda Tipo server
- L'installazione dovrebbe essere completata correttamente
- Nella pagina completa, lasciare selezionata la casella di controllo accanto a Esegui configurazione guidata prodotto e fare clic su Chiudi

Configurazione

Se si sta continuando dal passaggio precedente, la Configurazione guidata del prodotto SharePoint 2016 dovrebbe aprirsi automaticamente. se la casella non viene visualizzata o si sta eseguendo la configurazione in un secondo momento, aprire la procedura guidata di configurazione selezionando Start -> Prodotti SharePoint 2016 -> Configurazione guidata prodotto SharePoint 2016.

- Fare clic su Avanti sulla pagina di benvenuto
- Apparirà una finestra di dialogo modale che dice che alcuni servizi verranno riavviati durante la configurazione; non è stato ancora installato nulla, quindi fai clic su Sì
- Aggiungere il server database per la farm
    - Immettere il nome della macchina che esegue SQL Server; in questo caso, è la macchina locale
    - Immettere il nome del database di configurazione o mantenere il nome predefinito SharePoint_Config
    - Immettere il nome utente dell'utente del servizio di dominio che accederà al database (sotto forma di DOMINIO \ utente) * Immettere la password per l'utente del dominio
    - Fare clic su Avanti quando fatto
- Inserire la password della farm; questo verrà utilizzato quando si uniranno server aggiuntivi alla nuova farm
- Seleziona il ruolo Farm singolo server
- Configurare l'app Web di amministrazione centrale (in cui SharePoint verrà gestito dagli amministratori della farm) selezionare il numero di porta e selezionare il tipo di federazione di autenticazione (NTLM o Negotate (Kerberose))
- Rivedere le impostazioni nelle pagine finali e apportare le modifiche necessarie
- Quando sei pronto, esegui la configurazione che potrebbe richiedere alcuni minuti
- Al completamento, si aprirà la procedura guidata consentirà di aprire il sito di amministrazione centrale
- In caso di errore, è possibile esaminare i registri nella cartella% COMMONPROGRAMFILES% \ Microsoft Shared \ Web Server Extensions \ 16 \ LOG

Configurazione della fattoria

Una volta configurate l'app web centrale, il database di configurazione e l'amministratore centrale, sarete pronti per configurare la farm da utilizzare per gli utenti o lo sviluppo. È possibile aggiungere un segnalibro alla posizione del sito di Amministrazione centrale o accedervi tramite un collegamento nella stessa posizione della Configurazione guidata prodotto.

- Se si sta avviando la configurazione in un secondo momento, fare clic su Avvio veloce -> Procedure guidate di configurazione -> Configurazione guidata della farm
- Se stai avviando la procedura guidata dal passaggio di installazione, fai clic su Avvia la procedura guidata
- Scegli se vuoi far parte del programma di miglioramento del cliente facendo clic su Sì o No

- Nella pagina di configurazione della farm, selezionare l'account di dominio che eseguirà i servizi in background nella farm
    - Sebbene questo account possa essere uguale all'account del database, potrebbe anche essere diverso per la separazione di ruoli e privilegi
    - Inserisci l'account come DOMINIO \ utente
- Convalidare i servizi che si desidera disponibili nella farm nella pagina Servizi
- Crea la prima raccolta siti nella farm (questo passaggio può essere saltato e fatto in un secondo momento)
    - Immettere il titolo, la descrizione, l'indirizzo Web della raccolta siti (in genere il primo sito si trova nella radice del server) e il modello
    - La maggior parte delle cose può essere cambiata (titolo, descrizione) può essere cambiata facilmente, ma altri come l'URL web possono richiedere molto più lavoro per cambiare; il modello può anche non essere facilmente ripristinato, ma SharePoint consente una grande quantità di personalizzazioni che ti permettono di prendere qualsiasi modello di base e convertire lo stile e il layout del sito
- Al termine della configurazione, fare clic su Fine

La farm e la prima raccolta siti sono ora configurate per l'uso.

Creare una web part con SharePoint Framework

dev.office.com/sharepoint è un ottimo punto di partenza per SharePoint Framework.

SharePoint Framework è un moderno approccio client allo sviluppo di SharePoint inizialmente indirizzato a SharePoint Online in Office 365. Le parti Web create con SharePoint Framework sono un nuovo tipo di web part e possono essere rese disponibili per l'aggiunta su entrambe le pagine SharePoint esistenti e nuove pagine di SharePoint.

C'è un grande esempio di ciao mondo per questo processo ospitato su Costruisci la tua prima web part lato client di SharePoint (Hello World parte 1) . Tutti gli esempi su dev.office.com sono disponibili per i contributi della community attraverso github.

I passi fondamentali di Hello World in SharePoint Framework sono:

1. Genera lo scheletro del progetto con Yeoman SharePoint Generator .

   yo @ microsoft / SharePoint

2. Modifica il codice generato nell'editor di tua scelta. Il supporto per Visual Studio Code è forte su tutte le piattaforme.

3. Visualizza l'anteprima della web part utilizzando gulp e SharePoint Workbench locale

   servire il gulp

4. Anteprima nell'ambiente SharePoint Online

   Vai al seguente URL: ' https: //your-sharepoint-site/_layouts/workbench.aspx '

Log e registro ULS di SharePoint

Il servizio di registrazione unificata di SharePoint (ULS) offre funzionalità di supporto e debug per entrambi gli operatori e gli sviluppatori. Capire come leggere i registri è un primo passo importante per risolvere i problemi.

Tooling

Microsoft fornisce il Visualizzatore ULS per aiutare a leggere i registri ei registri vecchi che vengono scritti attualmente mentre la farm è in esecuzione. Può anche filtrare e applicare la formattazione ai registri per aiutare a restringere un problema.

Identificatore di correlazione

Per isolare un problema, è utile esaminare solo un particolare ID di correlazione. Ogni ID di correlazione è associato a una richiesta o un'azione end to end del sistema (come un jobber di tempo). Se si verifica un problema con una pagina Web sottoposta a rendering, l'individuazione della richiesta nei registri ULS e l'isolamento con l'ID di correlazione specifico rimuove tutto il rumore dagli altri registri, contribuendo a individuare il problema.

Aggiunta di SPMonitoredScope al mio codice

Un modo per calcolare la registrazione aggiuntiva e il monitoraggio delle prestazioni è aggiungere SPMonitoredScope al codice.

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

Questo codice registrerà l'inizio e la fine delle richieste così come alcuni dati sulle prestazioni. Costruire il proprio monitor personalizzato che implementa ISPScopedPerformanceMonitor consente di impostare il livello di tracciamento o il tempo massimo di esecuzione per un set di codice.

Leggi Iniziare con sharepoint online: https://riptutorial.com/it/sharepoint/topic/950/iniziare-con-sharepoint

Capitolo 2: App di SharePoint

introduzione

SharePoint Hosted App

Osservazioni

Riferimento richiesto dal sito: http://www.letsharepoint.com/what-is-user-information-list-in-sharepoint-2013/

Examples

SharePoint 2013: accesso ai dati del servizio profili utente tramite JSOM in SharePoint 2013

SharePoint 2013: accesso ai dati del servizio profili utente tramite JSOM in SharePoint 2013

In questo articolo, impareremo a gestire o accedere all'applicazione UPS (User Profile Service) utilizzando JSOM (Javascript Object Model) e creare un'app di base. Prima di iniziare, passiamo prima alla terminologia UPS di base.

Profilo utente: contiene tutte le informazioni delle persone in un'organizzazione in modo organizzato. Visualizza tutte le proprietà come AccountName, FirstName, LastName, WorkEmail ecc. Relative a un utente.

Applicazione del servizio profili utente: viene considerata una posizione centralizzata per archiviare tutti i profili utente e consente inoltre agli amministratori di configurare o gestire profili, sincronizzazione profilo, sito personale, tag social, ecc. Può inoltre estrarre informazioni dai servizi di directory come Active Directory.

Sito personale: un sito personalizzato per utenti individuali per gestire le proprie informazioni e archiviare documenti, collegamenti, ecc. Fornisce ricche funzionalità di rete e social consentendo agli utenti di condividere informazioni su se stessi o le loro attività. Il mio sito è accessibile facendo clic su Nome utente nell'angolo in alto a destra della pagina di SharePoint.

Gestire e accedere ai dati del profilo utente

Dato che lavoreremo con JSOM, possiamo eseguire solo operazioni 'Leggi' con un'eccezione che l'immagine del profilo può essere cambiata usando JSOM (o CSOM o REST)

* Codice lato server consente di leggere / scrivere entrambe le operazioni.

Recupera le proprietà del profilo utente usando JSOM

Consente di creare un'app di hosting di SharePoint e recuperare le informazioni dell'utente in

quell'app

Avvia Visual Studio 2013 e seleziona "App per SharePoint 2013" da Nuovo progetto. Dopo aver selezionato sopra il tipo di progetto, ti viene presentata una finestra per collegarti al sito di SharePoint e selezionare il tipo di app da distribuire (vedi screenshot qui sotto). Qui ho fornito l'URL del sito di sviluppo di SharePoint Online e l'App ospitata SharePoint. Fai clic su Fine.

3.) Una volta creato il progetto, verrà visualizzato un set di cartelle / filer aggiunto in Esplora soluzioni aggiunto per impostazione predefinita al progetto.

4.) Se apri la pagina "Default.aspx", troverai alcune librerie JavaScript già aggiunte alla pagina.

Qui abbiamo bisogno di aggiungere un'altra libreria per iniziare a lavorare con i profili utente

Leggi App di SharePoint online: https://riptutorial.com/it/sharepoint/topic/9876/app-di-sharepoint

Capitolo 3: Creazione di un'app ospitata dal provider

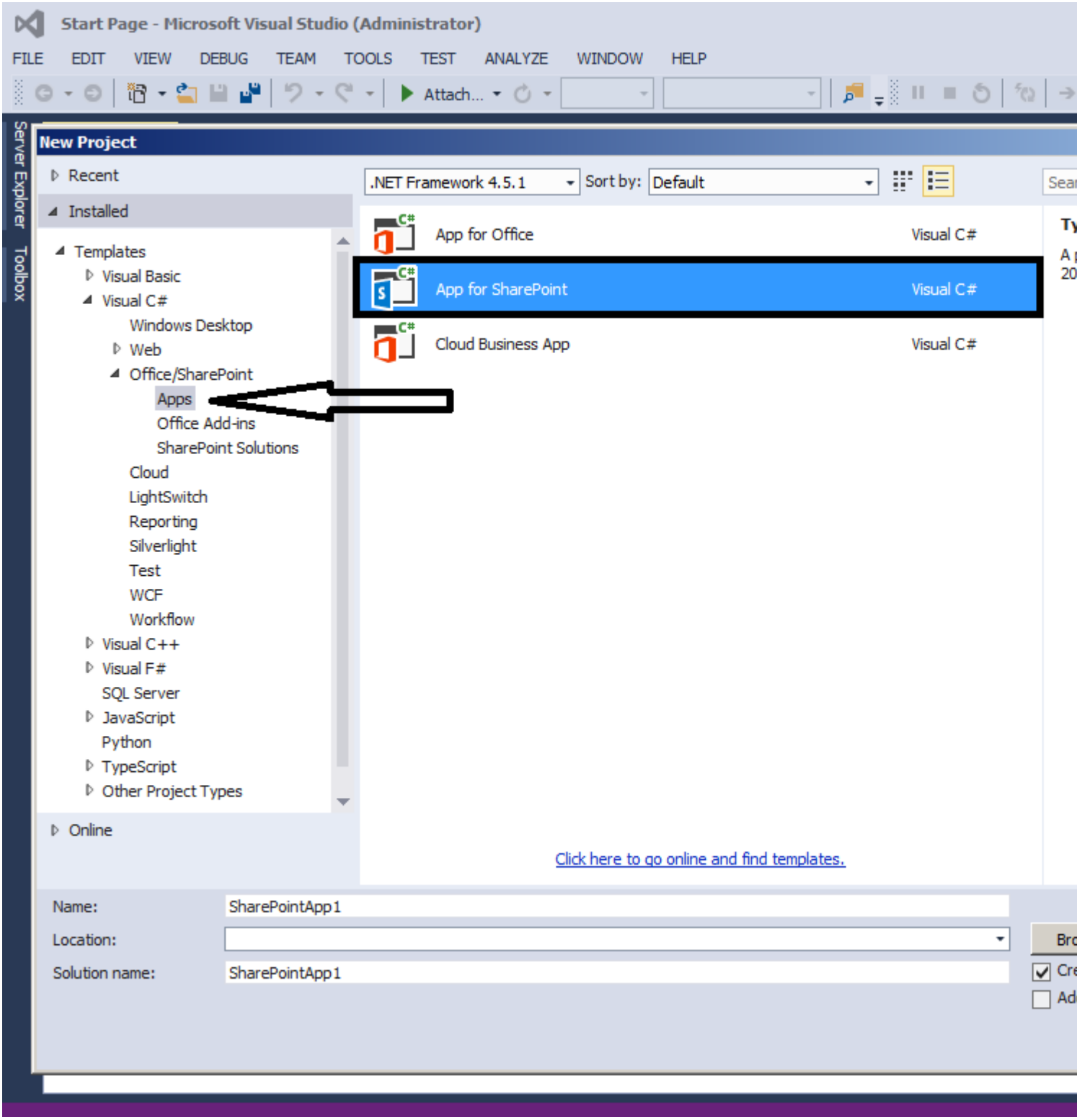
Examples

Impostazione dell'ambiente di sviluppo

Per iniziare con App Development abbiamo bisogno di Visual Studio 2013 o versione successiva. Scarica l'ultima versione di community o expression da qui>
https://www.visualstudio.com/products/free-developer-offers-vs

Una volta scaricato e installato

> Apri e fai **clic su crea nuovo progetto**
>
> espandi la sezione Office / SharePoint dovresti vedere un'opzione per App come mostrato di seguito.



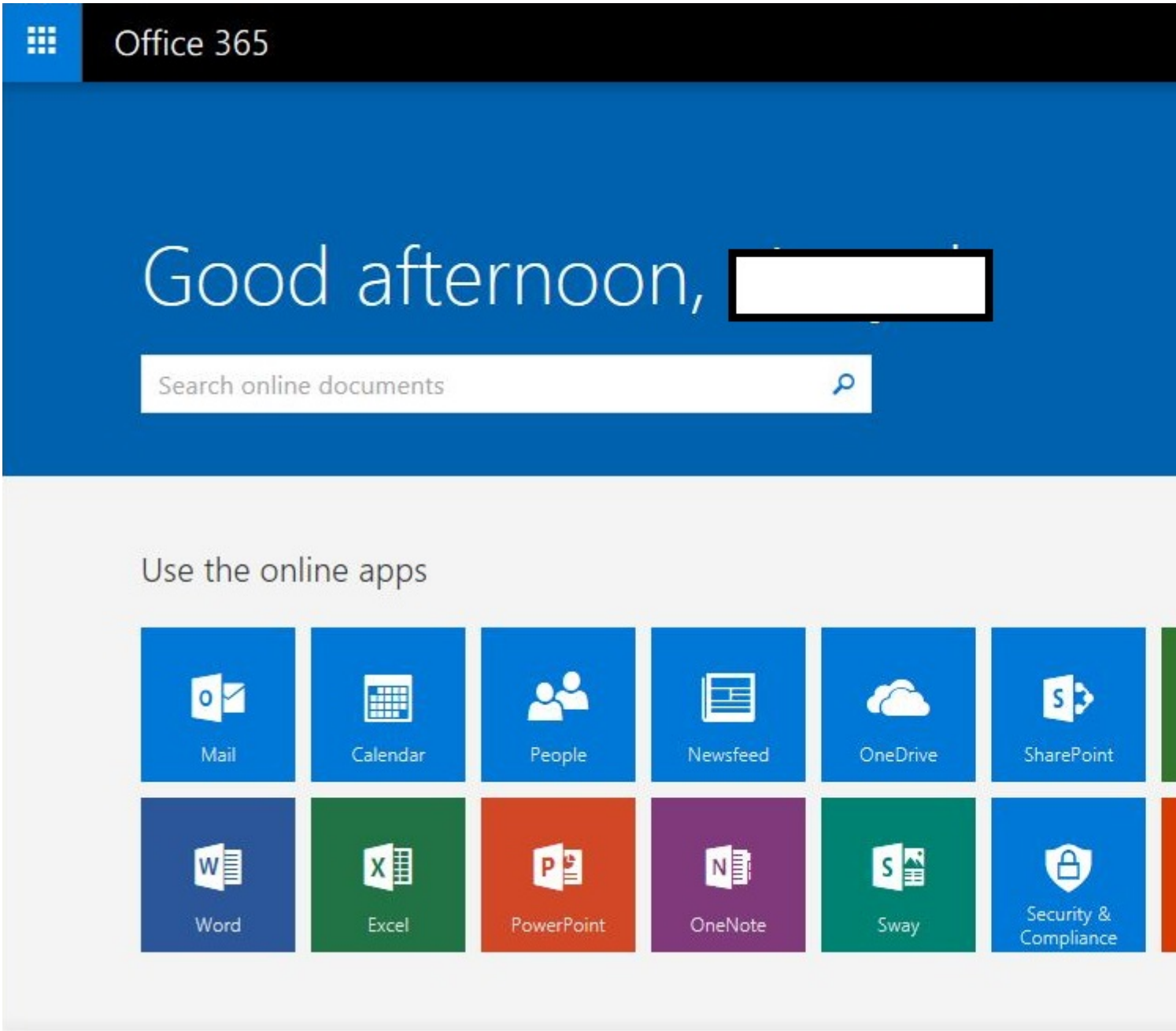
Se l'opzione App non è disponibile Chiudere il VS, scaricare e installare gli **Strumenti per gli sviluppatori di Microsoft Office** https://www.visualstudio.com/en-us/features/office-tools-vs.aspx

Preparazione per il sito degli sviluppatori

Una volta che abbiamo uno studio visivo, abbiamo bisogno di un sito per sviluppatori per distribuire app in SharePoint. Il modo più semplice è quello di ottenere è> Registrati per un account di sviluppatore di Office 365 gratuito per un anno https://profile.microsoft.com/RegSysProfileCenter/wizardnp.aspx?wizid=14b845d0-938c-45af-

b061-f798fbb4d170&lcid=1033

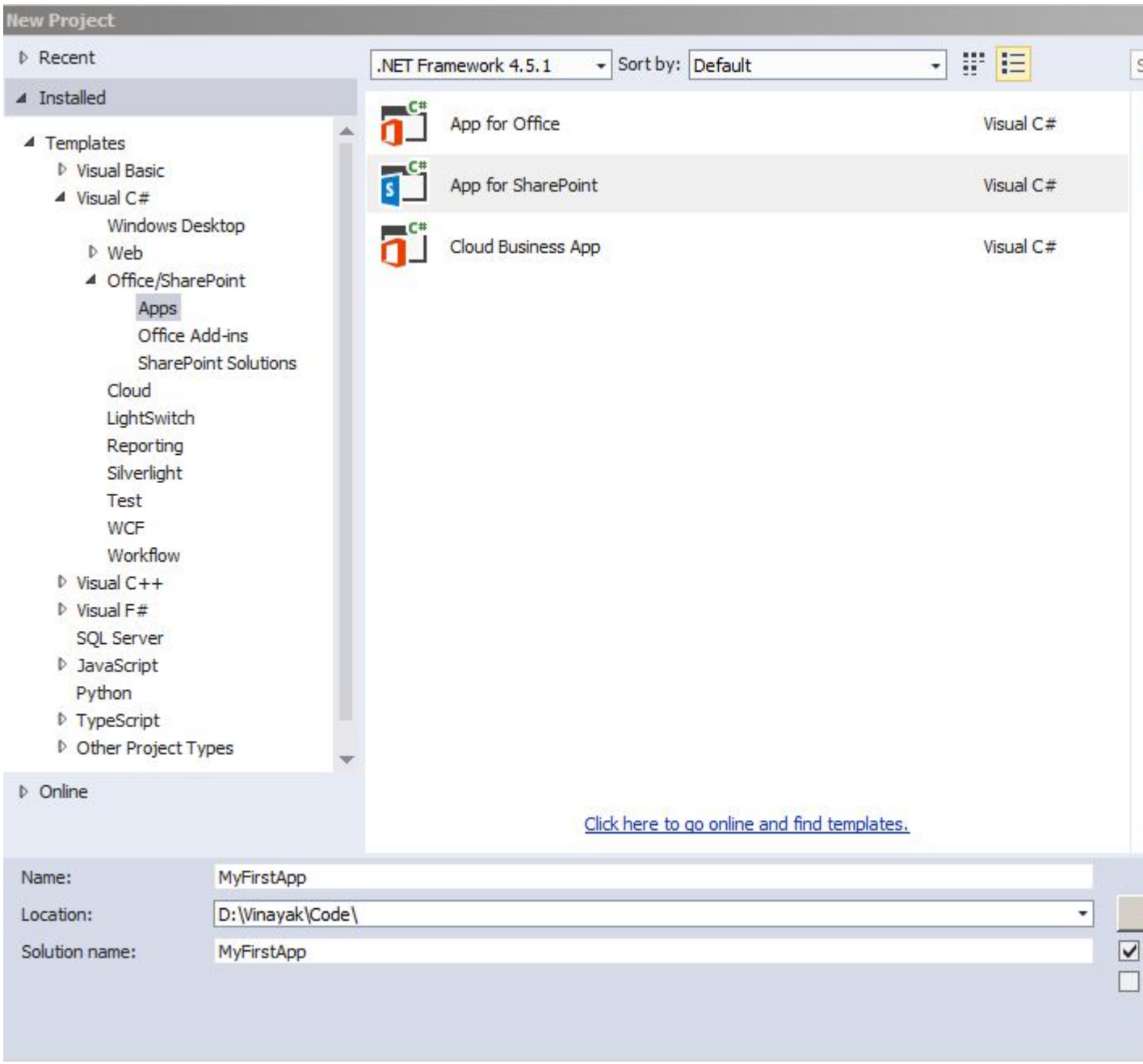
Una volta che il processo di registrazione è finito https://www.office.com / centro URL per tutta la tua app



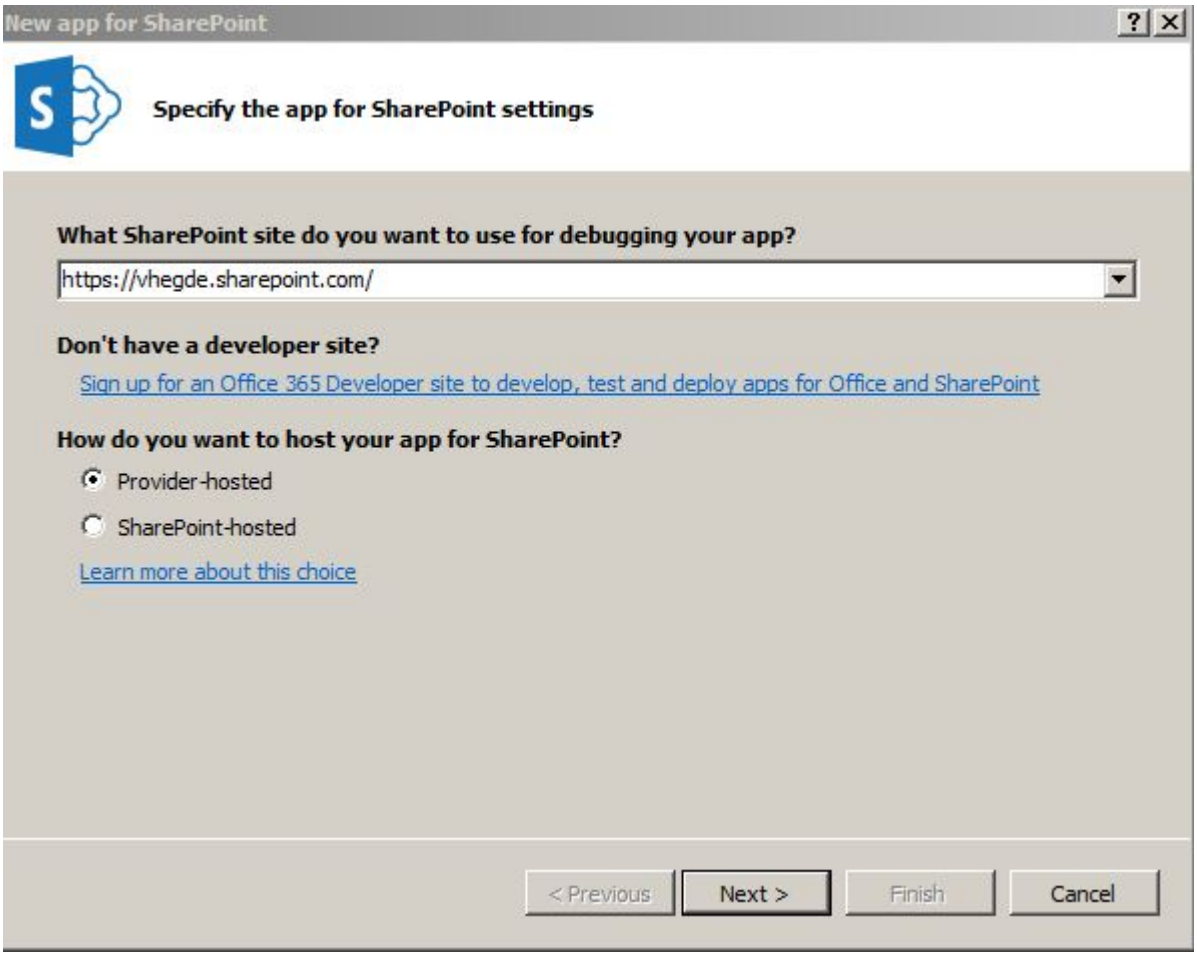
Crea app in Visual Studio

Iniziamo con la creazione della nostra prima app

1. Apri lo studio visivo e> crea un nuovo progetto
2. Inserisci nome e posizione

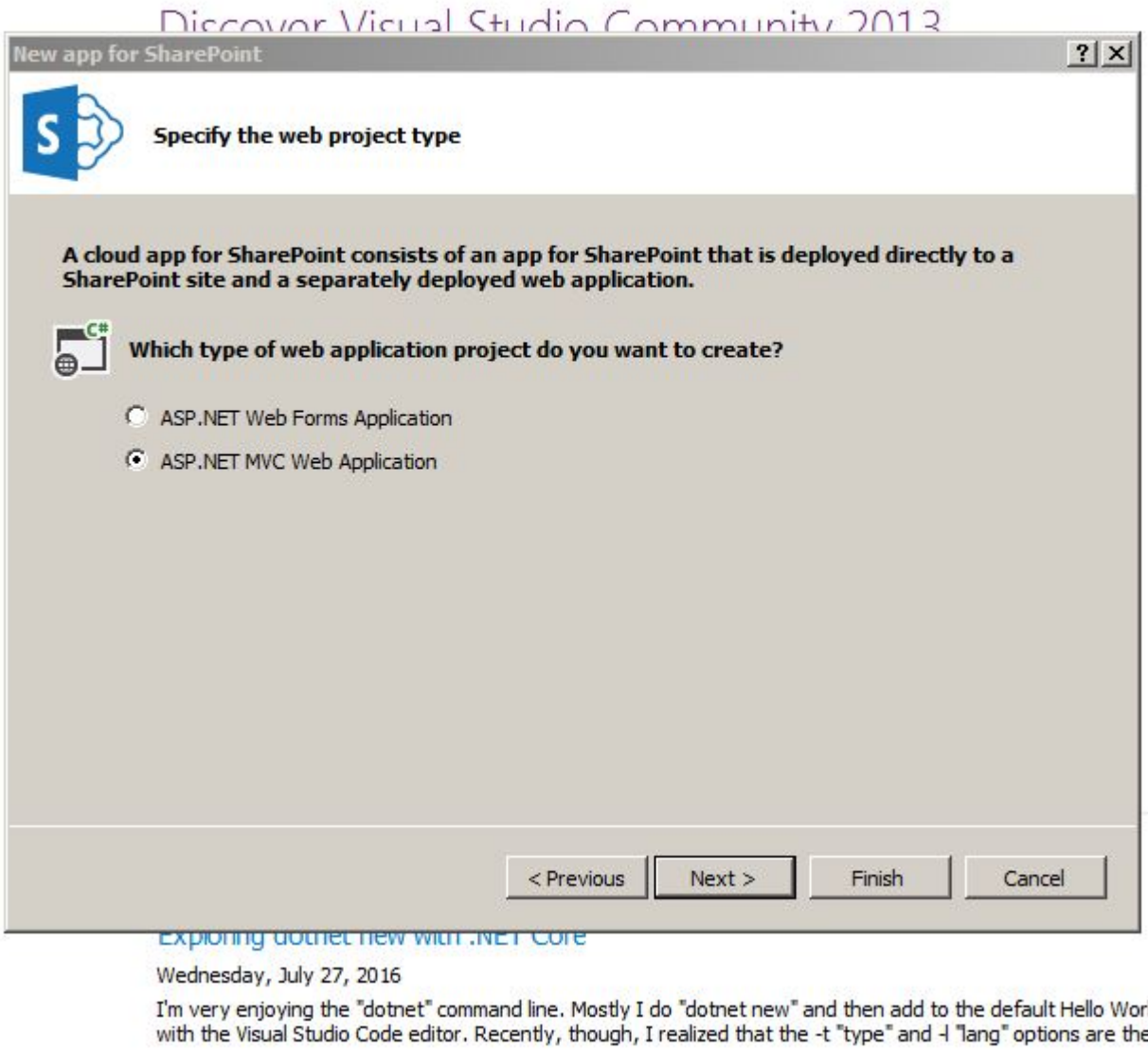


3. Inserisci l'URL del tuo sito sviluppatore creato nel passaggio precedente e seleziona Provider ospitato



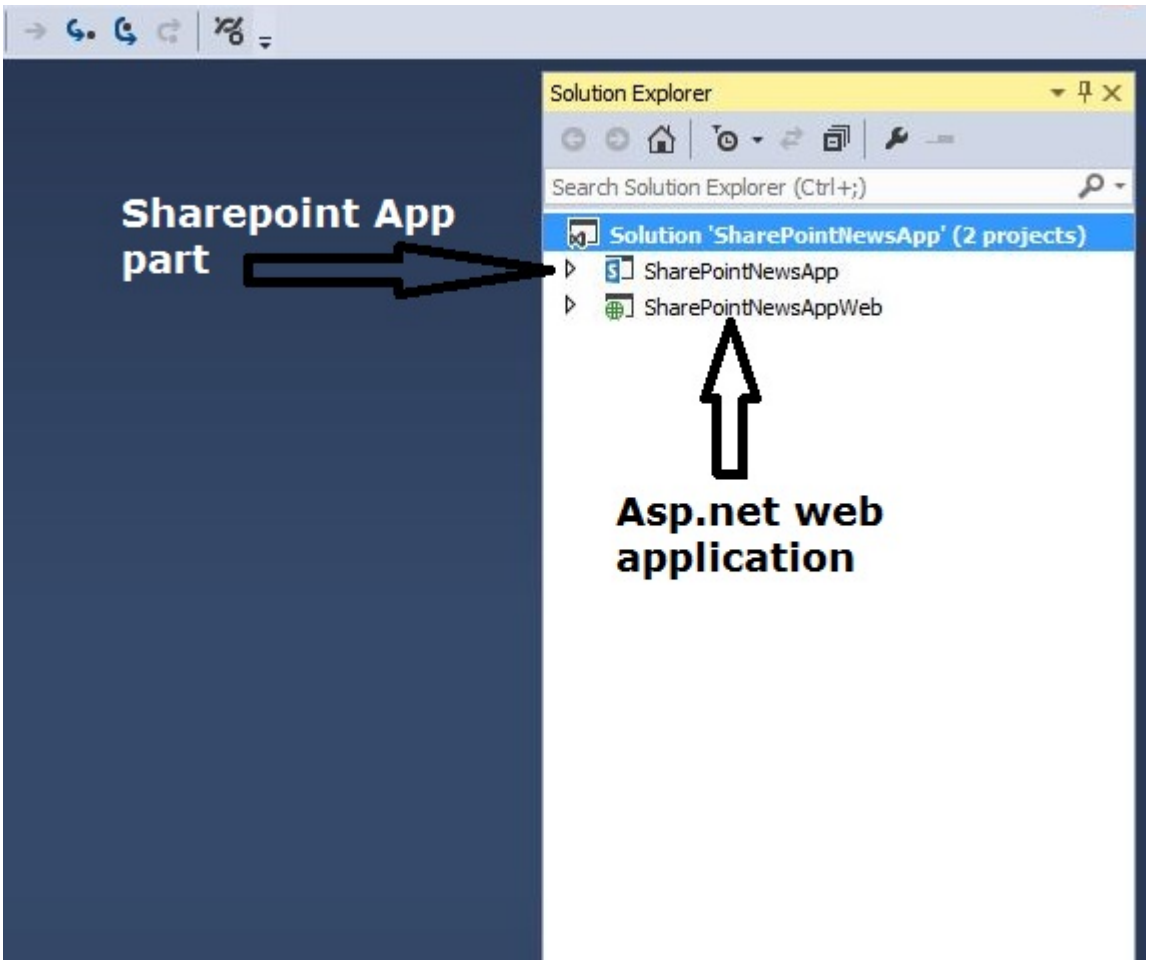
4. Si aprirà un popup che avrà come login

5. Il prossimo passo sarà come per il tipo di applicazione, selezionare MVC o Webform. Sto selezionando MCV qui



6. In Come si desidera che il componente aggiuntivo esegua l'autenticazione ?, scegliere Usa servizio di controllo di accesso di Windows Azure. Fare clic su Fine

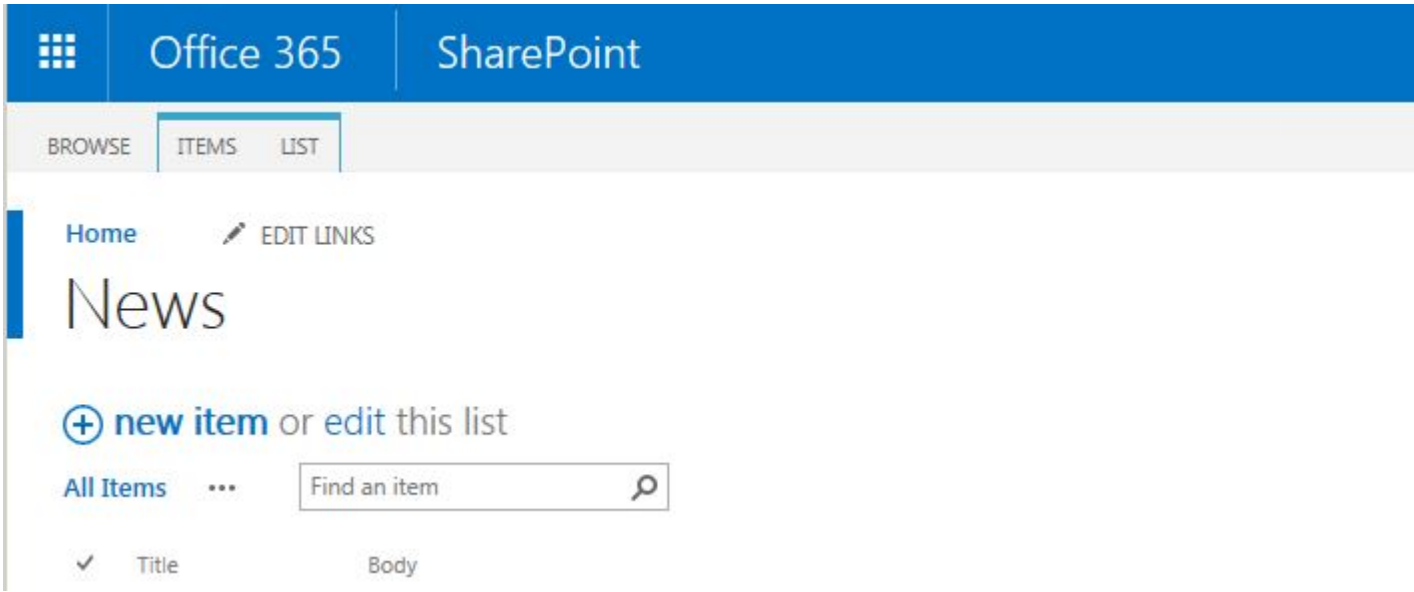
7. In solution explorer possiamo vedere che 2 progetti sono stati creati. Uno è la parte dell'app di SharePoint e un'altra è l'app Web di asp.net



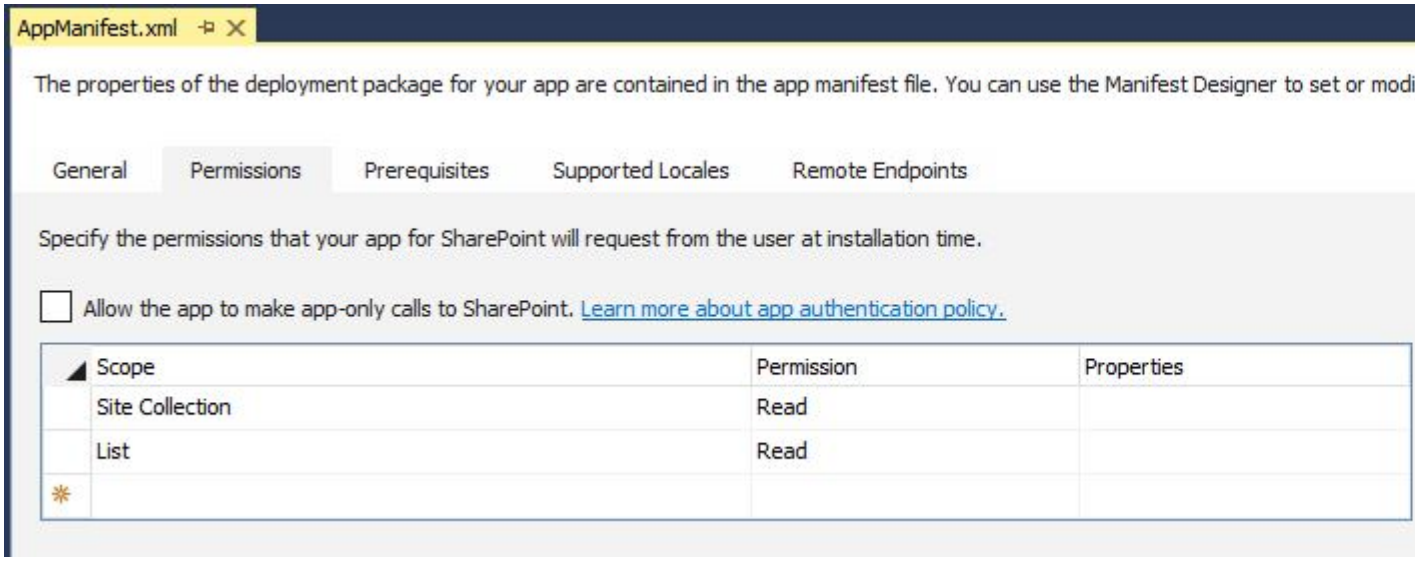
Iniziamo la codifica

Qui sto prendendo l'esempio di un'app di notizie di base

1. Apri il sito degli sviluppatori di SharePoint e crea un elenco per archiviare i nostri articoli di notizie
2. Creare un elenco personalizzato e aggiungere altre 3 colonne Corpo, Estivo, ThumbnailImageUrl



3. Torna alla nostra app di SharePoint, apri il file AppManifest.xml, fai clic sulla scheda di autorizzazione e concedi l'autorizzazione di lettura alla raccolta siti e salvala.



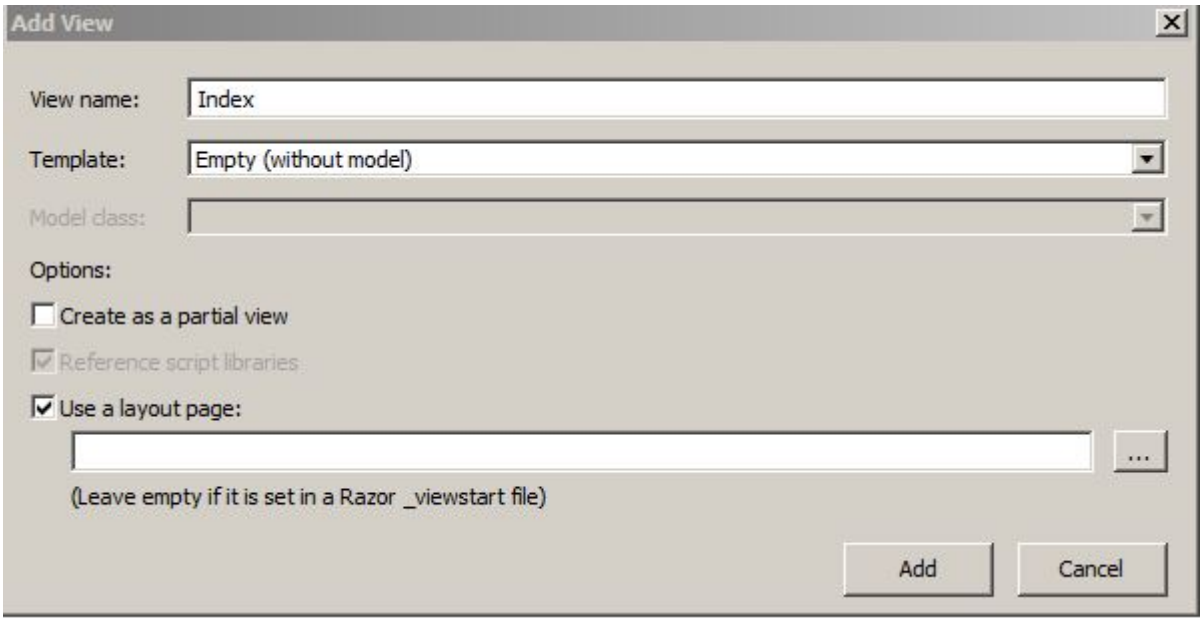
4. Aprire HomeController dall'applicazione Web, nel mio caso è un'applicazione MVC. Se stai creando un'app webform, il codice dovrebbe essere nella pagina default.aspx.cs

5. Di seguito è riportato lo snippet di codice per ottenere le ultime notizie dall'elenco. Ecco come dovrebbe apparire la nostra pagina indice.

```
[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. Ora fai clic destro su **Indice** e fai clic su **Aggiungi vista.** Quindi fare clic su Aggiungi



7. Ora apri il file **Index.cshtml** da **Views> Home directory**

8. Di seguito è riportato lo snippet di codice per il file 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">
        <img class="img-responsive" style="max-height:200px;max-width:100%;"
src="@item.Thumbnail" alt="@item.Title" />
    </a>
</div>
<div class="col-xs-9 panel-default">
    <div class="panel-heading">
        <h4><a href="/home/aticle?ArticleId=@item.Id">@item.Title.ToUpper()</a></h4>
    </div>
```

```
    <div class="panel-body">
        <p>@item.Summery</p>
    </div>
</div>
```

9. Fare clic con il tasto destro sulla cartella Modello nella soluzione e aggiungere un file di classe CS. Aggiungi sotto Classi di modelli

```
 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 Summery { 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. Utilizzare il tasto F5 per distribuire ed eseguire il componente aggiuntivo. Se viene visualizzata una finestra di avviso di sicurezza che richiede di considerare attendibile il certificato Localhost autofirmato, selezionare Sì.

E ora la prima app è pronta

Creazione di una pagina completa dell'articolo

Abbiamo già creato la prima pagina che mostrerà tutti gli articoli di notizie. Questa pagina mostrerà l'articolo completo.

1. Aggiungi un altro metodo di azione a 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. Fare nuovamente clic con il tasto destro su Azione e creare una vista con lo stesso nome Nome metodo di azione. Nel mio caso, la vista si chiamerà **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>
```

Questo è il codice per la pagina completa dell'articolo che mostra il corpo dell'articolo

Leggi Creazione di un'app ospitata dal provider online:
https://riptutorial.com/it/sharepoint/topic/6301/creazione-di-un-app-ospitata-dal-provider

Capitolo 4: Lavorare con JavaScript Client Object Model (JSOM)

Osservazioni

sfondo

Il modello di oggetti JavaScript è stato introdotto in SharePoint 2010. Espone sul lato client molti degli oggetti che in precedenza erano accessibili solo attraverso il codice lato server o tramite servizi Web dedicati.

Incorporamento di JavaScript nelle pagine di SharePoint

In SharePoint 2013 è possibile inserire il codice JavaScript in una web part Script Editor.

In SharePoint 2010 è possibile utilizzare la proprietà "collegamento contenuto" di una web part Editor contenuto per collegarsi a un file HTML che contiene lo script incorporato.

Riferimento dell'oggetto

I costruttori, i metodi e le proprietà di tutti gli oggetti trovati nello spazio dei nomi `SP` sono documentati nel riferimento del modello di oggetto client SharePoint 2013 qui .

Il riferimento del modello di oggetto client JavaScript di SharePoint 2010 è disponibile qui .

Pattern di programmazione asincrona di JSOM

Quando si utilizza il modello di oggetto client JavaScript, il codice generalmente ha il seguente schema:

1. Ottenere un oggetto `ClientContext` .
2. Utilizzare l'oggetto `ClientContext` per recuperare oggetti che rappresentano entità nel modello di oggetti di SharePoint, come elenchi, cartelle, viste.
3. Accodare le istruzioni da eseguire rispetto agli oggetti. Queste istruzioni non sono ancora state trasmesse al server.
4. Utilizzare la funzione di `load` per comunicare a `ClientContext` quali informazioni si desidera ricevere dal server.
5. Richiamare la funzione `ClientContext` dell'oggetto `executeQueryAsync` per inviare le istruzioni in coda al server, passando due funzioni di callback per l'esecuzione in caso di esito positivo o negativo.
6. Nella funzione di callback, lavorare con i risultati restituiti dal server.

alternative

Le alternative lato client di JSOM includono i servizi Web di SharePoint, gli endpoint REST e il modello di oggetto client .NET .

Examples

Ottenere i tipi di contenuto della libreria usando il nome della libreria

```
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());
}
```

Elimina un elemento in un elenco

```
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());});
}
```

Creazione di elementi o cartelle

Creazione di voci di elenco

```
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'esempio sopra mostra che una voce di elenco viene creata eseguendo quanto segue:

1. Chiama il metodo `addItem` di un oggetto lista per ottenere un oggetto oggetto
2. Chiamare il metodo `set_item` sull'oggetto item dell'elenco risultante per impostare ogni valore di campo come desiderato
3. Chiamare il metodo di `update` sull'oggetto dell'elenco per indicare che le modifiche devono essere eseguite
4. Chiamare il metodo `executeQueryAsync` dell'oggetto contesto client per eseguire le istruzioni in coda

Si noti che **non è** necessario passare il nuovo oggetto oggetto al metodo di `load` del contesto del client per creare l'elemento. Questo passaggio è necessario solo se si desidera recuperare uno dei valori del campo dell'articolo dal server.

Creazione di cartelle

La creazione di una cartella è simile all'aggiunta di un elemento a un elenco. La differenza è che bisogna prima creare un oggetto `ListItemCreationInformation` e impostare la proprietà `underlyingObjectType` su `SP.FileSystemObjectType.folder` e la proprietà `leafName` sul nome desiderato della nuova cartella.

L'oggetto viene quindi passato come parametro nel metodo `addItem` sulla libreria per creare la cartella.

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

Per `executeQueryAsync` la modifica, richiamare il metodo `ClientContext` dell'oggetto `ClientContext` tramite il quale è stata `ClientContext` l' `ClientContext` alla libreria.

L'esempio completo di seguito crea una cartella con un nome basato sul timestamp corrente, quindi apre quella cartella in una finestra di dialogo 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());});
}
```

Ottieni informazioni utente correnti

```
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());})
}
```

Ottieni un elemento della lista per 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());
        }
    );
}
```

Ottieni elementi della lista tramite CAML Query

Esempio di base

Utilizzare il `set_viewXml` metodo dell'oggetto SP.CamlQuery per specificare una query CAML per recuperare gli elementi.

```
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());});
}
```

Paging i risultati di una query CAML

È possibile sfruttare l'elemento `RowLimit` in una query CAML per recuperare solo un sottoinsieme di risultati con ciascuna query.

Utilizzare il metodo `get_listItemCollectionPosition` di una raccolta di voci di elenco per recuperare la posizione corrente, quindi utilizzare tale valore come parametro in un metodo

set_listItemCollectionPosition dell'oggetto `set_listItemCollectionPosition` per recuperare il successivo batch di risultati.

```
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());
    }
}
```

Leggi Lavorare con JavaScript Client Object Model (JSOM) online: https://riptutorial.com/it/sharepoint/topic/1316/lavorare-con-javascript-client-object-model--jsom-

Capitolo 5: Lavorare con le finestre di dialogo modali con JavaScript

Sintassi

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

Parametri

opzioni Proprietà	Descrizione
titolo	Una stringa che contiene il titolo della finestra di dialogo
url	Una stringa che contiene l'URL della pagina che appare nella finestra di dialogo. È necessario specificare **URL** o **html** . **l'url** ha la precedenza su **html** .
html	Un elemento HTML da visualizzare all'interno della finestra di dialogo.
X	L'offset x della finestra di dialogo come valore intero.
y	L'offset y della finestra di dialogo come valore intero.
larghezza	La larghezza della finestra di dialogo come valore intero. Se non specificato e **autosize** è **falso,** la larghezza è impostata su 768px
altezza	L'altezza della finestra di dialogo come valore intero. Se non specificato e **autosize** è **falso,** l'altezza è impostata su 576px
allowMaximize	Un valore booleano che specifica se deve essere mostrato il pulsante **Ingrandisci** .
SHOWMAXIMIZED	Un valore booleano che specifica se la finestra di dialogo si apre ingrandita.
showClose	Un valore booleano che specifica se il pulsante **Chiudi** viene visualizzato nella finestra di dialogo.
dimensione dell'auto	Un valore booleano che specifica se la piattaforma di dialogo gestisce automaticamente il ridimensionamento della finestra di dialogo.

opzioni Proprietà	Descrizione
dialogReturnValueCallback	Un puntatore a funzione che specifica la funzione di callback di ritorno. La funzione accetta due parametri: una *finestra* di *dialogo* di tipo SP.UI.DialogResult Enumeration e un oggetto *returnValue* che contiene tutti i dati restituiti dalla finestra di dialogo.
args	Un oggetto che contiene dati che vengono passati alla finestra di dialogo.

Osservazioni

Lo spazio `SP.UI.ModalDialog` nomi `SP.UI.ModalDialog` stato introdotto nel modello a oggetti JavaScript con SharePoint 2010 ed è disponibile nelle successive versioni di SharePoint 2013, Office365 e 2016.

Materiale di riferimento aggiuntivo:

- Riferimento MSDN per SP.UI.ModalDialog.showModalDialog (opzioni)
- Riferimento MSDN per enumerazione SP.UI.DialogResult

Examples

Esegui un'azione quando una finestra di dialogo è chiusa

```
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;
        }
    }
}
```

Mostra una pagina esistente in una finestra di dialogo

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

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

Mostra una finestra di dialogo personalizzata

```
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);
}
```

Leggi Lavorare con le finestre di dialogo modali con JavaScript online: https://riptutorial.com/it/sharepoint/topic/6868/lavorare-con-le-finestre-di-dialogo-modali-con-javascript

Capitolo 6: Major Releases

Examples

SharePoint 2016

Numero di build	Descrizione	Prodotto
16.0.4366.1000	Aggiornamento cumulativo aprile 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 Beta 2	SharePoint Server 2016

SharePoint 2013

Numero di build	Descrizione
15.0.4623.1001	Giugno 2014
15.0.4631.1001	Luglio 2014
15.0.4641.1001	Agosto 2014
15.0.4649.1001	Settembre 2014
15.0.4659.1001	Ottobre 2014
15.0.4667.1000	Novembre 2014
15.0.4675.1000	Dicembre 2014
15.0.4693.1001	Febbraio 2015
15.0.4701.1001	Marzo 2015
15.0.4711.1000	Aprile 2015 (SP1 REQ)
15.0.4719.1002	Maggio 2015
15.0.4727.1001	Giugno 2015
15.0.4737.1000	Luglio 2015
15.0.4745.1000	Agosto 2015

Numero di build	Descrizione
15.0.4753.1003	Settembre 2015
15.0.4763.1002	Ottobre 2015
15.0.4771.1000	Novembre 2015
15.0.4779.1000	Dicembre 2015
15.0.4787.1000	MS16-004
15.0.4787.1000	Januar 2016
15.0.4797.1001	Febbraio 2016
15.0.4805.1000	Marzo 2016
15.0.4815.1000	Aprile 2016
15.0.4823.1003	Maggio 2016
15.0.4833.1000	Giugno 2016

Fonte: numeri di build di SharePoint 2013 e CU

Leggi Major Releases online: https://riptutorial.com/it/sharepoint/topic/2737/major-releases

Capitolo 7: Rendering lato client SharePoint 2013

introduzione

Il rendering lato client (CSR) è un nuovo concetto introdotto in SharePoint 2013. Fornisce un meccanismo che consente di utilizzare il proprio rendering di output per un set di controlli ospitati in una pagina di SharePoint (visualizzazioni elenco, moduli elenco e risultati di ricerca). Il rendering del sito client è semplicemente quando i dati vengono trasformati utilizzando il client anziché il server. Ciò significa utilizzare tecnologie lato client, come HTML e JavaScript, piuttosto che dover scrivere XSLT.

Examples

Cambia il collegamento ipertestuale di campi / colonne all'interno della visualizzazione elenco utilizzando CSR

L'esempio seguente mostra come modificare il collegamento ipertestuale per il campo " **ID** " e " **Titolo (LinkTitle)** " all'interno della visualizzazione elenco utilizzando CSR.

Passaggio 1: crea un file JS e incolla sotto il codice

```
(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');

})();
```

Passaggio 2: Vai alle proprietà della web part della Visualizzazione elenco e aggiungi il riferimento JS Link a questo file js appena creato (ad esempio ~ sitecollection / SiteAssets / CSRCodeFile.js)

(Nota: indica il tuo JSlink solo in questo formato. "~ Sitecollection / YourJSfFilePath".)

Passaggio 3: Appy and Done

Nascondi colonna dalla visualizzazione elenco di SharePoint utilizzando CSR.

Questo esempio mostra come nascondere un campo "Data" dalla visualizzazione elenco di SharePoint utilizzando 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');

})();
```

Applicare le convalide su Nuovo / Modifica modulo oggetto utilizzando CSR

Supponiamo di avere un elenco di SharePoint e ha quattro campi vale. Titolo, Nome completo, Email, Numero di cellulare ecc. Ora se vuoi applicare la convalida personalizzata nel modulo Nuovo / Modifica elemento, puoi farlo facilmente con il codice CSR. Il sotto menzionato può convalidare le seguenti condizioni nei moduli:

- Valori vuoti nei campi
- Verifica formato ID email con espressione regolare
- Formato numero di cellulare Controllare con espressione regolare
- Il campo Nome completo non deve contenere valori numerici

Passaggio: 1 Creare un file JS, ad esempio `CSRValidations.js` e copia incolla seguendo il codice nel file 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>";
    }

    // Custom validation object to validate email format
    emailValidator = function () {
        emailValidator.prototype.Validate = function (value) {
```

```
            var isError = false;
            var errorMessage = "";

            //Email format Regex expression
            //var emailRejex = /\S+@\S+\.\S+/;
            var emailRejex = /^(([^<>()[\]HYPERLINK
"\\.,;:\s@\"\\.,;:\s@\"]+(\.[^<>()[\]HYPERLINK "\\.,;:\s@\"\\.,;:\s@\"]+)*)|(\".+\"))@((\[[0-
9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;

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

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

        };
    };

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

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

        var formCtx = SPClientTemplates.Utility.GetFormContextForCurrentField(ctx);
        // Register a callback just before submit.

        formCtx.registerGetValueCallback(formCtx.fieldName, function () {
            return document.getElementById('inpTitle').value;
        });

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

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

        formCtx.registerClientValidator(formCtx.fieldName, validators);

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

    // Custom validation object to validate title format
    titleValidator = function () {
        titleValidator.prototype.Validate = function (value) {
            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 ,.'-]+$/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>";
    }
```

Passaggio: 2 Apri il modulo Nuovo elemento nel browser. Modifica la pagina e modifica la web part.

Passaggio: 3 Nelle proprietà delle partizioni Web, Vai a Varie -> Collegamento JS -> incollare il percorso del file js (ad esempio ~ sitecollection / SiteAssets / CSRValidations.js)

Passaggio: 4 Salvare le proprietà e la pagina della web part.

Cambia il nome visualizzato della colonna nella visualizzazione elenco utilizzando CSR

Vi sono casi in cui è necessario modificare il nome visualizzato della colonna in una visualizzazione elenco

ad esempio, il nome della colonna visualizzato nella vista è "IsApprovalNeeded" e si desidera visualizzare come "È necessaria l'approvazione?".

Ovviamente puoi cambiare il nome visualizzato di una colonna cambiando il titolo della colonna nelle impostazioni dell'elenco, ma se vuoi mantenerlo come è nelle impostazioni dell'elenco e modificarlo solo nell'anteprima della pagina, puoi farlo usando CSR (lato client-rendering).

Ecco il codice ...

```
(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');

})();
```

Leggi Rendering lato client SharePoint 2013 online:
https://riptutorial.com/it/sharepoint/topic/8317/rendering-lato-client-sharepoint-2013

Capitolo 8: Servizi REST

Osservazioni

URL endpoint del servizio REST

L'API di accesso client REST è stata introdotta per la prima volta in SharePoint 2010, ma è stata notevolmente ampliata in SharePoint 2013. L'API REST in SharePoint 2010 è accessibile tramite il servizio Web ListData all'indirizzo `/_vti_bin/ListData.svc` . SharePoint 2013 ha introdotto gli `/_api/lists/` e `/_api/web` endpoint, che si comportano in modo leggermente diverso.

Gli URL degli endpoint sopra indicati devono essere preceduti da `http://server/site` cui il `server` rappresenta il nome del server e il `site` rappresenta il nome o il percorso del sito specifico.

URL di esempio per ...	SharePoint 2010	SharePoint 2013
Recupero di una lista:	`/_vti_bin/ListData.svc/ListName`	`/_api/lists('ListGuid')`
Recupero di un oggetto:	`/_vti_bin/ListData.svc/ListName(1)`	`/_api/lists('ListGuid')/items(1)`
Recupero di un Web:	(nessun equivalente)	`/_api/web`

Nonostante le differenze nell'accedere ad elenchi e voci di elenco, lavorare con questi risultati è molto simile in entrambe le versioni.

Si noti che il servizio `ListData.svc` è ancora disponibile in SharePoint 2013 per compatibilità con le versioni precedenti.

Invio di richieste REST

Una richiesta REST può essere inviata tramite un XMLHttpRequest JavaScript nativo o tramite il costrutto wrapper jQuery AJAX.

Sintassi XMLHttpRequest

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

jQuery AJAX Sintassi

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

Per ulteriori dettagli sull'invio di richieste tramite AJAX, consultare la documentazione JavaScript AJAX .

Examples

Lavorare con le liste

Ottenere elementi della lista

Questo esempio mostra come recuperare tutti gli elementi della lista e iterare attraverso di essi. Puoi usare il parametro `top` per richiedere un certo numero di risultati. È inoltre possibile utilizzare il parametro `select` per selezionare determinati campi (`$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);
        }
    });
    }
```

Ottenere un singolo elemento della lista

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);
        }
    });
    }
```

Ottieni elementi elenco con colonne di ricerca

A volte, potresti avere una struttura di lista simile a questa:

Tabella di elenco degli animali

Nome	genere	Descrizione
Titolo	String (testo)	Nome dell'animale
Età	Numero	Quanti anni ha l'animale?
Valore	Moneta	Valore dell'animale
genere	*Ricerca (tabella Tipi di animali)*	Campo di ricerca (fornisce il menu a discesa delle scelte dalla tabella Tipi di animali)

Tabella dei tipi di animali

Nome	genere	Descrizione
Titolo	String (testo)	Nome della specie / tipo di animale (ex maiale)
NumLegs	Numero	Numero di zampe sull'animale

Probabilmente non è l'esempio più serio ma il problema qui è ancora valido. Quando si utilizza la normale richiesta per recuperare i valori dall'elenco di SharePoint, per il `Type` di animale, si otterrà solo un campo chiamato `TypeId` nella risposta JSON. Al fine di espandere questi elementi in una sola chiamata AJAX, nei parametri URL sono richiesti alcuni extra markup.

Questo esempio si applica a più di semplici colonne di ricerca. Quando si utilizzano le colonne `People/Groups` , sono essenzialmente solo ricerche, quindi è possibile estrarre facilmente elementi come `Title` , `EMail` e altri.

Codice di esempio

Nota importante : quando si definiscono i campi che si desidera recuperare dalle colonne di ricerca, è necessario inserire come prefisso il nome del campo con il nome del campo di ricerca nella tabella originale. Ad esempio, se si desidera recuperare l'attributo `NumLegs` dalla colonna di ricerca, è necessario digitare `Type/NumLegs` .

JavaScript

```
// webUrl: 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 getItems(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 getItems 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();
```

Aggiunta di selezioni a un campo di ricerca multivalore

In questo esempio si presuppone che la colonna di ricerca sia denominata `MultiLookupColumnName` e che si desideri impostare il campo di ricerca multipla per cercare gli elementi con ID 1 e 2.

Utilizzando 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
});
```

Utilizzando 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);
```

Elementi di elenco di paging restituiti da una query

Per simulare il paging usando REST puoi fare quanto segue:

1. Usa il parametro `$skip=n` per saltare le prime `n` voci in base al parametro `$orderby`

2. Usa il parametro `$top=n` per restituire le prime `n` voci in base ai parametri `$orderby` e `$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();
            }
        }
    );
}
```

Recupera un ID dell'elemento appena creato nell'elenco di SharePoint

Questo esempio mostra come recuperare un ID di un elemento appena creato utilizzando l'API REST di SharePoint.

Nota :

listName : questa variabile contiene il nome del tuo elenco.

newItemBody - Questo sarà il corpo della richiesta per l'aggiunta di un nuovo elemento nell'elenco.

es. var newItemBody = {__metadata: {'type': 'SP.Data.MyListNameItem'}, Titolo: '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
    });
```

Come eseguire le operazioni CRUD utilizzando l'interfaccia REST di SharePoint 2010

Creare

Per eseguire un'operazione di CREAZIONE tramite REST, è necessario eseguire le seguenti azioni:

Crea una richiesta HTTP usando il verbo `POST` . Utilizzare l'URL del servizio dell'elenco a cui si desidera aggiungere un'entità come destinazione per il POST. Imposta il tipo di contenuto su `application/json` . Serializzare gli oggetti JSON che rappresentano i nuovi elementi dell'elenco come una stringa e aggiungere questo valore all'esempio JavaScript del corpo della richiesta:

```
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);
        }
    });
}
```

uso

```
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));
  }
);
```

Leggere

Per eseguire un'operazione di lettura tramite REST, è necessario eseguire le seguenti azioni:

Creare una richiesta HTTP usando il verbo `GET` . Utilizzare l'URL del servizio dell'elemento di elenco a cui si desidera aggiungere un'entità come destinazione per GET. Imposta il tipo di contenuto su `application/json` . Esempio di 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);
        }
    });
}
```

uso

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

Aggiornare

Per aggiornare un'entità esistente, è necessario eseguire le seguenti azioni:

Crea una richiesta HTTP usando il verbo `POST` . Aggiungi un'intestazione del `X-HTTP-Method` con un valore di `MERGE` . Utilizzare l'URL del servizio dell'elemento dell'elenco che si desidera aggiornare come destinazione per l'intestazione POST Aggiungi un `If-Match` con un valore dell'ETag originale dell'entità o *. Esempio di JavaScript:

```
function updateListItem(webUrl,listName,itemId,itemProperties,success, failure)
{
   getListItemById(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);
   });

}
```

uso

```
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));
  }
);
```

Elimina

Per eliminare un'entità, è necessario eseguire le seguenti azioni:

Crea una richiesta HTTP usando il verbo `POST` . Aggiungi un'intestazione del `X-HTTP-Method` con un valore di `DELETE` . Utilizza l'URL del servizio dell'elemento dell'elenco che desideri aggiornare come destinazione per l'intestazione POST Aggiungi un `If-Match` con un valore dell'ETag originale dell'entità. Esempio di 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);
   });
}
```

uso

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

Leggi Servizi REST online: https://riptutorial.com/it/sharepoint/topic/3045/servizi-rest

Capitolo 9: Utilizzo del modello a oggetti lato server gestito (full trust)

Osservazioni

Gerarchia concettuale

Nella gerarchia concettuale di SharePoint, le **raccolte siti** contengono **siti** , che a loro volta contengono **elenchi** . Una raccolta siti (`SPSite`) non ha `RootWeb` utente esplicita, ma contiene sempre un sito di livello principale (accessibile tramite la proprietà `RootWeb`) ed eventualmente altri siti secondari in tale sito radice. Un sito o web (`SPWeb`) ha un'interfaccia utente e contiene elenchi / raccolte documenti (`SPList`), pagine con webparts e articoli / documenti (`SPListItem`).

Avvertenze sul lato server

- Per creare un'applicazione che utilizza il modello di oggetti lato server di SharePoint, nel progetto di Visual Studio è necessario aggiungere un riferimento all'assembly Microsoft.SharePoint che è elencato in Framework Assembly.
- Le applicazioni che utilizzano il modello Server Side Object (full-trust) possono essere eseguite solo su un server Windows che ospita SharePoint.
- Non è possibile connettersi a un server SharePoint diverso da quello su cui è in esecuzione l'applicazione.

Examples

Hello World (ottenere il titolo del sito)

2013

SharePoint 2013 e le versioni più recenti sono solo a 64 bit e quindi l'assembly / programma deve essere creato anche per il processore a 64 bit.

Subito dopo la creazione del progetto, è necessario spostare il **target** della **piattaforma** da **Any CPU** a **x64** altrimenti si verificherà un errore.

```
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);
            }
        }
    }
}
```

Ciclo continuo attraverso l'intera farm di SharePoint

Utilizzo di PowerShell eseguito da un server Web di 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()
        }
    }
}
```

Recupera elementi di elenco

```
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
    }
}
```

Recupera elementi usando il paging

```
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);
}
```

Ottieni la lista per 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);
}
```

Creazione di una voce di elenco

Quando si crea un nuovo elemento di lista, i suoi campi possono essere impostati usando la sintassi simile agli array di stringhe. Nota che questi campi non sono creati al volo e sono definiti dallo schema della lista. Questi campi (o colonne) devono esistere sul server altrimenti la creazione fallirà. Tutte le voci dell'elenco avranno il campo Titolo. Alcuni elenchi possono avere campi obbligatori che devono essere compilati prima che l'articolo venga pubblicato nell'elenco.

In questo esempio, la lista sta usando il modello degli annunci. Oltre al campo del titolo, l'elenco include il campo Corpo che visualizzerà il contenuto dell'annuncio nella lista.

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

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

Leggi Utilizzo del modello a oggetti lato server gestito (full trust) online: https://riptutorial.com/it/sharepoint/topic/7543/utilizzo-del-modello-a-oggetti-lato-server-gestito--full-trust-

Capitolo 10: Utilizzo del modello CSOM (Managed Client Side Model)

Osservazioni

- La maggior parte degli esempi proviene da MSDN .
- Per creare un'applicazione client gestita .NET che utilizza il modello di oggetto client, è necessario impostare i riferimenti a due DLL di librerie client: Microsoft.SharePoint.Client.dll e Microsoft.SharePoint.Client.Runtime.dll. È possibile trovarlo in% Programmi% \ File comuni \ Microsoft Shared \ estensioni server Web \ 16 \ cartella ISAPI o server SharePoint.
- oppure Installa il pacchetto NuGet di Microsoft.SharePointOnline.CSOM, che funzionerà "on prem" così come in SP O365.
- La maggior parte delle proprietà sono proprietà del valore e prima di accedervi è necessario chiamare esplicitamente clientContext.Load () e clientContext.ExecuteQuery (). Maggiori informazioni qui: Chiama Load ed ExecuteQuery prima di accedere a Proprietà valore

Examples

Ciao mondo (ottenendo il titolo del sito)

Tutte le versioni di SharePoint sono basate su Sites (SPSite (SSOM) o Site (CSOM)) e Web (SPWeb (SSOM) o Web (CSOM)). Nell'interfaccia utente non viene eseguito il rendering di un sito sebbene contenga metadati e funzionalità applicate ai suoi figli. Un web è il blocco base che rende l'interfaccia utente all'utente che accede al sito. Tutti i siti hanno un Web principale che contiene informazioni e / o metadati come le librerie di documenti. Questo esempio mostra una chiamata di base per recuperare il web che si trova sul server `MyServer` sotto i `sites` percorsi virtuali.

```
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. Recupero delle proprietà di un sito 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. Recupero solo delle proprietà specificate di un sito 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. Aggiornamento del titolo e della descrizione di un sito 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. Creazione di un sito 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);
```

Elenco. Recupero di tutte le proprietà di tutti gli elenchi in un sito 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());
}
```

Elenco. Recupero solo delle proprietà specificate degli elenchi

```
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"));
}
```

Elenco. Memorizzazione di elenchi recuperati in una raccolta

```
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"));
}
```

Elenco. Recupero di campi elenco da un sito 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);
        }
    }
}
```

Elenco. Creazione e aggiornamento di un elenco

```
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();
```

Elenco. Aggiungere un campo a un elenco

```
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();
```

Elenco. Cancellare una lista

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

List oList = oWebsite.Lists.GetByTitle("My Announcements List");

oList.DeleteObject();

clientContext.ExecuteQuery();
```

Articolo. Recupero di oggetti da un elenco

```
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"]);
}
```

Articolo. Recupero di oggetti (usando il metodo Include)

Questo esempio mostra come recuperare elementi dal server e ottenere proprietà più profonde di

ciascun elemento dell'elenco. Per impostazione predefinita, il server restituirà solo la quantità minima di dati per rappresentare l'oggetto. Spetta al chiamante richiedere ulteriori informazioni dal server.

```
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);
}
```

Articolo. Recupero di campi specifici da un numero specificato di elementi

```
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"]);
}
```

Articolo. Recupero di elementi da tutti gli elenchi in un sito 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);
}
```

Articolo. Recupero di oggetti usando la posizione di raccolta degli articoli della lista

```
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");
}
```

Articolo. Creazione di una voce di elenco

Quando si crea un nuovo elemento di lista, i suoi campi possono essere impostati usando la sintassi simile agli array di stringhe. Nota che questi campi non sono creati al volo e sono definiti dallo schema della lista. Questi campi (o colonne) devono esistere sul server altrimenti la creazione fallirà. Tutte le voci dell'elenco avranno il campo Titolo. Alcuni elenchi possono avere campi obbligatori che devono essere compilati prima che l'articolo venga pubblicato nell'elenco.

In questo esempio, la lista sta usando il modello degli annunci. Oltre al campo del titolo, l'elenco include il campo Corpo che visualizzerà il contenuto dell'annuncio nella lista.

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

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

oListItem.Update();

clientContext.ExecuteQuery();
```

Articolo. Aggiornamento di una voce di elenco

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

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

oListItem.Update();

clientContext.ExecuteQuery();
```

Articolo. Eliminazione di una voce di elenco

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

oListItem.DeleteObject();

clientContext.ExecuteQuery();
```

Gruppi. Recupero di tutti gli utenti da un gruppo di 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);
}
```

Gruppi. Recupero di proprietà specifiche degli utenti

```
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);
}
```

Gruppi. Recupero di tutti gli utenti in tutti i gruppi di una raccolta siti

```
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);
    }
}
```

Gruppi. Aggiunta di un utente a un gruppo di 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();
```

Ruoli. Creazione di una definizione di ruolo

```
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);
```

Ruoli. Assegnare un utente a un ruolo in un sito 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);
```

Ruoli. Creazione di un gruppo di SharePoint e aggiunta del gruppo a un ruolo

```
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);
}
```

Autorizzazioni. Rompere l'eredità della sicurezza di una lista

```
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();
```

Autorizzazioni. Rompere l'ereditarietà della sicurezza di un documento e aggiungere un utente come lettore

```
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();
```

Autorizzazioni. Rompere l'ereditarietà della sicurezza di un documento e modificare le autorizzazioni di un utente

```
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();
```

Azione personalizzata Aggiunta di un'azione personalizzata dell'utente per gli elementi dell'elenco

```
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();
```

Azione personalizzata Modifica di un'azione personalizzata dell'utente

```
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();
    }
}
```

Azione personalizzata Aggiunta di un'azione personalizzata dell'utente alle azioni del sito di un sito 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. Aggiornamento del titolo di una web part

```
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. Aggiunta di una web part a una pagina

```
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> </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. Eliminazione di una web part da una pagina

```
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();
```

Contesto. Utilizzo di una cache delle credenziali per un'esecuzione elevata del codice

Mentre il side-code del server può essere eseguito con privilegi elevati, non esiste un metodo equivalente per elevare i privilegi nel codice lato client (per ovvi motivi di sicurezza). In alternativa, è possibile specificare credenziali per emulare l'accesso di un account utente o servizio specifico.

Per specificare le credenziali, creare e popolare un oggetto `CredentialCache` , quindi assegnarlo alla proprietà `Credentials` dell'oggetto `ClientContext` .

L'esempio seguente emula l'account del pool di applicazioni e presuppone un ambiente SharePoint 2013 locale con 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
```

```
}
```

Si noti che la concessione dei privilegi elevati dell'account del pool di applicazioni in SharePoint è contraria alle best practice, ma che al suo posto potrebbero essere utilizzate tutte le credenziali di rete pertinenti.

Leggi Utilizzo del modello CSOM (Managed Client Side Model) online: https://riptutorial.com/it/sharepoint/topic/2679/utilizzo-del-modello-csom--managed-client-side-model-

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con sharepoint	Community, Marco, Ryan Gregg, Thriggle, Tom Resing, Zach Koehne
2	App di SharePoint	Sunil sahu
3	Creazione di un'app ospitata dal provider	vinayak hegde
4	Lavorare con JavaScript Client Object Model (JSOM)	Thriggle, yngrdyn
5	Lavorare con le finestre di dialogo modali con JavaScript	Thriggle
6	Major Releases	jjr2527, MikhailSP
7	Rendering lato client SharePoint 2013	Rohit Waghela, Yayati
8	Servizi REST	Aaron, Brock Davis, ocelotsloth, R4mbi, Rohit Waghela, Thriggle
9	Utilizzo del modello a oggetti lato server gestito (full trust)	Lukáš Nešpor, Thriggle
10	Utilizzo del modello CSOM (Managed Client Side Model)	InvoiceGuy, Lukáš Nešpor, MikhailSP, RamenChef, Thriggle, Zach Koehne