Бесплатная электронная книга

учусь sharepoint

Free unaffiliated eBook created from
Stack Overflow contributors.

#sharepoint

Около

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.

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

глава 1: Начало работы с sharepoint

замечания

SharePoint может ссылаться на один или несколько продуктов из семейства Microsoft SharePoint.

- **SharePoint Foundation** . Это была основная технология для всех сайтов SharePoint и больше не доступна для SharePoint 2016
- **SharePoint Server** : это локальная версия SharePoint. Вы можете развернуть один или несколько серверов SharePoint. Он предлагает дополнительные функции в SharePoint Foundation, такие как возможности BI, управление корпоративным контентом и многое другое.
- **SharePoint Online** : облачная версия SharePoint. Клиенту не нужно заботиться о инфраструктуре сервера или масштабируемости.

Office 365 - это отдельное предложение Microsoft, которое включает службу SharePoint Online, хотя не все планы поддерживают все функции SharePoint.

Следующие ссылки предоставляют обширные сопоставления функций между доступными версиями SharePoint:

- SharePoint 2013 в режиме реального времени и SharePoint 2016: доступность объектов по локальным планам SharePoint
- Возможности SharePoint в Office 365: доступность функций в планах SharePoint
- Возможности SharePoint в SharePoint Online (без Office 365): доступность функций в автономных планах SharePoint
- Объединенное сравнение функций между SharePoint 2013 и SharePoint Online: http://www.buckleyplanet.com/2014/06/sharepoint-online-vs-onprem-feature-comparison.html

Версии

Версия	Официальное название	Дата выхода
Pre-2003	Сервер 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

Версия	Официальное название	Дата выхода
2016	Microsoft SharePoint Server 2016	2016-05-01

Examples

Установка SharePoint 2016 для односерверной фермы

Вступление

SharePoint 2016 - это версия семейства продуктов SharePoint версии 16. Он был выпущен 4 мая 2016 года. В этом примере рассматривается установка SharePoint 2016 с использованием конфигурации Single Server Farm. Эта конфигурация охватывает основы настройки фермы SharePoint без необходимости иметь несколько серверов. Обратите внимание, что описанные сценарии односерверной фермы обычно ограничиваются разработками и очень небольшими производственными сценариями.

Требования

Перед установкой SharePoint необходимо настроить базовую среду. SharePoint хранит документы, а также метаданные, журналы, пользовательские приложения, настройки и многое другое. Убедитесь, что у вас есть достаточное дисковое пространство и оперативная память, доступная выше требований базовой линии.

- 4 ядра на 64-разрядных совместимых процессорах
- 12 - 24 ГБ ОЗУ (в зависимости от развертывания теста или prod)
- 80 ГБ жесткий диск для системы
- 100 ГБ жесткий диск в качестве второго диска
- Сервер с 64-битным Windows Server 2012 R2 или Технический просмотр «Порог»
- SQL Server 2014 или SQL Server 2016
- .NET Framework 4.5.2 или .NET Framework 4.6
- Домен подключился к компьютеру и делегировал учетные записи фермы

Все другие предварительные условия могут быть установлены вручную или выполнены с помощью установщика SharePoint Preprise, входящего в состав установки SharePoint.

Монтаж

- Запустите программу установки предварительных условий; он может запросить перезагрузку сервера, прежде чем продолжить

- Запустите Setup.exe из установки SharePoint
- Введите лицензионный ключ
- Принять лицензионное соглашение
- Выберите «Завершить» на вкладке «Тип сервера»
- Настройка должна завершиться успешно
- На полной странице оставьте флажок рядом с Мастере настройки продукта и нажмите Закрыть

конфигурация

Если вы продолжаете с предыдущего шага, мастер настройки продуктов SharePoint 2016 должен открываться автоматически. если окно не отображается или вы используете конфигурацию позже, откройте мастер настройки, перейдя в меню Пуск -> SharePoint 2016 Products -> Мастер настройки продукта SharePoint 2016.

- Нажмите «Далее» на странице приветствия
- Появится модальное диалоговое окно с сообщением некоторых служб, которые я должен перезапустить во время конфигурации; ничего еще не установлено, поэтому нажмите «Да».
- Добавить сервер базы данных для фермы
    - Введите имя машины, на которой запущен SQL Server; в этом случае это локальная машина
    - Введите имя базы данных конфигурации или сохраните имя по умолчанию SharePoint_Config
    - Введите имя пользователя пользователя службы домена, который будет обращаться к базе данных (в форме DOMAIN \ user). * Введите пароль для пользователя домена.
    - Нажмите дальше, когда закончите.
- Введите пароль фермы; это будет использоваться при подключении дополнительных серверов к новой ферме
- Выберите роль Single Server Farm
- Настройте Центральное администраторское веб-приложение (где SharePoint будет управляться администраторами фермы) выберите номер порта и выберите тип федерации аутентификации (NTLM или Negotate (Kerberose)).
- Просмотрите настройки на последних страницах и внесите необходимые изменения
- Когда будете готовы, запустите конфигурацию, которая может занять несколько минут
- По завершении работы вы откроете мастер, который позволит вам открыть сайт Центра администрирования
- При сбое вы можете исследовать журналы в папке% COMMONPROGRAMFILES% \ Microsoft Shared \ Web Server Extensions \ 16 \ LOG

Настройка фермы

После того, как настроены центральное веб-приложение, база данных конфигурации и центральный администратор, вы будете готовы настроить ферму для использования для пользователей или разработки. Вы можете пометить местоположение центра администрирования или получить доступ к нему с помощью ярлыка в том же месте, что и мастер настройки продукта.

- Если позднее вы запустите конфигурацию, нажмите «Быстрый запуск» -> «Мастера настройки» -> «Мастер настройки фермы»
- Если вы запустите мастер с этапа установки, нажмите «Запустить мастер»
- Выберите, хотите ли вы быть частью программы улучшения клиентов, нажав «Да» или «Нет».
- На странице конфигурации фермы выберите учетную запись домена, на которой будут выполняться фоновые службы на ферме
    - Хотя эта учетная запись может совпадать с учетной записью базы данных, они могут также отличаться для разделения ролей и привилегий
    - Введите учетную запись как DOMAIN \ user
- Подтвердите, какие службы вы хотите получить в ферме на странице «Услуги»
- Создайте первый семейство сайтов в ферме (этот шаг можно пропустить и позже)
    - Введите заголовок, описание, веб-адрес семейства сайтов (обычно первый сайт находится в корне сервера), а шаблон
    - Большинство вещей можно изменить (название, описание) можно легко изменить, но другим, таким как веб-URL, может потребоваться гораздо больше работы для изменения; шаблон также не может быть легко откат, но SharePoint позволяет большое количество настроек, которые позволяют вам использовать любой базовый шаблон и преобразовать стиль и макет сайта
- Когда вы закончите настройку, нажмите «Готово»

Ферма и первый семейство сайтов теперь настроены для использования.

Создайте веб-часть с помощью SharePoint Framework

dev.office.com/sharepoint - отличное место для работы с SharePoint Framework.

SharePoint Framework - это современный подход на стороне клиента к SharePoint Development, первоначально ориентированный на SharePoint Online в Office 365. Веб-части, созданные с помощью SharePoint Framework, представляют собой новый тип веб-части, и они могут быть доступны для добавления на обеих существующих страницах SharePoint и новые страницы SharePoint.

Для этого процесса есть большой привет приветственный пример. Создайте свою первую веб-часть на стороне клиента SharePoint (Hello World, часть 1) . Все примеры на

dev.office.com доступны для вкладов сообщества через github.

Основные шаги Hello World в SharePoint Framework:

1. Создайте скелет проекта с помощью Yeoman SharePoint Generator .

   yo @ microsoft / SharePoint

2. Измените сгенерированный код в редакторе по вашему выбору. Поддержка Visual Studio Code сильна на разных платформах.

3. Предварительный просмотр веб-части с помощью gulp и локальной SharePoint Workbench

   подача глотки

4. Предварительный просмотр в среде SharePoint Online

   Перейдите по следующему URL-адресу: « https: //your-sharepoint-site/_layouts/workbench.aspx »

Журналы и журналы SharePoint ULS

Служба унифицированных журналов SharePoint (ULS) предоставляет возможности поддержки и отладки как для операционных систем, так и для разработчиков. Понимание того, как читать журналы, является важным первым шагом к решению проблем.

механическая обработка

Microsoft предоставляет средство просмотра ULS для чтения старых журналов и журналов, которые в настоящее время записываются при запуске фермы. Он также может фильтровать и применять форматирование в журналах, чтобы уменьшить проблему.

Идентификатор корреляции

Чтобы изолировать проблему, полезно только посмотреть на конкретный идентификатор корреляции. Каждый идентификатор корреляции связан с запросом или от конца до конца действия системы (например, время работы). Если возникает проблема с визуализируемой веб-страницей, поиск запроса в журналах ULS и выделение его на определенный идентификатор корреляции устраняет все шумы из других журналов, помогая выявить проблему.

Добавление SPMonitoredScope в мой код

Один из способов увеличить регистрацию и некоторый мониторинг производительности - добавить SPMonitoredScope к вашему коду.

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

Этот код будет записывать начало и конец ваших запросов, а также некоторые данные о производительности. Создавая собственный пользовательский монитор, который реализует ISPScopedPerformanceMonitor, вы можете установить уровень трассировки или максимальное время выполнения для набора кода.

Прочитайте Начало работы с sharepoint онлайн: https://riptutorial.com/ru/sharepoint/topic/950/начало-работы-с-sharepoint

глава 2: Основные выпуски

Examples

SharePoint 2016

Номер сборки	Описание	Товар
16.0.4366.1000	Накопительное обновление Апрель 2016 года	SharePoint Server 2016
16.0.4336.1000	RTM	SharePoint Server 2016
16.0.4327.1000	Релиз-кандидат	SharePoint Server 2016
16.0.4266.1001	16.0.4306.1002 Beta 2	SharePoint Server 2016

SharePoint 2013

Номер сборки	Описание
15.0.4623.1001	Июнь 2014 г.
15.0.4631.1001	Июль 2014 г.
15.0.4641.1001	Август 2014 года
15.0.4649.1001	Сентябрь 2014 года
15.0.4659.1001	Октябрь 2014 г.
15.0.4667.1000	Ноябрь 2014 г.
15.0.4675.1000	Декабрь 2014 года
15.0.4693.1001	Февраль 2015 г.
15.0.4701.1001	Март 2015 г.
15.0.4711.1000	Апрель 2015 г. (SP1 REQ)
15.0.4719.1002	Май 2015 г.

Номер сборки	Описание
15.0.4727.1001	Июнь 2015 г.
15.0.4737.1000	Июль 2015 г.
15.0.4745.1000	Август 2015 г.
15.0.4753.1003	Сентябрь 2015 г.
15.0.4763.1002	Октябрь 2015 г.
15.0.4771.1000	Ноябрь 2015 г.
15.0.4779.1000	Декабрь 2015 г.
15.0.4787.1000	MS16-004
15.0.4787.1000	Januar 2016
15.0.4797.1001	Февраль 2016 года
15.0.4805.1000	Март 2016 г.
15.0.4815.1000	Апрель 2016 года
15.0.4823.1003	Май 2016 г.
15.0.4833.1000	Июнь 2016 г.

Источник: сборщики данных SharePoint 2013 и CU

Прочитайте Основные выпуски онлайн: https://riptutorial.com/ru/sharepoint/topic/2737/основные-выпуски

глава 3: Приложение SharePoint

Вступление

Приложение для хостинга SharePoint

замечания

Ссылка, требуемая с сайта: http://www.letsharepoint.com/what-is-user-information-list-in-sharepoint-2013/

Examples

SharePoint 2013: доступ к данным службы профилей пользователей с помощью JSOM в SharePoint 2013

SharePoint 2013: доступ к данным службы профилей пользователей с помощью JSOM в SharePoint 2013

В этой статье мы научимся управлять или использовать приложение службы профилей пользователей (ИБП) с использованием JSOM (объектная модель Javascript) и создавать базовое приложение. Прежде чем мы начнем, сначала переходим к базовой терминологии ИБП.

Профиль пользователя - он имеет всю информацию людей в организации организованным образом. Он отображает все свойства, такие как AccountName, FirstName, LastName, WorkEmail и т. Д., Связанные с пользователем.

Приложение службы профилей пользователей - считается централизованным местом хранения всех пользовательских профилей, а также позволяет администраторам настраивать профили, синхронизацию профилей, мой сайт, социальные тэги и т. Д. Он также может извлекать информацию из служб каталогов, таких как Active Directory.

Мой сайт - персональный сайт для индивидуального пользователя для управления своей информацией и хранения документов, ссылок и т. Д. Он обеспечивает богатые сетевые и социальные функции, позволяя пользователям обмениваться информацией о себе или своей деятельности. Мой сайт доступен, щелкнув «Имя пользователя» в верхнем правом углу страницы SharePoint.

Управление и доступ к данным профиля пользователя

Поскольку мы собираемся работать с использованием JSOM, мы можем выполнять только операции «Read» с исключением, что изображение профиля может быть изменено с

помощью JSOM (или CSOM или REST)

* Код на стороне сервера позволяет читать / записывать обе операции.

Получение свойств профиля пользователя с помощью JSOM

Позволяет создавать хостинговое приложение SharePoint и получать информацию о пользователе в этом приложении -

Запустите Visual Studio 2013 и выберите «Приложение для SharePoint 2013» из New Project. После выбора типа проекта вам будет предоставлено окно для подключения к сайту SharePoint и выберите тип приложения для развертывания (см. Ниже скриншот). Здесь я предоставил URL-адрес сайта mu SharePoint Online и выбранное приложение Hosted Host. Нажмите «Готово».

3.) После создания проекта вы увидите набор файлов / папок, добавленных в Solution Explorer по умолчанию в проект.

4.) Если вы откроете страницу «Default.aspx», вы найдете некоторые библиотеки JavaScript, уже добавленные на страницу.

Здесь нам нужно добавить еще одну библиотеку, чтобы начать работу с профилями пользователей

Прочитайте Приложение SharePoint онлайн: https://riptutorial.com/ru/sharepoint/topic/9876/приложение-sharepoint

глава 4: Работа с модальными диалоговыми окнами с JavaScript

Синтаксис

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

параметры

Варианты недвижимости	Описание
заглавие	Строка, содержащая заголовок диалога
URL	Строка, содержащая URL-адрес страницы, которая появляется в диалоговом окне. Необходимо указать **URL-адрес** или **html** . **url** имеет приоритет над **html** .
HTML	HTML-элемент для отображения в диалоговом окне.
Икс	X-смещение диалога в виде целочисленного значения.
Y	У-смещение диалога в виде целочисленного значения.
ширина	Ширина диалога как целочисленное значение. Если unspecified и **autosize false,** ширина устанавливается равной 768px
рост	Высота диалога как целочисленное значение. Если unspecified и **autosize false,** высота устанавливается равной 576px
allowMaximize	Булевое значение, указывающее, должна ли отображаться кнопка **Maximize** .
showMaximized	Логическое значение, определяющее, открывается ли диалоговое окно.
showClose	Логическое значение, указывающее, появляется ли в диалоговом окне кнопка **Закрыть** .

Варианты недвижимости	Описание
авто размер	Логическое значение, указывающее, будет ли диалоговая платформа автоматически обрабатывать диалоги.
dialogReturnValueCallback	Указатель функции, который задает функцию обратного обратного вызова. Функция принимает два параметра: *dialogResult* типа SP.UI.DialogResult Enumeration и объект *returnValue,* который содержит любые данные, возвращаемые диалогом.
арг	Объект, содержащий данные, которые передаются в диалог.

замечания

`SP.UI.ModalDialog` имен `SP.UI.ModalDialog` было введено в `SP.UI.ModalDialog` модель JavaScript с SharePoint 2010 и доступно в следующих версиях SharePoint 2013, Office365 и 2016.

Дополнительные справочные материалы:

- Ссылка MSDN для SP.UI.ModalDialog.showModalDialog (параметры)
- Ссылка MSDN для перечисления SP.UI.DialogResult

Examples

Выполните действие, когда диалоговое окно закрыто

```
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;
        }
    }
}
```

Показать существующую страницу в диалоге

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

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

Показать пользовательский диалог

```
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);
}
```

Прочитайте Работа с модальными диалоговыми окнами с JavaScript онлайн: https://riptutorial.com/ru/sharepoint/topic/6868/работа-с-модальными-диалоговыми-окнами-с-javascript

глава 5: Работа с объектной моделью клиента JavaScript (JSOM)

замечания

Фон

Объектная модель JavaScript была представлена в SharePoint 2010. Она предоставляет на стороне клиента многие из объектов, которые ранее были доступны только через серверный код или через выделенные веб-службы.

Встраивание JavaScript в страницы SharePoint

В SharePoint 2013 вы можете разместить свой JavaScript в веб-части редактора сценариев.

В SharePoint 2010 вы можете использовать свойство «content link» веб-части Редактора контента для ссылки на HTML-файл, содержащий встроенный скрипт.

Ссылка на объект

Конструкторы, методы и свойства всех объектов, найденных в пространстве имен `SP` , описаны здесь в ссылке на объектную модель клиента SharePoint 2013.

Ссылка на объектную модель клиента JavaScript 2010 доступна здесь .

Шаблон асинхронного программирования JSOM

При использовании объектной модели клиента JavaScript код обычно принимает следующий шаблон:

1. Получить объект `ClientContext` .
2. Используйте объект `ClientContext` для извлечения объектов, представляющих объекты в объектной модели SharePoint, например списки, папку, представления.
3. Инструкции об очередях, выполняемые против объектов. Эти инструкции еще не передаются на сервер.
4. Используйте функцию `load` чтобы сообщить `ClientContext` какую информацию вы хотите получать с сервера.
5. `ClientContext` функцию `executeQueryAsync` объекта `ClientContext` , чтобы отправить на сервер запрошенные очереди, передав две функции обратного вызова для успешного выполнения или отказа.
6. В функции обратного вызова работайте с результатами, возвращаемыми с сервера.

альтернативы

Клиентские альтернативы JSOM включают веб-службы SharePoint, конечные точки REST и объектную модель .NET .

Examples

Получение типов содержимого библиотеки с использованием имени библиотеки

```
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());
}
```

Удаление элемента в списке

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

Создание элементов или папок

Создание элементов списка

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

В приведенном выше примере показано, что элемент списка создается путем выполнения следующих действий:

1. Вызовите метод `addItem` объекта списка, чтобы получить объект item
2. Вызовите метод `set_item` в результирующем объекте элемента списка, чтобы задать каждое значение поля по желанию
3. Вызовите метод `update` объекта объекта списка, чтобы указать, что изменения должны быть зафиксированы
4. Вызовите `executeQueryAsync` объекта контекста клиента для выполнения заданий в очереди

Обратите внимание, что вам **не** нужно передавать объект нового объекта в метод `load` контекста клиента для создания элемента. Этот шаг необходим только в том случае, если вы хотите получить какие-либо значения поля этого элемента с сервера.

Создание папок

Создание папки аналогично добавлению элемента в список. Разница заключается в том, что сначала нужно создать `ListItemCreationInformation` объект и установить его `underlyingObjectType` свойство `SP.FileSystemObjectType.folder` и его `leafName` собственности на нужное имя новой папки.

Затем объект передается в качестве параметра в методе `addItem` в библиотеке для создания папки.

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

Чтобы зафиксировать изменение, вызовите `executeQueryAsync` объекта `ClientContext` через который была `ClientContext` доступ к библиотеке.

Полный пример ниже создает папку с именем, основанным на текущей временной отметке, затем открывает эту папку в модальном диалоговом окне.

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

Получить информацию о текущем пользователе

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

Получить элемент списка по идентификатору

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

Получить элементы списка по запросу CAML

Основной пример

Используйте `set_viewXml` метод объекта SP.CamlQuery указать CAML запрос для извлечения элементов.

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

Подкачка результатов запроса CAML

Вы можете использовать элемент `RowLimit` в запросе CAML для получения только одного подмножества результатов с каждым запросом.

Используйте метод `get_listItemCollectionPosition` коллекции элементов списка для извлечения текущей позиции, а затем используйте это значение в качестве параметра в объекте set_listItemCollectionPosition объекта `set_listItemCollectionPosition` для получения следующей партии результатов.

```
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());
    }
```

```
}
```

Прочитайте Работа с объектной моделью клиента JavaScript (JSOM) онлайн: https://riptutorial.com/ru/sharepoint/topic/1316/работа-с-объектной-моделью-клиента-javascript--jsom-

глава 6: Работа с управляемой клиентской боковой объектной моделью (CSOM)

замечания

- Большинство примеров из MSDN .
- Чтобы создать управляемое клиентское приложение .NET, использующее объектную модель клиента, вы должны установить ссылки на две библиотеки DLL библиотеки: Microsoft.SharePoint.Client.dll и Microsoft.SharePoint.Client.Runtime.dll. Вы можете найти его в папке% ProgramFiles% \ Common Files \ Microsoft Shared \ веб-сервера \ 16 \ ISAPI или на вашем сервере SharePoint.
- или установите пакет Microsoft.SharePointOnline.CSOM NuGet, который будет работать «на предварительном этапе», а также в SP O365.
- Большинство свойств - это свойства ценности, и перед их доступом вам нужно явно вызвать clientContext.Load () и clientContext.ExecuteQuery (). Дополнительная информация здесь: вызов Load и ExecuteQuery перед доступом к свойствам свойств

Examples

Привет мир (получение названия сайта)

Все версии SharePoint основаны на сайтах (SPSite (SSOM) или сайте (CSOM)) и в Web (SPWeb (SSOM) или Web (CSOM)). Сайт не отображается в пользовательском интерфейсе, хотя он содержит метаданные и функции, которые применяются к его дочерним элементам. Веб - это основной строительный блок, который отображает пользовательский интерфейс для доступа пользователя к сайту. На всех сайтах есть корневая сеть, в которой хранятся информация и / или метаданные, такие как библиотеки документов. В этом примере показан базовый вызов для извлечения веб- `MyServer` расположенной на сервере `MyServer` на `sites` виртуального пути.

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

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

```
            {
                // To make it easier to read the code, pull the target web
                // context off of the client context and store in a variable
                Web oWebsite = clientContext.Web;
                // Tell the client context we want to request information about the
                // Web from the server
                clientContext.Load(oWebsite);
                // After we are done creating the batch of information we need from the sever,
                // request the data from SharePoint
                clientContext.ExecuteQuery();
                // Print the results of the query
                Console.WriteLine("Title: {0} Description: {1}", oWebsite.Title,
oWebsite.Description);
            }
        }
    }
}
```

Web. Получение свойств веб-сайта

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

Web. Получение только определенных свойств веб-сайта

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

Web. Обновление названия и описания веб-сайта

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

Web. Создание веб-сайта

```
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);
```

Список. Получение всех свойств всех списков на веб-сайте

```
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());
}
```

Список. Получение только указанных свойств списков

```
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"));
}
```

Список. Хранение извлеченных списков в коллекции

```
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"));
}
```

Список. Получение полей списка с веб-сайта

```
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);
        }
    }
}
```

Список. Создание и обновление списка

```
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();
```

Список. Добавление поля в список

```
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();
```

Список. Удаление списка

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

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

oList.DeleteObject();

clientContext.ExecuteQuery();
```

Вещь. Получение элементов из списка

```
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"]);
}
```

Вещь. Получение элементов (с помощью метода Include)

В этом примере показано, как извлекать элементы с сервера, а также получать более глубокие свойства каждого элемента списка. По умолчанию сервер будет возвращать минимальный объем данных для представления объекта. Пользователь должен запросить дополнительную информацию с сервера.

```
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);
}
```

Вещь. Получение определенных полей из определенного количества элементов

```
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"]);
}
```

Вещь. Извлечение элементов из всех списков на веб-сайте

```
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);
}
```

Вещь. Получение элементов с использованием позиции коллекции элементов списка

```
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");
}
```

Вещь. Создание элемента списка

При создании нового элемента списка его поля могут быть установлены с использованием синтаксиса, аналогичного строковым массивам. Обратите внимание, что эти поля не создаются «на лету» и определяются схемой списка. Эти поля (или столбцы) должны существовать на сервере, иначе создание не будет выполнено. Все элементы списка будут иметь поле Title. Некоторые списки могут иметь обязательные поля, которые должны быть заполнены до того, как элемент будет опубликован в списке.

В этом примере в списке используется шаблон объявлений. Помимо поля заголовка, список включает поле «Тело», в котором будет отображаться содержимое объявления в списке.

```
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();
```

Вещь. Обновление элемента списка

```
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();
```

Вещь. Удаление элемента списка

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

oListItem.DeleteObject();
```

```
clientContext.ExecuteQuery();
```

Группы. Получение всех пользователей из группы SharePoint

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

clientContext.Load(collUser);

clientContext.ExecuteQuery();

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

Группы. Получение определенных свойств пользователей

```
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);
}
```

Группы. Получение всех пользователей во всех группах семейства сайтов

```
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);
    }
}
```

Группы. Добавление пользователя в группу SharePoint

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

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

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

clientContext.ExecuteQuery();
```

Роли. Создание определения роли

```
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);
```

Роли. Назначение пользователя роли на веб-сайте

```
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);
```

Роли. Создание группы SharePoint и добавление группы к роли

```
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);
}
```

Права доступа. Нарушение наследования безопасности списка

```
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();
```

Права доступа. Нарушение наследования безопасности документа и добавление пользователя в качестве читателя

```
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();
```

Права доступа. Нарушение наследования безопасности документа и изменение разрешений пользователя

```
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();
```

Пользовательское действие. Добавление пользовательского действия для элементов списка

```
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();
```

Пользовательское действие. Изменение пользовательского действия

```
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();
    }
}
```

Пользовательское действие. Добавление пользовательских действий к действиям сайта на веб-сайте

```
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();
```

Веб-часть. Обновление заголовка веб-части

```
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();
```

Веб-часть. Добавление веб-части на страницу

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

string xmlWebPart = "<?xml version=\"1.0\" encoding=\"utf-8\"?>" +
    "<WebPart xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"" +
    " xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\"" +
    " xmlns=\"http://schemas.microsoft.com/WebPart/v2\">" +
    "<Title>My Web Part</Title><FrameType>Default</FrameType>" +
    "<Description>Use for formatted text, tables, and images.</Description>" +
    "<IsIncluded>true</IsIncluded><ZoneID></ZoneID><PartOrder>0</PartOrder>" +
    "<FrameState>Normal</FrameState><Height /><Width /><AllowRemove>true</AllowRemove>" +
    "<AllowZoneChange>true</AllowZoneChange><AllowMinimize>true</AllowMinimize>" +
    "<AllowConnect>true</AllowConnect><AllowEdit>true</AllowEdit>" +
    "<AllowHide>true</AllowHide><IsVisible>true</IsVisible><DetailLink /><HelpLink />" +
    "<HelpMode>Modeless</HelpMode><Dir>Default</Dir><PartImageSmall />" +
```

```
    "<MissingAssembly>Cannot import this Web Part.</MissingAssembly>" +
    "<PartImageLarge>/_layouts/images/mscontl.gif</PartImageLarge><IsIncludedFilter />" +
    "<Assembly>Microsoft.SharePoint, Version=13.0.0.0, Culture=neutral, " +
    "PublicKeyToken=94de0004b6e3fcc5</Assembly>" +
    "<TypeName>Microsoft.SharePoint.WebPartPages.ContentEditorWebPart</TypeName>" +
    "<ContentLink xmlns=\"http://schemas.microsoft.com/WebPart/v2/ContentEditor\" />" +
    "<Content xmlns=\"http://schemas.microsoft.com/WebPart/v2/ContentEditor\">" +
    "<![CDATA[This is a first paragraph!<DIV> </DIV>And this is a second
paragraph.]]></Content>" +
    "<PartStorage xmlns=\"http://schemas.microsoft.com/WebPart/v2/ContentEditor\"
/></WebPart>";

WebPartDefinition oWebPartDefinition = limitedWebPartManager.ImportWebPart(xmlWebPart);

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

oClientContext.ExecuteQuery();
```

Веб-часть. Удаление веб-части со страницы

```
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();
```

Контекст. Использование кеша учетных данных для повышенного выполнения кода

Хотя серверный код может работать с повышенными привилегиями, нет эквивалентного метода для повышения привилегий в клиентском коде (для очевидных причин безопасности). В качестве альтернативы вы можете указать учетные данные для эмуляции доступа определенного пользователя или учетной записи службы.

Чтобы указать учетные данные, создать и заполнить объект `CredentialCache` , назначьте его `ClientContext Credentials` объекта `ClientContext` .

Приведенный ниже пример эмулирует учетную запись пула приложений и предполагает среду среды SharePoint 2013 с NTLM.

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

using (ClientContext ctx = new ClientContext("https://onpremises.local/sites/demo/"))
{
    // need the web object
    ctx.Load(ctx.Web);
    ctx.ExecuteQuery();

    // here the default network credentials relate to the identity of the account
    // running the App Pool of your web application.
    CredentialCache credCache = new CredentialCache();
    cc.Add(new Uri(ctx.Web.Url), "NTLM", CredentialCache.DefaultNetworkCredentials);

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

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

Обратите внимание, что предоставление привилегий учетной записи с правами пула приложений в SharePoint противоречит наилучшей практике, но на их месте можно использовать любые соответствующие сетевые учетные данные.

Прочитайте Работа с управляемой клиентской боковой объектной моделью (CSOM) онлайн: https://riptutorial.com/ru/sharepoint/topic/2679/работа-с-управляемой-клиентской-боковой-объектной-моделью--csom-

глава 7: Работа с управляемой серверной моделью (полное доверие)

замечания

Концептуальная иерархия

В концептуальной иерархии SharePoint семейства **сайтов** содержат **сайты** , которые, в свою очередь, содержат **списки** . В `SPSite` сайтов (`SPSite`) нет явного пользовательского интерфейса, но он всегда содержит один сайт корневого уровня (доступный через свойство `RootWeb`) и, возможно, дополнительные подсайты под этим корневым сайтом. Сайт или веб-сайт (`SPWeb`) имеет пользовательский интерфейс и содержит библиотеки списков / документов (`SPList`), страницы с веб-страницами и элементы / документы (`SPListItem`).

Предостережения на стороне сервера

- Чтобы создать приложение, использующее объектную модель на стороне сервера SharePoint, в проекте Visual Studio вы должны добавить ссылку на сборку Microsoft.SharePoint, которая указана в разделе «Рамочные сборки».
- Приложения, использующие объектную модель на стороне сервера (полное доверие), могут выполняться только на сервере Windows Server, на котором размещается SharePoint.
- Вы не можете подключиться к серверу SharePoint, отличному от того, на котором работает приложение.

Examples

Hello World (получение названия сайта)

2013

SharePoint 2013 и более новые версии имеют только 64-разрядные версии, поэтому сборка / программа также должна быть построена для 64-битного процессора.

Сразу после создания вашего проекта необходимо перейти с **целевого объекта платформы** с **любого CPU** на **x64,** иначе будет возникать ошибка.

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

Цитирование через всю ферму SharePoint

Использование PowerShell выполняется с веб-сервера SharePoint:

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

Получить элементы списка

```
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
    }
}
```

Получение элементов с помощью подкачки

```
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);
}
```

Получить список по URL-адресу

```
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);
}
```

Создание элемента списка

При создании нового элемента списка его поля могут быть установлены с использованием синтаксиса, аналогичного строковым массивам. Обратите внимание, что эти поля не создаются «на лету» и определяются схемой списка. Эти поля (или столбцы) должны существовать на сервере, иначе создание не будет выполнено. Все элементы списка будут иметь поле Title. Некоторые списки могут иметь обязательные поля, которые должны быть заполнены до того, как элемент будет опубликован в списке.

В этом примере в списке используется шаблон объявлений. Помимо поля заголовка, список включает поле «Тело», в котором будет отображаться содержимое объявления в списке.

```
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();
}
```

Прочитайте Работа с управляемой серверной моделью (полное доверие) онлайн: https://riptutorial.com/ru/sharepoint/topic/7543/работа-с-управляемой-серверной-моделью--

полное-доверие-

глава 8: Рендеринг на стороне клиента SharePoint 2013

Вступление

Клиентская рендеринг (CSR) - это новая концепция, которая внедрена в SharePoint 2013. Она предоставляет вам механизм, позволяющий использовать собственный рендеринг вывода для набора элементов управления, размещенных на странице SharePoint (виды списка, формы списка и результаты поиска). Рендеринг сайта клиента - это просто, когда данные преобразуются с использованием клиента, а не сервера. Это означает использование клиентских технологий, таких как HTML и JavaScript, вместо необходимости писать XSLT.

Examples

Изменение гиперссылки полей / столбцов внутри списка с использованием CSR

В следующем примере показано, как изменить гиперссылку для поля « **ID** » и « **Заголовок (LinkTitle)** » внутри списка, используя CSR.

Шаг 1: Создайте JS-файл и вставьте ниже кода

```
(function () {

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

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

```
            }
        };
        SPClientTemplates.TemplateManager.RegisterTemplateOverrides(ctxForm);
    }
    ExecuteOrDelayUntilScriptLoaded(registerRenderer, 'clienttemplates.js');

})();
```

Шаг 2: Свойства веб-части GoTo в представлении «Список» и добавьте ссылку JS Link в этот вновь созданный файл js (например, ~ sitecollection / SiteAssets / CSRCodeFile.js)

(Примечание. Обратитесь к своему JSlink только в этом формате. ~ Sitecollection / YourJSfFilePath ".)

Шаг 3: Appy и Done

Скрыть столбец из списка SharePoint, используя CSR.

В этом примере показано, как скрыть поле «Дата» в представлении списка SharePoint с помощью CSR.

```
(function () {

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

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

})();
```

Применить валидации в форме New / Edit Item Form, используя CSR

Предположим, у нас есть список SharePoint, и у него есть четыре поля: Title, Full Name, Email, Mobile Number и т. Д. Теперь, если вы хотите применить специальную проверку в форме New / Edit Item, вы можете легко сделать это с помощью кода CSR. Приведенные ниже могут подтвердить следующие условия в формах:

- Пустые значения в полях
- Проверка формата идентификатора электронной почты с регулярным выражением
- Формат мобильного номера Проверка с регулярным выражением
- Поле Full Name не должно содержать числовые значения

Шаг: 1 Создайте JS-файл, скажем `CSRValidations.js` и скопируйте вставку следующий код в JS-файле

```
        (function () {

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

        SPClientTemplates.TemplateManager.RegisterTemplateOverrides(fieldContext);

    })();

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

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

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

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

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

        formCtx.registerClientValidator(formCtx.fieldName, validators);

        return "<span dir='none'><input type='text' value='" + formCtx.fieldValue + "'
```

```
maxlength='255' id='inpEmail' class='ms-long'> \ <br><span id='spnEmailError' class='ms-
formvalidation ms-csrformvalidation'></span></span>";
    }

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

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

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

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

        };
    };

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

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

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

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

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

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

        formCtx.registerClientValidator(formCtx.fieldName, validators);

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

    // Custom validation object to validate title format
```

```
    titleValidator = function () {
        titleValidator.prototype.Validate = function (value) {
            var isError = false;
            var errorMessage = "";

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

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

        };
    };

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

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

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

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

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

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

        formCtx.registerClientValidator(formCtx.fieldName, validators);

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

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

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

            if (value.trim() == "") {
                isError = true;
                errorMessage = "You must specify a value for this required field.";
            }else if (!mobilePhoneRejex.test(value) && value.trim()) {
                isError = true;
```

```
                errorMessage = "Please enter valid mobile phone number";
            }

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

        };
    };

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

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

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

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

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

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

        formCtx.registerClientValidator(formCtx.fieldName, validators);

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

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

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

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

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

        };
    };
```

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

Шаг: 2 Откройте форму «Новый элемент» в браузере. Отредактируйте страницу и отредактируйте веб-часть.

Шаг 3 В свойствах веб-части перейдите в раздел Разное -> JS link -> вставьте путь к вашему js-файлу (например, ~ sitecollection / SiteAssets / CSRValidations.js)

Шаг: 4 Сохраните свойства и страницу веб-части.

Изменить отображаемое имя столбца в виде списка с помощью CSR

Бывают случаи, когда вам нужно изменить отображаемое имя столбца в виде списка

например, имя столбца, отображаемое в представлении «IsApprovalNeeded», и вы хотите выглядеть как «Требуется ли одобрение?».

Вы можете, конечно, изменить отображаемое имя столбца, изменив заголовок столбца в настройках списка, но если вы хотите сохранить его так, как он есть в настройках списка, и изменить его только на предварительном просмотре страницы, вы можете сделать это, используя CSR (Client-Side-Rendering).

Вот код ...

```
(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');

})();
```

Прочитайте Рендеринг на стороне клиента SharePoint 2013 онлайн:
https://riptutorial.com/ru/sharepoint/topic/8317/рендеринг-на-стороне-клиента-sharepoint-2013

глава 9: Создание приложения, размещенного провайдером

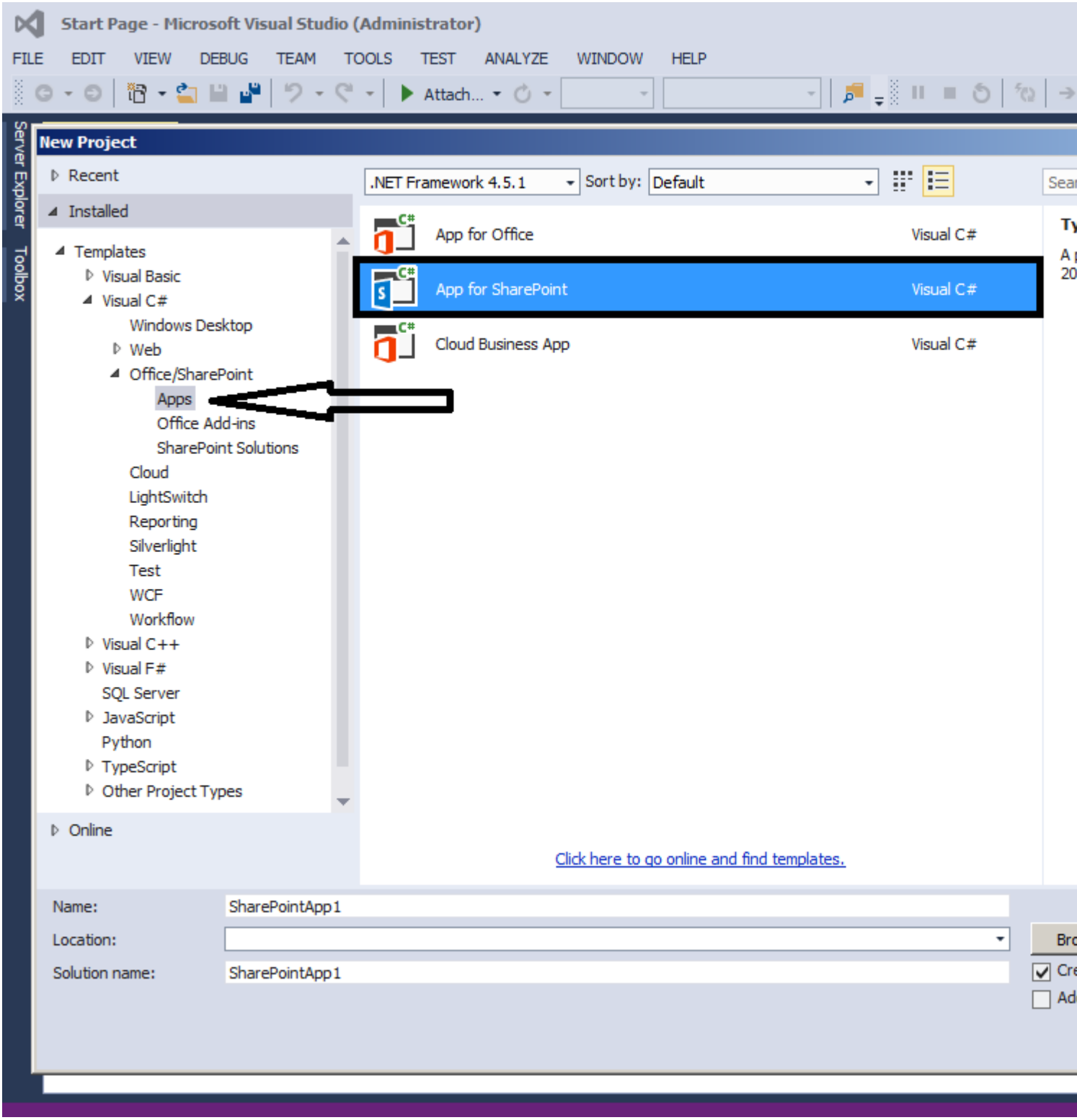
Examples

Настройка среды разработки

Для начала с App Development нам нужна Visual studio 2013 или более поздняя версия. Загрузите последнюю версию сообщества или выражения здесь> https://www.visualstudio.com/products/free-developer-offers-vs

Как только он был загружен и установлен

> Открыть и **клик создать новый проект**
>
> в разделе Office / SharePoint вы должны увидеть опцию для приложения, как показано ниже.



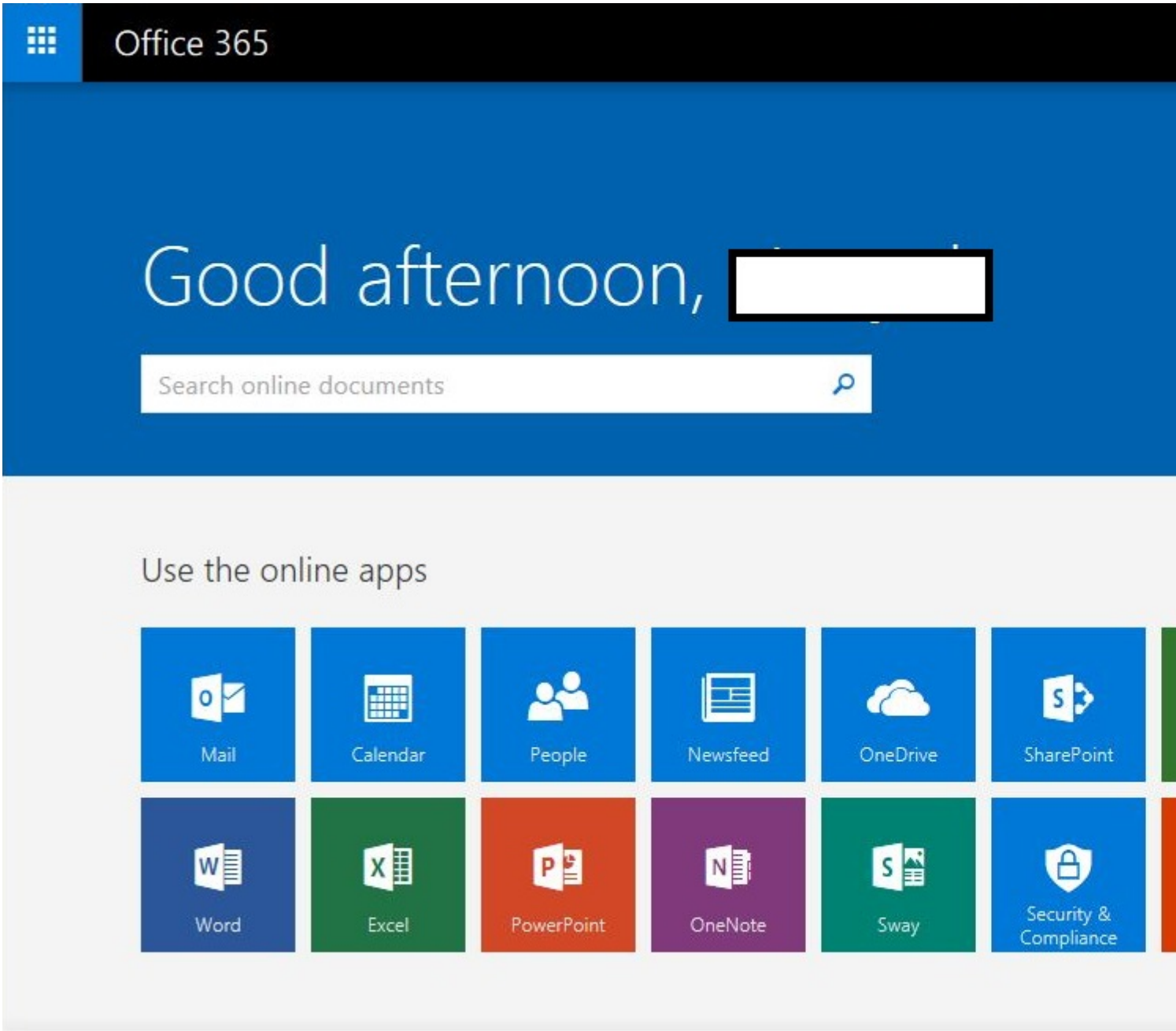
Если параметр «Приложение» недоступен, закройте VS, загрузите и установите **Microsoft Office Developer Tools** https://www.visualstudio.com/en-us/features/office-tools-vs.aspx

Подготовка к сайту разработчика

Когда у нас есть визуальная студия, нам нужен сайт разработчика для развертывания приложений в SharePoint. Самый простой способ - это> зарегистрироваться на бесплатную однолетнюю учетную запись разработчика Office 365 https://profile.microsoft.com/RegSysProfileCenter/wizardnp.aspx?wizid=14b845d0-938c-45af-

b061-f798fbb4d170&lcid=1033

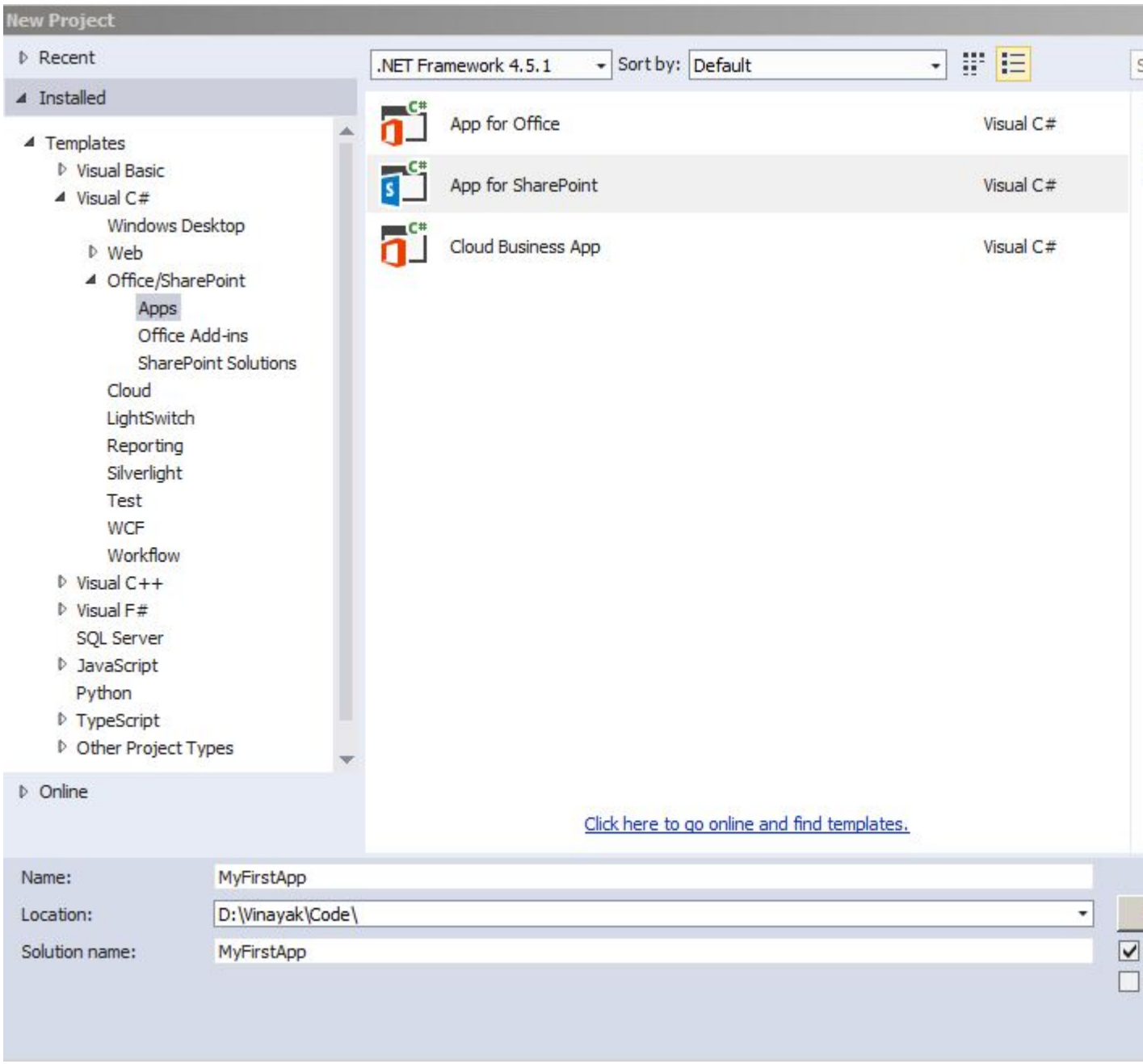
После завершения процесса регистрации https://www.office.com/ URL центра для всего вашего приложения



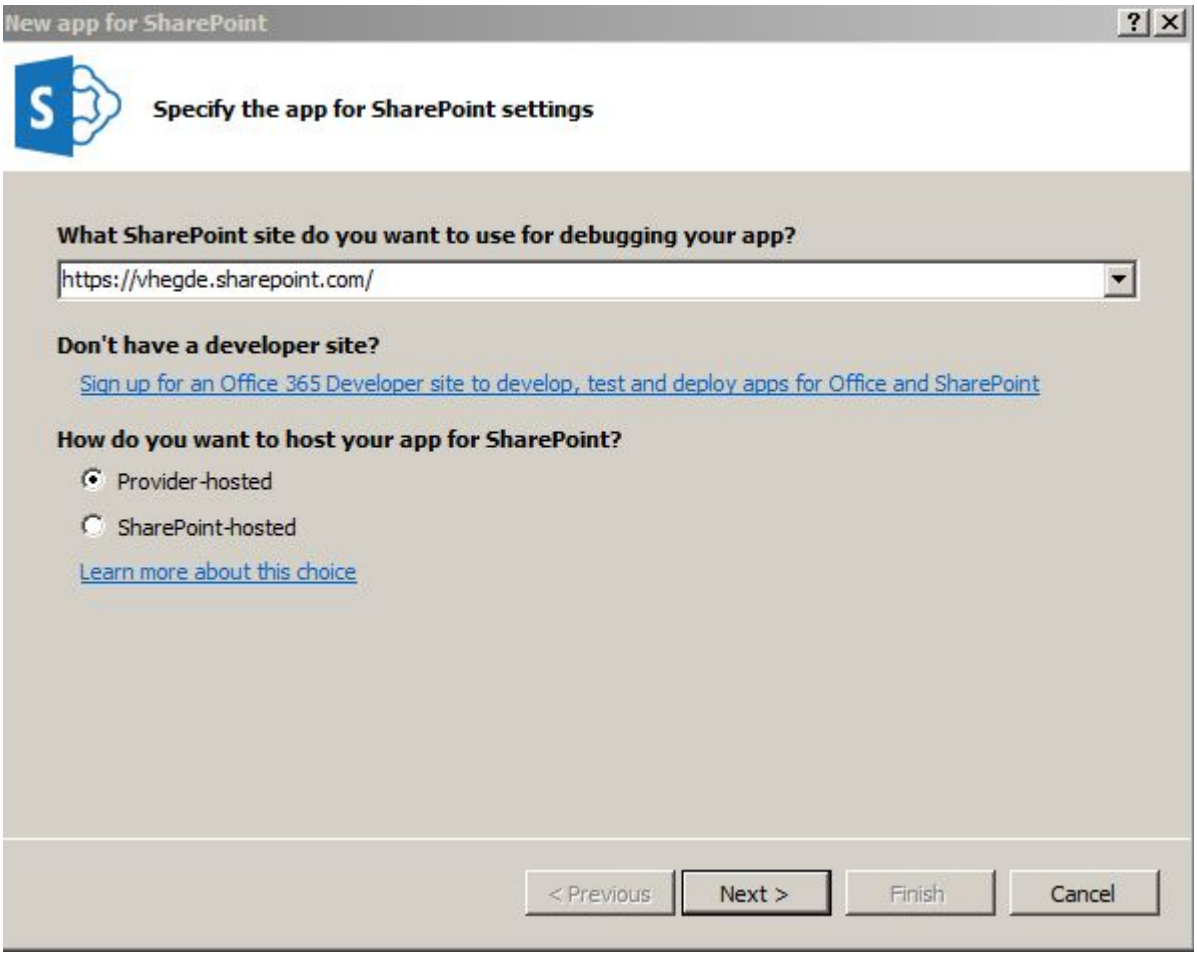
Создать приложение в Visual Studio

Давайте начнем с создания нашего первого приложения

1. Открыть визуальную студию и> создать новый проект
2. Введите имя и местоположение

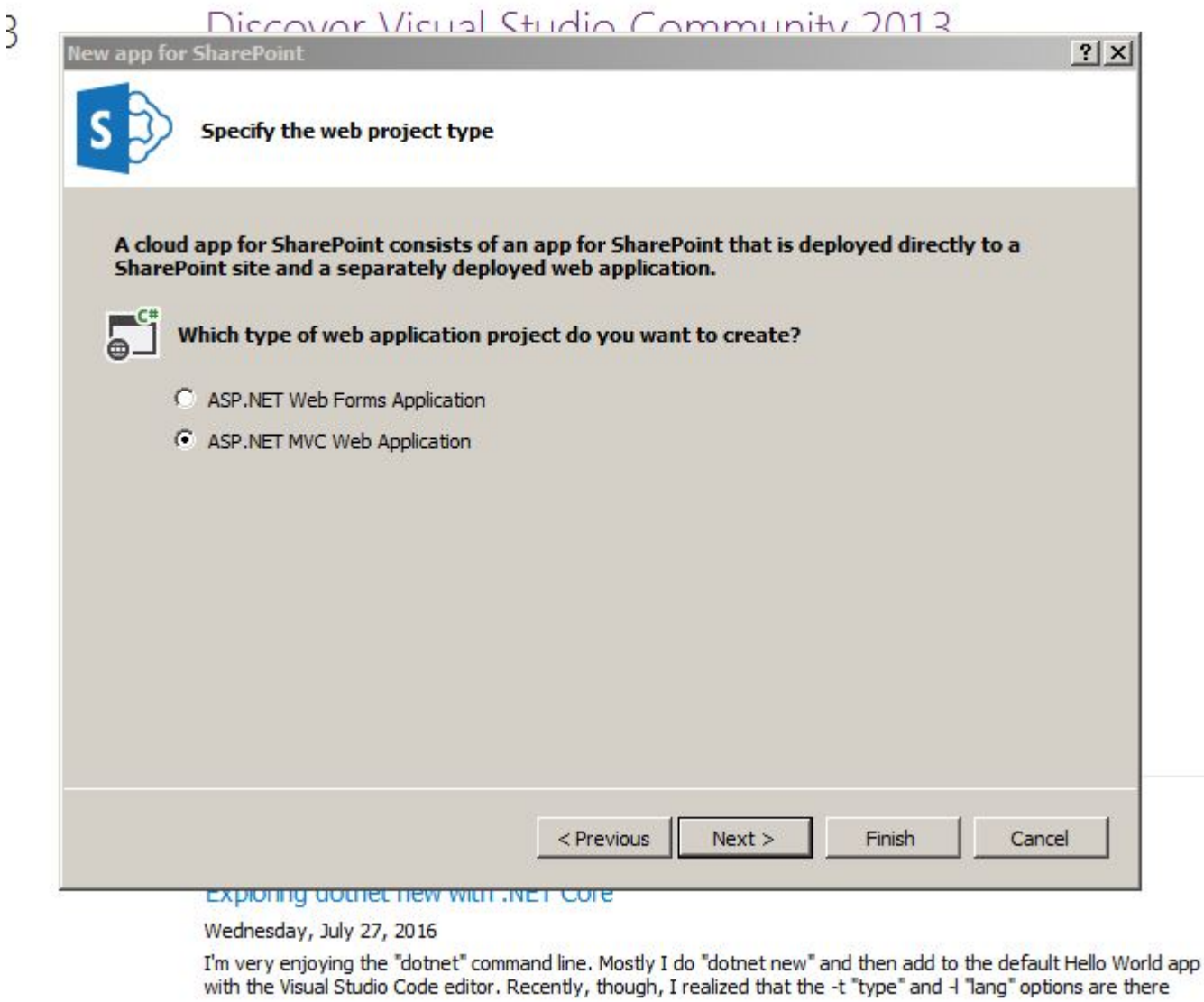


3. Введите URL-адрес вашего сайта разработчика, созданный на предыдущем шаге, и выберите Host-Host



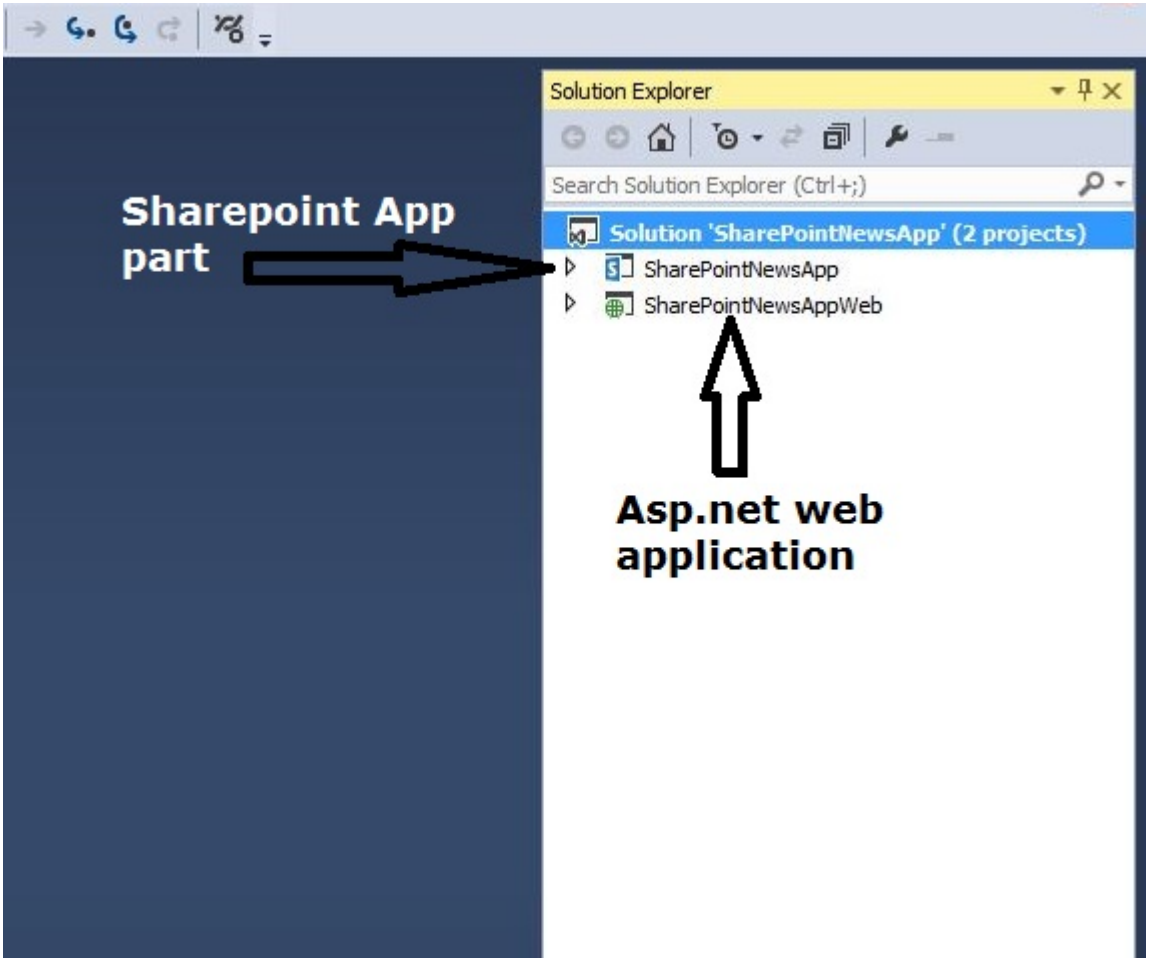
4. Всплывающее окно откроется, как для входа в систему

5. Следующим шагом будет как тип приложения, либо выбрать MVC или Webform. Я выбираю MCV здесь



6. В разделе «Как вы хотите, чтобы ваша надстройка была аутентифицирована?», Выберите «Использовать Windows Azure Access Control Service» и нажмите «Готово».

7. В обозревателе решений мы видим, что 2 проекта были созданы. Одна из них - часть приложения SharePoint, а другая - веб-приложение asp.net.

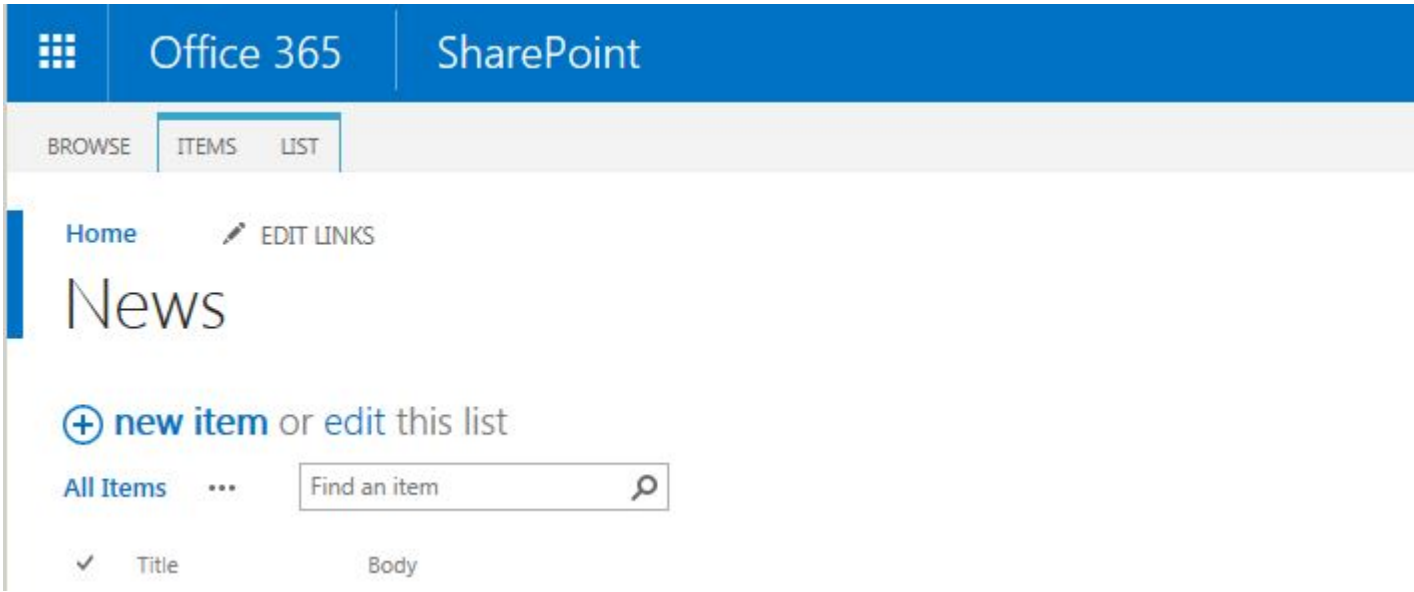


Позволяет начать кодирование

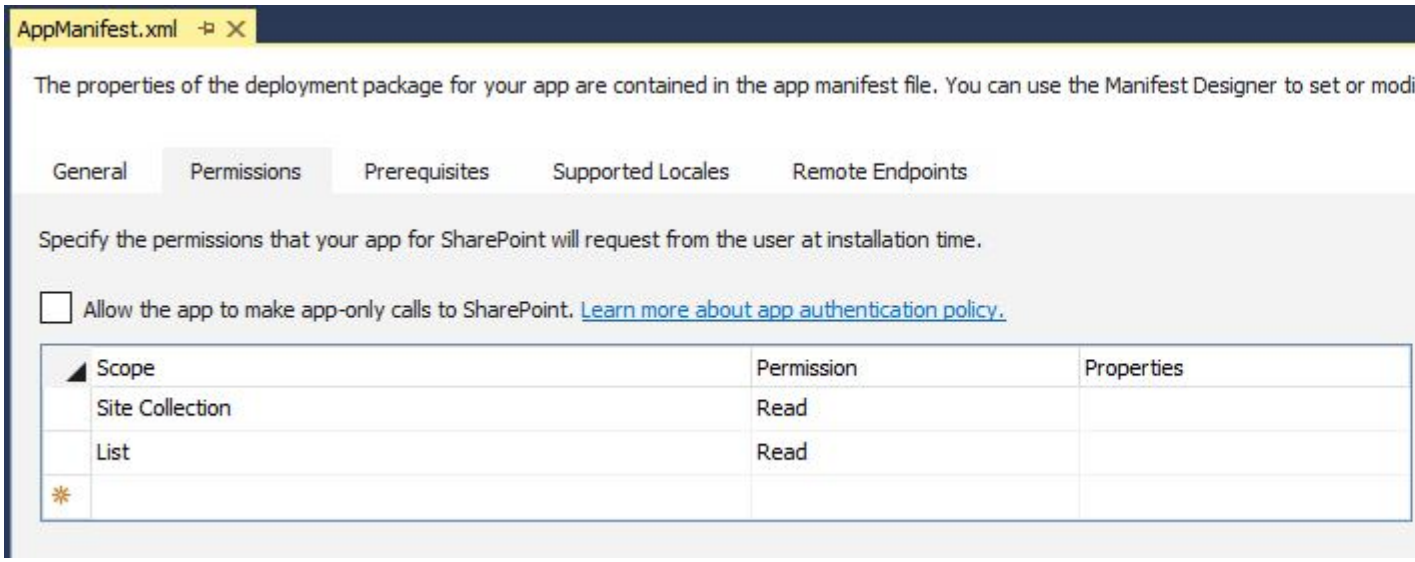
Здесь я беру пример базового приложения новостей

1. Откройте сайт разработчика SharePoint и создайте список для хранения наших новостных статей

2. Создайте собственный список и добавьте еще 3 столбца Body, Summery, ThumbnailImageUrl



3. Вернитесь в наше приложение SharePoint, откройте файл AppManifest.xml, щелкните вкладку «Разрешение» и дайте разрешение на чтение для семейства сайтов и сохраните его.



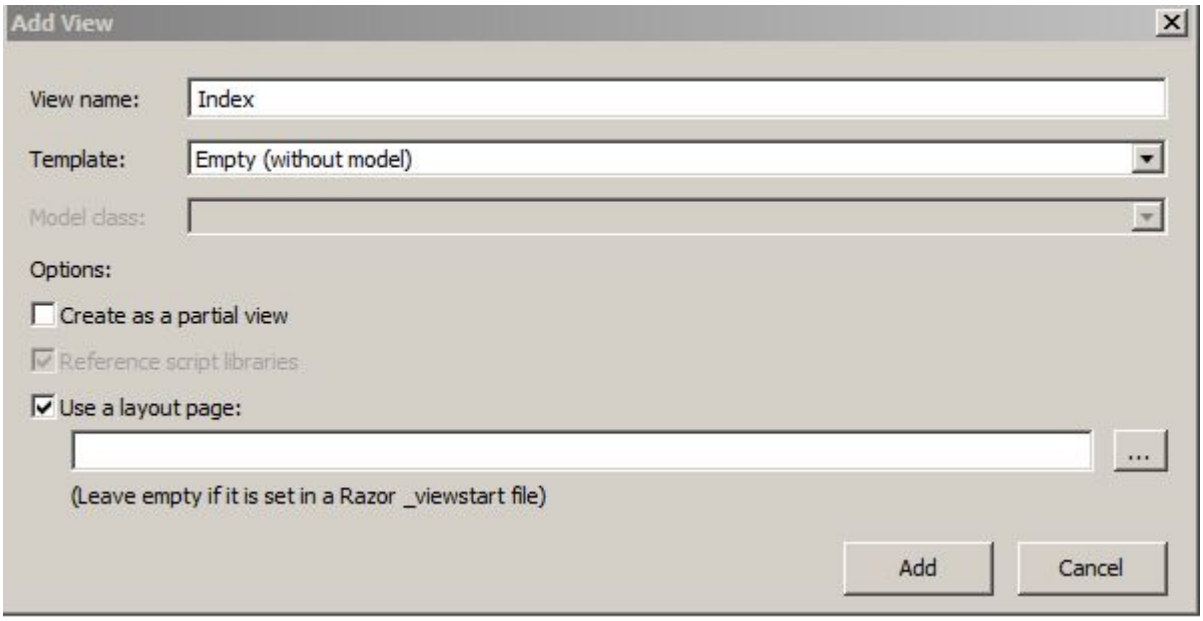
4. Откройте HomeController из веб-приложения, в моем случае это приложение MVC. Если вы создаете приложение webform, тогда код должен быть в файле default.aspx.cs

5. Ниже приведен фрагмент кода для получения последних новостей из списка. Это будет выглядеть наша индексная страница.

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

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

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

            clientContext.ExecuteQuery();

            ViewBag.UserName = spUser.Title;

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

            clientContext.ExecuteQuery();

            if (newsItems != null)
            {
```

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

    return View(newsList);
}
```

6. Теперь щелкните правой кнопкой мыши по **индексу** и нажмите « **Добавить вид».** Затем нажмите «Добавить».



7. Теперь откройте файл **Index.cshtml** From **Views> Home directory**

8. Ниже приведен фрагмент кода для файла index.cshtml.

```
@model List<SharePointNewsAppWeb.Models.NewsList>
 @{
 ViewBag.Title = "My News - browse latest news";
}
 <br />
 @foreach (var item in Model)
 {
 <div class="row panel panel-default">
  <div class="col-xs-3">
    <a href="/home/aticle?ArticleId=@item.Id">
        <img class="img-responsive" style="max-height:200px;max-width:100%;"
src="@item.Thumbnail" alt="@item.Title" />
    </a>
```

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

9. Щелкните правой кнопкой мыши папку Model в вашем решении и добавьте файл класса CS. Добавьте ниже классы моделей

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

 namespace SharePointNewsAppWeb.Models
{
  public class NewsApp
   {

   }
 public class NewsList
  {

public int Id { get; set; }

public string Title { get; set; }

public string Summery { get; set; }

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

  public int Id { get; set; }

  public string Title { get; set; }

  public string Body { get; set; }

 }
}
```

10. Используйте ключ F5 для развертывания и запуска надстройки. Если вы видите окно Security Alert, которое просит вас доверять самоподписанному сертификату Localhost, выберите «Да».

И теперь первое приложение готово

Создание полной страницы статьи

Мы уже создали первую страницу, на которой будут показаны все новостные статьи. На

этой странице будет показана полная статья.

1. Добавить еще один метод действий для HomeController

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

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

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

            clientContext.ExecuteQuery();

            ViewBag.UserName = spUser.Title;

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

            clientContext.ExecuteQuery();

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

2. Снова щелкните правой кнопкой мыши на Action и создайте представление с тем же именем Имя метода. В моем случае View будет называться **Story**

```
  @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>
```

Это код для полной страницы статьи, на которой показан текст статьи о новостях

Прочитайте Создание приложения, размещенного провайдером онлайн: https://riptutorial.com/ru/sharepoint/topic/6301/создание-приложения--размещенного-провайдером

глава 10: Услуги REST

замечания

URL-адрес конечной точки службы REST

API доступа клиента REST был впервые представлен в SharePoint 2010, но был значительно расширен в SharePoint 2013. API REST в SharePoint 2010 доступен через веб-службу `/_vti_bin/ListData.svc` URL-адресу `/_vti_bin/ListData.svc` . В SharePoint 2013 были введены URL-адреса `/_api/lists/` and `/_api/web` endpoint, которые ведут себя несколько иначе.

Вышеуказанным URL-адресам конечных точек должен предшествовать `http://server/site` где `server` представляет имя сервера, а `site` представляет собой имя или путь к определенному сайту.

Пример URL для ...	SharePoint 2010	SharePoint 2013
Получение списка:	`/_vti_bin/ListData.svc/ListName`	`/_api/lists('ListGuid')`
Получение элемента:	`/_vti_bin/ListData.svc/ListName(1)`	`/_api/lists('ListGuid')/items(1)`
Получение веб-страницы:	(без эквивалента)	`/_api/web`

Несмотря на различия в доступе к спискам и элементам списка, работа с этими результатами очень схожа в обеих версиях.

Обратите внимание, что служба `ListData.svc` по-прежнему доступна в SharePoint 2013 для обратной совместимости.

Отправка запросов REST

Запрос REST может быть отправлен через собственный XMLHttpRequest JavaScript или через конструкцию оболочки jQuery AJAX.

Синтаксис XMLHttpRequest

```
var xhr = new XMLHttpRequest();
```

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

Синтаксис JQuery AJAX

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

Подробнее о отправке запросов через AJAX см. В документации JavaScript AJAX .

Examples

Работа со списками

Получение элементов списка

В этом примере показано, как извлекать все элементы списка и проходить через них. Вы можете использовать `top` параметр для запроса определенного количества результатов. Вы также можете использовать параметр `select` для выбора определенных полей (`$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);
        }
    });
    }
```

Получение отдельного элемента списка

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);
        }
    });
    }
```

Получить элементы списка с колонками поиска

Иногда у вас может быть структура списка, которая выглядит так:

Список листинга животных

название	Тип	Описание
заглавие	Строка (текст)	Название животного
Возраст	Число	Сколько лет животное
Значение	валюта	Значение животного
Тип	*Поиск (таблица типов животных)*	Поле поиска (дает раскрывающийся список вариантов из таблицы типов животных)

Таблица видов животных

название	Тип	Описание
заглавие	Строка (текст)	Название вида / вида животных (например, свиньи)
NumLegs	Число	Количество ног на животном

Вероятно, это не самый серьезный пример, но проблема здесь остается в силе. Когда вы используете обычный запрос для извлечения значений из списка SharePoint, для `Type` животного вы получите только поле `TypeId` в ответе JSON. Чтобы развернуть эти элементы только в одном вызове AJAX, в параметрах URL требуется дополнительная разметка.

Этот пример применим не только к столбцам поиска. Когда вы используете столбцы «

`People/Groups` , они, по сути, просто ищут, поэтому вы можете легко переносить такие предметы, как `Title` , `EMail` и другие.

Пример кода

Важное примечание . Когда вы определяете поля, которые хотите вернуть из столбцов поиска, вы должны указать имя поля с именем поля поиска в исходной таблице. Например, если вы хотите вернуть атрибут `NumLegs` из столбца поиска, вы должны ввести `Type/NumLegs` .

JavaScript

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

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

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

    return $.ajax(ajaxOptions);
}

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

    var selectFields = [
        "Title",
        "Age",
```

```
        "Value",
        "Type/Title",
        "Type/NumLegs"
    ];

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

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

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

// Start the entire process
getAnimals();
```

Добавление выбора в многозначное поле поиска

В этом примере предполагается, что столбец поиска называется `MultiLookupColumnName` и что вы хотите настроить поле поиска нескольких поисковых `MultiLookupColumnName` на элементы с идентификаторами 1 и 2.

Использование jQuery AJAX

2010

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

2013

```
var listGuid = "id-of-list-to-update"; // use list GUID here
```

```
var lookupGuid = "id-of-lookup-list"; // use lookup list GUID here
var idOfItemToUpdate = 1;
var url = "/server/site/_api/lists('"+ listGuid + "')/items("+ idOfItemToUpdate + ")";
var data = JSON.stringify({
    MultiLookupColumnName:[
        {__metadata:{uri:"http://yoursiteurl/_api/lists('" + lookupGuid + "')/items(1)"}},
        {__metadata:{uri:"http://yoursiteurl/_api/lists('" + lookupGuid + "')/items(2)"}}
    ]
});
$.ajax({
    method: 'POST',
    url: url,
    contentType: 'application/json',
    headers: {
      "X-HTTP-Method" : "MERGE",
      "If-Match" : "*"
    },
    data: data
});
```

Использование 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);
```

Элементы списка подкачки, возвращаемые из запроса

Чтобы смоделировать пейджинг с помощью REST, вы можете сделать следующее:

1. Используйте параметр `$skip=n` чтобы пропустить первые `n` записей в соответствии с параметром `$orderby`
2. Используйте параметр `$top=n` чтобы вернуть верхние `n` записей в соответствии с параметрами `$orderby` и `$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();
            }
        }
    );
}
```

Получить идентификатор вновь созданного элемента в списке SharePoint

В этом примере показано, как получить идентификатор вновь созданного элемента с помощью API REST SharePoint.

Замечания :

listName - эта переменная содержит имя вашего списка.

newItemBody - это будет ваш орган запроса для добавления нового элемента в список.

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

Как выполнять операции CRUD с использованием интерфейса REST SharePoint 2010

Создайте

Чтобы выполнить операцию Create через REST, вы должны выполнить следующие действия:

Создайте HTTP-запрос, используя глагол `POST` . Используйте URL-адрес службы, в который вы хотите добавить объект в качестве цели для POST. Задайте тип содержимого для `application/json` . Сериализуйте объекты JSON, которые представляют ваши новые элементы списка как строку, и добавьте это значение в тело запроса. Пример JavaScript:

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

использование

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

Читать

Чтобы выполнить операцию чтения через REST, вы должны выполнить следующие действия:

Создайте HTTP-запрос, используя `GET` глагол. Используйте URL-адрес службы элемента списка, к которому вы хотите добавить объект в качестве цели для GET. Задайте тип содержимого для `application/json` . Пример JavaScript:

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

использование

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

Обновить

Чтобы обновить существующий объект, вы должны выполнить следующие действия:

Создайте HTTP-запрос, используя глагол `POST` . Добавьте заголовок `X-HTTP-Method` со значением `MERGE` . Используйте URL-адрес службы элемента списка, который вы хотите обновить, в качестве цели для POST. Добавить заголовок `If-Match` со значением исходного ETag объекта или *. Пример JavaScript:

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

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

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

}
```

использование

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

удалять

Чтобы удалить объект, необходимо выполнить следующие действия:

Создайте HTTP-запрос, используя глагол `POST` . Добавьте заголовок `X-HTTP-Method` со значением `DELETE` . Используйте URL-адрес службы элемента списка, который вы хотите обновить, в качестве цели для POST. Добавить заголовок `If-Match` со значением исходного ETag объекта. Пример JavaScript:

```
function deleteListItem(webUrl, listName, itemId, success, failure) {
    getListItemById(webUrl,listName,itemId,function(item){
        $.ajax({
            url: item.__metadata.uri,
            type: "POST",
            headers: {
                "Accept": "application/json;odata=verbose",
                "X-Http-Method": "DELETE",
                "If-Match": item.__metadata.etag
            },
            success: function (data) {
                success();
            },
            error: function (data) {
                failure(data.responseJSON.error);
            }
        });
    },
   function (error) {
       failure(error);
   });
}
```

использование

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

Прочитайте Услуги REST онлайн: https://riptutorial.com/ru/sharepoint/topic/3045/услуги-rest

кредиты

S. No	Главы	Contributors
1	Начало работы с sharepoint	Community, Marco, Ryan Gregg, Thriggle, Tom Resing, Zach Koehne
2	Основные выпуски	jjr2527, MikhailSP
3	Приложение SharePoint	Sunil sahu
4	Работа с модальными диалоговыми окнами с JavaScript	Thriggle
5	Работа с объектной моделью клиента JavaScript (JSOM)	Thriggle, yngrdyn
6	Работа с управляемой клиентской боковой объектной моделью (CSOM)	InvoiceGuy, Lukáš Nešpor, MikhailSP, RamenChef, Thriggle, Zach Koehne
7	Работа с управляемой серверной моделью (полное доверие)	Lukáš Nešpor, Thriggle
8	Рендеринг на стороне клиента SharePoint 2013	Rohit Waghela, Yayati
9	Создание приложения, размещенного провайдером	vinayak hegde
10	Услуги REST	Aaron, Brock Davis, ocelotsloth, R4mbi, Rohit Waghela, Thriggle