



Kostenloses eBook

LERNEN

sharepoint

Free unaffiliated eBook created from
Stack Overflow contributors.

#sharepoint

Inhaltsverzeichnis

Über.....	1
Kapitel 1: Erste Schritte mit Sharepoint.....	2
Bemerkungen.....	2
Versionen.....	2
Examples.....	3
Installation von SharePoint 2016 für eine Serverfarm.....	3
Einführung.....	3
Bedarf.....	3
Installation.....	3
Aufbau.....	4
Farmkonfiguration.....	4
Erstellen Sie ein Webpart mit dem SharePoint Framework.....	5
SharePoint ULS-Protokolle und -Protokollierung.....	6
Werkzeugbau.....	6
Korrelationskennung.....	6
SPMonitoredScope zu My Code hinzufügen.....	6
Kapitel 2: Arbeiten mit JavaScript Client Object Model (JSOM).....	7
Bemerkungen.....	7
Examples.....	8
Bibliotheksinhaltstypen über den Bibliotheksnamen abrufen.....	8
Löschen Sie ein Element in einer Liste.....	8
Erstellen von Elementen oder Ordnern.....	8
Listenelemente erstellen.....	8
Ordner erstellen.....	9
Aktuelle Benutzerinformationen abrufen.....	10
Holen Sie sich ein Listenelement nach ID.....	10
Listenelemente nach CAML-Abfrage abrufen.....	11
Basisbeispiel.....	11
Paging der Ergebnisse einer CAML-Abfrage.....	11

Kapitel 3: Arbeiten mit Managed Client Side Object Model (CSOM)	13
Bemerkungen.....	13
Examples.....	13
Hallo Welt (bekommt den Seitentitel).....	13
Netz. Abrufen der Eigenschaften einer Website.....	14
Netz. Nur angegebene Eigenschaften einer Website abrufen.....	14
Netz. Titel und Beschreibung einer Website aktualisieren.....	14
Netz. Erstellen einer Website.....	14
Liste. Abrufen aller Eigenschaften aller Listen einer Website.....	15
Liste. Nur angegebene Eigenschaften von Listen abrufen.....	15
Liste. Abgerufene Listen in einer Sammlung speichern.....	16
Liste. Listenfelder von einer Website abrufen.....	16
Liste. Liste erstellen und aktualisieren.....	16
Liste. Hinzufügen eines Feldes zu einer Liste.....	17
Liste. Liste löschen.....	17
Artikel. Elemente aus einer Liste abrufen.....	17
Artikel. Elemente abrufen (mithilfe der Include-Methode).....	18
Artikel. Bestimmte Felder aus einer bestimmten Anzahl von Elementen abrufen.....	18
Artikel. Elemente aus allen Listen einer Website abrufen.....	18
Artikel. Abrufen von Elementen mithilfe der Listenelementsammlungsposition.....	19
Artikel. Listenelement erstellen.....	20
Artikel. Aktualisieren eines Listenelements.....	20
Artikel. Löschen eines Listenelements.....	20
Gruppen. Abrufen aller Benutzer aus einer SharePoint-Gruppe.....	20
Gruppen. Abrufen bestimmter Eigenschaften von Benutzern.....	21
Gruppen. Abrufen aller Benutzer in allen Gruppen einer Websitesammlung.....	21
Gruppen. Hinzufügen eines Benutzers zu einer SharePoint-Gruppe.....	22
Rollen Erstellen einer Rollendefinition.....	22
Rollen Zuweisen eines Benutzers zu einer Rolle auf einer Website.....	22
Rollen Erstellen einer SharePoint-Gruppe und Hinzufügen der Gruppe zu einer Rolle.....	23
Berechtigungen. Brechen der Sicherheitsvererbung einer Liste.....	23
Berechtigungen. Brechen der Sicherheitsvererbung eines Dokuments und Hinzufügen eines Benu.....	23
Berechtigungen. Aufheben der Sicherheitsvererbung eines Dokuments und Ändern der Berechtig.....	24

Benutzerdefinierte Aktion Benutzerdefinierte Aktion für Listenelemente hinzufügen.....	24
Benutzerdefinierte Aktion Benutzerdefinierte Aktion ändern.....	25
Benutzerdefinierte Aktion Hinzufügen einer benutzerdefinierten Benutzeraktion zu den Websi.....	25
Webpart. Titel eines Webparts aktualisieren.....	26
Webpart. Webpart zu einer Seite hinzufügen.....	26
Webpart. Löschen eines Webparts von einer Seite.....	27
Kontext. Verwenden eines Cache mit Berechtigungsnachweisen für die erhöhte Ausführung von	27
Kapitel 4: Arbeiten mit modalen Dialogfeldern mit JavaScript.....	29
Syntax.....	29
Parameter.....	29
Bemerkungen.....	30
Examples.....	30
Führen Sie eine Aktion aus, wenn ein Dialogfeld geschlossen ist.....	30
Vorhandene Seite in einem Dialog anzeigen.....	30
Ein benutzerdefiniertes Dialogfeld anzeigen.....	31
Kapitel 5: Clientseitige Wiedergabe von SharePoint 2013.....	32
Einführung.....	32
Examples.....	32
Ändern Sie den Hyperlink von Feldern / Spalten in der Listenansicht mithilfe von CSR.....	32
Spalte mithilfe der CSR aus der SharePoint-Listenansicht ausblenden.....	33
Gültigkeitsprüfungen im Formular "Neu" / "Formular bearbeiten" mit CSR anwenden.....	33
Ändern Sie den Anzeigenamen der Spalte in der Listenansicht mithilfe von CSR.....	38
Kapitel 6: Erstellen einer vom Provider gehosteten App.....	40
Examples.....	40
Entwicklungsumgebung einstellen.....	40
Vorbereitung für die Entwicklerseite.....	41
App in Visual Studio erstellen.....	42
Beginnen wir mit der Codierung.....	46
Vollständige Artikelseite erstellen.....	49
Kapitel 7: Hauptversionen.....	52
Examples.....	52
SharePoint 2016.....	52

SharePoint 2013.....	52
Kapitel 8: Mit dem verwalteten Serverseitenobjektmodell arbeiten (voll vertrauenswürdig).....	54
Bemerkungen.....	54
Konzeptionelle Hierarchie.....	54
Vorsichtsmaßnahmen auf dem Server.....	54
Examples.....	54
Hallo Welt (Website-Titel erhalten).....	54
Durchlaufen der gesamten SharePoint-Farm.....	55
Listenelemente abrufen.....	55
Rufen Sie Elemente mit Paging ab.....	55
Liste per URL abrufen.....	56
Listenelement erstellen.....	56
Kapitel 9: REST-Dienste.....	57
Bemerkungen.....	57
URLs für den REST-Service-Endpunkt.....	57
Senden von REST-Anforderungen.....	57
XMLHttpRequest-Syntax.....	57
jQuery AJAX-Syntax.....	57
Examples.....	58
Mit Listen arbeiten.....	58
Listenelemente mit Suchspalten abrufen.....	59
Tierauflistungstabelle.....	59
Tabelle der Tierarten.....	59
Beispielcode.....	59
Hinzufügen von Auswahlmöglichkeiten zu einem mehrwertigen Suchfeld.....	61
Auslagerunglistenelemente, die von einer Abfrage zurückgegeben wurden.....	62
Rufen Sie eine ID des neu erstellten Elements in der SharePoint-Liste ab.....	63
So führen Sie CRUD-Vorgänge mithilfe der SharePoint 2010-REST-Schnittstelle durch.....	64
Kapitel 10: SharePoint App.....	68
Einführung.....	68
Bemerkungen.....	68

Examples.....	68
SharePoint 2013: Zugriff auf Benutzerprofildienstdaten mithilfe von JSOM in SharePoint 201.....	68
Credits.....	70



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

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

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

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

Kapitel 1: Erste Schritte mit Sharepoint

Bemerkungen

SharePoint kann sich auf ein oder mehrere Produkte aus der Microsoft SharePoint-Produktfamilie beziehen.

- **SharePoint Foundation** : Dies war die zugrunde liegende Technologie für alle SharePoint-Websites und steht für SharePoint 2016 nicht mehr zur Verfügung
- **SharePoint Server** : Dies ist die lokale Version von SharePoint. Sie können einen oder mehrere SharePoint-Server bereitstellen. Es bietet zusätzliche Funktionen für SharePoint Foundation, z. B. BI-Funktionen, Enterprise Content Management und mehr
- **SharePoint Online** : Cloud-basierte Version von SharePoint. Der Kunde muss sich nicht um die Serverinfrastruktur oder die Skalierbarkeit kümmern.

Office 365 ist ein separates Microsoft-Angebot, das den SharePoint Online-Dienst umfasst, obwohl nicht alle Pläne alle SharePoint-Funktionen unterstützen.

Die folgenden Links bieten umfangreiche Funktionsvergleiche zwischen verfügbaren SharePoint-Versionen:

- Lokales SharePoint 2013-System im Vergleich zu SharePoint 2016-lokalem Standort: [Verfügbarkeit von Funktionen für lokale SharePoint-Pläne](#)
- SharePoint-Funktionen in Office 365: [Verfügbarkeit von Funktionen für SharePoint-Pläne](#)
- SharePoint-Funktionen in SharePoint Online (ohne Office 365): [Verfügbarkeit von Funktionen für eigenständige SharePoint-Pläne](#)
- Zusammengefasster Funktionsvergleich zwischen SharePoint 2013 und SharePoint Online: <http://www.buckleyplanet.com/2016/06/sharepoint-online-vs-onprem-feature-comparison.html>

Versionen

Ausführung	Offizieller Name	Veröffentlichungsdatum
Vor 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

Installation von SharePoint 2016 für eine Serverfarm

Einführung

SharePoint 2016 ist die Version 16 der SharePoint-Produktfamilie. Es wurde am 4. Mai 2016 veröffentlicht. In diesem Beispiel wird die Installation von SharePoint 2016 mithilfe der Konfiguration mit einer Serverfarm beschrieben. Diese Konfiguration umfasst die Grundlagen zum Einrichten einer SharePoint-Farm, ohne dass mehrere Server erforderlich sind. Beachten Sie, dass die behandelten Szenarien einer einzelnen Serverfarm normalerweise auf Entwicklungsszenarien und sehr kleine Produktionsszenarien beschränkt sind.

Bedarf

Vor der Installation von SharePoint muss die Basisumgebung eingerichtet werden. SharePoint speichert Dokumente sowie Metadaten, Protokolle, benutzerdefinierte Anwendungen, Anpassungen und vieles mehr. Stellen Sie sicher, dass über den Basisanforderungen genügend Speicherplatz und RAM zur Verfügung stehen.

- 4 Kerne auf einem 64-Bit-kompatiblen Prozessor
- 12 - 24 GB RAM (je nach Test- oder Produktbereitstellung)
- 80 GB Festplatte für das System
- 100 GB Festplatte als zweites Laufwerk
- Server mit 64-Bit-Windows Server 2012 R2 oder Technical Preview-Schwellenwert
- SQL Server 2014 oder SQL Server 2016
- .NET Framework 4.5.2 oder .NET Framework 4.6
- Domäne trat Computer und delegierten Farmdienstkonten bei

Alle anderen Voraussetzungen können manuell installiert oder mithilfe des in der SharePoint-Installation enthaltenen Installationsprogramms für SharePoint-Voraussetzungen ausgeführt werden.

Installation

- Führen Sie das Installationsprogramm für Voraussetzungen aus. Möglicherweise muss der Server neu gestartet werden, bevor Sie fortfahren
- Führen Sie Setup.exe aus der SharePoint-Installation aus
- Geben Sie den Lizenzschlüssel ein
- Akzeptieren Sie die Lizenzvereinbarung
- Wählen Sie auf der Registerkarte Server Type "Complete" aus
- Das Setup sollte erfolgreich abgeschlossen werden
- Lassen Sie auf der vollständigen Seite das Kontrollkästchen neben dem Assistenten zum

Ausführen von Produktkonfigurationen aktiviert, und klicken Sie auf Schließen

Aufbau

Wenn Sie mit dem vorherigen Schritt fortfahren, sollte der SharePoint 2016-Produktkonfigurations-Assistent automatisch geöffnet werden. Wenn das Feld nicht angezeigt wird oder Sie die Konfiguration später ausführen, öffnen Sie den Konfigurationsassistenten über Start -> SharePoint 2016-Produkte -> SharePoint 2016-Produktkonfigurations-Assistent.

- Klicken Sie auf der Begrüßungsseite auf Weiter
- In einem modalen Dialogfeld werden einige Dienste angezeigt, die während der Konfiguration neu gestartet werden können. Es wurde noch nichts installiert, also klicken Sie auf Ja
- Fügen Sie den Datenbankserver für die Farm hinzu
 - Geben Sie den Namen des Computers ein, auf dem SQL Server ausgeführt wird. In diesem Fall handelt es sich um den lokalen Computer
 - Geben Sie den Namen der Konfigurationsdatenbank ein oder behalten Sie den Standardnamen SharePoint_Config bei
 - Geben Sie den Benutzernamen des Domänendienstbenutzers ein, der auf die Datenbank zugreifen soll (in Form von DOMÄNE \ Benutzer). * Geben Sie das Kennwort für den Domänenbenutzer ein
 - Klicken Sie auf Weiter, wenn Sie fertig sind
- Geben Sie das Farmpasswort ein. Dies wird verwendet, wenn zusätzliche Server mit der neuen Farm verbunden werden
- Wählen Sie die Rolle Single Server Farm aus
- Konfigurieren Sie die Central Admin-Webanwendung (über die SharePoint von den Farmadministratoren verwaltet wird). Wählen Sie die Portnummer und den Typ des Authentifizierungsverbundes aus (NTLM oder Negotiate (Kerberos)).
- Überprüfen Sie die Einstellungen auf den letzten Seiten und nehmen Sie gegebenenfalls Änderungen vor
- Wenn Sie fertig sind, führen Sie die Konfiguration aus. Dies kann einige Minuten dauern
- Wenn Sie fertig sind, öffnen Sie den Assistenten, um die Central Admin-Site zu öffnen
- Bei einem Fehler können Sie die Protokolle im Ordner% COMMONPROGRAMFILES% \ Microsoft Shared \ Web Server Extensions \ 16 \ LOG untersuchen

Farmkonfiguration

Sobald die zentrale Web-App, die Konfigurationsdatenbank und der zentrale Administrator eingerichtet sind, können Sie die Farm zur Verwendung für Benutzer oder für die Entwicklung konfigurieren. Sie können den Speicherort der Central Admin-Site mit einem Lesezeichen versehen oder über eine Verknüpfung an demselben Speicherort wie der Produktkonfigurations-Assistent darauf zugreifen.

- Wenn Sie die Konfiguration später starten, klicken Sie auf Schnellstart -> Konfigurationsassistenten -> Farmkonfigurationsassistent

- Wenn Sie den Assistenten vom Installationsschritt aus starten, klicken Sie auf Assistenten starten
- Wählen Sie aus, ob Sie am Kundenverbesserungsprogramm teilnehmen möchten, indem Sie auf Ja oder Nein klicken
- Wählen Sie auf der Farmkonfigurationsseite das Domänenkonto aus, das Hintergrunddienste in der Farm ausführen soll
 - Dieses Konto kann zwar mit dem Datenbankkonto identisch sein, kann sich jedoch bei der Trennung von Rollen und Berechtigungen unterscheiden
 - Geben Sie das Konto als DOMÄNE \ Benutzer ein
- Überprüfen Sie die Dienste, die in der Farm verfügbar sein sollen, auf der Seite Dienste
- Erstellen Sie die erste Websitesammlung in der Farm (dieser Schritt kann zu einem späteren Zeitpunkt übersprungen werden).
 - Geben Sie den Titel, die Beschreibung und die Webadresse der Websitesammlung (normalerweise die erste Website befindet sich im Serverstammverzeichnis) und die Vorlage ein
 - Die meisten Dinge können geändert werden (Titel, Beschreibung). Sie können leicht geändert werden. Andere, wie die Web-URL, benötigen jedoch viel mehr Arbeit, um sie zu ändern. Die Vorlage kann auch nicht ohne weiteres rückgängig gemacht werden. In SharePoint sind jedoch zahlreiche Anpassungen möglich, sodass Sie jede Basisvorlage verwenden und den Stil und das Layout der Website konvertieren können
- Wenn Sie mit der Konfiguration fertig sind, klicken Sie auf Fertig stellen

Die Farm und die erste Websitesammlung sind jetzt für die Verwendung konfiguriert.

Erstellen Sie ein Webpart mit dem SharePoint Framework

dev.office.com/sharepoint ist ein großartiger Ort, um mit dem SharePoint Framework zu beginnen.

Das SharePoint Framework ist ein moderner, clientseitiger Ansatz für die SharePoint-Entwicklung, der ursprünglich auf Office Online in Office 365 ausgerichtet war. Mit dem SharePoint Framework erstellte Webparts sind ein neuer Typ von Webparts, die sowohl auf vorhandenen SharePoint-Seiten als auch auf SharePoint hinzugefügt werden können neue SharePoint-Seiten.

Für diesen Prozess, der unter [Erstellen Sie Ihren ersten SharePoint-Client-seitigen Webpart \(Hello World Teil 1\)](#) gehostet wird, gibt es ein gutes Beispiel für die Welt. Alle Beispiele auf dev.office.com stehen für Community-Beiträge über github zur Verfügung.

Die grundlegenden Schritte von Hello World im SharePoint Framework sind:

1. Generieren Sie das Skelett des Projekts mit [dem Yeoman SharePoint Generator](#) .

```
yo @ microsoft / SharePoint
```

2. Bearbeiten Sie den generierten Code in einem Editor Ihrer Wahl. Die Unterstützung für [Visual Studio Code](#) ist auf allen Plattformen stark.

3. Zeigen Sie eine Vorschau des Webparts mit gulp und der lokalen SharePoint Workbench an

```
Schluck servieren
```

4. Vorschau in Ihrer SharePoint Online-Umgebung

Gehen Sie zur folgenden URL: ' https://your-sharepoint-site/_layouts/workbench.aspx '

SharePoint ULS-Protokolle und -Protokollierung

Der Unified Logging Service (ULS) von SharePoint bietet Support- und Debugging-Funktionen für Benutzer und Entwickler. Das Verständnis der Lesung der Protokolle ist ein wichtiger erster Schritt zur Behebung von Problemen.

Werkzeugbau

Microsoft stellt den [ULS Viewer](#) bereit, um alte Protokolle und Protokolle zu lesen, in die aktuell geschrieben wird, während die Farm ausgeführt wird. Es kann auch filtern und die Formatierung auf Protokolle anwenden, um ein Problem einzugrenzen.

Korrelationskennung

Um ein Problem zu isolieren, ist es hilfreich, nur eine bestimmte Korrelations-ID zu betrachten. Jede Korrelations-ID ist einer Anforderung oder Ende-an-Ende-Aktion des Systems zugeordnet (z. B. einem Zeitberater). Wenn bei der Wiedergabe einer Webseite ein Problem auftritt, wird durch das Auffinden der Anforderung in den ULS-Protokollen und durch das Isolieren der entsprechenden Korrelations-ID das gesamte Rauschen aus den anderen Protokollen entfernt, wodurch das Problem lokalisiert werden kann.

SPMonitoredScope zu My Code hinzufügen

Eine Möglichkeit zum Hinzufügen der Protokollierung und einiger Leistungsüberwachung ist das Hinzufügen von `SPMonitoredScope` zu Ihrem Code.

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

Dieser Code protokolliert den Beginn und das Ende Ihrer Anforderungen sowie einige Leistungsdaten. Wenn Sie Ihren eigenen benutzerdefinierten Monitor erstellen, der `ISPScopePerformanceMonitor` implementiert, können Sie den Trace-Level oder die maximale Ausführungszeit für einen Satz von Code festlegen.

Erste Schritte mit Sharepoint online lesen: <https://riptutorial.com/de/sharepoint/topic/950/erste-schritte-mit-sharepoint>

Kapitel 2: Arbeiten mit JavaScript Client Object Model (JSOM)

Bemerkungen

Hintergrund

Das JavaScript-Objektmodell wurde in SharePoint 2010 eingeführt. Es macht auf Clientseite viele Objekte verfügbar, auf die zuvor nur durch serverseitigen Code oder über dedizierte Webdienste zugegriffen werden konnte.

Einbetten von JavaScript in SharePoint-Seiten

In SharePoint 2013 können Sie Ihr JavaScript in einem Skript-Editor-Webpart einfügen.

In SharePoint 2010 können Sie die Eigenschaft "Content Link" eines Content Editor-Webparts verwenden, um eine Verknüpfung zu einer HTML-Datei herzustellen, die Ihr eingebettetes Skript enthält.

Objektreferenz

Die Konstrukteure, Methoden und Eigenschaften aller Objekte in dem gefundenen `SP` - Namespace in der Sharepoint 2013 - Clientobjektmodell Referenz dokumentieren [hier](#) .

Die Referenz zum SharePoint 2010-JavaScript-Clientobjektmodell ist [hier](#) verfügbar.

Das asynchrone Programmiermuster von JSOM

Bei der Verwendung des JavaScript-Clientobjektmodells folgt Code im Allgemeinen dem folgenden Muster:

1. `ClientContext` Sie ein `ClientContext` Objekt.
2. Verwenden Sie das `ClientContext` Objekt, um Objekte abzurufen, die Entitäten im SharePoint-Objektmodell darstellen, z. B. Listen, Ordner und Ansichten.
3. Warteschlangenweisungen für die Objekte. Diese Anweisungen werden noch nicht an den Server übermittelt.
4. Verwenden Sie die `load` Funktion sagen `ClientContext` , welche Informationen Sie vom Server empfangen möchten.
5. Rufen Sie das `ClientContext` Objekt `executeQueryAsync` Funktion , um die Warteschlangen Anweisungen an den Server zu senden, vorbei an zwei Callback - Funktionen auf Erfolg oder Misserfolg laufen.
6. Arbeiten Sie in der Rückruffunktion mit den vom Server zurückgegebenen Ergebnissen.

Alternativen

Zu den clientseitigen Alternativen zu JSOM gehören die Webdienste, [REST-Endpunkte](#) von

Examples

Bibliotheksinhaltstypen über den Bibliotheksnamen abrufen

```
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());
}
```

Löschen Sie ein Element in einer Liste

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

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

Erstellen von Elementen oder Ordnern

Listenelemente erstellen

```
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());
});
}

```

Das obige Beispiel zeigt, dass ein Listenelement mit den folgenden Schritten erstellt wird:

1. Rufen Sie die `addItem` Methode eines `addItem` auf, um ein `addItem`
2. Rufen Sie die Methode `set_item` für das resultierende Listenelementobjekt auf, um jeden `set_item` wie gewünscht `set_item`
3. Rufen Sie die `update` für das Listenelementobjekt auf, um anzugeben, dass die Änderungen festgeschrieben werden sollen
4. Rufen Sie die Methode `executeQueryAsync` des Client- `executeQueryAsync` auf, um die Anweisungen in der Warteschlange auszuführen

Beachten Sie, dass Sie **nicht** passieren müssen, um das neue Element - Objekt an den Client - Kontext `load` das Element Methode zu erstellen. Dieser Schritt ist nur erforderlich, wenn Sie Feldwerte des Elements vom Server abrufen möchten.

Ordner erstellen

Das Erstellen eines Ordners ähnelt dem Hinzufügen eines Elements zu einer Liste. Der Unterschied besteht darin, dass man zuerst ein erstellen muß `ListItemCreationInformation` Objekt und legen Sie seine `underlyingObjectType` Eigenschaft auf `SP.FileSystemObjectType.folder` und seine `leafName` Eigenschaft auf den gewünschten Namen des neuen Ordners.

Das Objekt wird dann in der `addItem` Methode der Bibliothek als Parameter `addItem`, um den Ordner zu erstellen.

```

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

```

`executeQueryAsync` zum `ClientContext` der Änderung die `executeQueryAsync` Methode des `ClientContext` Objekts auf, über das auf die Bibliothek zugegriffen wurde.

Das vollständige Beispiel unten erstellt einen Ordner mit einem Namen, der auf dem aktuellen Zeitstempel basiert, und öffnet diesen Ordner in einem modalen Dialogfeld.

```

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());});
}

```

Aktuelle Benutzerinformationen abrufen

```

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());});
}

```

Holen Sie sich ein Listenelement nach 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());
}
);
}

```

Listenelemente nach CAML-Abfrage abrufen

Basisbeispiel

Verwenden Sie die `set_viewXml` Methode des `SP.CamlQuery`-Objekts, um eine CAML-Abfrage zum Abrufen von Elementen anzugeben.

```

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 der Ergebnisse einer CAML-Abfrage

Sie können das `RowLimit` Element in einer CAML-Abfrage dazu verwenden, mit jeder Abfrage nur eine Teilmenge der Ergebnisse abzurufen.

Verwenden Sie die `get_listItemCollectionPosition` Methode einer Listenelementauflistung, um die

aktuelle Position abzurufen, und verwenden Sie diesen Wert als Parameter in der `set_listItemCollectionPosition` Methode eines `SP.CamlQuery`-Objekts, um den nächsten Stapel von Ergebnissen abzurufen.

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

Arbeiten mit JavaScript Client Object Model (JSOM) online lesen:

<https://riptutorial.com/de/sharepoint/topic/1316/arbeiten-mit-javascript-client-object-model-jsom>

Kapitel 3: Arbeiten mit Managed Client Side Object Model (CSOM)

Bemerkungen

- Die meisten Beispiele stammen aus [MSDN](#) .
- Zum Erstellen einer von .NET verwalteten Clientanwendung, die das Clientobjektmodell verwendet, müssen Sie Verweise auf zwei Clientbibliotheks-DLLs festlegen: Microsoft.SharePoint.Client.dll und Microsoft.SharePoint.Client.Runtime.dll. Sie finden es im Ordner% ProgramFiles% \ Gemeinsame Dateien \ Microsoft Shared \ Webserver-Erweiterungen \ 16 \ ISAPI oder auf Ihrem SharePoint-Server.
- oder Installieren Sie das Microsoft.SharePointOnline.CSOM-NuGet-Paket, das sowohl unter "Prem" als auch in SP O365 funktioniert.
- Die meisten Eigenschaften sind Werteigenschaften. Bevor Sie darauf zugreifen, müssen Sie clientContext.Load () und clientContext.ExecuteQuery () explizit aufrufen. Weitere Informationen finden Sie hier: [Laden und Ausführen von Abfragen vor dem Zugriff auf Werteigenschaften](#)

Examples

Hallo Welt (bekommt den Seitentitel)

Alle SharePoint-Versionen basieren auf Sites (SPSite (SSOM) oder Site (CSOM)) und Webs (SPWeb (SSOM) oder Web (CSOM)). Eine Site wird nicht in der Benutzeroberfläche dargestellt, obwohl sie Metadaten und Features enthält, die auf ihre untergeordneten Elemente angewendet werden. Ein Web ist der grundlegende Baustein, der dem Benutzer, der auf die Site zugreift, eine Benutzeroberfläche darstellt. Alle Sites verfügen über ein Stammweb, das Informationen und / oder Metadaten wie Dokumentbibliotheken enthält. Dieses Beispiel zeigt eine Basisrufsteuerung die Bahn zu holen auf dem Server `MyServer` unter den virtuellen Pfad `sites` .

```
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);
    }
}
}
}

```

Netz. Abrufen der Eigenschaften einer Website

```

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

```

Netz. Nur angegebene Eigenschaften einer Website abrufen

```

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);

```

Netz. Titel und Beschreibung einer Website aktualisieren

```

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();

```

Netz. Erstellen einer Website

```

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

ClientContext clientContext = new ClientContext(siteUrl);

```

```

Web oWebsite = clientContext.Web;

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

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

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

clientContext.ExecuteQuery();

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

```

Liste. Abrufen aller Eigenschaften aller Listen einer Website

```

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

clientContext.Load(collList);

clientContext.ExecuteQuery();

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

```

Liste. Nur angegebene Eigenschaften von Listen abrufen

```

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

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

clientContext.ExecuteQuery();

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

```

Liste. Abgerufene Listen in einer Sammlung speichern

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

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

clientContext.ExecuteQuery();

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

Liste. Listenfelder von einer Website abrufen

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

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

clientContext.ExecuteQuery();

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

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

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

Liste. Liste erstellen und aktualisieren

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

ListCreationInformation listCreationInfo = new ListCreationInformation();
```

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

List oList = oWebsite.Lists.Add(listCreationInfo);

clientContext.ExecuteQuery();
```

Liste. Hinzufügen eines Feldes zu einer Liste

```
ClientContext clientContext = new ClientContext(siteUrl);

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

SP.Field oField = oList.Fields.AddFieldAsXml("<Field DisplayName='MyField' Type='Number' />",
    true, AddFieldOptions.DefaultValue);

SP.FieldNumber fieldNumber = clientContext.CastTo<FieldNumber>(oField);
fieldNumber.MaximumValue = 100;
fieldNumber.MinimumValue = 35;

fieldNumber.Update();

clientContext.ExecuteQuery();
```

Liste. Liste löschen

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

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

oList.DeleteObject();

clientContext.ExecuteQuery();
```

Artikel. Elemente aus einer Liste abrufen

```
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"]);
}
```

Artikel. Elemente abrufen (mithilfe der Include-Methode)

Dieses Beispiel zeigt, wie Sie Elemente vom Server abrufen und tiefere Eigenschaften für jedes Listenelement erhalten. Standardmäßig gibt der Server nur die Mindestdatenmenge zurück, um das Objekt darzustellen. Es liegt am Anrufer, zusätzliche Informationen vom Server anzufordern.

```
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);
}
```

Artikel. Bestimmte Felder aus einer bestimmten Anzahl von Elementen abrufen

```
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"]);
}
```

Artikel. Elemente aus allen Listen einer Website abrufen


```

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);
}

```

Artikel. Abrufen von Elementen mithilfe der Listenelementsammlungsposition

```

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");
}

```

Artikel. Listenelement erstellen

Beim Erstellen eines neuen Listenelements können die Felder mit einer Syntax ähnlich der String-Arrays festgelegt werden. Beachten Sie, dass diese Felder nicht direkt erstellt werden und vom Schema der Liste definiert werden. Diese Felder (oder Spalten) müssen auf dem Server vorhanden sein, andernfalls schlägt die Erstellung fehl. Alle Listenelemente haben das Titelfeld. Einige Listen enthalten möglicherweise erforderliche Felder, die ausgefüllt werden müssen, bevor das Element in der Liste veröffentlicht wird.

In diesem Beispiel verwendet die Liste die Ankündigungsvorlage. Neben dem Titelfeld enthält die Liste das Textfeld, in dem der Inhalt der Ankündigung in der Liste angezeigt wird.

```
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();
```

Artikel. Aktualisieren eines Listenelements

```
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();
```

Artikel. Löschen eines Listenelements

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

oListItem.DeleteObject();

clientContext.ExecuteQuery();
```

Gruppen. Abrufen aller Benutzer aus einer SharePoint-Gruppe

```
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);
}

```

Gruppen. Abrufen bestimmter Eigenschaften von Benutzern

```

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);
}

```

Gruppen. Abrufen aller Benutzer in allen Gruppen einer Websitesammlung

```

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);
    }
}

```

Gruppen. Hinzufügen eines Benutzers zu einer SharePoint-Gruppe

```
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();
```

Rollen Erstellen einer Rollendefinition

```
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);
```

Rollen Zuweisen eines Benutzers zu einer Rolle auf einer Website

```
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);

```

Rollen Erstellen einer SharePoint-Gruppe und Hinzufügen der Gruppe zu einer Rolle

```

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);
}

```

Berechtigungen. Brechen der Sicherheitsvererbung einer Liste

```

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

oList.BreakRoleInheritance(true, false);

oContext.ExecuteQuery();

```

Berechtigungen. Brechen der Sicherheitsvererbung eines Dokuments und Hinzufügen eines Benutzers als Leser

```

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();

```

Berechtigungen. Aufheben der Sicherheitsvererbung eines Dokuments und Ändern der Berechtigungen eines Benutzers

```

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();

```

Benutzerdefinierte Aktion Benutzerdefinierte Aktion für Listenelemente hinzufügen

```

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();

```

Benutzerdefinierte Aktion Benutzerdefinierte Aktion ändern

```

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();
    }
}

```

Benutzerdefinierte Aktion Hinzufügen einer benutzerdefinierten Benutzeraktion zu den Websiteaktionen einer Website

```

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();
```

Webpart. Titel eines Webparts aktualisieren

```
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();
```

Webpart. Webpart zu einer Seite hinzufügen

```
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/mscont1.gif</PartImageLarge><IsIncludedFilter />" +
    "<Assembly>Microsoft.SharePoint, Version=13.0.0.0, Culture=neutral, " +
    "PublicKeyToken=94de0004b6e3fcc5</Assembly>" +
    "<TypeName>Microsoft.SharePoint.WebPartPages.ContentEditorWebPart</TypeName>" +
    "<ContentLink xmlns=\"http://schemas.microsoft.com/WebPart/v2/ContentEditor\" />" +
    "<Content xmlns=\"http://schemas.microsoft.com/WebPart/v2/ContentEditor\">" +
    "<![CDATA[This is a first paragraph!<DIV>&nbsp;</DIV>And this is a second" +
    "paragraph.]]></Content>" +
    "<PartStorage xmlns=\"http://schemas.microsoft.com/WebPart/v2/ContentEditor\"
```



```

/></WebPart>";

WebPartDefinition oWebPartDefinition = limitedWebPartManager.ImportWebPart(xmlWebPart);

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

oClientContext.ExecuteQuery();

```

Webpart. Löschen eines Webparts von einer Seite

```

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();

```

Kontext. Verwenden eines Cache mit Berechtigungsnachweisen für die erhöhte Ausführung von Code

Während Server-Side-Code mit erhöhten Berechtigungen ausgeführt werden kann, gibt es keine entsprechende Methode zum Erhöhen von Berechtigungen in clientseitigem Code (aus offensichtlichen Sicherheitsgründen). Alternativ können Sie Anmeldeinformationen angeben, um den Zugriff eines bestimmten Benutzers oder Dienstkontos zu emulieren.

Erstellen Sie zum `ClientContext` von `Credentials` ein `CredentialCache` Objekt, `ClientContext` es aus, und weisen Sie es der `Credentials` Eigenschaft Ihres `ClientContext` Objekts zu.

Das folgende Beispiel emuliert das Anwendungspoolkonto und setzt eine lokale SharePoint 2013-Umgebung mit NTLM voraus.

```

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
}
```

Beachten Sie, dass das Erteilen erweiterter Berechtigungen für das Anwendungspoolkonto in SharePoint gegen die bewährte Methode verstößt, dass jedoch relevante Netzwerkanmeldeinformationen verwendet werden können.

Arbeiten mit Managed Client Side Object Model (CSOM) online lesen:

<https://riptutorial.com/de/sharepoint/topic/2679/arbeiten-mit-managed-client-side-object-model--csom->

Kapitel 4: Arbeiten mit modalen Dialogfeldern mit JavaScript

Syntax

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

Parameter

Optionen Eigenschaft	Beschreibung
Titel	Eine Zeichenfolge, die den Titel des Dialogs enthält
URL	Eine Zeichenfolge, die die URL der Seite enthält, die im Dialogfeld angezeigt wird. Es muss entweder URL oder HTML angegeben werden. URL hat Vorrang vor HTML .
html	Ein HTML-Element, das im Dialog angezeigt werden soll.
x	Der X-Versatz des Dialogs als ganzzahliger Wert.
y	Der y-Versatz des Dialogs als ganzzahliger Wert.
Breite	Die Breite des Dialogs als ganzzahliger Wert. Wenn unspecified und autosize false ist, wird die Breite auf 768px gesetzt
Höhe	Die Höhe des Dialogs als ganzzahliger Wert. Wenn nicht angegeben und autosize falsch ist, wird die Höhe auf 576px gesetzt
allowMaximize	Ein boolescher Wert, der angibt, ob die Schaltfläche " Maximieren " angezeigt werden soll.
showMaximized	Ein boolescher Wert, der angibt, ob der Dialog maximiert geöffnet wird.
showClose	Ein boolescher Wert, der angibt, ob die Schaltfläche Schließen im Dialogfeld angezeigt wird.
automatische Größenanpassung	Ein boolescher Wert, der angibt, ob die Dialogplattform die Größenanpassung von Dialogen automatisch übernimmt.
dialogReturnValueCallback	Ein Funktionszeiger, der die Return Callback-Funktion angibt.

Optionen Eigenschaft	Beschreibung
	Die Funktion <i>akzeptiert</i> zwei Parameter: ein <i>dialogResult</i> vom Typ SP.UI.DialogResult-Enumeration und ein <i>returnValue</i> -Objekt, das alle vom Dialog zurückgegebenen Daten enthält.
args	Ein Objekt, das Daten enthält, die an das Dialogfeld übergeben werden.

Bemerkungen

Der Namespace `SP.UI.ModalDialog` wurde in das [JavaScript-Objektmodell](#) mit SharePoint 2010 eingeführt und ist in den nachfolgenden SharePoint-Versionen 2013, Office365 und 2016 verfügbar.

Zusätzliche Referenzmaterialien:

- [MSDN-Referenz für SP.UI.ModalDialog.showModalDialog \(Optionen\)](#)
- [MSDN-Referenz für SP.UI.DialogResult-Enumeration](#)

Examples

Führen Sie eine Aktion aus, wenn ein Dialogfeld geschlossen ist

```
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;
        }
    }
}
```

Vorhandene Seite in einem Dialog anzeigen

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

function showDialog() {
```

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

Ein benutzerdefiniertes Dialogfeld anzeigen

```
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);  
}
```

Arbeiten mit modalen Dialogfeldern mit JavaScript online lesen:

<https://riptutorial.com/de/sharepoint/topic/6868/arbeiten-mit-modalen-dialogfeldern-mit-javascript>

Kapitel 5: Clientseitige Wiedergabe von SharePoint 2013

Einführung

Das Client Side Rendering (CSR) ist ein neues Konzept, das in SharePoint 2013 eingeführt wurde. Es bietet Ihnen einen Mechanismus, mit dem Sie Ihren eigenen Ausgabe-Renderer für eine Reihe von Steuerelementen verwenden können, die auf einer SharePoint-Seite gehostet werden (Listenansichten, Listenformulare) und Suchergebnisse). Client-Site-Rendering ist einfach, wenn die Daten mit dem Client und nicht mit dem Server umgewandelt werden. Dies bedeutet, dass clientseitige Technologien wie HTML und JavaScript verwendet werden müssen, anstatt XSLT schreiben zu müssen.

Examples

Ändern Sie den Hyperlink von Feldern / Spalten in der Listenansicht mithilfe von CSR

Das folgende Beispiel zeigt, wie Sie den Hyperlink für das Feld " ID " und " Titel (LinkTitle) " in der **Listenansicht** mithilfe von CSR **ändern** .

Schritt 1: Erstellen Sie eine JS-Datei und fügen Sie den folgenden Code ein

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

```

Schritt 2: Gehen Sie zu den Webpart-Eigenschaften der Listenansicht und fügen Sie der neu erstellten js-Datei eine JS-Link-Referenz hinzu (z. B. ~ sitecollection / SiteAssets / CSRCODEFILE.js).

(Hinweis: Verweisen Sie Ihren JSlink nur in diesem Format. "~ Sitecollection / YourJSFilePath".)

Schritt 3: Appy und Fertig

Spalte mithilfe der CSR aus der SharePoint-Listenansicht ausblenden.

In diesem Beispiel wird veranschaulicht, wie ein Feld "Datum" mithilfe der CSR aus der SharePoint-Listenansicht ausgeblendet wird.

```

(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');
}) ();

```

Gültigkeitsprüfungen im Formular "Neu" / "Formular bearbeiten" mit CSR anwenden

Angenommen, wir haben eine SharePoint-Liste und vier Felder, nämlich. Titel, vollständiger Name, E-Mail-Adresse, Handynummer usw. Wenn Sie nun die benutzerdefinierte Validierung im Formular "Neues Element / Element bearbeiten" anwenden möchten, können Sie dies ganz einfach mit CSR-Code tun. Das Folgende kann die folgenden Bedingungen in Formularen bestätigen:

- Leere Werte in Feldern

- E-Mail-ID-Formatprüfung mit regulärem Ausdruck
- Handy-Nummernformat Überprüfen Sie den regulären Ausdruck
- Das Feld für den vollständigen Namen darf keine numerischen Werte enthalten

Schritt: 1 Erstellen Sie eine JS-Datei, sagen Sie `CSRValidations.js` und kopieren Sie den folgenden Code in die JS-Datei

```
(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 ,.'-]+$&#x2F;i;

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

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

    };

};

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

```

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

Schritt: 2 Öffnen Sie das neue Elementformular im Browser. Seite bearbeiten und Webpart bearbeiten.

Schritt: 3 Gehen Sie in den Eigenschaften des Webparts zu Verschiedenes -> JS-Link -> fügen Sie den Pfad Ihrer js-Datei ein (zB ~ sitecollection / SiteAssets / CSRValidations.js)

Schritt: 4 Speichern Sie die Webpart-Eigenschaften und die Seite.

Ändern Sie den Anzeigenamen der Spalte in der Listenansicht mithilfe von CSR

Es gibt Fälle, in denen Sie den Anzeigenamen der Spalte in einer Listenansicht ändern müssen

ZB ist der in der Ansicht angezeigte Spaltenname "IsApprovalNeeded" und Sie möchten als "Ist eine Genehmigung erforderlich?" angezeigt werden.

Sie können den Anzeigenamen einer Spalte natürlich ändern, indem Sie den Spaltentitel in den Listeneinstellungen ändern. Wenn Sie jedoch den Spaltennamen in den Listeneinstellungen beibehalten und ihn nur in der Seitenvorschau ändern möchten, können Sie dies tun CSR (Client-Side-Rendering).

Hier ist der Code ...

```
(function () {

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

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

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

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

Clientseitige Wiedergabe von SharePoint 2013 online lesen:

<https://riptutorial.com/de/sharepoint/topic/8317/clientseitige-wiedergabe-von-sharepoint-2013>

Kapitel 6: Erstellen einer vom Provider gehosteten App

Examples

Entwicklungsumgebung einstellen

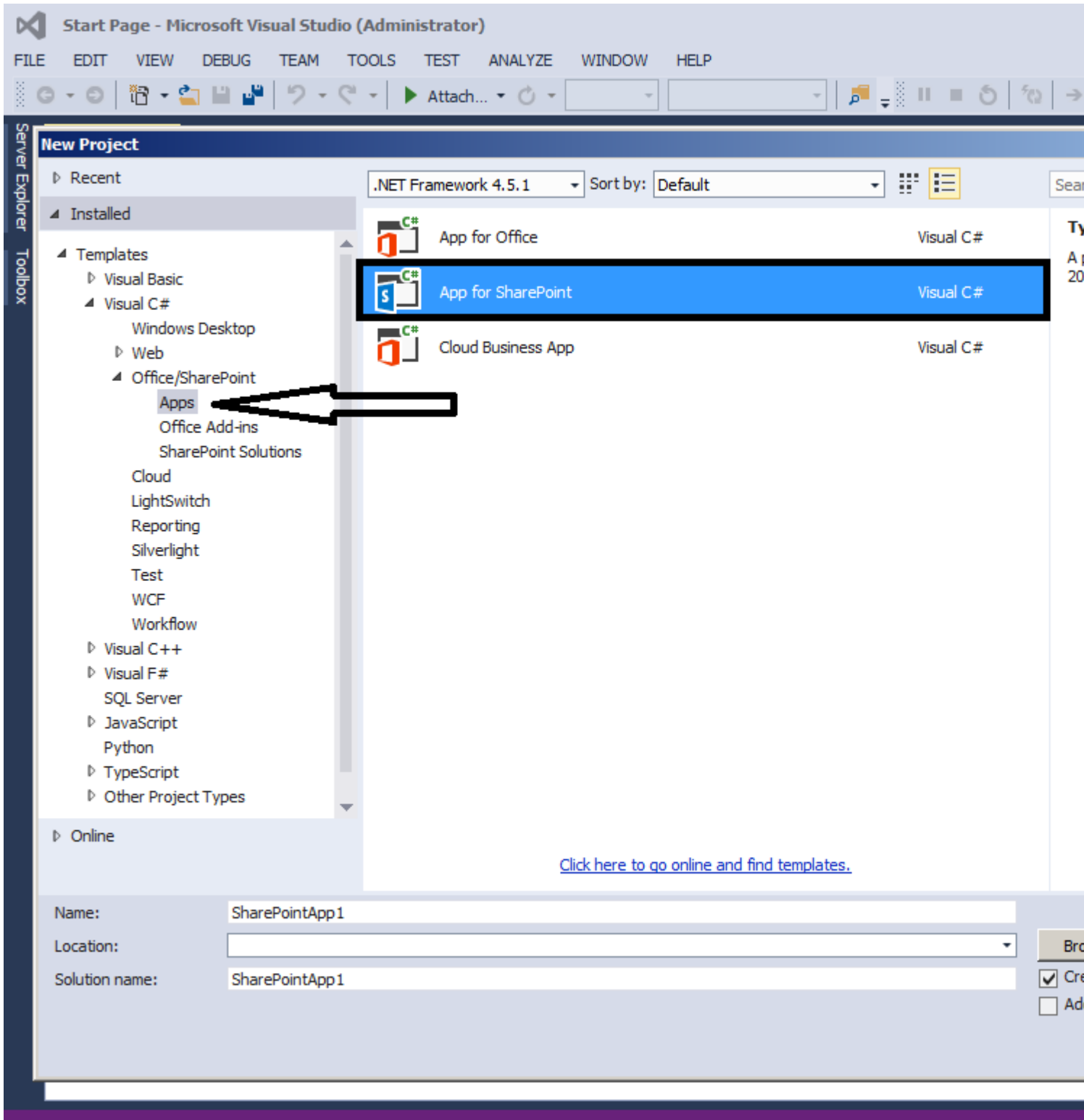
Um mit App Development zu beginnen, benötigen wir Visual Studio 2013 oder eine höhere Version. Laden Sie die neueste Community- oder Ausdrucksversion von hier>

<https://www.visualstudio.com/products/free-developer-offers-vs> herunter

Sobald es heruntergeladen und installiert wurde

Öffnen Sie und **klicken Sie auf Neues Projekt erstellen**

Erweitern Sie den Abschnitt Office / SharePoint. Sie sollten eine Option für App sehen (siehe unten).

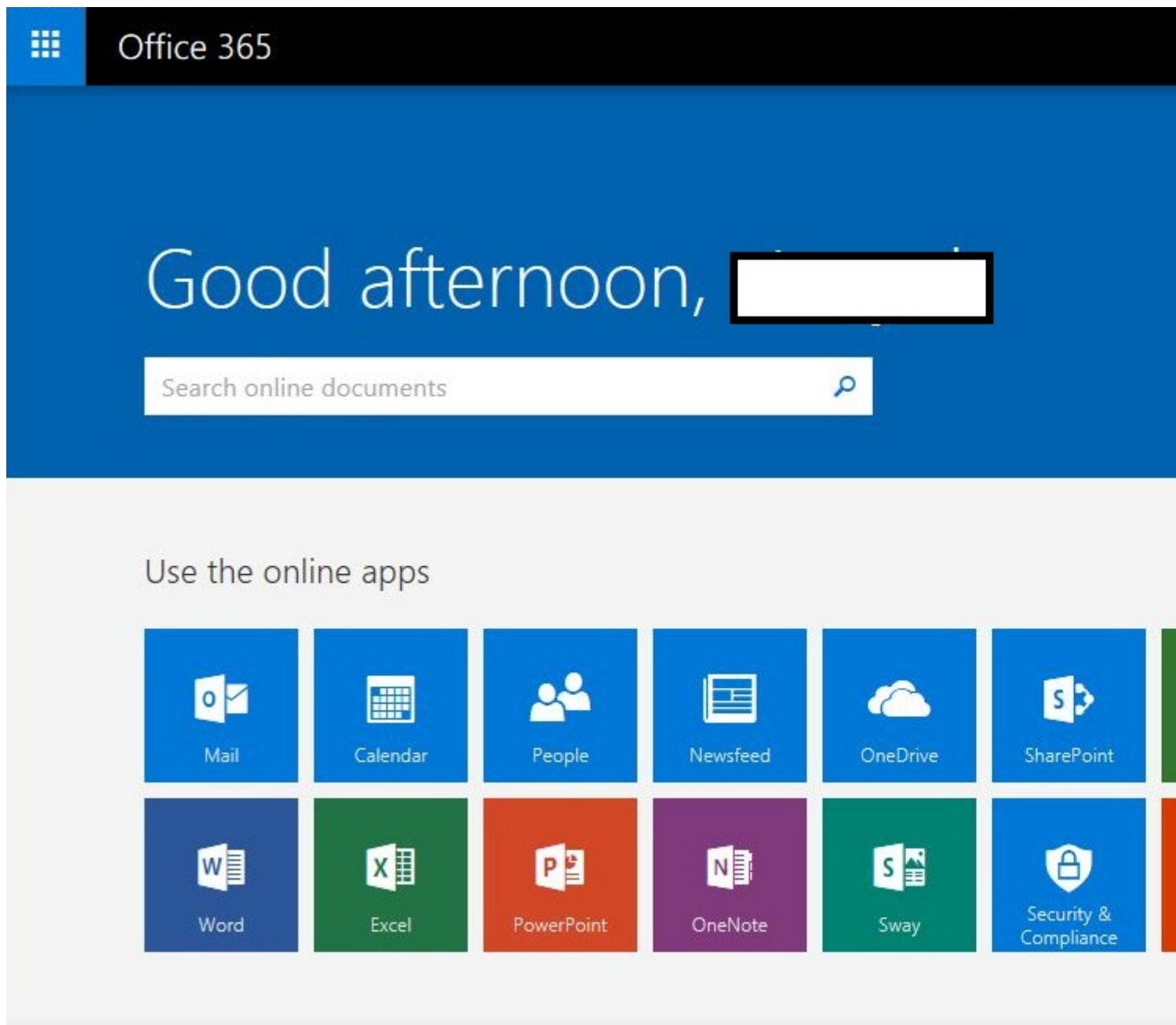


Wenn keine App-Option verfügbar ist Schließen Sie das VS und laden Sie die **Microsoft Office Developer Tools** <https://www.visualstudio.com/en-us/features/office-tools-vs.aspx> herunter

Vorbereitung für die Entwicklerseite

Sobald wir Visual Studio haben, benötigen wir eine Entwicklerseite, um Apps für SharePoint bereitzustellen. Am einfachsten erhalten Sie > Ein kostenloses Office 365-Entwicklerkonto für ein Jahr (<https://profile.microsoft.com/RegSysProfileCenter/wizardnp.aspx?wizid=14b845d0-938c-45af-b061-f798fbb4d170&lcid=1033>

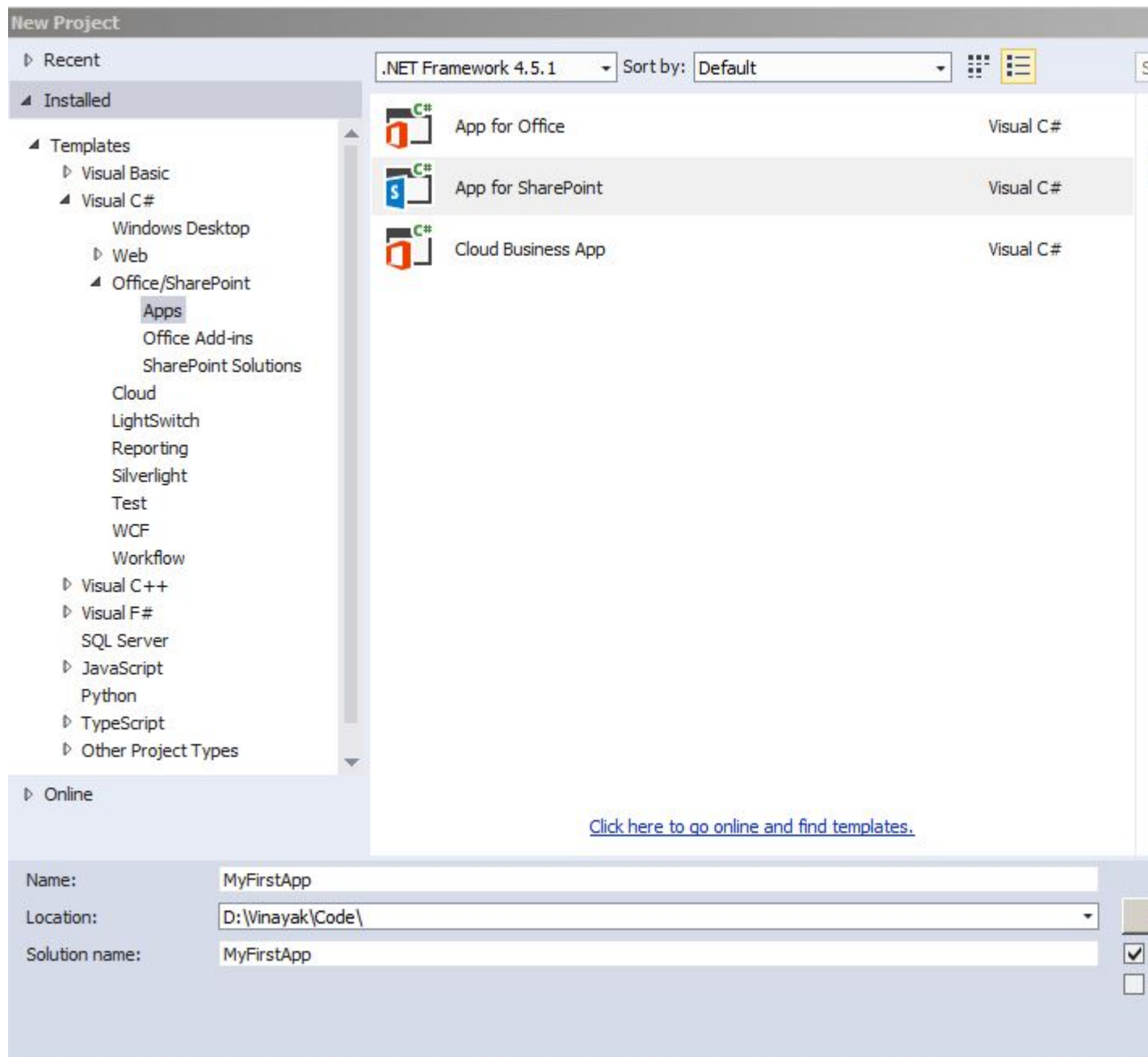
Sobald der Anmeldeprozess abgeschlossen ist, <https://www.office.com/> center URL für alle Ihre App



App in Visual Studio erstellen


Beginnen wir mit der Erstellung unserer ersten App

1. Visual Studio öffnen und> neues Projekt erstellen
2. Geben Sie Name und Ort ein



3. Geben Sie die URL der Entwicklersite ein, die Sie im vorherigen Schritt erstellt haben, und wählen Sie Provider-Hosted aus

New app for SharePoint ? ×

 Specify the app for SharePoint settings

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

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

How do you want to host your app for SharePoint?

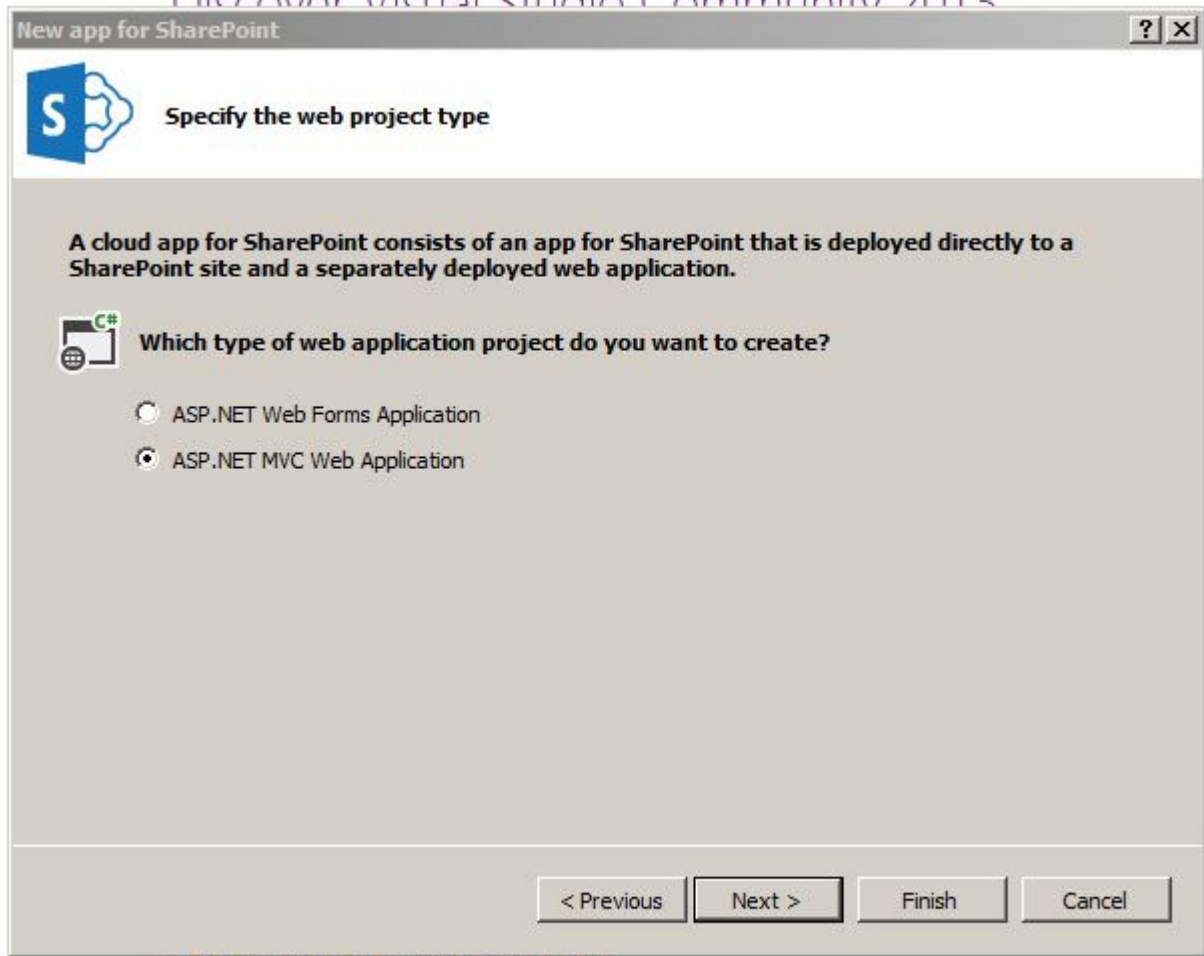
Provider-hosted
 SharePoint-hosted

[Learn more about this choice](#)

4. Es öffnet sich ein Popup mit dem Login

5. Als nächsten Schritt wird wie für den Anwendungstyp MVC oder Webform ausgewählt. Ich wähle hier MCV aus

3

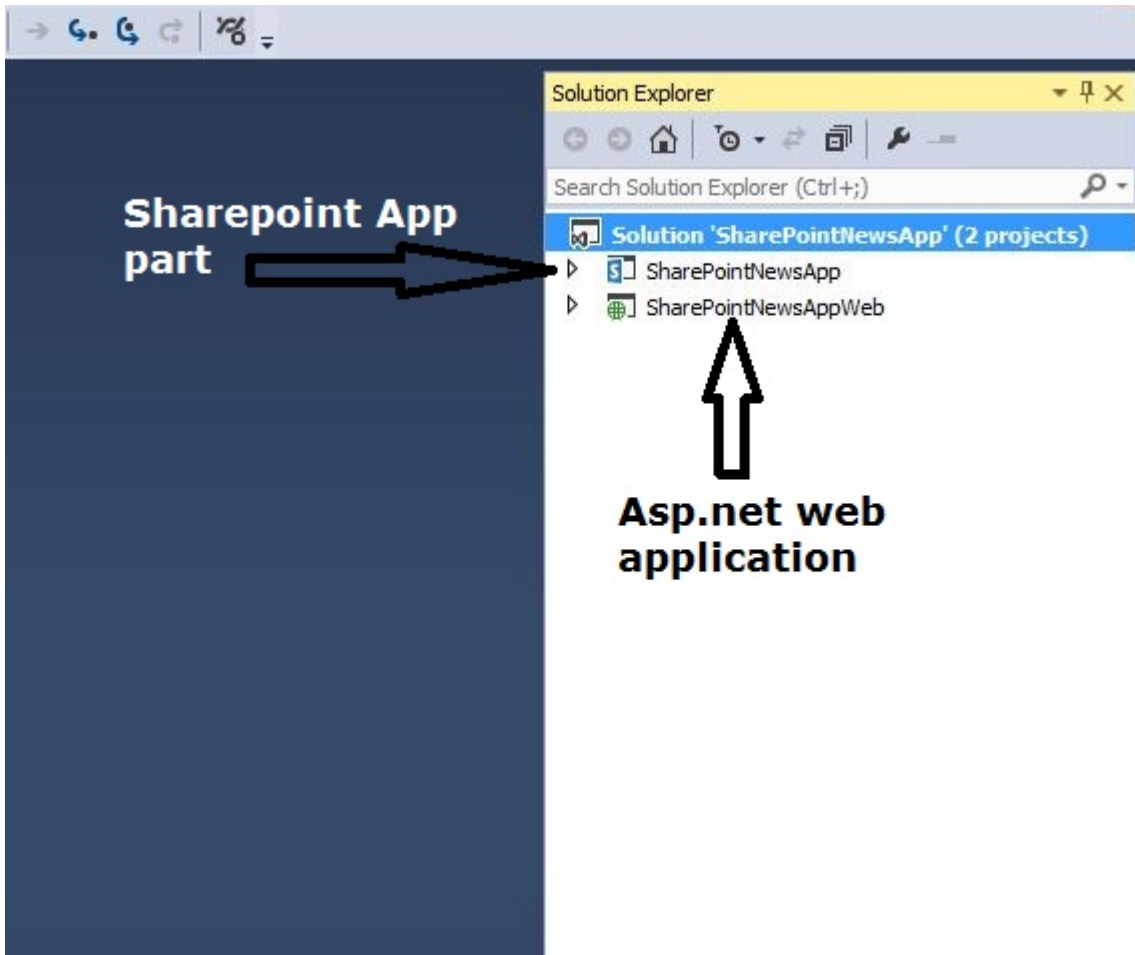


[Exploring dotnet new with .NET Core](#)

Wednesday, July 27, 2016

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

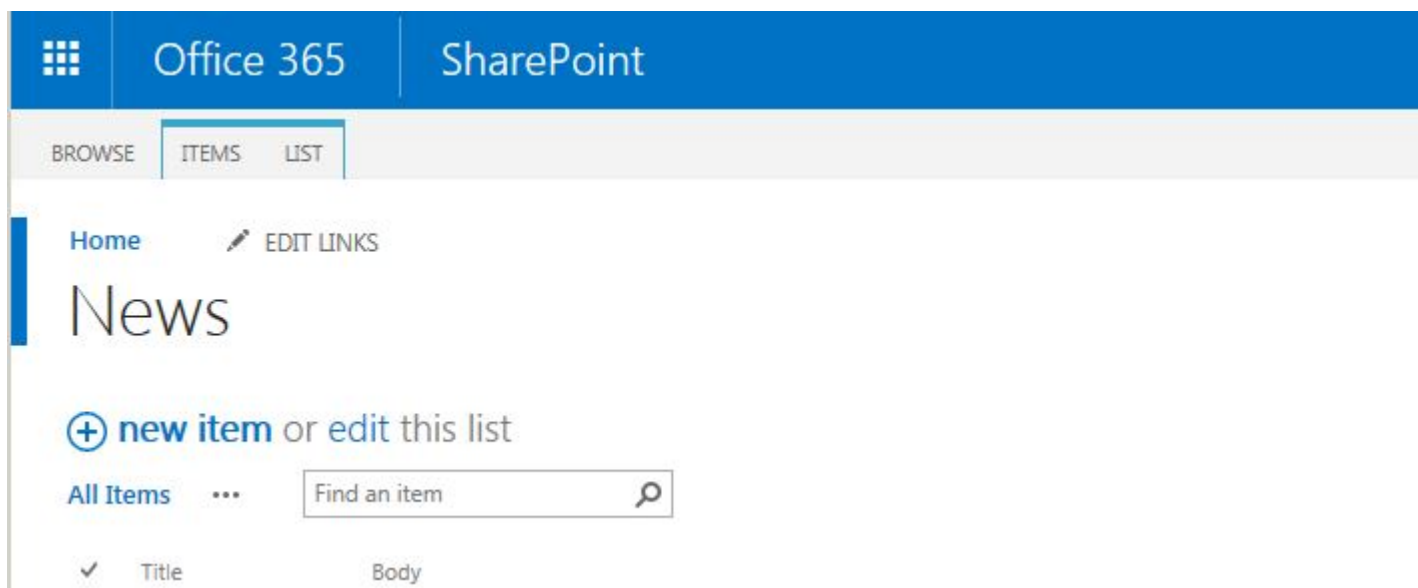
6. Wählen Sie unter Wie soll sich das Add-In authentifizieren? Wählen Sie Windows Azure Access Control-Dienst verwenden aus, und klicken Sie auf Fertig stellen
7. Im Solution Explorer können wir sehen, dass 2 Projekte erstellt wurden. Eines ist SharePoint App-Teil und ein anderes ist Asp.net Web App



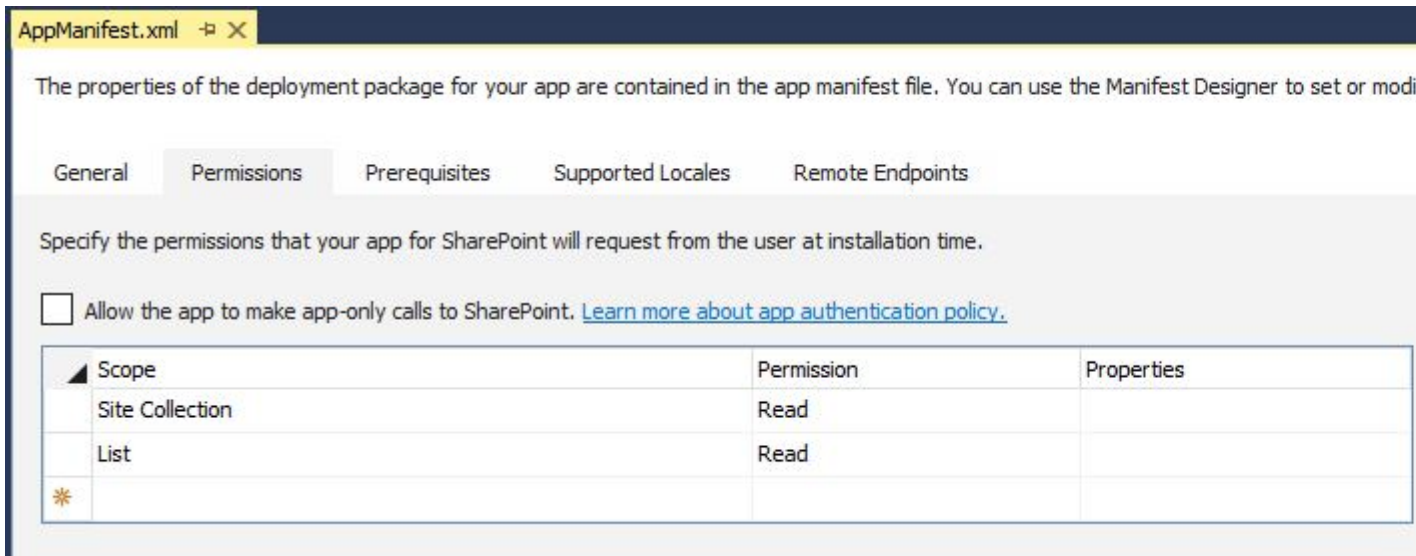
Beginnen wir mit der Codierung

Hier nehme ich das Beispiel einer einfachen Nachrichten-App

1. Öffnen Sie die SharePoint-Entwicklerseite, und erstellen Sie eine Liste zum Speichern unserer Nachrichtenartikel
2. Erstellen Sie eine benutzerdefinierte Liste und fügen Sie 3 weitere Spalten hinzu. Body, Summery, ThumbnaiImageUrl



3. Gehen Sie zurück zu unserer SharePoint-App, Öffnen Sie die AppManifest.xml-Datei, klicken Sie auf die Registerkarte Berechtigung, und geben Sie der Websitesammlung die Berechtigung Lesen, und speichern Sie sie.



4. Öffnen Sie HomeController über eine Webanwendung, in meinem Fall eine MVC-Anwendung. Wenn Sie eine Webform-App erstellen, sollte sich der Code auf der Seite default.aspx.cs befinden
5. Nachfolgend finden Sie den Code-Ausschnitt, um die neuesten Nachrichten aus der Liste zu erhalten. So sollte unsere Indexseite aussehen.

```
[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(1stProductItem.Id.ToString()),
                    Title = 1stProductItem.DisplayName.ToString(),
                    Summery = 1stProductItem["Summery"].ToString(),
                    Thumbnail = 1stProductItem["ThumbnailImageUrl"].ToString()
                });
        }
    }
}

return View(newsList);
}

```

6. Klicken **Sie** nun mit der rechten Maustaste auf **Index** und klicken **Sie** auf **Ansicht hinzufügen**. Klicken Sie dann auf Hinzufügen

7. Öffnen Sie nun die Datei **Index.cshtml** aus **Ansichten > Basisverzeichnis**

8. Nachstehend finden Sie den Code-Ausschnitt für die Datei **index.cshtml**

```

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

```

```

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

```

9. Klicken Sie in Ihrer Lösung mit der rechten Maustaste auf den Ordner Modell und fügen Sie eine CS-Klassendatei hinzu. Fügen Sie unten Modellklassen hinzu

```

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

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

    public int Id { get; set; }

    public string Title { get; set; }

    public string Summary { get; set; }

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

    public int Id { get; set; }

    public string Title { get; set; }

    public string Body { get; set; }

    }
}

```

10. Verwenden Sie die Taste F5, um das Add-In bereitzustellen und auszuführen. Wenn ein Sicherheitsalertfenster angezeigt wird, in dem Sie aufgefordert werden, dem selbstsignierten Localhost-Zertifikat zu vertrauen, wählen Sie Ja.

Und jetzt ist die erste App fertig

Vollständige Artikelseite erstellen

Wir haben bereits eine erste Seite erstellt, auf der alle Nachrichtenartikel angezeigt werden. Diese Seite zeigt den vollständigen Artikel.

1. Fügen Sie eine weitere Aktionsmethode zu HomeController hinzu

```
[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. Klicken Sie erneut mit der rechten Maustaste auf Aktion, und erstellen Sie eine Ansicht mit demselben Namen. Name der Aktionsmethode. In meinem Fall wird die Ansicht " **Absehen** " genannt

```
@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>
```

Dies ist der Code für die vollständige Artikelseite, die den Hauptteil des Nachrichtenartikels zeigt

Erstellen einer vom Provider gehosteten App online lesen:

<https://riptutorial.com/de/sharepoint/topic/6301/erstellen-einer-vom-provider-gehosteten-app>

Kapitel 7: Hauptversionen

Examples

SharePoint 2016

Build-Nummer	Beschreibung	Produkt
16.0.4366.1000	Kumulatives Update April 2016	SharePoint Server 2016
16.0.4336.1000	RTM	SharePoint Server 2016
16.0.4327.1000	Bewerber freigeben	SharePoint Server 2016
16.0.4266.1001	16.0.4306.1002 Beta 2	SharePoint Server 2016

SharePoint 2013

Build-Nummer	Beschreibung
15.0.4623.1001	Juni 2014
15.0.4631.1001	Juli 2014
15.0.4641.1001	August 2014
15.0.4649.1001	September 2014
15.0.4659.1001	Oktober 2014
15.0.4667.1000	November 2014
15.0.4675.1000	Dezember 2014
15.0.4693.1001	Februar 2015
15.0.4701.1001	März 2015
15.0.4711.1000	April 2015 (SP1-REQ)
15.0.4719.1002	Mai 2015
15.0.4727.1001	Juni 2015
15.0.4737.1000	Juli 2015
15.0.4745.1000	August 2015

Build-Nummer	Beschreibung
15.0.4753.1003	September 2015
15.0.4763.1002	Oktober 2015
15.0.4771.1000	November 2015
15.0.4779.1000	Dezember 2015
15.0.4787.1000	MS16-004
15.0.4787.1000	Januar 2016
15.0.4797.1001	Februar 2016
15.0.4805.1000	März 2016
15.0.4815.1000	April 2016
15.0.4823.1003	Mai 2016
15.0.4833.1000	Juni 2016

Quelle: [SharePoint 2013-Build-Nummern und -CUs](#)

Hauptversionen online lesen: <https://riptutorial.com/de/sharepoint/topic/2737/hauptversionen>

Kapitel 8: Mit dem verwalteten Serverseitenobjektmodell arbeiten (voll vertrauenswürdig)

Bemerkungen

Konzeptionelle Hierarchie

In der Sharepoint - konzeptuellen Hierarchie enthalten **Websitesammlungen Websites**, die wiederum **Listen** enthalten. Eine `SPSite` (`SPSite`) hat keine explizite Benutzeroberfläche, enthält jedoch immer eine Site auf Stammebene (auf die über die Eigenschaft `RootWeb`) und möglicherweise zusätzliche Unterwebsites unter dieser Root-Site. Eine Site oder Web (`SPWeb`) hat eine Benutzeroberfläche und enthält Listen / Dokumentbibliotheken (`SPList`), Seiten mit Webparts und Elemente / Dokumente (`SPListItem`).

Vorsichtsmaßnahmen auf dem Server

- Um eine Anwendung zu erstellen, die das serverseitige SharePoint-Objektmodell verwendet, müssen Sie in Ihrem Visual Studio-Projekt einen Verweis auf die Microsoft.SharePoint-Assembly hinzufügen, die unter Framework Assemblies aufgeführt ist.
- Anwendungen, die das serverseitige Objektmodell (vollständig vertrauenswürdig) verwenden, können nur auf einem Windows Server ausgeführt werden, der SharePoint hostet.
- Sie können keine Verbindung zu einem anderen SharePoint-Server als dem herstellen, auf dem die Anwendung ausgeführt wird.

Examples

Hallo Welt (Website-Titel erhalten)

2013

SharePoint 2013 und neuere Versionen sind nur 64-Bit, daher muss die Assembly / das Programm auch für einen 64-Bit-Prozessor erstellt werden.

Direkt nach der Erstellung Ihres Projekts müssen Sie das **Plattformziel** von **Any CPU** auf **x64** umstellen, andernfalls tritt ein Fehler auf.

```
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);
            }
        }
    }
}

```

Durchlaufen der gesamten SharePoint-Farm

Verwenden von PowerShell, das von einem SharePoint-Webserver ausgeführt wird:

```

$wacoll = get-spwebapplication
foreach($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()
        }
    }
}

```

Listenelemente abrufen

```

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
    }
}

```

Rufen Sie Elemente mit Paging ab

```

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);
}

```

Liste per URL abrufen

```

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);
}

```

Listenelement erstellen

Beim Erstellen eines neuen Listenelements können die Felder mit einer Syntax ähnlich der String-Arrays festgelegt werden. Beachten Sie, dass diese Felder nicht direkt erstellt werden und vom Schema der Liste definiert werden. Diese Felder (oder Spalten) müssen auf dem Server vorhanden sein, andernfalls schlägt die Erstellung fehl. Alle Listenelemente haben das Titelfeld. Einige Listen enthalten möglicherweise erforderliche Felder, die ausgefüllt werden müssen, bevor das Element in der Liste veröffentlicht wird.

In diesem Beispiel verwendet die Liste die Ankündigungsvorlage. Neben dem Titelfeld enthält die Liste das Textfeld, in dem der Inhalt der Ankündigung in der Liste angezeigt wird.

```

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();
}

```

Mit dem verwalteten Serverseitenobjektmodell arbeiten (voll vertrauenswürdig) online lesen: <https://riptutorial.com/de/sharepoint/topic/7543/mit-dem-verwalteten-serverseitenobjektmodell-arbeiten--voll-vertrauenswurdig->

Kapitel 9: REST-Dienste

Bemerkungen

URLs für den REST-Service-Endpunkt

Die REST-Clientzugriffs-API wurde erstmals in SharePoint 2010 eingeführt, wurde jedoch in SharePoint 2013 erheblich erweitert. Auf die REST-API in [SharePoint 2010](#) kann über den ListData-Webdienst unter der URL `/_vti_bin/ListData.svc` . In [SharePoint 2013](#) wurden die URLs für die Endpunkte `/_api/lists/` und `/_api/web` , die sich etwas anders verhalten.

Den obigen Endpunkt-URLs sollte `http://server/site` vorangestellt werden, wobei `server` den Namen des Servers und `site` den Namen oder den Pfad zu der bestimmten Site darstellt.

Beispiel-URL für ...	SharePoint 2010	SharePoint 2013
Eine Liste abrufen:	<code>/_vti_bin/ListData.svc/ListenName</code>	<code>/_api/lists('ListGuid')</code>
Artikel abrufen:	<code>/_vti_bin/ListData.svc/ListenName(1)</code>	<code>/_api/lists('ListGuid')/items(1)</code>
Ein Web abrufen:	(kein Äquivalent)	<code>/_api/web</code>

Trotz der Unterschiede beim Zugriff auf Listen und Listenelemente ist das Arbeiten mit diesen Ergebnissen in beiden Versionen sehr ähnlich.

Beachten Sie, dass der Dienst `ListData.svc` aus `ListData.svc` der `ListData.svc` in SharePoint 2013 noch verfügbar ist.

Senden von REST-Anforderungen

Eine REST-Anforderung kann über ein natives JavaScript XMLHttpRequest oder über das jQuery AJAX-Wrapper-Konstrukt übermittelt werden.

XMLHttpRequest-Syntax

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

jQuery AJAX-Syntax

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

Weitere Informationen zum Senden von Anforderungen über AJAX finden Sie in [der JavaScript-Dokumentation zu AJAX](#) .

Examples

Mit Listen arbeiten

Listenelemente abrufen

Dieses Beispiel zeigt, wie alle Listenelemente abgerufen und durchlaufen werden. Mit dem Parameter `top` können Sie eine bestimmte Anzahl von Ergebnissen abfragen. Sie können den `select` Parameter auch verwenden, um bestimmte Felder `$select=id, Title, uri ($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);
    }
  });
}
```

Einzelnen Listeneintrag abrufen

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);
    }
});
}

```

Listenelemente mit Suchspalten abrufen

Manchmal haben Sie möglicherweise eine Listenstruktur, die wie folgt aussieht:

Tierauflistungstabelle

Name	Art	Beschreibung
Titel	Zeichenfolge (Text)	Name des Tieres
Alter	Nummer	Wie alt ist das Tier?
Wert	Währung	Wert des Tieres
Art	<i>Nachschlagen</i> (<i>Tierartentabelle</i>)	Nachschlagefeld (Dropdown-Auswahl aus der Animal Types Table)

Tabelle der Tierarten

Name	Art	Beschreibung
Titel	Zeichenfolge (Text)	Name der Art / Tierart (zB Schwein)
NumLegs	Nummer	Anzahl der Beine am Tier

Wahrscheinlich nicht das ernsteste Beispiel, aber das Problem ist hier immer noch gültig. Wenn Sie die übliche Anforderung zum Abrufen von Werten aus der SharePoint-Liste für den `Type` des Tieres verwenden, erhalten Sie nur ein Feld mit dem Namen `TypeId` in der JSON-Antwort. Um diese Elemente in nur einem einzigen AJAX-Aufruf zu erweitern, sind einige zusätzliche Markierungen in den URL-Parametern erforderlich.

Dieses Beispiel gilt auch für mehr als nur Suchspalten. Wenn Sie `People/Groups`, handelt es sich im Wesentlichen nur um Suchvorgänge, sodass Sie Elemente wie `Title`, `Email` problemlos `Email` können.

Beispielcode

Wichtiger Hinweis : Wenn Sie die Felder definieren, die Sie aus den Suchspalten abrufen

möchten, müssen Sie dem Namen des Felds den Namen des Suchfelds in der ursprünglichen Tabelle voranstellen. Wenn Sie beispielsweise das `NumLegs` Attribut aus der `NumLegs` möchten, müssen Sie `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 getItem(webUrl, listTitle, selectFields, expandFields, callback) {
    var endpointUrl = webUrl + "/_api/web/lists/getbytitle('" + listTitle + "')/items";
    endpointUrl += '?$select=' + selectFields.join(",");
    endpointUrl += '&$expand=' + expandFields.join(",");
    return executeRequest(endpointUrl, 'GET', callback);
}

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

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

    return $.ajax(ajaxOptions);
}

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

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

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

```

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

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

// Start the entire process
getAnimals();

```

Hinzufügen von Auswahlmöglichkeiten zu einem mehrwertigen Suchfeld

In diesem Beispiel wird davon `MultiLookupColumnName`, dass Ihre `MultiLookupColumnName` Namen `MultiLookupColumnName` und dass Sie Ihr Multi-Lookup-Feld so einrichten möchten, dass nach den Elementen mit den IDs 1 und 2 `MultiLookupColumnName`.

JQuery AJAX verwenden

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
});
```

Verwendung von 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);
```

Auslagerungselemente, die von einer Abfrage zurückgegeben wurden

Um das Paging mit REST zu simulieren, haben Sie folgende Möglichkeiten:

1. Verwenden Sie den Parameter `$orderby $skip=n`, um die ersten `n` Einträge gemäß dem Parameter `$orderby` zu überspringen

2. Verwenden Sie den Parameter `$top=n` , um die ersten `n` Einträge gemäß den Parametern `$orderby` und `$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();
            }
        }
    );
}
```

Rufen Sie eine ID des neu erstellten Elements in der SharePoint-Liste ab

In diesem Beispiel wird gezeigt, wie eine ID eines neu erstellten Elements mithilfe der SharePoint-REST-API abgerufen wird.

Hinweis :

listName - Diese Variable enthält den Namen Ihrer Liste.

newItemBody - Dies ist Ihre Anforderung, um ein neues Element in der Liste hinzuzufügen.

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

```

So führen Sie CRUD-Vorgänge mithilfe der SharePoint 2010-REST-Schnittstelle durch

Erstellen

Um einen Create-Vorgang über REST durchzuführen, müssen Sie die folgenden Aktionen ausführen:

Erstellen Sie eine HTTP-Anforderung mit dem `POST` Verb. Verwenden Sie die Service-URL der Liste, zu der Sie eine Entität hinzufügen möchten, als Ziel für den POST. Setzen Sie den Inhaltstyp auf `application/json`. Serialisieren Sie die JSON-Objekte, die Ihre neuen Listenelemente als Zeichenfolge darstellen, und fügen Sie diesen Wert zum JavaScript-Beispiel des Anforderungskörpers hinzu:

```

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);
        }
    });
}

```

Verwendungszweck

```

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));
    }
);

```

Lesen

Um einen Lesevorgang über REST auszuführen, müssen Sie die folgenden Aktionen ausführen:

Erstellen Sie eine HTTP-Anforderung mit dem Verb `GET`. Verwenden Sie die Service-URL des Listenelements, zu dem Sie eine Entität hinzufügen möchten, als Ziel für den GET. Setzen Sie den Inhaltstyp auf `application/json`. JavaScript-Beispiel:

```

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);
        }
    });
}

```

Verwendungszweck

```

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

```

Aktualisieren

Um eine vorhandene Entität zu aktualisieren, müssen Sie die folgenden Aktionen ausführen:

Erstellen Sie eine HTTP-Anforderung mit dem `POST` Verb. Fügen Sie einen `X-HTTP-Method` Header mit dem Wert `MERGE`. Verwenden Sie die Service-URL des Listenelements, das Sie aktualisieren möchten, als Ziel für den POST. Fügen Sie einen `If-Match` Header mit einem Wert des ursprünglichen ETag der Entität (*) hinzu. JavaScript-Beispiel:

```

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);
});
}

```

Verwendungszweck

```

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));
}
);

```

Löschen

Um eine Entität zu löschen, müssen Sie die folgenden Aktionen ausführen:

Erstellen Sie eine HTTP-Anforderung mit dem `POST` Verb. Fügen Sie einen `X-HTTP-Method` Header mit dem Wert `DELETE` . Verwenden Sie die Service-URL des Listenelements, das Sie aktualisieren möchten, als Ziel für den `POST`. Fügen Sie einen `If-Match` Header mit einem Wert des ursprünglichen ETag der Entität hinzu. JavaScript-Beispiel:

```

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);
});
}
```

Verwendungszweck

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

REST-Dienste online lesen: <https://riptutorial.com/de/sharepoint/topic/3045/rest-dienste>

Kapitel 10: SharePoint App

Einführung

Gehostete SharePoint-App

Bemerkungen

Erforderliche Referenz von Website: <http://www.letsharepoint.com/what-is-user-information-list-in-sharepoint-2013/>

Examples

SharePoint 2013: Zugriff auf Benutzerprofildienstdaten mithilfe von JSOM in SharePoint 2013

SharePoint 2013: Zugriff auf Benutzerprofildienstdaten mithilfe von JSOM in SharePoint 2013

In diesem Artikel erfahren Sie, wie Sie mithilfe von JSOM (Javascript Object Model) die Anwendung "User Profile Service (UPS)" verwalten oder darauf zugreifen und eine grundlegende App erstellen. Bevor wir beginnen, lassen Sie uns zunächst die grundlegende USV-Terminologie durchgehen.

Benutzerprofil - Es enthält alle Informationen von Personen in einer Organisation auf organisierte Weise. Es zeigt alle Eigenschaften wie AccountName, Vorname, Nachname, WorkEmail usw. an, die sich auf einen Benutzer beziehen.

Benutzerprofildienst-Anwendung - Es wird als zentraler Speicherort für das Speichern aller Benutzerprofile betrachtet. Administratoren können außerdem Profile, Profilsynchronisierung, Meine Website, soziale Tags usw. konfigurieren oder verwalten. Außerdem können Informationen aus Verzeichnisdiensten wie Active Directory abgerufen werden.

Meine Website - Eine personalisierte Website für individuelle Benutzer zum Verwalten ihrer Informationen und zum Speichern von Dokumenten, Links usw. Sie bietet umfassende Netzwerk- und soziale Funktionen, indem Benutzer Informationen über sich selbst oder ihre Aktivitäten austauschen können. Auf Meine Website können Sie durch Klicken auf Benutzername in der rechten oberen Ecke der SharePoint-Seite zugreifen.

Verwalten und Zugriff auf Benutzerprofildaten

Da wir mit JSOM arbeiten werden, können wir nur Leseoperationen ausführen, mit der Ausnahme, dass das Profilbild mit JSOM (oder CSOM oder REST) geändert werden kann.

* Server Side Code ermöglicht das Lesen / Schreiben beider Vorgänge.

Rufen Sie die Benutzerprofileigenschaften mit JSOM ab

Ermöglicht das Erstellen einer gehosteten SharePoint-App und das Abrufen von Benutzerinformationen in dieser App.

Starten Sie Visual Studio 2013 und wählen Sie "App für SharePoint 2013" unter Neues Projekt. Nachdem Sie den oben genannten Projekttyp ausgewählt haben, wird ein Fenster angezeigt, in dem Sie eine Verbindung zur SharePoint-Website herstellen können. Wählen Sie den App-Typ aus, der bereitgestellt werden soll (siehe Abbildung unten). Hier habe ich die URL der Entwickler-Website von Microsoft SharePoint Online angegeben und die SharePoint Hosted App ausgewählt. Klicken Sie auf Fertig stellen.

3.) Nachdem das Projekt erstellt wurde, werden im Projektmappen-Explorer standardmäßig mehrere Ordner / Ordner hinzugefügt.

4.) Wenn Sie die Seite "Default.aspx" öffnen, finden Sie bereits einige JavaScript-Bibliotheken, die der Seite hinzugefügt wurden.

Hier müssen wir eine weitere Bibliothek hinzufügen, um mit den Benutzerprofilen arbeiten zu können

SharePoint App online lesen: <https://riptutorial.com/de/sharepoint/topic/9876/sharepoint-app>

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit Sharepoint	Community , Marco , Ryan Gregg , Thriggle , Tom Resing , Zach Koehne
2	Arbeiten mit JavaScript Client Object Model (JSOM)	Thriggle , yngrdyn
3	Arbeiten mit Managed Client Side Object Model (CSOM)	InvoiceGuy , Lukáš Nešpor , MikhailSP , RamenChef , Thriggle , Zach Koehne
4	Arbeiten mit modalen Dialogfeldern mit JavaScript	Thriggle
5	Clientseitige Wiedergabe von SharePoint 2013	Rohit Waghela , Yayati
6	Erstellen einer vom Provider gehosteten App	vinayak hegde
7	Hauptversionen	jlr2527 , MikhailSP
8	Mit dem verwalteten Serverseitenobjektmodell arbeiten (voll vertrauenswürdig)	Lukáš Nešpor , Thriggle
9	REST-Dienste	Aaron , Brock Davis , ocelotsloth , R4mbi , Rohit Waghela , Thriggle
10	SharePoint App	Sunil sahu