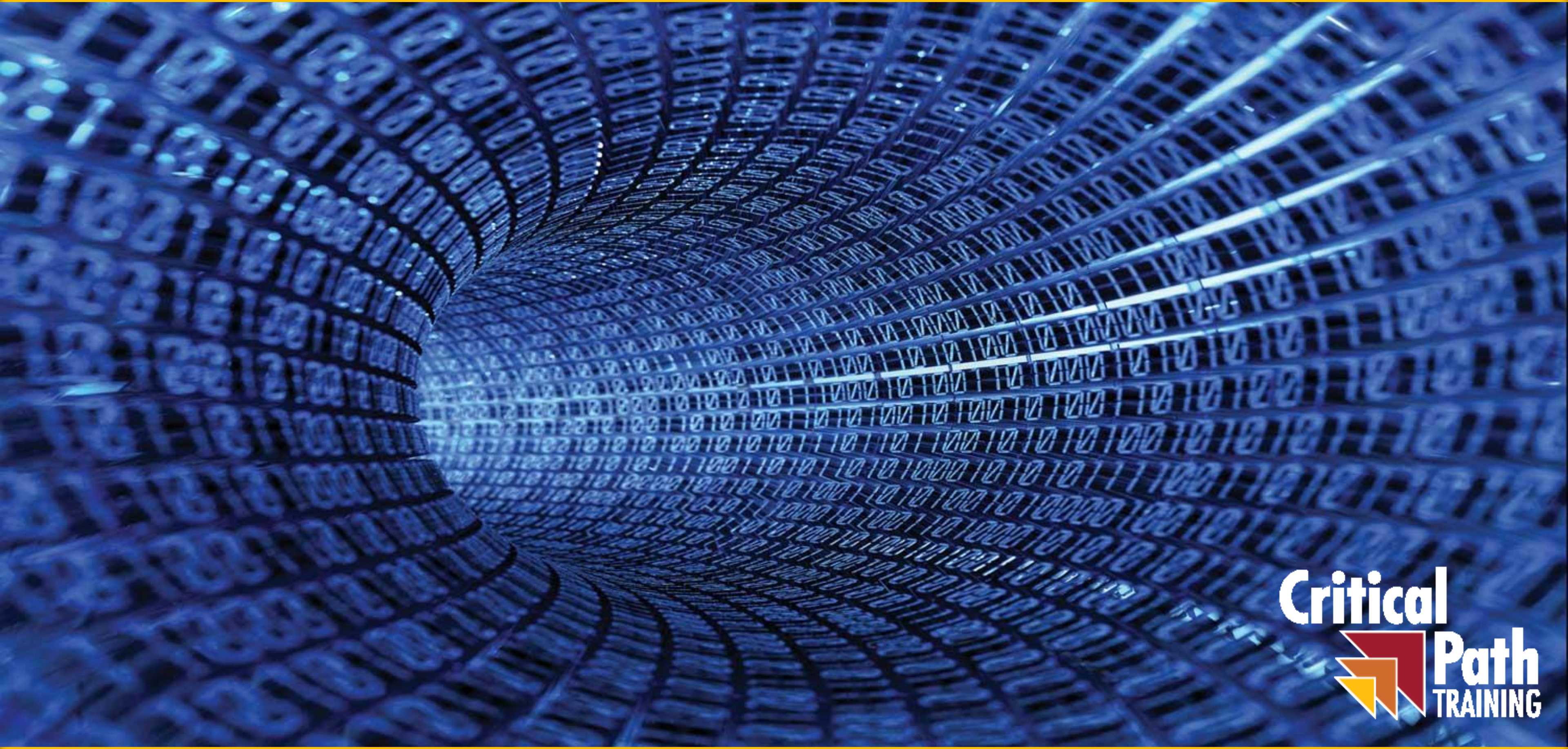
Developing SharePoint Add-ins



Agenda

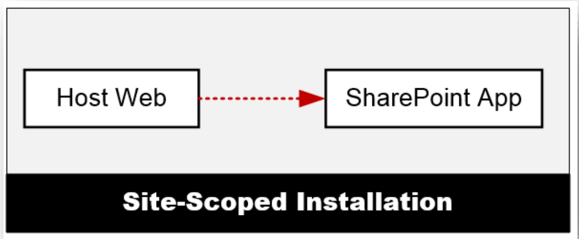
- SharePoint Add-in Model Overview
- SharePoint-hosted Add-in
- Programming the SharePoint REST API
- SharePoint Add-in Security
- Provider-hosted Add-ins
- Acquiring and Managing Access Tokens

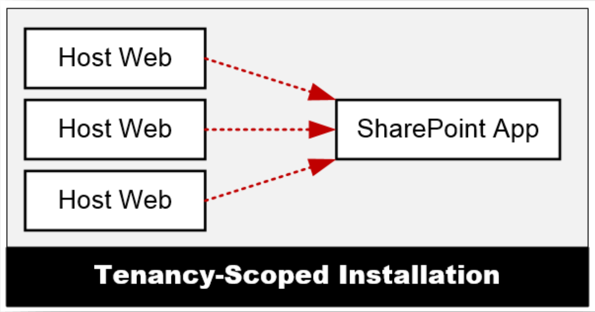
SharePoint App Add-in Model

- SharePoint 2013 introduced new development model
  - Originally introduced as "SharePoint App” model
  - Marketing folks renamed "SharePoint App” to "SharePoint Add-in”
- Add-in model designed to replace farm solutions
  - Add-ins designed to supported SPO and SharePoint on-premises
  - Add-in code not allowed to run on SharePoint host server
  - Add-in talks to SharePoint using REST and CSOM
  - Add-in authenticates and establishes add-in identity
  - Add-in has permissions independent of user
  - Add-ins deployed to catalogs using publishing scheme

Add-in Installation Scopes

- Site-scoped Installation
  - Add-in installed in SharePoint site which becomes **host web**
  - Add-in can be installed multiple times across site collections
  - Each installed instance of an add-in gets its own app web

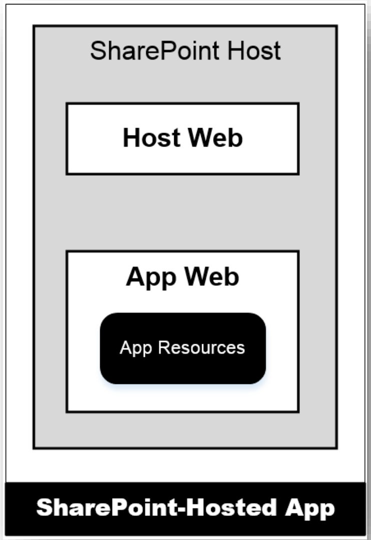




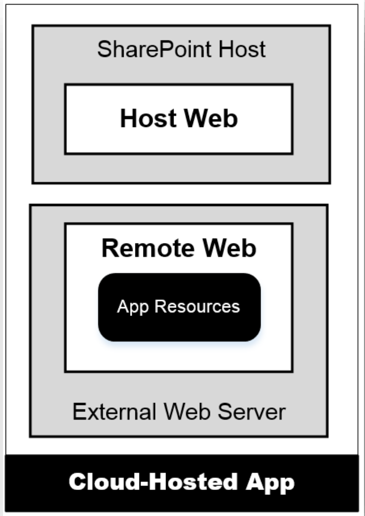
- Tenancy-scoped Installation
  - Provides centralized approach to app deployment & management
  - Requires app to first be installed in an app catalog site
  - Once installed, the app is then configured for use multiple sites
  - Tenancy install scoped to web application in on-premises farms

Hosting Options for SharePoint Add-ins

- SharePoint-Hosted Add-ins
  - App resources added to SharePoint host
  - Stored in child site known as **app web**
  - Add-in can have only client-side code
  - Add-in cannot have server-side code

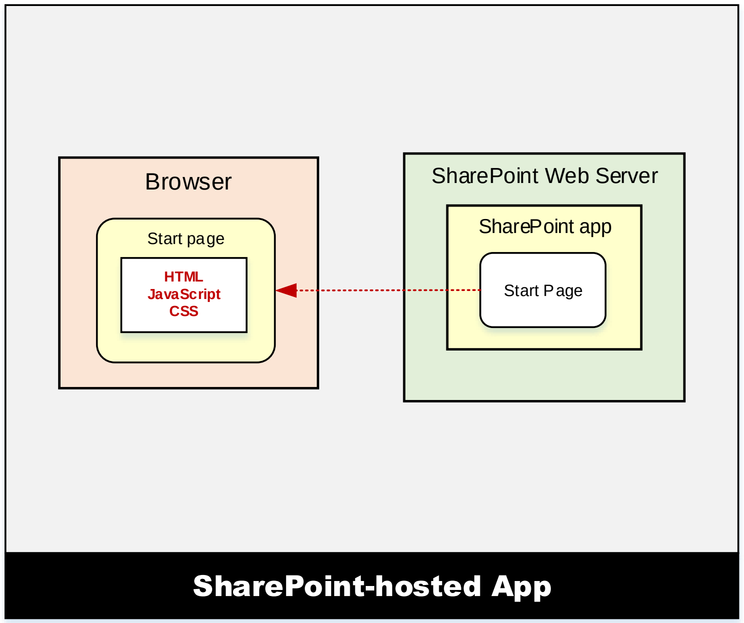


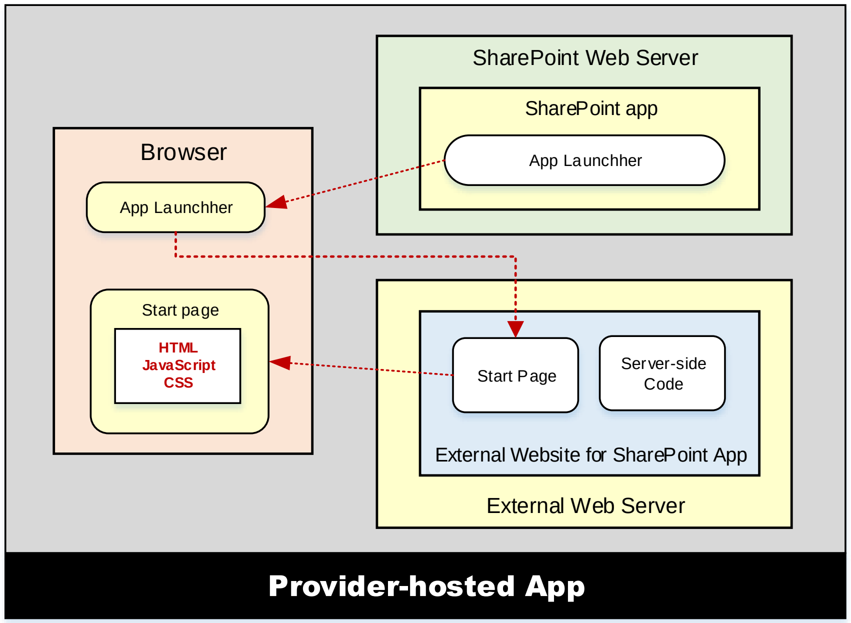
- Provider-Hosted Add-ins
  - Add-in pages deployed to remote server
  - Remote site known as **remote web**
  - Add-in can have client-side code
  - Add-in can have server-side .NET code



Add-in Start Page

- Every SharePoint add-in requires a start page
  - Start page provides entry point into add-in
  - SharePoint adds app launcher to Site Contents in host web
  - SharePoint-Hosted add-in start page hosted by SharePoint
  - Provider-Hosted add-in start page hosted in remote web





SharePoint Add-in User Interface Design

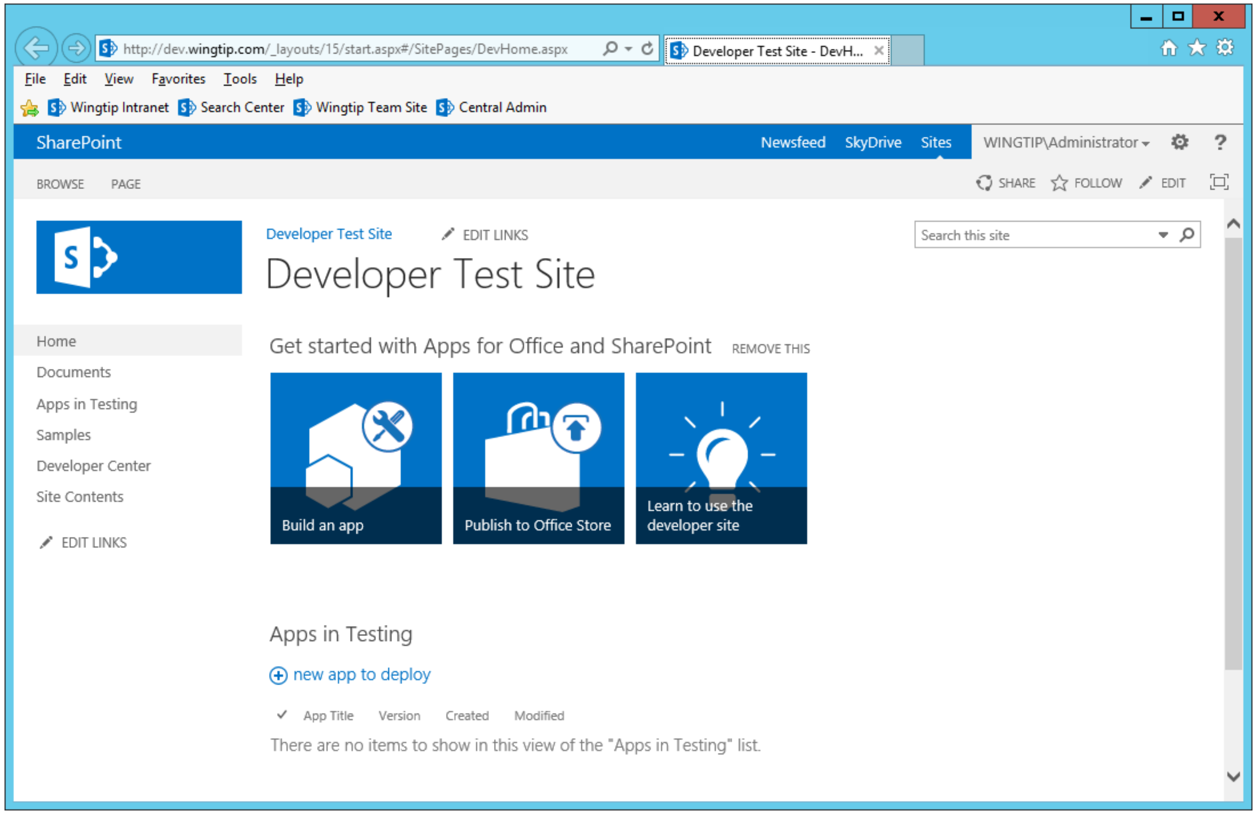
- Start page (required)
  - Represents user entry point into add-in
  - Can be implemented with .aspx file or .htm file
- Add-in Parts
  - External page (e.g. from app web) surfaced in host web
  - Displayed on host web pages using iFrame
- User Custom Actions
  - URL-based command surfaced in host web
  - Used to create ECB commands and ribbon controls

DEMO

Working with Add-ins from the Business User Perspective

Developer Sites

- Allows for <u>remote</u> add-in installation by Visual Studio
  - Required for testing add-ins in SharePoint Online environment



Developer Test Site

Get started with Apps for Office and SharePoint

Apps in Testing

There are no items to show in this view of the "Apps in Testing" list.

DEMO

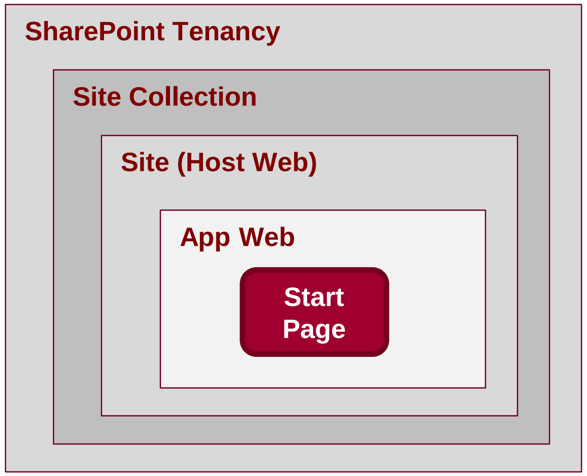
Creating a Developer Site

Agenda

- ✓ SharePoint Add-in Model Overview
- ➢ SharePoint-hosted Add-in
- Programming the SharePoint REST API
- SharePoint Add-in Security
- Provider-hosted Add-ins
- Acquiring and Managing Access Tokens

SharePoint-hosted Add-in Architecture

- SharePoint-hosted app fundamentals
  - SharePoint host creates app web during installation
  - App start page and resources are added into app web
  - All app logic must be written in client-side JavaScript
  - App authentication happens behind the scenes



App Web (aka Add-in Web)

- App web is created during app installation
  - App web created as child to site where app is installed
- SharePoint-hosted apps must create app web
  - App must add start page and related resources
  - App can add other SharePoint elements (e.g. lists)
- Provider-hosted apps *can* create app web
  - Provider-hosted apps will not create app web by default
  - Provider-hosted app can create app web if needed

App Web Hosting Domain

- App web pages served out of isolated domain
  - Isolates JavaScript code on app web pages
  - Allows SharePoint to authenticate callbacks from app

```
https://mytenant-ee060af276f95a.sharepoint.com/MyFirstApp
```

- URL to app web made up of 4 parts
  - **Tenancy name:** mytenant
  - **APPUID:** ee060af276f95a
  - **App web hosting domain:** sharepoint.com
  - **App name:** MyFirstApp

Start Page URL

- Dynamic tokens used in start page URL
  - SharePoint-Hosted apps use **~appWebUrl** token

    `~appWebUrl/Pages/Default.aspx`

  - All apps should use **{StandardTokens}** token

    `~appWebUrl/Pages/Default.aspx?{StandardTokens}`

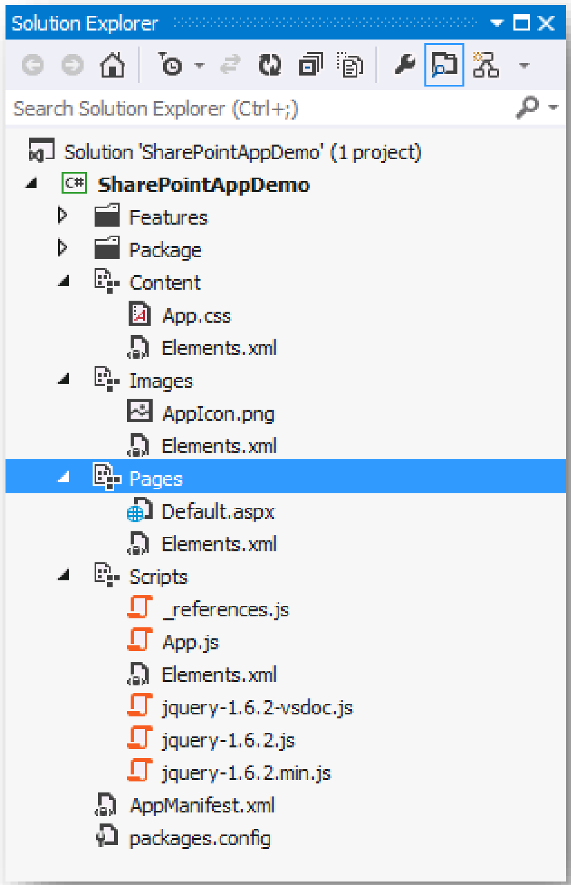
{StandardTokens}

- Start Page URL contains {StandardTokens}
  - Dynamic placeholder for querystring parameters

Parameter	Purpose
SPHostUrl	URL back to host web
SPAppWebUrl	URL to app web
SPLanguage	Language in use (e.g. en-US)
SPClientTag	Client cache control number for the current website.
SPProductNumber	Version of SharePoint (e.g. 15.0.4433.1011)

Modules in a SharePoint-Hosted Add-in

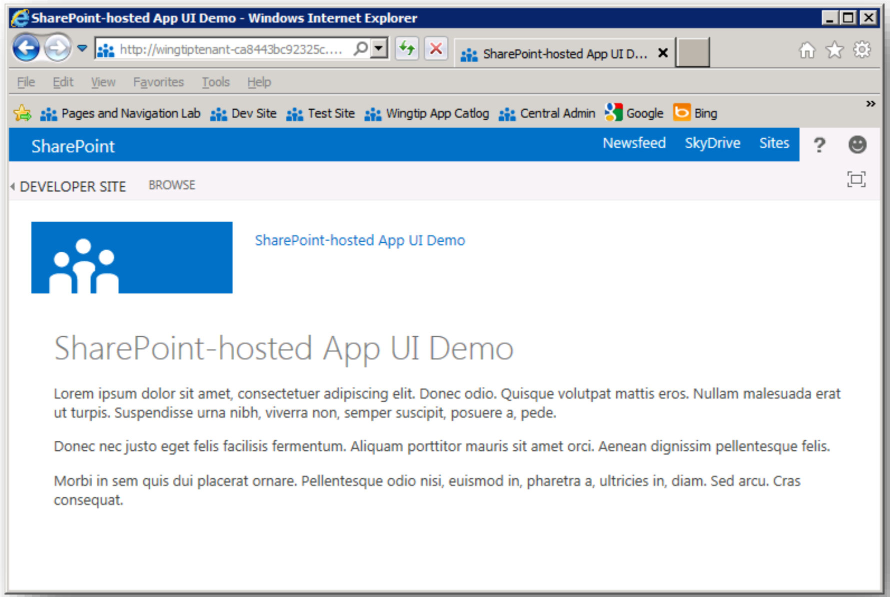
- Visual Studio adds Modules to each new project
  1. Content
  2. Images
  3. Pages
  4. Scripts



```
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'SharePointAppDemo' (1 project)
◢ SharePointAppDemo
    ▷ Features
    ▷ Package
    ◢ Content
        App.css
        Elements.xml
    ◢ Images
        AppIcon.png
        Elements.xml
    ◢ Pages
        Default.aspx
        Elements.xml
    ◢ Scripts
        _references.js
        App.js
        Elements.xml
        jquery-1.6.2-vsdoc.js
        jquery-1.6.2.js
        jquery-1.6.2.min.js
    AppManifest.xml
    packages.config
```

App.master

- App web uses `app.master` by default
  - Gives app SharePoint look and feel
  - Provides app with required link back to host web
  - Does not have Site Actions menu or top link bar
  - Does <u>not</u> support adding Office 365 app launcher
  - Should not be used for add-ins targeting SharePoint Online


SharePoint-hosted App UI Demo - Windows Internet Explorer
http://wingtiptenant-ca8443bc92325c....
SharePoint-hosted App UI D...
File Edit View Favorites Tools Help
Pages and Navigation Lab
Dev Site
Test Site
Wingtip App Catlog
Central Admin
Google
Bing
SharePoint
Newsfeed
SkyDrive
Sites
DEVELOPER SITE
BROWSE
SharePoint-hosted App UI Demo

SharePoint-hosted App UI Demo

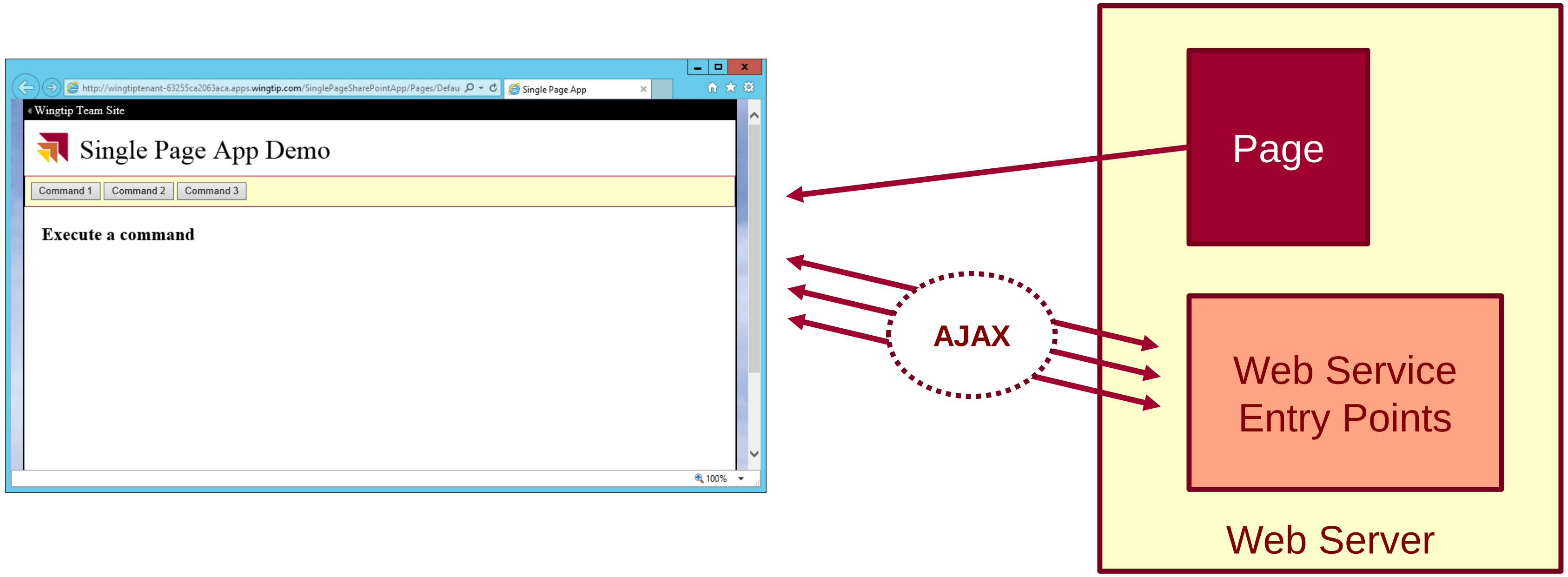
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec odio. Quisque volutpat mattis eros. Nullam malesuada erat ut turpis. Suspendisse urna nibh, viverra non, semper suscipit, posuere a, pede.

Donec nec justo eget felis facilisis fermentum. Aliquam porttitor mauris sit amet orci. Aenean dignissim pellentesque felis.

Morbi in sem quis dui placerat ornare. Pellentesque odio nisi, euismod in, pharetra a, ultricies in, diam. Sed arcu. Cras consequat.

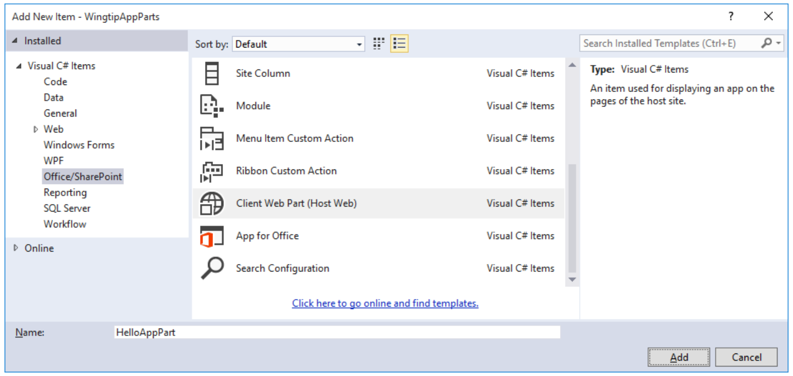
Single Page App (SPA) Model

- Web applications often designed as SPAs
  - Design leads to better and more fluid user experience
  - Request data posted to start page is always there
  - JavaScript variables do not unload/reload
  - App makes AJAX calls and uses client-side JavaScript



Adding Add-in Parts to a Project

- Add new item based on Client Web Part project item



Add New Item - WingtipAppParts

Installed
- Visual C# Items
  - Code
  - Data
  - General
  - Web
  - Windows Forms
  - WPF
  - Office/SharePoint
  - Reporting
  - SQL Server
  - Workflow
- Online

Sort by: Default

Site Column	Visual C# Items
Module	Visual C# Items
Menu Item Custom Action	Visual C# Items
Ribbon Custom Action	Visual C# Items
Client Web Part (Host Web)	Visual C# Items
App for Office	Visual C# Items
Search Configuration	Visual C# Items

Click here to go online and find templates.

Search Installed Templates (Ctrl+E)

Type: Visual C# Items

An item used for displaying an app on the pages of the host site.

Name: HelloAppPart

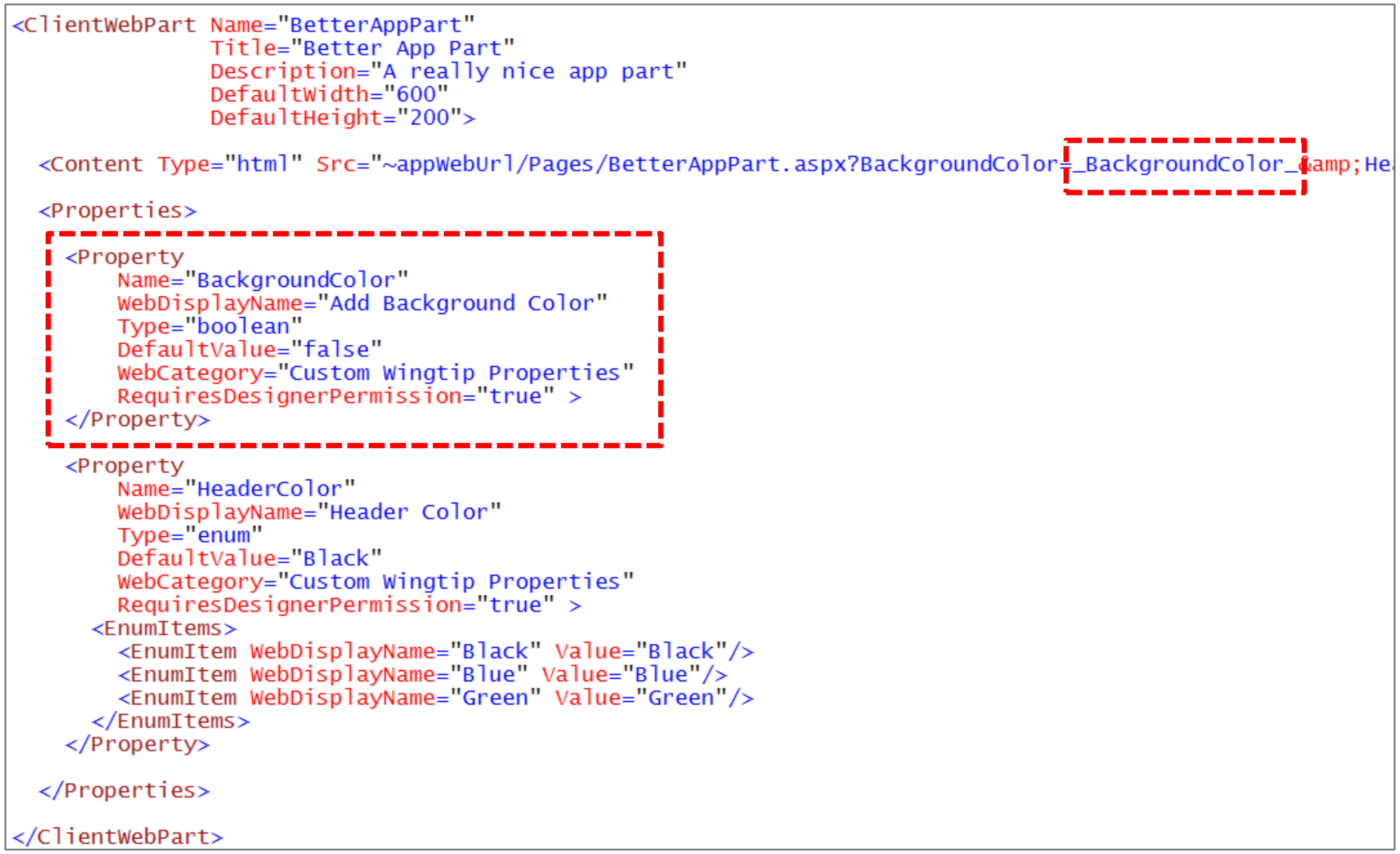
Add Cancel

- App part requires ClientWebPart definition in element.xml

```xml
<?xml version="1.0" encoding="utf-8"?>

<Elements xmlns="http://schemas.microsoft.com/sharepoint/">

  <ClientWebPart
    Name="HelloAppPart"
    Title="Hello App Part"
    Description="A simple little app part"
    DefaultWidth="600"
    DefaultHeight="200">

    <Content Type="html" Src="~appWebUrl/Pages/HelloAppPart.htm?{StandardTokens}" />

    <Properties></Properties>

  </ClientWebPart>

</Elements>
```

Add-in Parts with Custom Properties

- Add-in part can define custom properties
  - Property defined using Property element
  - Property value sent to add-in part using query string



```xml
<ClientWebPart Name="BetterAppPart"
               Title="Better App Part"
               Description="A really nice app part"
               DefaultWidth="600"
               DefaultHeight="200">

  <Content Type="html" Src="~appWebUrl/Pages/BetterAppPart.aspx?BackgroundColor=_BackgroundColor_&amp;He

  <Properties>

    <Property
        Name="BackgroundColor"
        WebDisplayName="Add Background Color"
        Type="boolean"
        DefaultValue="false"
        WebCategory="Custom Wingtip Properties"
        RequiresDesignerPermission="true" >
    </Property>

    <Property
        Name="HeaderColor"
        WebDisplayName="Header Color"
        Type="enum"
        DefaultValue="Black"
        WebCategory="Custom Wingtip Properties"
        RequiresDesignerPermission="true" >
      <EnumItems>
        <EnumItem WebDisplayName="Black" Value="Black"/>
        <EnumItem WebDisplayName="Blue" Value="Blue"/>
        <EnumItem WebDisplayName="Green" Value="Green"/>
      </EnumItems>
    </Property>

  </Properties>

</ClientWebPart>
```

Resizing Add-in Parts

- Add-in part displayed in host web using inside iFrame
  - IFrame given initial width and height
  - Dynamic resizing often required to avoid scrollbars
  - Resizing add-in part requires postMessage call to host

```
function resizeAppPart() {

  var pageWidth = $("#appPartContainer").width() + 20;
  var pageHeight = $("#appPartContainer").height() + 20;

  var SPHostUrl = getQueryStringParameter("SPHostUrl");
  var senderId = getQueryStringParameter("SenderId");
  var message = "<message senderId=" + senderId + ">" +
                  "resize(" + pageWidth + ", " + pageHeight + ")" +
                "</message>";

  parent.postMessage(message, SPHostUrl);


}
```

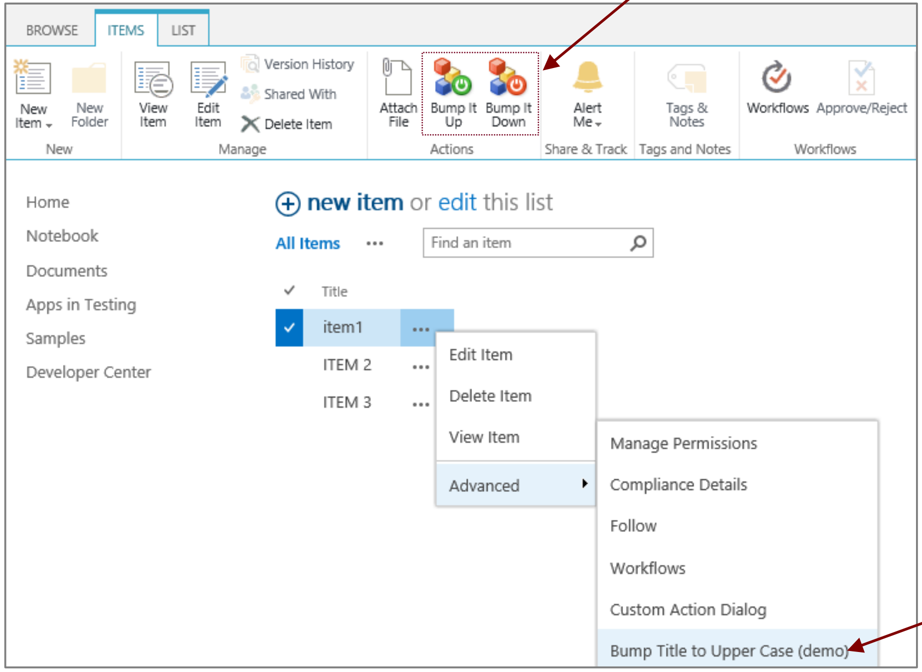
DEMO

Developing Add-in Parts

Creating User Custom Actions

- User custom actions used to add commands to host web
  - Custom action can create ECB menu items and ribbon buttons

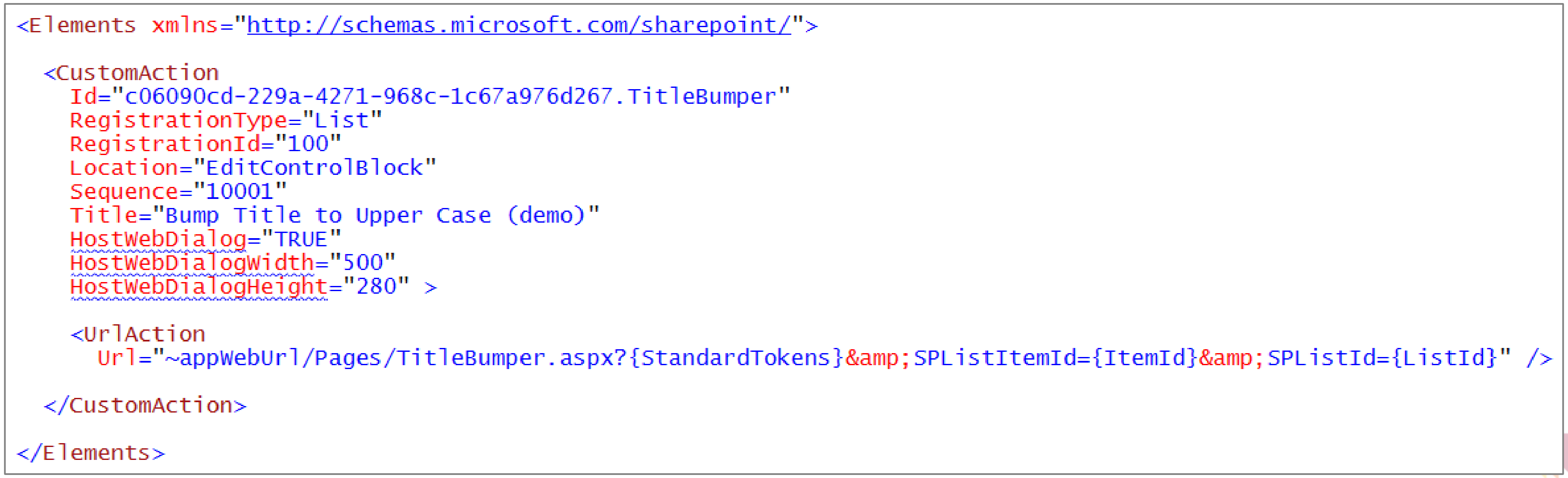
Custom ribbon buttons



Custom ECB menu item buttons

Creating User Custom Actions

- User custom actions used to add menu items to host web
  - Custom action can create ECB menu items and ribbon buttons
  - Created using declarative CustomAction element
  - UrlAction links to page in app web or remote web
  - UrlAction Url attribute cannot contain any JavaScript code
  - HostWebDialog attribute displays page in model dialog in host web



```
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">

  <CustomAction
    Id="c06090cd-229a-4271-968c-1c67a976d267.TitleBumper"
    RegistrationType="List"
    RegistrationId="100"
    Location="EditControlBlock"
    Sequence="10001"
    Title="Bump Title to Upper Case (demo)"
    HostWebDialog="TRUE"
    HostWebDialogWidth="500"
    HostWebDialogHeight="280" >

    <UrlAction
      Url="~appWebUrl/Pages/TitleBumper.aspx?{StandardTokens}&amp;SPListItemId={ItemId}&amp;SPListId={ListId}" />

  </CustomAction>

</Elements>
```

URL Tokens for User Custom Actions

- Certain tokens must be used with certain actions
  - Token use changes between ECB actions and Ribbon actions

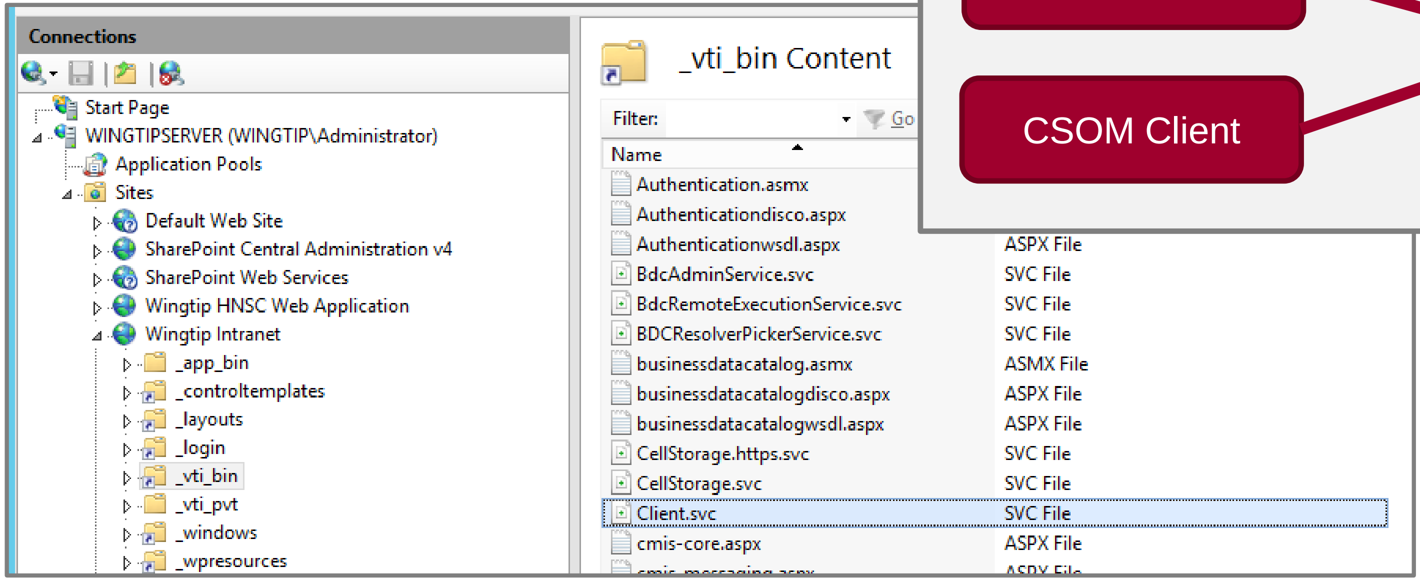
Token	Purpose
{AppWebUrl}	URL of the app web in an app for SharePoint
{HostLogoUrl}	Logo for the host web of an app for SharePoint
{HostTitle}	Title of the host web of an app for SharePoint
{HostUrl}	URL of the host web of an app for SharePoint
{ItemId}	Integer-based ID of item in a list or library **(ECB menu actions only)**
{SelectedItemId}	Array of item IDs in a list or library **(Ribbon menu actions only)**
{ItemUrl}	URL of target item being acted upon **(ECB menu actions only)**
{SelectedItemUrl}	URL array of target items being acted upon (**Ribbon menu actions only)**
{Language}	current language/culture of the host web of an app for SharePoint
{ListId}	ID of the current list (a GUID).
{RecurrenceId}	Recurrence index of a recurring event
{Site}	URL of the current website
{SiteCollection}	URL of the parent site of the current website
{SiteUrl}	URL of the current website

Agenda

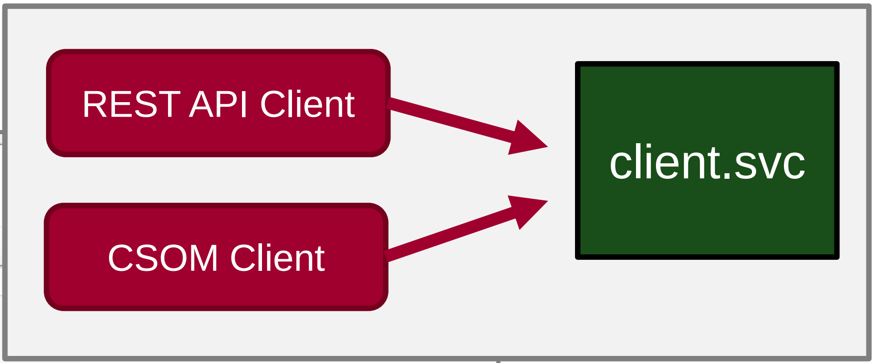
- ✓ SharePoint Add-in Model Overview
- ✓ SharePoint-hosted Add-in
- ➢ Programming the SharePoint REST API
- SharePoint Add-in Security
- Provider-hosted Add-ins
- Acquiring and Managing Access Tokens

SharePoint REST API Architecture

- REST API entry point is client.svc
  - In SharePoint 2010, client.svc only used by CSOM
  - In SharePoint 2013, client.svc used by CSOM and REST API



Name	
Authentication.asmx	
Authenticationdisco.aspx	
Authenticationwsdl.aspx	ASPX File
BdcAdminService.svc	SVC File
BdcRemoteExecutionService.svc	SVC File
BDCResolverPickerService.svc	SVC File
businessdatacatalog.asmx	ASMX File
businessdatacatalogdisco.aspx	ASPX File
businessdatacatalogwsdl.aspx	ASPX File
CellStorage.https.svc	SVC File
CellStorage.svc	SVC File
Client.svc	SVC File
cmis-core.aspx	ASPX File



SharePoint REST URLs and the _api Alias

- SharePoint REST API provides _api alias
  - The **_api** alias maps to **_vti_bin/client.svc**
  - Alias used to make SharePoint REST API URLs cleaner
  - Alias serves to decouple URLs from underlying architecture
- This URL works but it is not recommended
  - **http://intranet.wingtip.com/_vti_bin/client.svc/web**
- SharePoint REST API URLs should be created with _api
  - **http://intranet.wingtip.com/_api/web**

Anatomy of a SharePoint REST URL

- SharePoint REST made up of three parts
  - Base URI

    `http://intranet.wingtip.com/_api`
  - Target SharePoint Object

    `web`
  - Query String Parameter options

    `?$select=Id,Title,MasterUrl`

```
http://intranet.wingtip.com/_api/web/?$select=Id,Title,MasterUrl
```

Mapping SharePoint Objects to URLs

SharePoint Object	Object mapping
Site Collection	`site`
Site	`web`
Lists collection	`web/lists`
List by ID	`web/lists(guid'402cd788-9c5c-4931-92d6-09f18efb368c')`
List by Title	`web/lists/getByTitle('Customers')`
List property	`web/lists/getByTitle('Customers')/Title`
List items collection	`web/lists/getByTitle('Customers')/items`
List item	`web/lists/getByTitle('Customers')/items(1)`
List item property	`web/lists/getByTitle('Customers')/items(1)/FirstName`

ODATA Formats and the Accept Header

- OData v3 only supports OData verbose format

```
accept: application/json;odata=verbose
```

- OData v4 supports also minimal metadata format

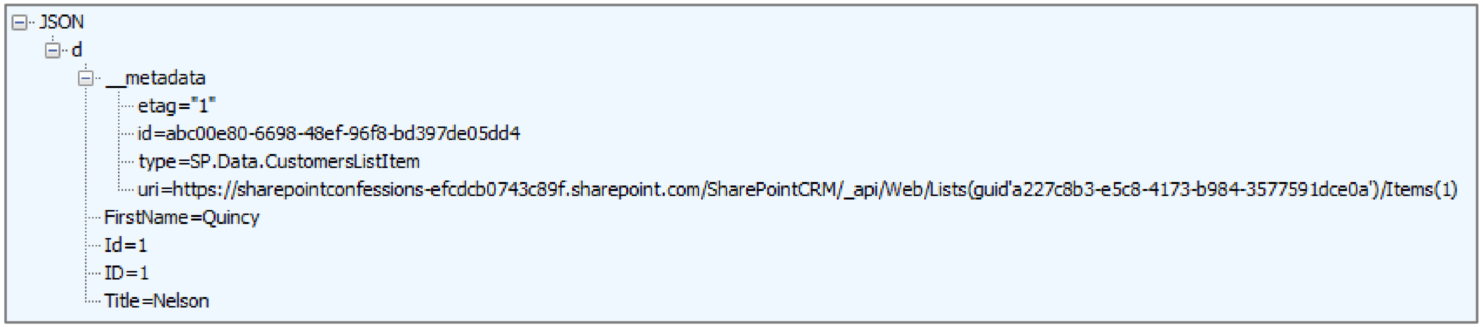
```
accept: application/json
accept: application/json;odata=minimalmetadata
```

- OData v4 also support no metadata format

```
accept: application/json;odata=nometadata
```

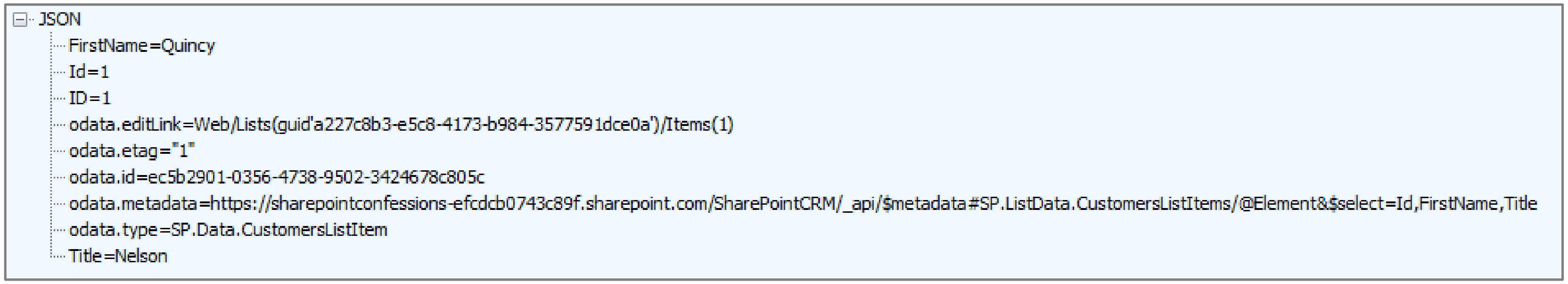
Comparing JSON Formats

- When using **application/json;odata=verbose**



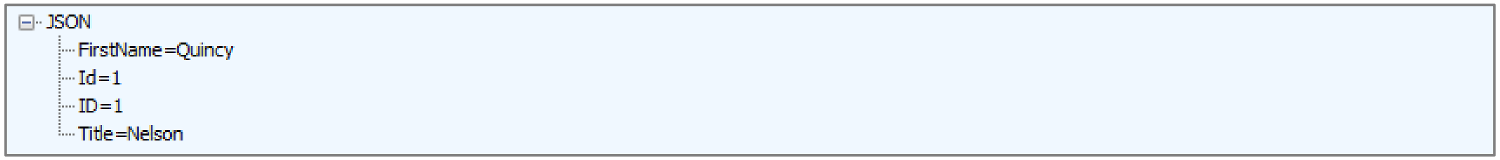
```
JSON
  d
    __metadata
      etag="1"
      id=abc00e80-6698-48ef-96f8-bd397de05dd4
      type=SP.Data.CustomersListItem
      uri=https://sharepointconfessions-efcdcb0743c89f.sharepoint.com/SharePointCRM/_api/Web/Lists(guid'a227c8b3-e5c8-4173-b984-3577591dce0a')/Items(1)
    FirstName=Quincy
    Id=1
    ID=1
    Title=Nelson
```

- When using **application/json** or **application/json;odata=minimalmetadata**



```
JSON
  FirstName=Quincy
  Id=1
  ID=1
  odata.editLink=Web/Lists(guid'a227c8b3-e5c8-4173-b984-3577591dce0a')/Items(1)
  odata.etag="1"
  odata.id=ec5b2901-0356-4738-9502-3424678c805c
  odata.metadata=https://sharepointconfessions-efcdcb0743c89f.sharepoint.com/SharePointCRM/_api/$metadata#SP.ListData.CustomersListItems/@Element&$select=Id,FirstName,Title
  odata.type=SP.Data.CustomersListItem
  Title=Nelson
```

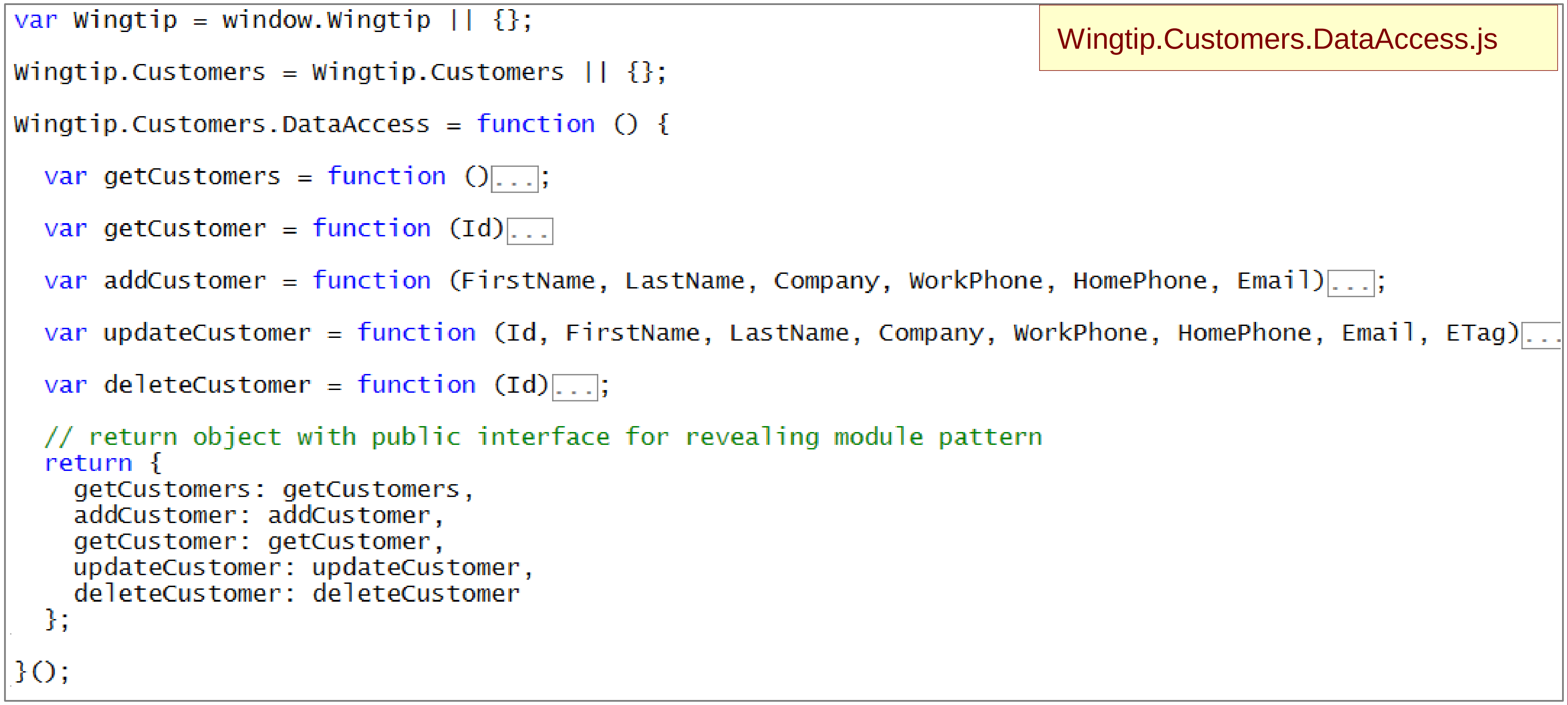
- When using **application/json;odata=nometadata**



```
JSON
  FirstName=Quincy
  Id=1
  ID=1
  Title=Nelson
```

Separating UI Code form Data Access Code

- You should not intermingle UI code and data access code
  - Do this leads to unmaintainable spaghetti code
  - Data access code should be in separate JavaScript library file
  - Best practice is to use the JavaScript revealing module pattern

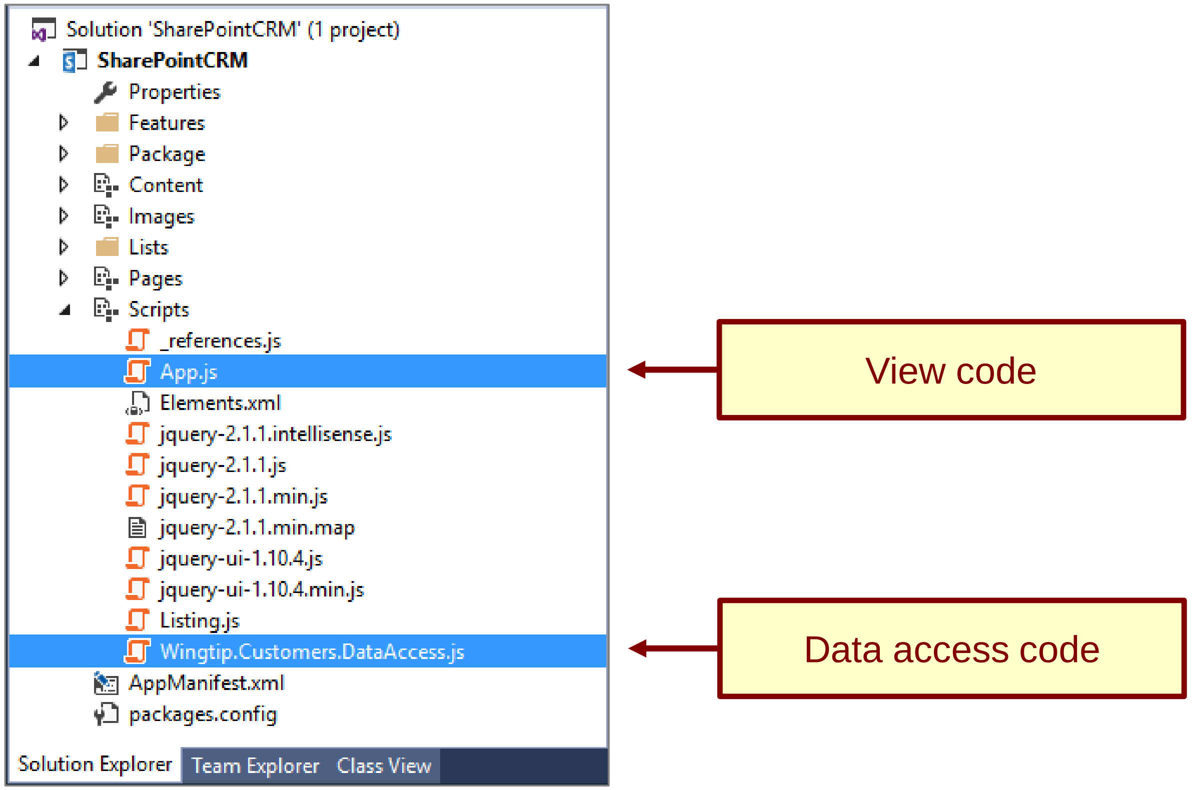


Wingtip.Customers.DataAccess.js

```
var Wingtip = window.Wingtip || {};

Wingtip.Customers = Wingtip.Customers || {};

Wingtip.Customers.DataAccess = function () {

  var getCustomers = function ()...;

  var getCustomer = function (Id)...

  var addCustomer = function (FirstName, LastName, Company, WorkPhone, HomePhone, Email)...;

  var updateCustomer = function (Id, FirstName, LastName, Company, WorkPhone, HomePhone, Email, ETag)...

  var deleteCustomer = function (Id)...;

  // return object with public interface for revealing module pattern
  return {
    getCustomers: getCustomers,
    addCustomer: addCustomer,
    getCustomer: getCustomer,
    updateCustomer: updateCustomer,
    deleteCustomer: deleteCustomer
  };

}();
```

Creating A Reusable Data Access Library

- Data access code is kept separate from view (i.e. UI code)
  - Data access code returns promise to view code
  - View code encapsulated from details of data access
  - View code responsible for updating user interface when required



Solution 'SharePointCRM' (1 project)
SharePointCRM
Properties
Features
Package
Content
Images
Lists
Pages
Scripts
_references.js
App.js
Elements.xml
jquery-2.1.1.intellisense.js
jquery-2.1.1.js
jquery-2.1.1.min.js
jquery-2.1.1.min.map
jquery-ui-1.10.4.js
jquery-ui-1.10.4.min.js
Listing.js
Wingtip.Customers.DataAccess.js
AppManifest.xml
packages.config

Service Root URI for the App Web

- Creating the App Web's Service Root URI
  - Use URL relative to **Pages** folder

```
var restURI = "../_api/web/?$select=Id,Title,Url"
```

  - Use URL created from **SPAppWebUrl** query string parameter

```
var restURI = getQueryStringParameter("SPAppWebUrl") +
              "/_api/web/?$select=Id,Title,Url"
```

  - Use URL created from **_spPageContextInfo.webAbsoluteUrl**

```
var restURI = _spPageContextInfo.webAbsoluteUrl +
              "/_api/web/?$select=Id,Title,Url"
```

Querying a List in the App Web

- Create reusable data access function

```
var getCustomers = function () {

  // create URI for request
  var requestUri = "../_api/web/lists/getByTitle('Customers')/items" +
                   "?$select=ID,FirstName,Title,Company,WorkPhone,HomePhone,Email" +
                   "&$orderby=Title,FirstName";

  // send call across network
  return $.ajax({
    url: requestUri,
    type: "GET",
    headers: { "accept": "application/json" }
  });

};
```

- Call the reusable data access function from the add-in's UI code

```
Wingtip.Customers.DataAccess.getCustomers().then(function (data) {
  // get OData result from data.value
  var customers = data.value;
  // create HTML table using OData result
  var table = $("<table>", { ID: "customersTable" });
  // add table header row
  table.append($("<thead>")
          .append($("<td>").text("First Name"))
          .append($("<td>").text("Last Name")));
      // add table data rows
  for (var customer = 1; customer < customers.length; customer++) {
    table.append($("<tr>")
          .append($("<td>").text(customers[customer].FirstName))
          .append($("<td>").text(customers[customer].Title)));
  }
  // append HTML table to div on page
  $("#content_box").append(table);
});
```

Service Root URI for the Host Web

- This code will fail because it attempts a cross-domain call

```
var restURI = getQueryStringParameter("SPHostUrl") +
              "/_api/web/?$select=Id,Title,Url";
```

- This code works in some but not all scenarios

```
var restURI = "/_api/web/?$select=Id,Title,Url";
```

```
var restURI = "../../_api/web/?$select=Id,Title,Url"
```

- This code works in all scenarios and this is what you should use

```
var restURI = "../_api/SP.AppContextSite(@target)/web/" +
              "?$select=Id,Title,Url" +
              "&@target='" + getQueryStringParameter("SPHostUrl") + "'";
```

Querying for Lists within the Host Web

- Use SP.AppContextSite in URI to access host web from app web
  - Call gets routed through app web so there is no cross-domain call
  - SP.AppContextSite allows you to program against site in host web domain

```
var getLists = function () {

  var requestUri = "../_api/SP.AppContextSite(@target)/web/lists/" +
                   "?$filter=(Hidden eq false) and (BaseType eq 0)" +
                   "&$select=Id,Title, ItemCount, EntityTypeName, ListItemEntityTypeFullName" +
                   "&@target='" + hostWebUrl + "'";

  return $.ajax({
    url: requestUri,
    type: "GET",
    headers: { "accept": "application/json;odata=verbose" }
  });

};
```

Updating SharePoint Objects

- All write operations must pass valid request digest value
- You must include type metadata for inserts & updates
- Sometimes you must pass ETags for updates & deletes

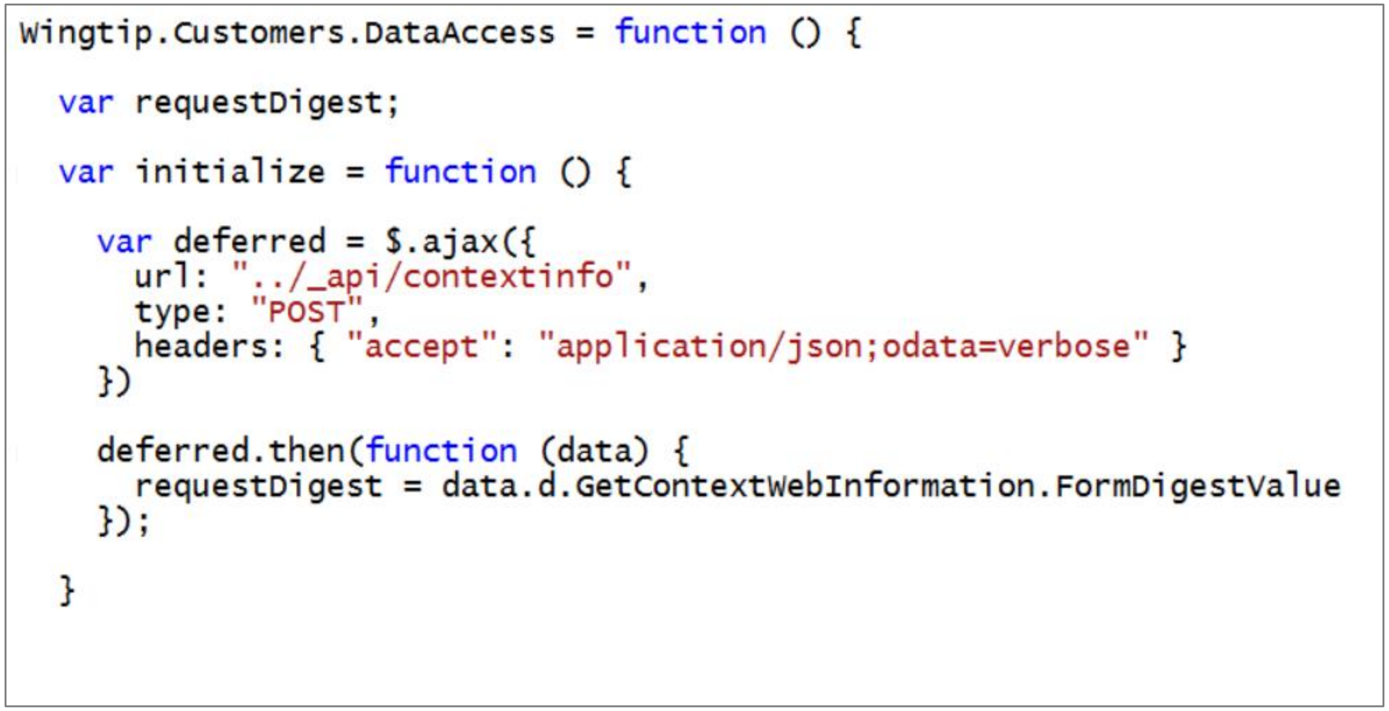
Understanding the Request Digest

- All SharePoint write operations require Request Digest
  - Provides security mechanism to protect again replay attacks
  - Request digest known to SharePoint old timers as “Form Digest”
  - SharePoint adds request digest element with ID __REQUESTDIGEST
  - Request digest value passed using X-RequestDigest header

```
var requestHeaders = {
  "accept": "application/json;odata=verbose",
  "X-RequestDigest": $("#__REQUESTDIGEST").val()
}
```

Caching the Request Digest

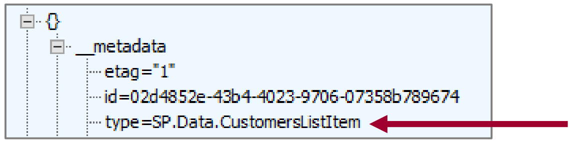
- Request digest queried using **/_api/contextinfo**



```
Wingtip.Customers.DataAccess = function () {

  var requestDigest;

  var initialize = function () {

    var deferred = $.ajax({
      url: "../_api/contextinfo",
      type: "POST",
      headers: { "accept": "application/json;odata=verbose" }
    })

    deferred.then(function (data) {
      requestDigest = data.d.GetContextWebInformation.FormDigestValue
    });

  }
```

Working with List Item Type Metadata

- Each SharePoint list has a unique type for its list items



```
{}
  __metadata
    etag="1"
    id=02d4852e-43b4-4023-9706-07358b789674
    type=SP.Data.CustomersListItem
```

- Verbose syntax requires **type** value be passed with inserts & updates
  - Type value can be omitted with non-verbose syntax (content-type=application/json)

```
var customerData = {
  __metadata: { "type": "SP.Data.CustomersListItem" },
  Title: LastName,
  FirstName: FirstName,
  Company: Company,
  WorkPhone: WorkPhone,
  HomePhone: HomePhone,
  Email: Email
};
```

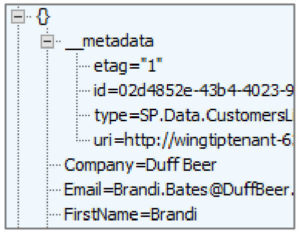
- **type** discoverable using **ListItemEntityTypeFullName** property

Adding a SharePoint List Item

```
var addCustomer = function (FirstName, LastName, Company, WorkPhone, HomePhone, Email) {

  var requestUri = "../_api/web/lists/getByTitle('Customers')/items";

  var requestHeaders = {
    "accept": "application/json;odata=verbose",
    "X-RequestDigest": $("#__REQUESTDIGEST").val()
  }

  var customerData = {
    __metadata: { "type": "SP.Data.CustomersListItem" },
    Title: LastName,
    FirstName: FirstName,
    Company: Company,
    WorkPhone: WorkPhone,
    HomePhone: HomePhone,
    Email: Email
  };

  var requestBody = JSON.stringify(customerData);

  return $.ajax({
    url: requestUri,
    type: "POST",
    contentType: "application/json;odata=verbose",
    headers: requestHeaders,
    data: requestBody,
  });

};
```

ETags and Optimistic Concurrency

- OData v2 requires items to carry ETags
  - ETag is integer value in that it identities version of item
  - ETag is automatically incremented with each update



```
{}
  __metadata
    etag="1"
    id=02d4852e-43b4-4023-9
    type=SP.Data.CustomersL
    uri=http://wingtiptenant-6
  Company=Duff Beer
  Email=Brandi.Bates@DuffBeer.
  FirstName=Brandi
```

- ETag use to support for optimistic concurrency control
  - ETag works to eliminate the “lost update” scenario
  - ETag must be tracked in order to post updates in most scenarios

```
// store item metadata values into hidden controls
$("#customer_id").val(data.d.ID);
$("#etag").val(data.d.__metadata.etag);
```

ETags and the If-Match Header

- Update and Delete operations require If-Match Header
  - Allows you to pass ETag value during an update
  - Update fails if ETag value changed due to update by other user

```
var requestHeaders = {
  "accept": "application/json;odata=verbose",
  "X-HTTP-Method": "MERGE",
  "X-RequestDigest": $("#__REQUESTDIGEST").val(),
  "If-Match": ETag
}
```

- You can pass wildcard (*) value inside If-Match Header
  - Done to disable optimistic concurrency control
  - This is commonly done with delete operations

```
var requestHeaders = {
  "accept": "application/json;odata=verbose",
  "X-RequestDigest": $("#__REQUESTDIGEST").val(),
  "If-Match": "*"
}
```

Updating a SharePoint List Item

```
var updateCustomer = function (Id, FirstName, LastName, Company, WorkPhone, HomePhone, Email, ETag) {

  var requestUri = "../_api/web/lists/getByTitle('Customers')/items(" + Id + ")";

  var requestHeaders = {
    "accept": "application/json;odata=verbose",
    "X-HTTP-Method": "MERGE",
    "X-RequestDigest": $("#__REQUESTDIGEST").val(),
    "If-Match": ETag
  }

  var customerData = {
    __metadata: { "type": "SP.Data.CustomersListItem" },
    Title: LastName,
    FirstName: FirstName,
    Company: Company,
    WorkPhone: WorkPhone,
    HomePhone: HomePhone,
    Email: Email
  };

  var requestBody = JSON.stringify(customerData);

  return $.ajax({
    url: requestUri,
    type: "POST",
    contentType: "application/json;odata=verbose",
    headers: requestHeaders,
    data: requestBody,
  });

};
```

Deleting a SharePoint List Item

```
var deleteCustomer = function (Id) {

  var requestUri = "../_api/web/lists/getByTitle('Customers')/items(" + Id + ")";

  var requestHeaders = {
    "accept": "application/json;odata=verbose",
    "X-RequestDigest": $("#__REQUESTDIGEST").val(),
    "If-Match": "*"
  }

  return $.ajax({
    url: requestUri,
    type: "DELETE",
    headers: requestHeaders,
  });

};
```

DEMO

Exploring the SharePointCRM App

Agenda

- ✓ SharePoint Add-in Model Overview
- ✓ SharePoint-hosted Add-in
- ✓ Programming the SharePoint REST API
- ➢ SharePoint Add-in Security
- Provider-hosted Add-ins
- Acquiring and Managing Access Tokens

Authentication and Authorization

- Authentication creates identity for principal
  - SharePoint 2010 only supports user authentication
  - SharePoint 2013 adds support to authenticate apps
  - SharePoint apps are given first class identities

- Authorization provides the access control
  - Used to verify an principal has the proper permission
  - SharePoint 2010 only supports user permissions
  - SharePoint 2013 adds support for app permissions

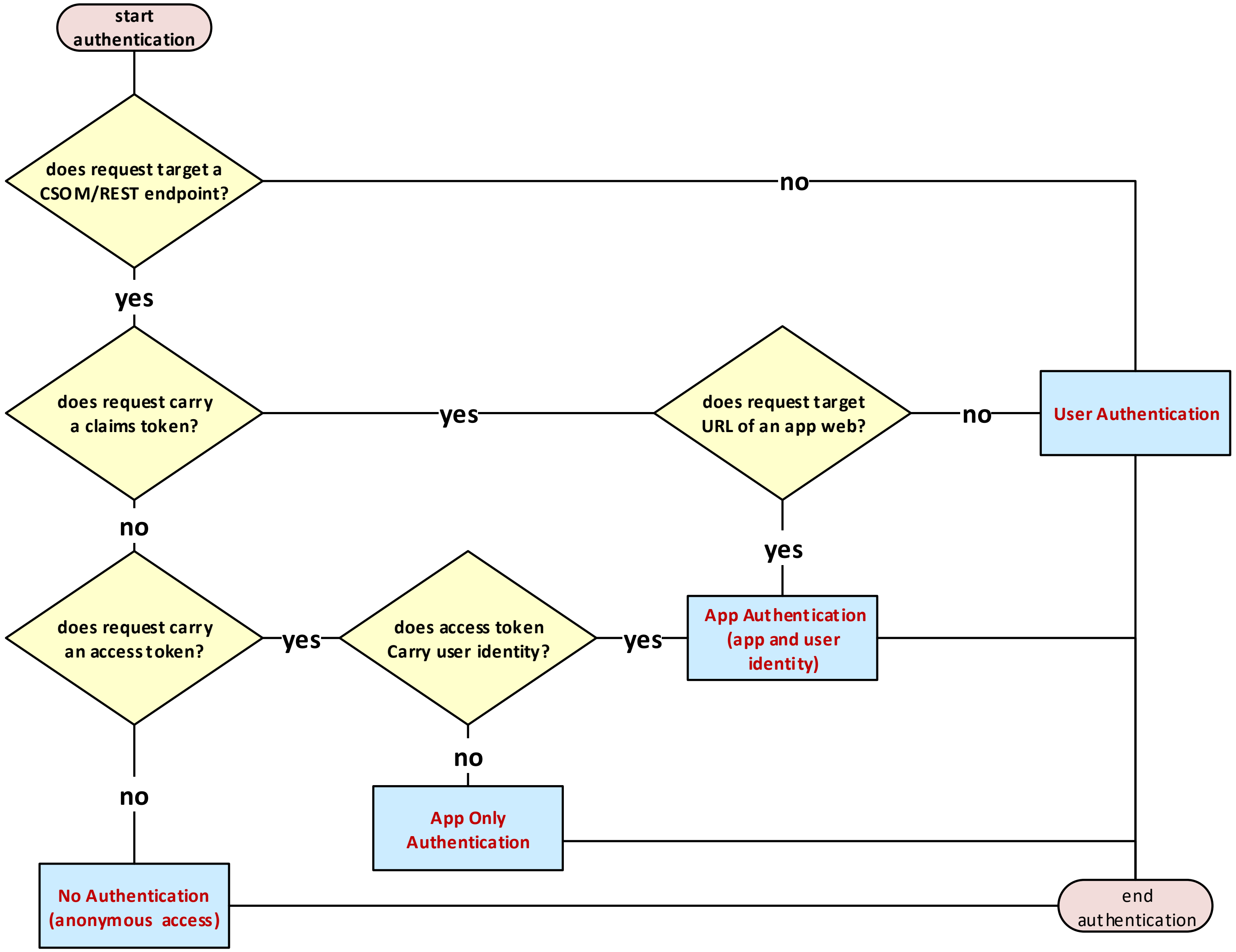
Internal Authentication

- Internal authentication is used if the following are true
  - Incoming call targets a CSOM or REST API endpoint
  - Incoming call carries claims token with established user identity
  - Incoming call targets URL of an exiting app web
- Important points about using internal authentication
  - It just works – no need to program in terms of access tokens
  - It's always used with client-side calls from pages in the app web
  - It can be used from remote web pages using cross domain library
  - It does not support app-only authentication to elevate privledge

External Authentication

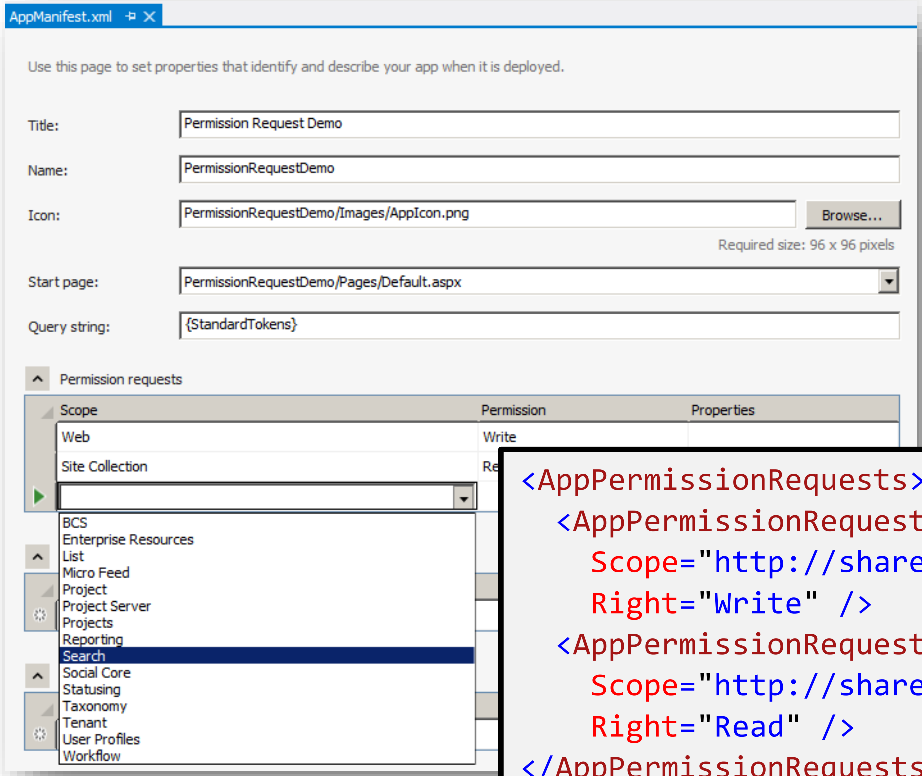
- In which scenarios does external authentication occur?
  - When server-side code in the remote web issues CSOM or REST API calls against the SharePoint host
  - Incoming calls free to target host web and other sites in tenancy
- How does it work?
  - App code must written to create and manage access tokens
  - Access token carries app identity
  - Access token can (and usually does) carry user identity as well
  - App must transmit access token when calling to SharePoint

SharePoint 2013 Authentication Flow


start authentication
does request target a CSOM/REST endpoint?
no
yes
does request carry a claims token?
yes
does request target URL of an app web?
no
User Authentication
no
yes
does request carry an access token?
yes
does access token Carry user identity?
yes
App Authentication (app and user identity)
no
no
App Only Authentication
No Authentication (anonymous access)
end authentication

Adding Permission Requests

- Permissions requests are added to app manifest
  - App manifest designer makes this relatively easy


AppManifest.xml

Use this page to set properties that identify and describe your app when it is deployed.

Title:	Permission Request Demo
Name:	PermissionRequestDemo
Icon:	PermissionRequestDemo/Images/AppIcon.png
Start page:	PermissionRequestDemo/Pages/Default.aspx
Query string:	{StandardTokens}

Required size: 96 x 96 pixels

Permission requests

Scope	Permission	Properties
Web	Write	
Site Collection	Re	

```
<AppPermissionRequests>
  <AppPermissionRequest
    Scope="http://sharepoint/content/sitecollection/web"
    Right="Write" />
  <AppPermissionRequest
    Scope="http://sharepoint/content/sitecollection"
    Right="Read" />
</AppPermissionRequests>
```

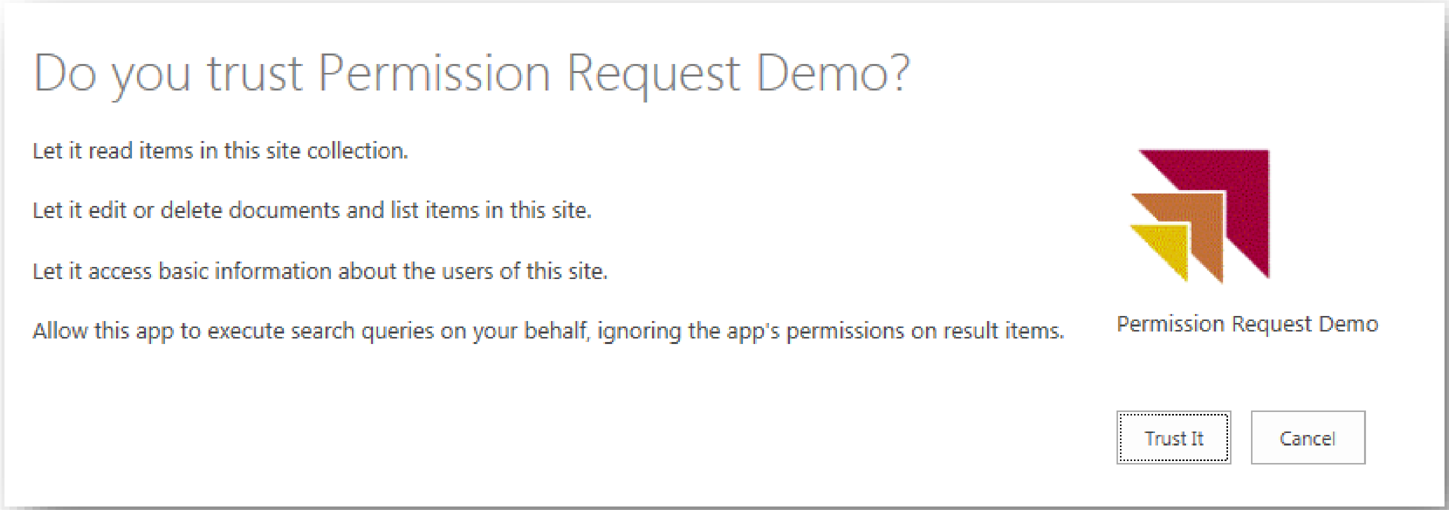
App-Only Permissions

- Used for two key scenarios
  - To call into SharePoint with permissions greater than the current user (elevation)
  - To call in to SharePoint when there is no current user
- Steps to accomplish this
  - Add AllowAppOnlyPolicy to AppManifest.xml
  - Write code to acquire an app only access token

```
<AppPermissionRequests AllowAppOnlyPolicy="true">
  <AppPermissionRequest Scope="http://sharepoint/content/sitecollection" Right="Read" />
  <AppPermissionRequest Scope="http://sharepoint/content/sitecollection/web/list" Right="FullControl" />
  <AppPermissionRequest Scope="http://sharepoint/search" Right="QueryAsUserIgnoreAppPrincipal" />
</AppPermissionRequests>
```

Granting Consent in SharePoint 2016

- User prompted to trust the app during installation
  - Trust It grants requested permissions to app
  - Cancel prevents app from being installed



Do you trust Permission Request Demo?

Let it read items in this site collection.

Let it edit or delete documents and list items in this site.

Let it access basic information about the users of this site.

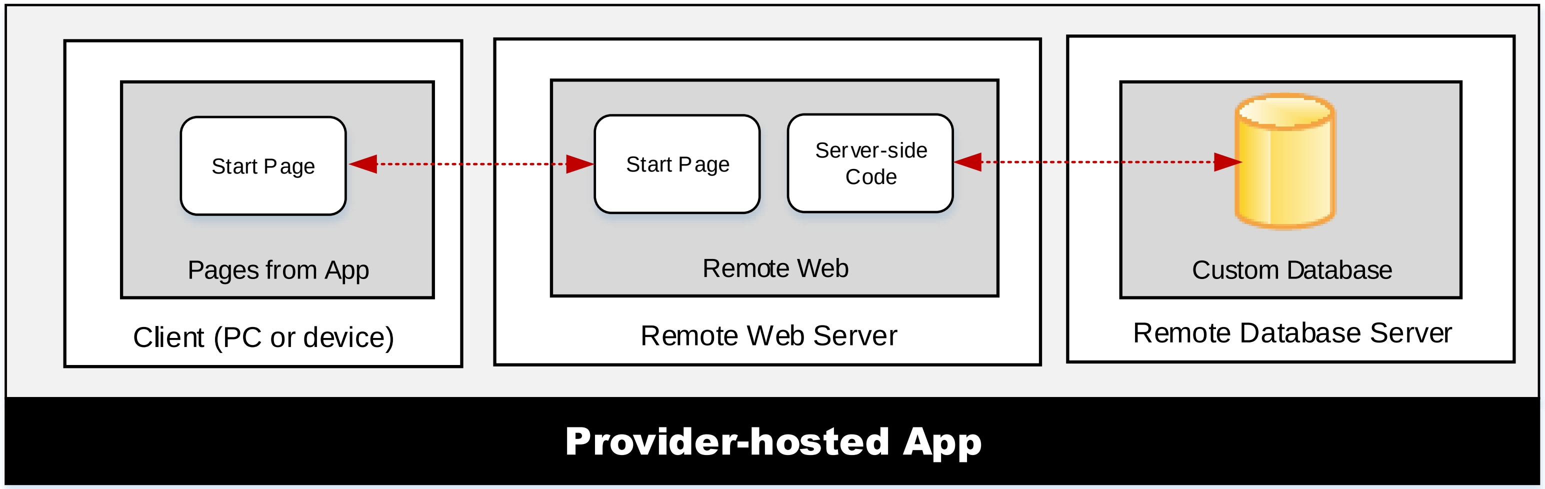
Allow this app to execute search queries on your behalf, ignoring the app's permissions on result items.

Agenda

- ✓ SharePoint Add-in Model Overview
- ✓ SharePoint-hosted Add-in
- ✓ SharePoint Add-in Security
- ➢ Provider-hosted Add-ins
- • Acquiring and Managing Access Tokens

Provider-Hosted App

- Developer responsible for deploying remote web
  - App deployed to remote web on remote web server
  - Developer deploys remote web prior to app installation
  - Developer often required to deploy database as well



Pros and Cons of Provider-hosted Apps

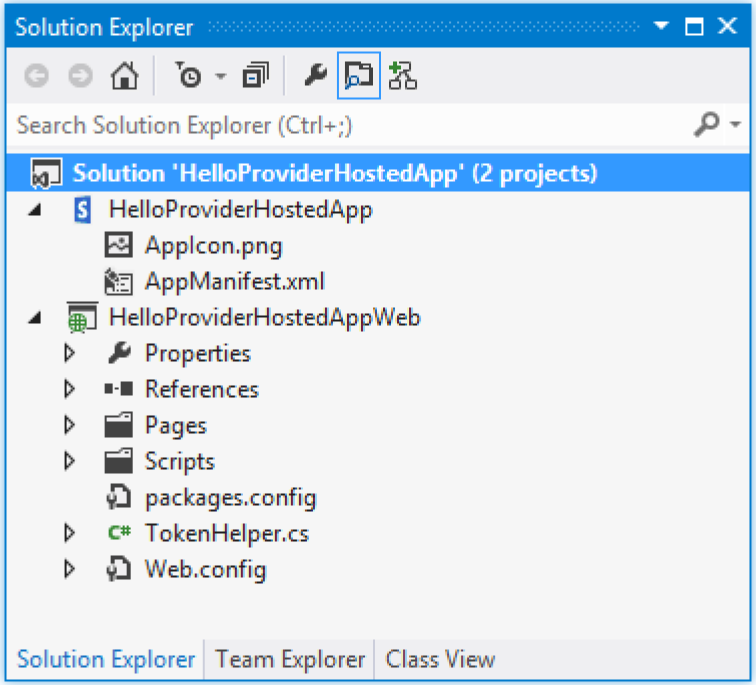
- Benefits of provider-hosted over SharePoint-hosted apps
  - You can write server-side .NET code using C# or VB.NET **([wahoo!])**
  - You can make HTTP calls w/o cross-domain scripting constraints
  - Your server-side code can access data in a custom database
  - You can leverage the support for remote event receivers
  - You can make CSOM/REST API calls using App-only permissions
  - Provider-hosted apps more strategic for Microsoft moving forward

- Negatives when compared to SharePoint-hosted apps
  - You must deploy and manage the remote web
  - Requires extra code to acquire and manage security tokens
  - Multi-tenant aware app design can introduce significant complexity

Provider-hosted App Projects

- Visual Studio create solution with two projects
  - SharePoint app project
  - ASP.NET Website project for remote web

    this project is known as the "web project"



```
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'HelloProviderHostedApp' (2 projects)
  HelloProviderHostedApp
    AppIcon.png
    AppManifest.xml
  HelloProviderHostedAppWeb
    Properties
    References
    Pages
    Scripts
    packages.config
    TokenHelper.cs
    Web.config
Solution Explorer | Team Explorer | Class View
```

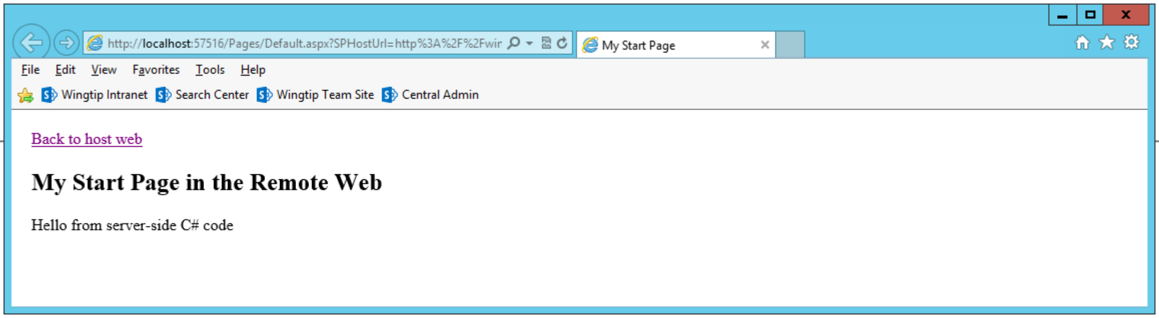
AppManifest.xml

- Provider-hosted app adds requirements to App Manifest
  - StartPage must point to page in remote web
  - AppPrincipal requires app authentication settings
  - External app authentication can be disabled using Internal setting

```
<App xmlns="http://schemas.microsoft.com/sharepoint/2012/app/manifest"
     Name="HelloProvider-HostedApp"
     ProductID="{8d587998-fdbf-4d97-a739-613a647bed83}"
     Version="1.0.0.0"
     SharePointMinVersion="15.0.0.0" >

  <Properties>
    <Title>Hello Provider-Hosted App</Title>
    <StartPage>~remoteAppUrl/Pages/Default.aspx?{StandardTokens}</StartPage>
  </Properties>

  <AppPrincipal>
    <!-- turn off external app authentication -->
    <Internal />
  </AppPrincipal>

</App>
```

A Sample Start Page

```
<%@ Page Language="C#" AutoEventWireup="true"
         CodeBehind="Default.aspx.cs" Inherits="HelloProviderHostedAppWeb.Pages.Default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">
    <title>My Start Page</title>
</head>

<body>
    <form id="form1" runat="server">
    <div>

      <!-- add HyperLink control to link back to host web-->
      <div><asp:HyperLink ID="linkHostWeb" runat="server">Back to host web</asp:HyperLink></div>

      <!-- add some HTML content to page -->
      <h2>My Start Page in the Remote Web</h2>

      <!-- -->
      <asp:PlaceHolder ID="PlaceHolderMain" runat="server"></asp:PlaceHolder>

    </div>
    </form>
</body>
</html>
```


http://localhost:57516/Pages/Default.aspx?SPHostUrl=http%3A%2F%2Fwir
My Start Page
File Edit View Favorites Tools Help
Wingtip Intranet
Search Center
Wingtip Team Site
Central Admin

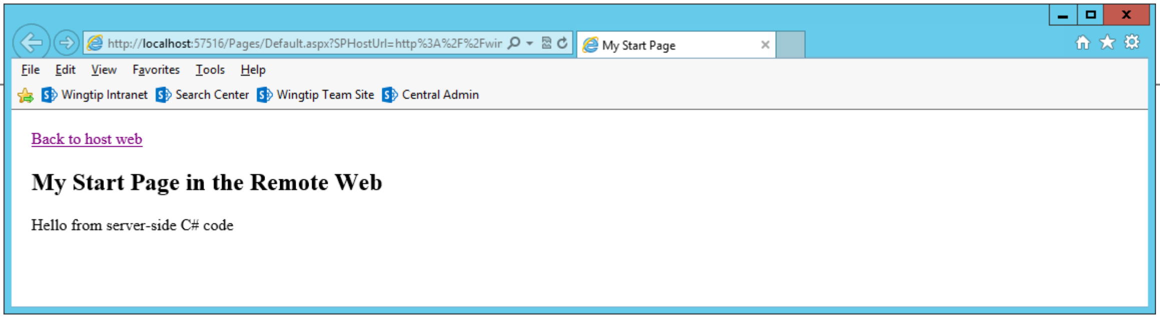
Back to host web

My Start Page in the Remote Web

Hello from server-side C# code

C# Code Behind Sample Start Page

```
namespace HelloProviderHostedAppWeb.Pages {
  public partial class Default : System.Web.UI.Page {

    protected void Page_Load(object sender, EventArgs e) {

      // delete all existing code added by Visual Studio - it requires authentication

      // Configure ASP.NET Hyperlink control with value from SPHostUrl querystring
      linkHostWeb.NavigateUrl = Request.QueryString["SPHostUrl"];

      // add some content to the page using server-side code
      PlaceHolderMain.Controls.Add( new LiteralControl("Hello from server-side C# code"));

    }
  }
}
```


http://localhost:57516/Pages/Default.aspx?SPHostUrl=http%3A%2F%2Fwir
My Start Page
File Edit View Favorites Tools Help
Wingtip Intranet Search Center Wingtip Team Site Central Admin

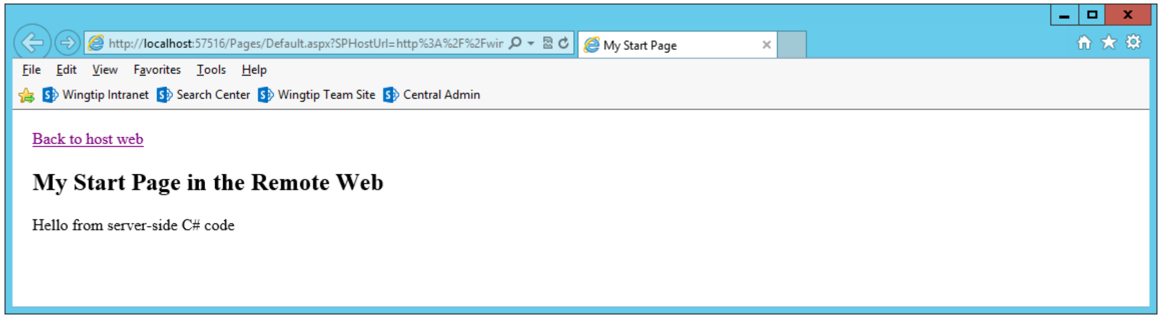
Back to host web

My Start Page in the Remote Web

Hello from server-side C# code

Debugging the Remote Web in IIS Express

- Visual Studio debugging involves IIS Express
  - URL created in **localhost** domain (e.g. **https://localhost:57516**)
  - Port number for Remote Web project selected automatically behind scenes



http://localhost:57516/Pages/Default.aspx?SPHostUrl=http%3A%2F%2Fwir

My Start Page

File Edit View Favorites Tools Help

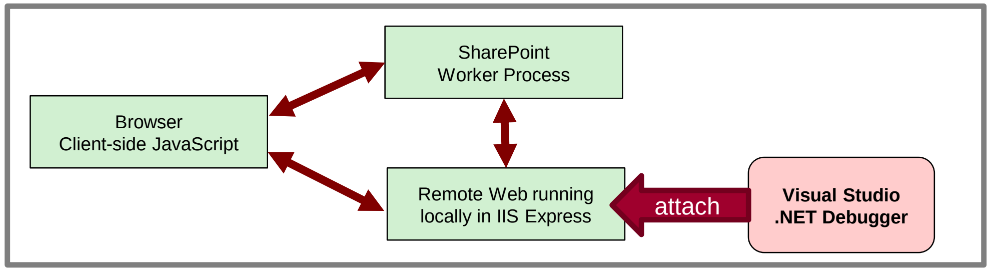
Wingtip Intranet Search Center Wingtip Team Site Central Admin

Back to host web

My Start Page in the Remote Web

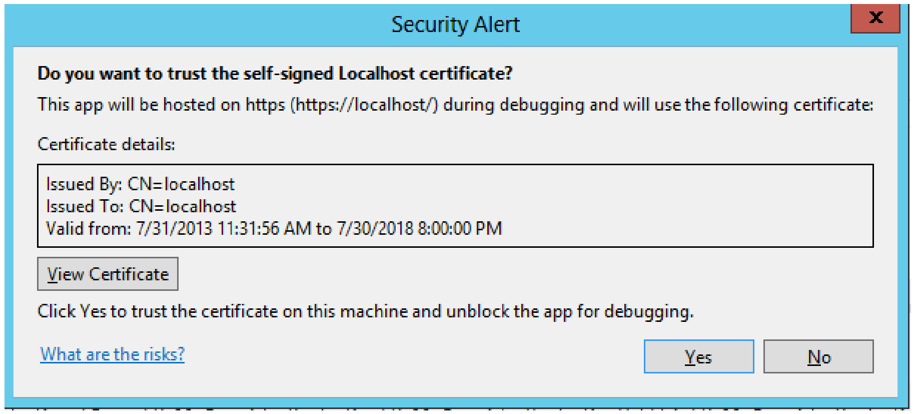
Hello from server-side C# code

- IIS Express debugging support allows you to debug…
  - Server-side C# code behind pages in the remote web
  - Web service calls from browser to remote web
  - Web service calls from SharePoint to remote web


SharePoint Worker Process
Browser Client-side JavaScript
Remote Web running locally in IIS Express
attach
Visual Studio .NET Debugger

Debugging Code in the Remote Web

- Remote Web can optionally use SSL
  - Pages can be served using HTTPS or HTTP
  - Use of SSL is usually preferred
  - IIS Express can configure SSL through https://localhost
  - Visual Studio registers self-signed certificate on first use


Security Alert

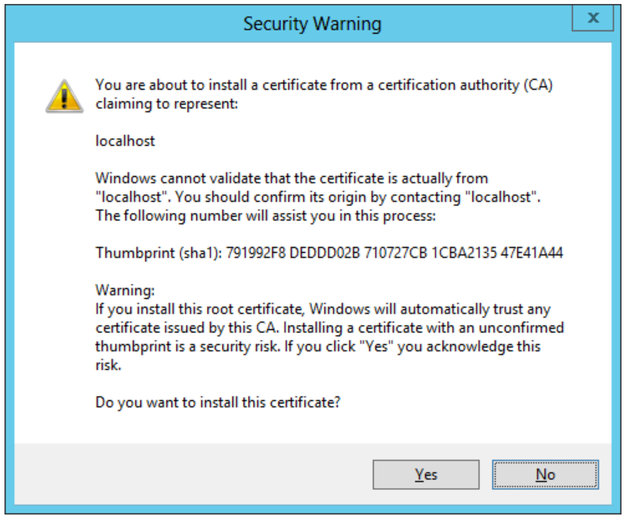
Do you want to trust the self-signed Localhost certificate?

This app will be hosted on https (https://localhost/) during debugging and will use the following certificate:

Certificate details:

Issued By: CN=localhost
Issued To: CN=localhost
Valid from: 7/31/2013 11:31:56 AM to 7/30/2018 8:00:00 PM

Click Yes to trust the certificate on this machine and unblock the app for debugging.


Security Warning

You are about to install a certificate from a certification authority (CA) claiming to represent:

localhost

Windows cannot validate that the certificate is actually from "localhost". You should confirm its origin by contacting "localhost". The following number will assist you in this process:

Thumbprint (sha1): 791992F8 DEDDD02B 710727CB 1CBA2135 47E41A44

Warning:
If you install this root certificate, Windows will automatically trust any certificate issued by this CA. Installing a certificate with an unconfirmed thumbprint is a security risk. If you click "Yes" you acknowledge this risk.

Do you want to install this certificate?

Web Forms Versus MVC

- ASP.NET provides two different platforms
  - ASP.NET Web Forms (e.g. ASPX files)
  - ASP.NET MVC
- MVC provides better platform for the web
  - More flexible routing
  - Lighter-weight
  - Richer templating
  - Better C# integration
  - Unit testing

Agenda

- ✓ SharePoint Add-in Model Overview
- ✓ SharePoint-hosted Add-in
- ✓ SharePoint Add-in Security
- ✓ Provider-hosted Add-ins
- ➢ Acquiring and Managing Access Tokens

App Principals

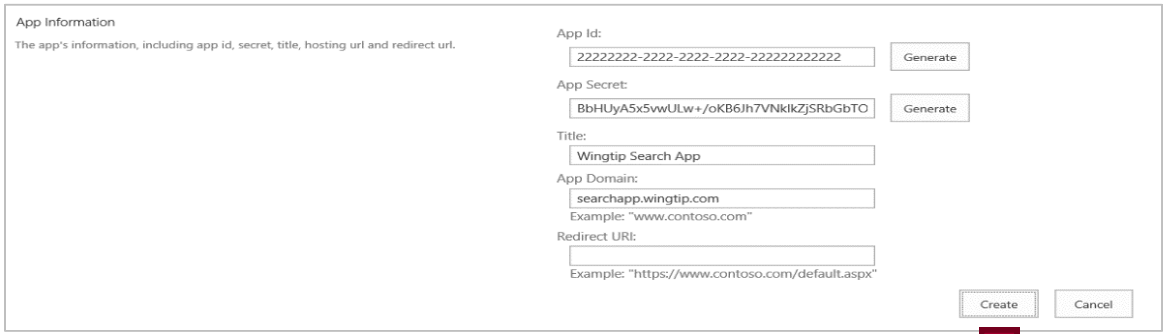
- External authentication requires app principals
  - App principal is a tenancy-scoped account for app identity
  - App principal identified using a GUID
  - App principals must be created in SharePoint host
- App principal properties
  - Client ID: GUID-based identifier for app principal
  - Client Secret: (not used in S2S)
  - App Host Domain: Base URL of remote web
  - Redirect URL: URL to a page used to configure on-the-fly security

Managing App Principals in SharePoint Online

- Get to know the built-in app management pages
  - AppRegNew.aspx
  - AppInv.com
  - AppPrincipals.aspx
- There is also management support using PowerShell
  - Use PowerShell cmdlets to administer SharePoint apps and app principals

Registering an App Security Principal

- Done automatically by Visual Studio during development
  - When you press {F5}, VS automatically registers app principal
  - Visual Studio also updates web.config file
- Can also be done using AppRegNew.aspx page
  - App deployment covered in more detail in App Publishing module



App Information

The app's information, including app id, secret, title, hosting url and redirect url.

App Id: 22222222-2222-2222-2222-222222222222

App Secret: BbHUyA5x5vwULw+/oKB6Jh7VNkIkZjSRbGbTO

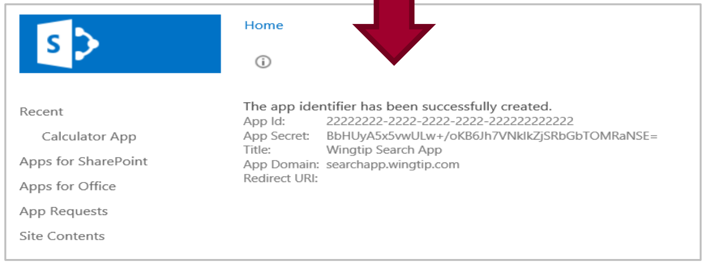
Title: Wingtip Search App

App Domain: searchapp.wingtip.com

Example: "www.contoso.com"

Redirect URI:

Example: "https://www.contoso.com/default.aspx"



The app identifier has been successfully created.

App Id: 22222222-2222-2222-2222-222222222222

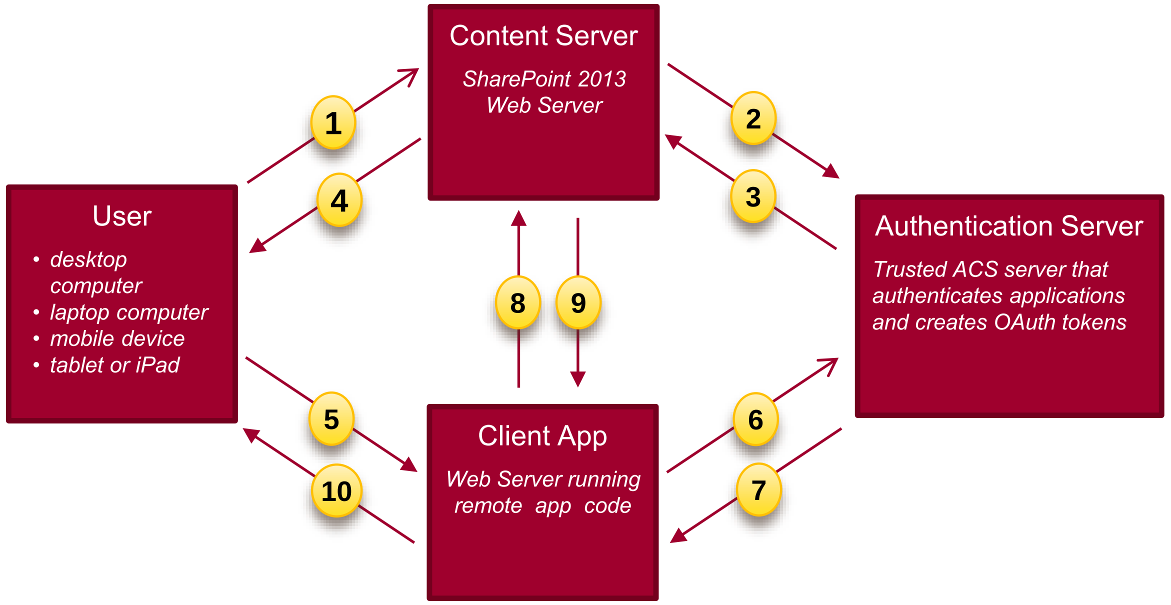
App Secret: BbHUyA5x5vwULw+/oKB6Jh7VNkIkZjSRbGbTOMRaNSE=

Title: Wingtip Search App

App Domain: searchapp.wingtip.com

Redirect URI:

OAuth Protocol Flow in SharePoint 2013


Content Server
SharePoint 2013 Web Server
User
• desktop computer
• laptop computer
• mobile device
• tablet or iPad
Authentication Server
Trusted ACS server that authenticates applications and creates OAuth tokens
Client App
Web Server running remote app code
1
2
3
4
5
6
7
8
9
10

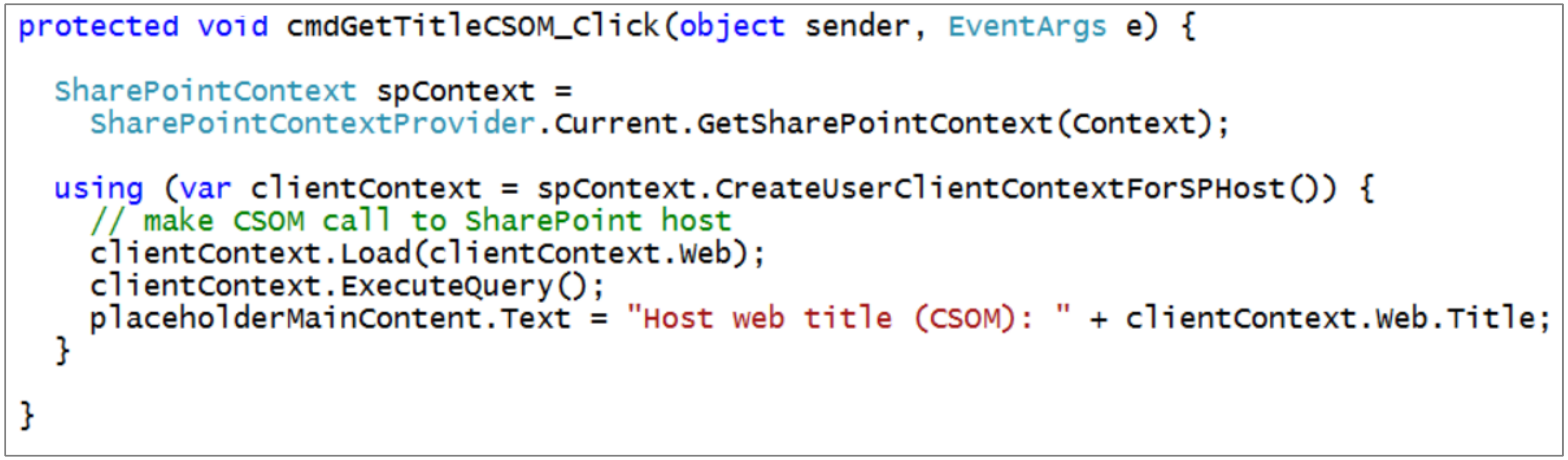
1. SharePoint authenticates user using claims
2. SharePoint requests context token for user
3. ACS returns context token
4. SharePoint pass context token to User
5. User POSTS to app passing context token
6. Client app is able to pull refresh token out of context token. Client app then passes refresh token to ACS to request OAuth token
7. ACS returns OAuth token to client app
8. Client App makes CSOM/REST calls to SharePoint site passing OAuth token
9. SharePoint returns site content to app
10. Client App returns HTML to user device

Security Tokens used with Azure ACS

- Context Token
  - Contextual information passed to app
- Refresh Token
  - Used by client app to acquire an access token
- Access Token
  - Token passed to SharePoint to app when using external authentication

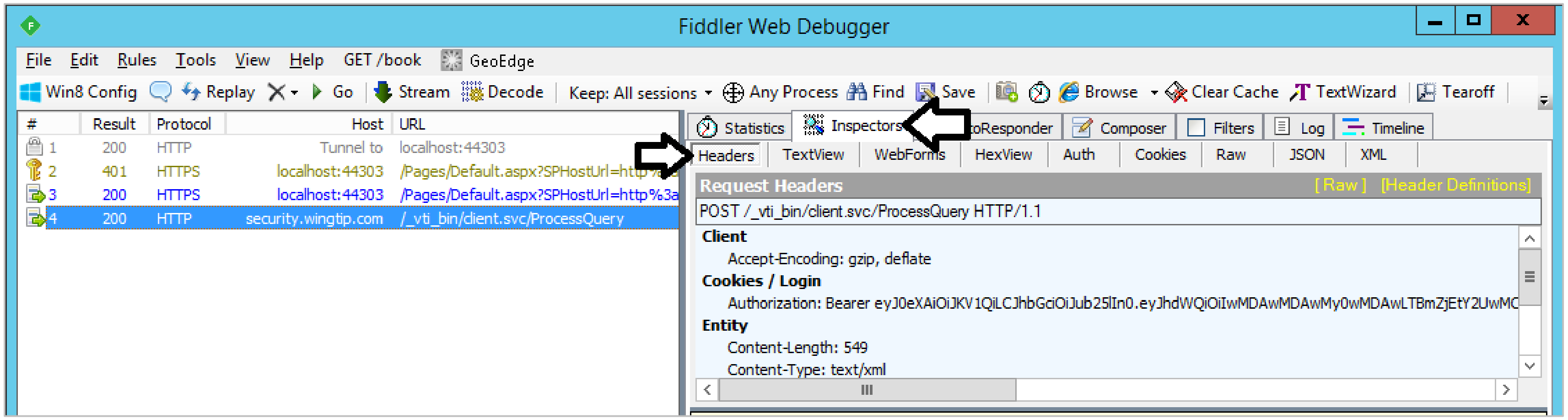
CSOM Calls using S2S Authentication

- **TokenHelper** class has methods specific to S2S
- **SharePointContext** has methods that are not S2S-specific



```
protected void cmdGetTitleCSOM_Click(object sender, EventArgs e) {

  SharePointContext spContext =
    SharePointContextProvider.Current.GetSharePointContext(Context);

  using (var clientContext = spContext.CreateUserClientContextForSPHost()) {
    // make CSOM call to SharePoint host
    clientContext.Load(clientContext.Web);
    clientContext.ExecuteQuery();
    placeholderMainContent.Text = "Host web title (CSOM): " + clientContext.Web.Title;
  }

}
```

Examining CSOM Calls using Fiddler



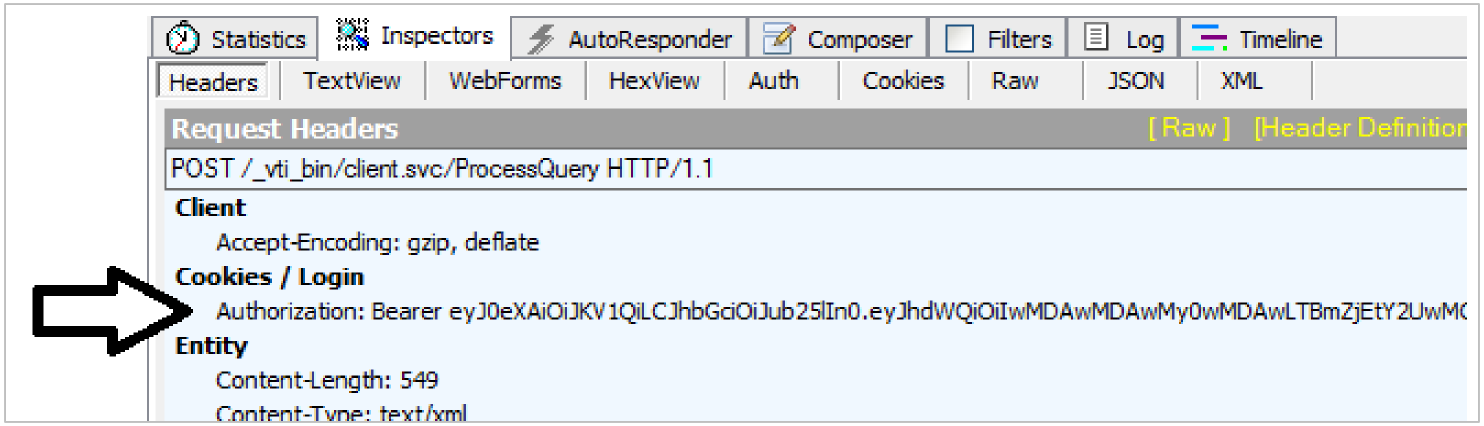
Fiddler Web Debugger
File Edit Rules Tools View Help GET /book GeoEdge
Win8 Config Replay Go Stream Decode Keep: All sessions Any Process Find Save Browse Clear Cache TextWizard Tearoff
Statistics Inspectors AutoResponder Composer Filters Log Timeline
Headers TextView WebForms HexView Auth Cookies Raw JSON XML
[Raw] [Header Definitions]

#	Result	Protocol	Host	URL
1	200	HTTP	Tunnel to	localhost:44303
2	401	HTTPS	localhost:44303	/Pages/Default.aspx?SPHostUrl=http%3a
3	200	HTTPS	localhost:44303	/Pages/Default.aspx?SPHostUrl=http%3a
4	200	HTTP	security.wingtip.com	/_vti_bin/client.svc/ProcessQuery

Request Headers

POST /_vti_bin/client.svc/ProcessQuery HTTP/1.1

Client
Accept-Encoding: gzip, deflate
Cookies / Login
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJub25lIn0.eyJhdWQiOiIwMDAwMDAwMy0wMDAwLTBmZjEtY2UwMC
Entity
Content-Length: 549
Content-Type: text/xml



Statistics Inspectors AutoResponder Composer Filters Log Timeline
Headers TextView WebForms HexView Auth Cookies Raw JSON XML
[Raw] [Header Definition

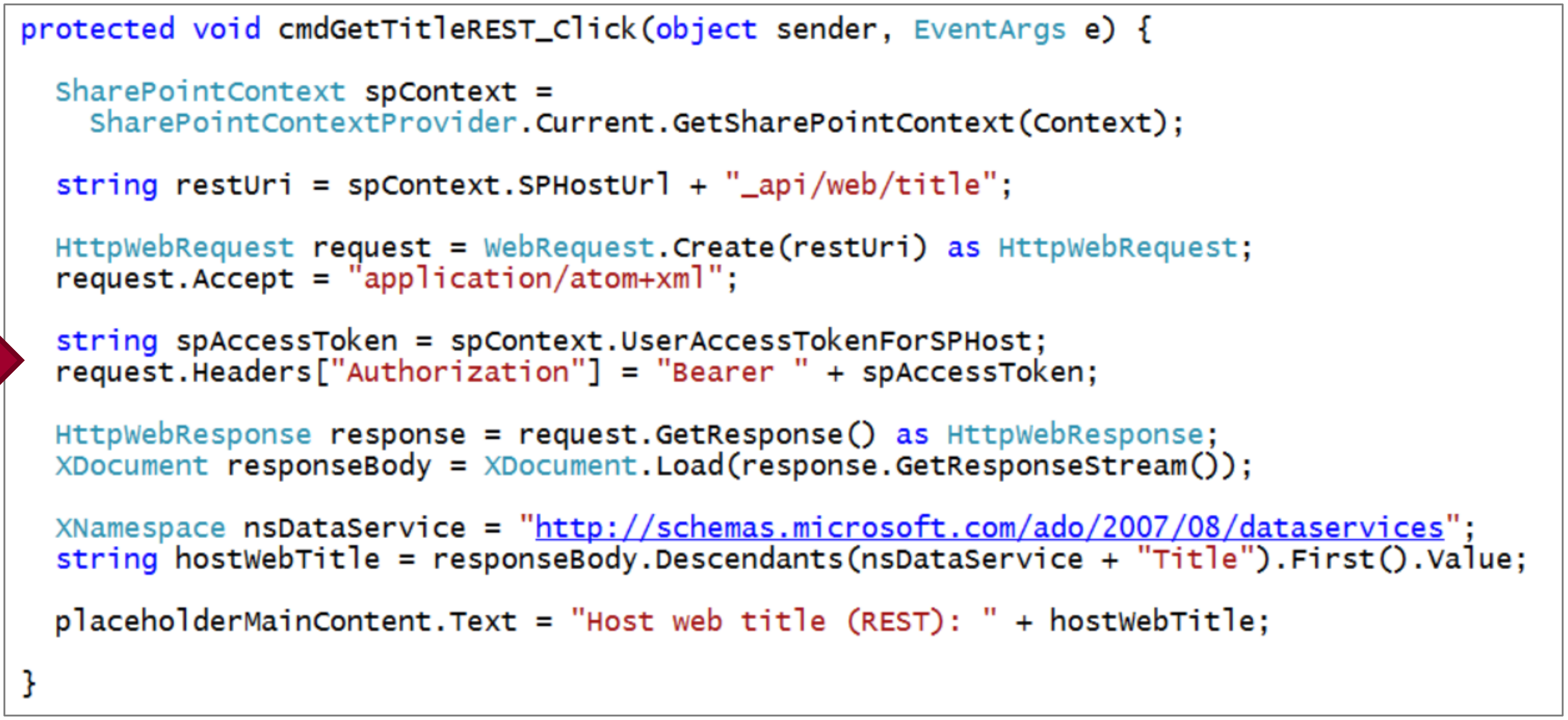
Request Headers

POST /_vti_bin/client.svc/ProcessQuery HTTP/1.1

Client
Accept-Encoding: gzip, deflate
Cookies / Login
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJub25lIn0.eyJhdWQiOiIwMDAwMDAwMy0wMDAwLTBmZjEtY2UwMC
Entity
Content-Length: 549
Content-Type: text/xml

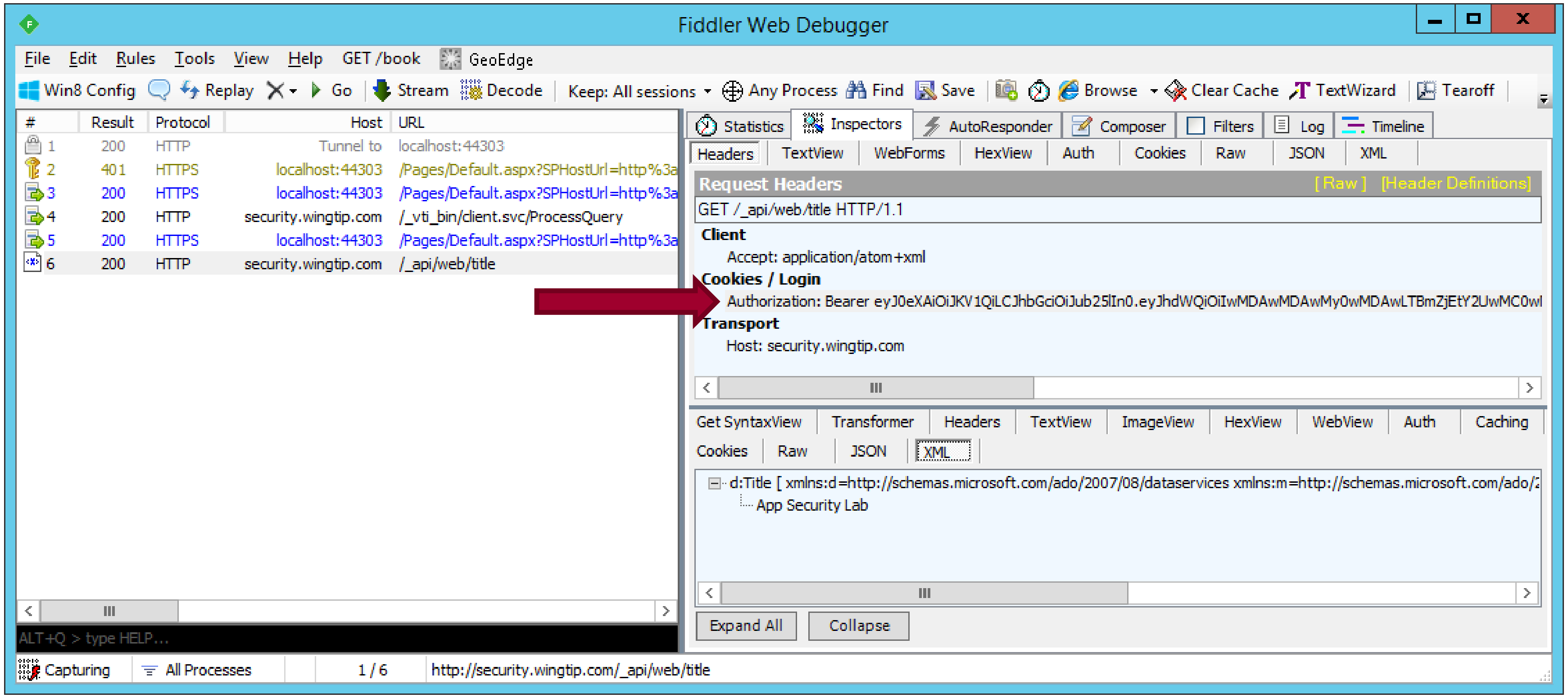
REST Calls using OAuth Authentication

- **Authorization** header must be added explicitly



```
protected void cmdGetTitleREST_Click(object sender, EventArgs e) {

  SharePointContext spContext =
    SharePointContextProvider.Current.GetSharePointContext(Context);

  string restUri = spContext.SPHostUrl + "_api/web/title";

  HttpWebRequest request = WebRequest.Create(restUri) as HttpWebRequest;
  request.Accept = "application/atom+xml";

  string spAccessToken = spContext.UserAccessTokenForSPHost;
  request.Headers["Authorization"] = "Bearer " + spAccessToken;

  HttpWebResponse response = request.GetResponse() as HttpWebResponse;
  XDocument responseBody = XDocument.Load(response.GetResponseStream());

  XNamespace nsDataService = "http://schemas.microsoft.com/ado/2007/08/dataservices";
  string hostWebTitle = responseBody.Descendants(nsDataService + "Title").First().Value;

  placeholderMainContent.Text = "Host web title (REST): " + hostWebTitle;

}
```

Examining REST Calls using Fiddler



Fiddler Web Debugger

#	Result	Protocol	Host	URL
1	200	HTTP	Tunnel to	localhost:44303
2	401	HTTPS	localhost:44303	/Pages/Default.aspx?SPHostUrl=http%3a
3	200	HTTPS	localhost:44303	/Pages/Default.aspx?SPHostUrl=http%3a
4	200	HTTP	security.wingtip.com	/_vti_bin/client.svc/ProcessQuery
5	200	HTTPS	localhost:44303	/Pages/Default.aspx?SPHostUrl=http%3a
6	200	HTTP	security.wingtip.com	/_api/web/title

Request Headers

```
GET /_api/web/title HTTP/1.1
Client
    Accept: application/atom+xml
Cookies / Login
    Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJub25lIn0.eyJhdWQiOiIwMDAwMDAwMy0wMDAwLTBmZjEtY2UwMC0wl
Transport
    Host: security.wingtip.com
```

```
d:Title [ xmlns:d=http://schemas.microsoft.com/ado/2007/08/dataservices xmlns:m=http://schemas.microsoft.com/ado/2
    App Security Lab
```

ALT+Q > type HELP...

Capturing | All Processes | 1 / 6 | http://security.wingtip.com/_api/web/title

Authentication with TokenHelper

- On-premises with S2S

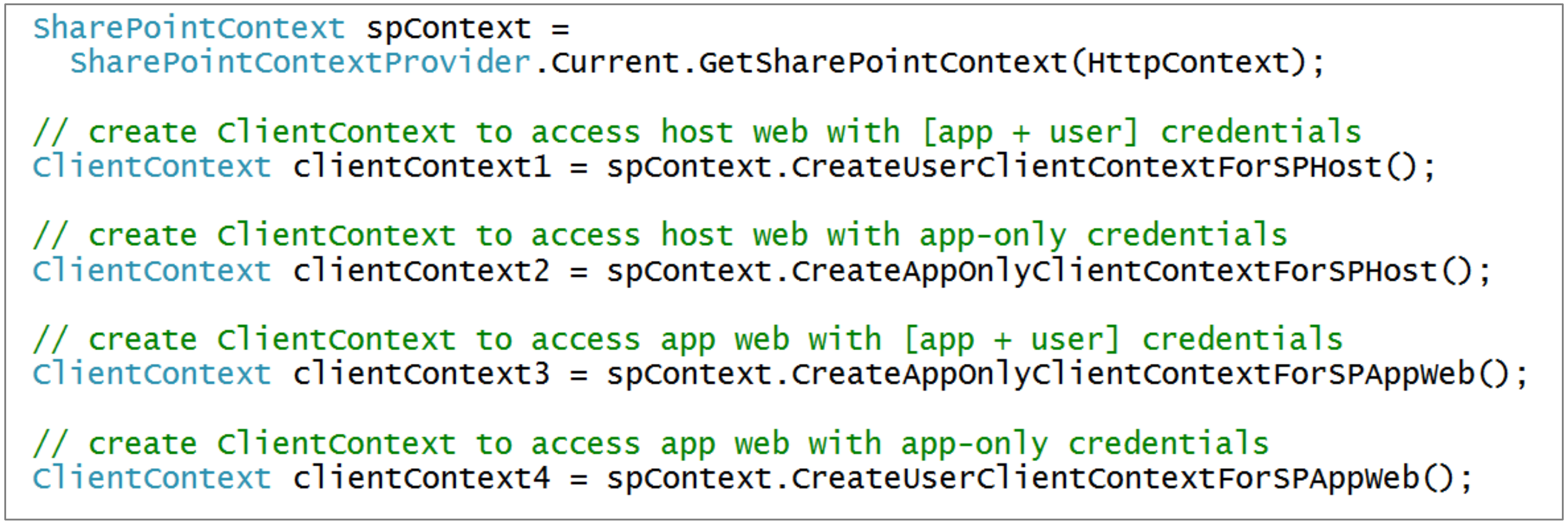
```
string hostWebUrl = Request.QueryString["SPHostUrl"];
Uri hostWebUri = new Uri(hostWebUrl);
WindowsIdentity userIdentity = Request.LogonUserIdentity;

ClientContext clientContext =
  TokenHelper.GetS2SClientContextWithWindowsIdentity(hostWebUri, userIdentity);
```

- In SharePoint Online with OAuth

```
string hostWebUrl = Request.QueryString["SPHostUrl"];
string remoteWebUrl = Request.Url.Authority;

string contextTokenString = TokenHelper.GetContextTokenFromRequest(Request);

ClientContext clientContext =
  TokenHelper.GetClientContextWithContextToken(hostWebUrl, contextTokenString, remoteWebUrl);

return clientContext;
```

Authentication with SharePointContext

- SharePointContext simplifies your code
  - Automatically tracks SharePoint query string variables
  - Abstracts away issues for OAuth vs S2S
  - Provides four ways to create ClientContext



```
SharePointContext spContext =
  SharePointContextProvider.Current.GetSharePointContext(HttpContext);

// create ClientContext to access host web with [app + user] credentials
ClientContext clientContext1 = spContext.CreateUserClientContextForSPHost();

// create ClientContext to access host web with app-only credentials
ClientContext clientContext2 = spContext.CreateAppOnlyClientContextForSPHost();

// create ClientContext to access app web with [app + user] credentials
ClientContext clientContext3 = spContext.CreateAppOnlyClientContextForSPAppWeb();

// create ClientContext to access app web with app-only credentials
ClientContext clientContext4 = spContext.CreateUserClientContextForSPAppWeb();
```

ClientContext Usage Pattern

- ClientContext is a disposable object
  - Should be disposed after you are done using it
  - Common to use within `using` statement

```
[SharePointContextFilter]
public ActionResult Index() {

  var spContext = SharePointContextProvider.Current.GetSharePointContext(HttpContext);

  using (ClientContext clientContext = spContext.CreateUserClientContextForSPHost()) {
    // work with ClientContext inside using statement
    Web site = clientContext.Web;
    clientContext.Load(site);
    clientContext.ExecuteQuery();
    ViewBag.HostWebTitle = site.Title;
    ViewBag.HostWebUrl = site.Url;
  }
  return View();
}
```

Summary

- ✓ SharePoint Add-in Model Overview
- ✓ SharePoint-hosted Add-in
- ✓ SharePoint Add-in Security
- ✓ Provider-hosted Add-ins
- ✓ Acquiring and Managing Access Tokens