

# Developing SharePoint-hosted Add-ins



# Agenda

- SharePoint Add-in Model
- SharePoint-hosted Add-in Architecture
- User Interface Design Techniques
- Developing App Parts
- Adding User Custom Actions



# Pain Points with SharePoint Solutions

- Custom code runs inside SharePoint environment
  - This poses risks and compromises scalability
- Custom solutions are SharePoint version dependent
  - Server-side code has must be adapted for each new version
  - SharePoint solutions hard code version specific paths to resources
  - Makes it harder to migrate to new versions of SharePoint
- Permissions model based entirely on user identity
  - You cannot configure permissions for the solution itself
  - Impersonation solves the too-little-permissions problem
  - Impersonation causes too-many-permissions problem
- SharePoint solutions are hard to manage
  - lack effective support for distribution, installation and upgrade



# What's in a Name?

- SharePoint 2013 introduced new development model
  - Originally introduced as "SharePoint App Model"
- In Q2 of 2015, Microsoft changed the name
  - "SharePoint Apps" renamed to "SharePoint Add-ins"
- Name change provides a potential source of confusion
  - "SharePoint App" and "SharePoint add-in" are the same thing
  - Visual Studio still calls them "SharePoint Apps"
  - SharePoint UI still calls them "SharePoint Apps"



# SharePoint Add-in Model Overview

- SharePoint Add-in model based on these assumptions:
  - Add-ins supported in Office 365 and in on-prem farms
  - Add-ins code never runs in SharePoint host
  - Add-ins talks to SharePoint using Web services
  - Add-ins is authenticated and has established identity
  - Add-in has permissions apart from user permissions
  - Add-ins published deployed to app catalog
  - Published add-ins easier to find, install and upgrade



# SharePoint Tenancies

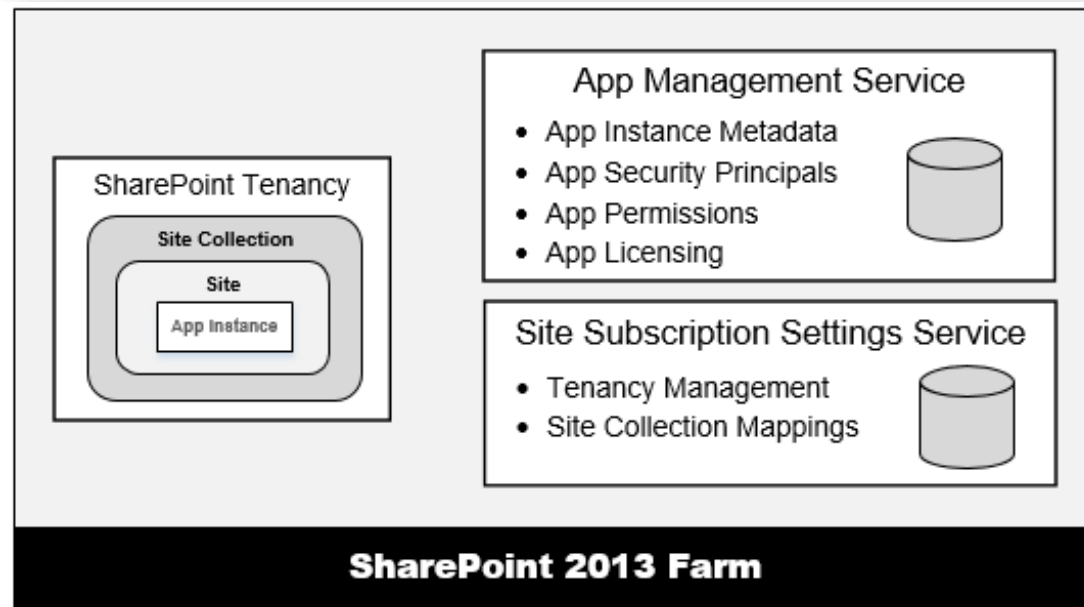
- A tenancy is a set of site collections
  - Configured and administrated as a unit
  - Created with administrative site collection
  - A scope for provisioning new site collection
  - Central concept to site management in Office 365
  - A requirement for installing SharePoint Add-ins
- What about tenancies in on-premises farms?
  - Most farms do not have explicitly created tenancies
  - To add support for SharePoint Add-ins, on-premises farm are configured with a farm-wide default tenancy





# Service Application Support for Add-ins

- Add-in support requires two service applications
  - App Management Service
  - Site Subscription Management Service

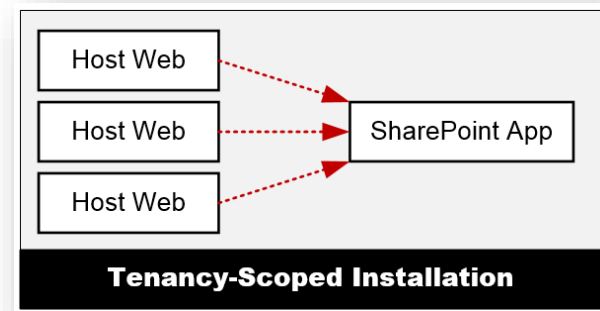
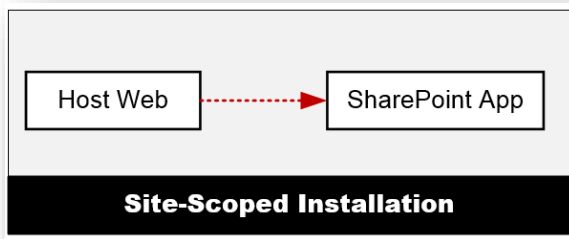


- These services must be created in on-premises farms to support Add-ins



# Add-in Installation Scopes

- Site-scoped Installation
  - Add-in installed in specific 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



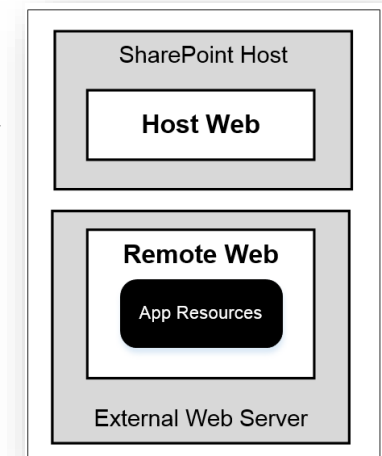
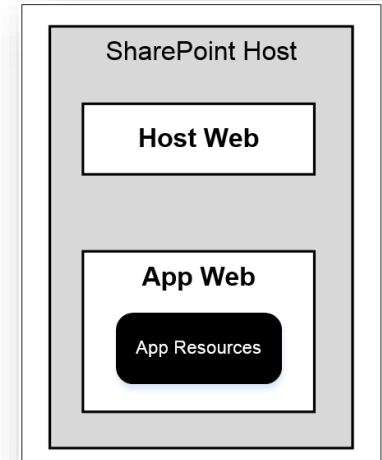
- Tenancy-scoped Installation
  - Provides centralized approach to Add-in deployment & management
  - Requires Add-in to first be installed in an app catalog site
  - Once installed, the Add-in is then configured for use multiple sites
  - Tenancy install scoped to web application in on-premises farms





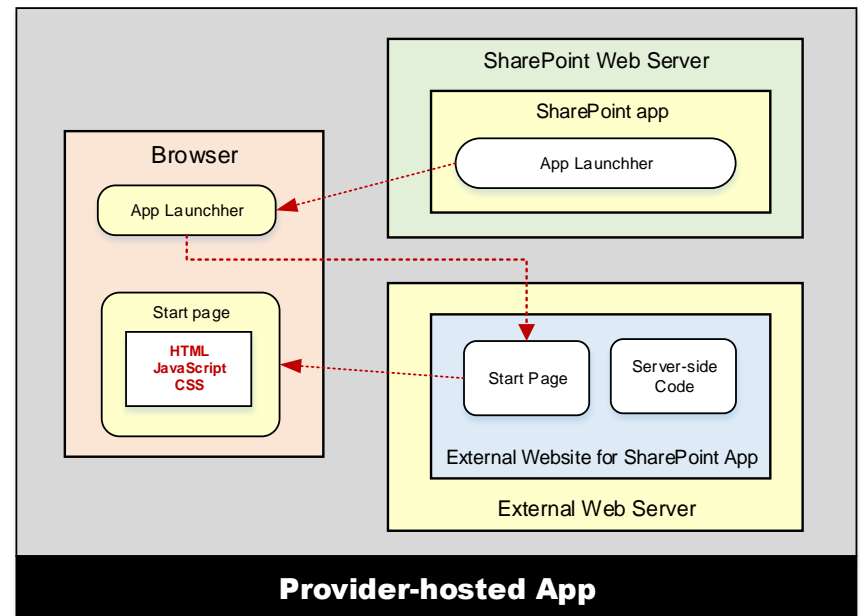
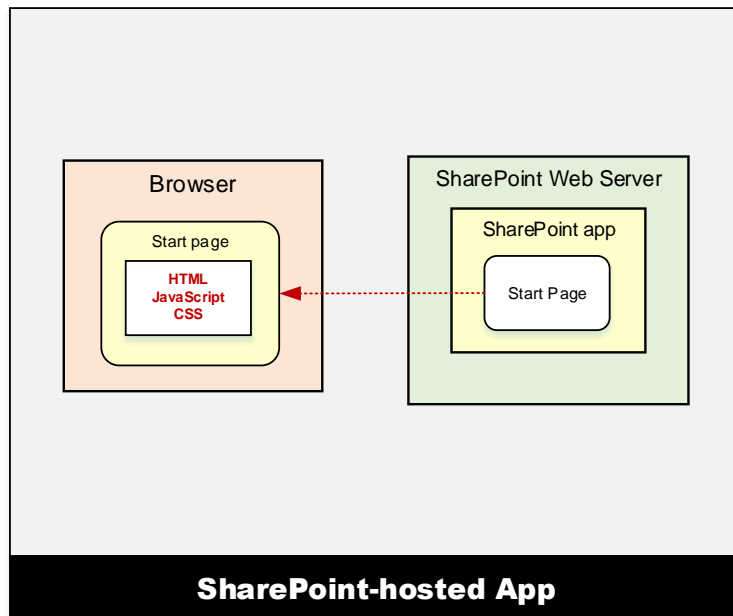
# Hosting Options for SharePoint Add-in

- SharePoint-Hosted Add-ins
  - Add-in resources added to SharePoint host
  - Stored in child site known as **app web**
  - Add-in can have client-side code
  - Add-in cannot have server-side code
- Provider-Hosted Add-ins
  - Add-in resources deployed on remote server
  - Remote site known as **remote web**
  - Add-in can have client-side code
  - Add-in can have server-side code



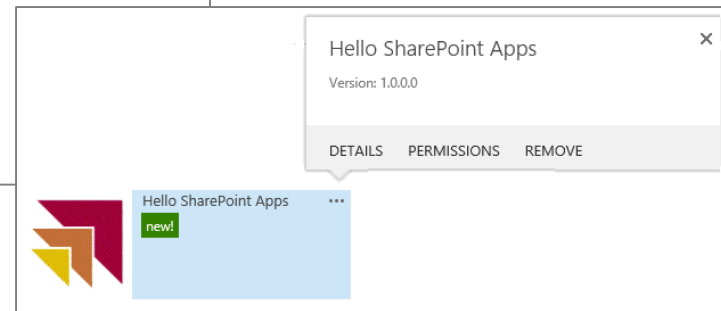
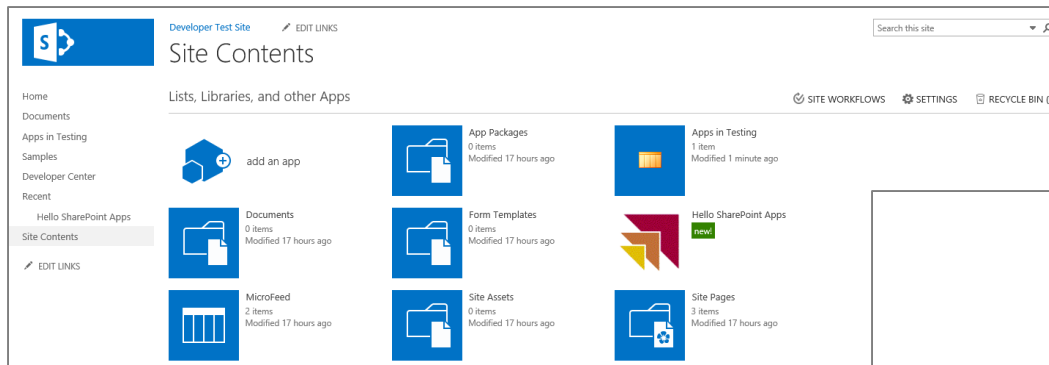
# Add-in Start Page

- Every 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



# User Experience with SharePoint Add-ins

- Users launch Add-in from tile on Site Contents
  - Add-ins grouped together with list & libraries
  - Add-in tile provides fly-out menu
  - Clicking on app tile redirects user to Add-in's start page





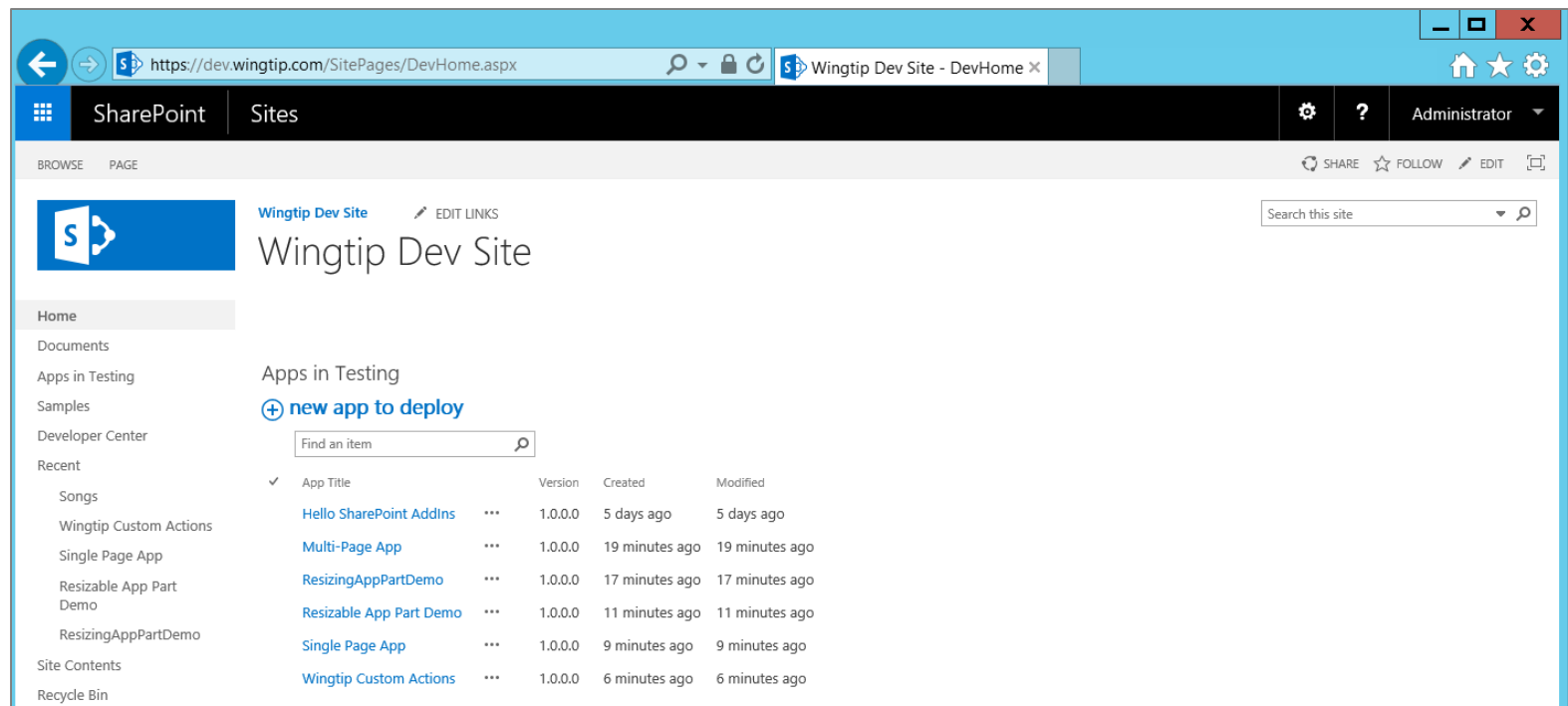
**DEMO**

## **Working with Add-ins from the Business User Perspective**



# Developer Sites

- Helpful in the development of SharePoint add-ins
  - Tracks a list of add-ins under development
  - Allows for remote deployment when using Visual Studio debugger





**DEMO**

## Creating a Developer Site

# Agenda

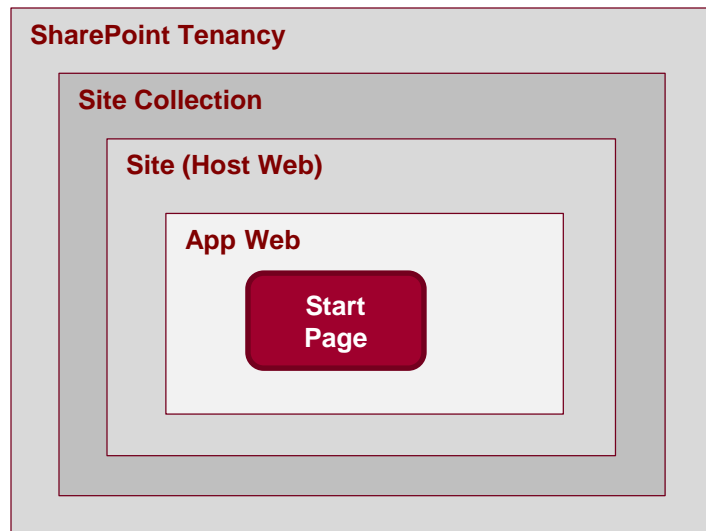
- ✓ SharePoint Add-in Model
- SharePoint-hosted Add-in Architecture
  - User Interface Design Techniques
  - Developing Add-in Parts
  - Adding User Custom Actions





# SharePoint-hosted Add-in Architecture

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



# App Web

- App web is created during add-in installation
  - App web created as child to site where Add-in is installed
- SharePoint-Hosted add-ins must create app web
  - Add-in requires home for start page & related resources
  - Add-in can add other SharePoint elements (e.g. lists)
- Provider-hosted Add-ins *can* create app web
  - Provider-hosted add-in doesn't get an app web by default
  - Provider-hosted add-in 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 add-in

```
https://wingtiptenant-f4c0ba5547a0ad.wingtip.com/MyLittleAddIn
```

- URL to app web made up of 4 parts
  - **Tenancy name:** wingtiptenant
  - **APPUIID:** f4c0ba5547a0ad
  - **App web hosting domain:** wingtip.com
  - **App name:** MyLittleAddin



# Start Page URL

- Dynamic tokens used in start page URL
  - SharePoint-Hosted add-ins use **~appWebUrl** token  
`~appweburl/Pages/Default.aspx`
  - All Add-ins should use **{StandardTokens}** token  
`~appweburl/Pages/Default.aspx?{StandardTokens}`



# {StandardTokens}

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

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



# Agenda

- ✓ SharePoint Add-in Model
- ✓ SharePoint-hosted Add-in Architecture
- User Interface Design Techniques
  - Developing App Parts
  - Adding User Custom Actions



# SharePoint Add-in User Interface Design

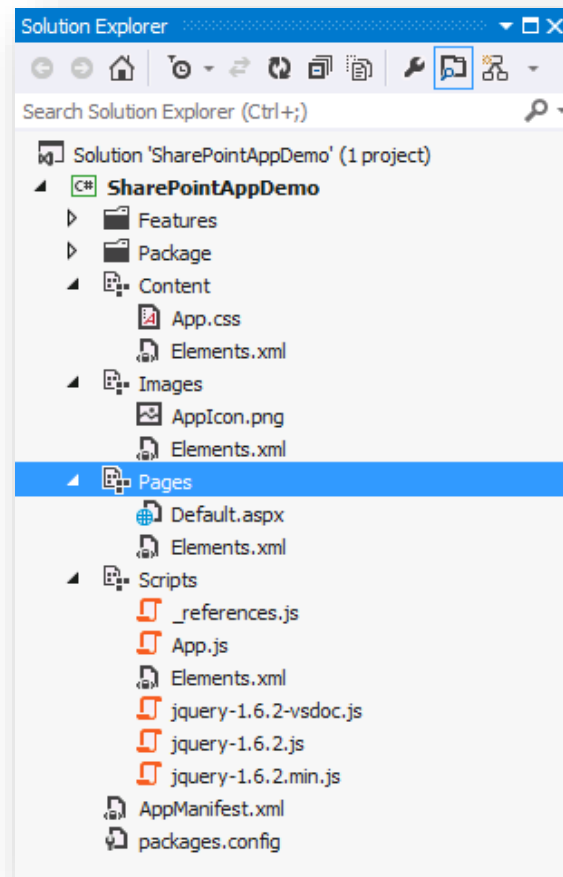
- Start page (required)
  - Represents user entry point into Add-in
  - Can be implemented with .aspx file or .htm file
- App part
  - External page (e.g. from app web) surfaced in host web
  - Displayed on host web pages using iFrame
- User custom action
  - URL-based command surfaced in host web
 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





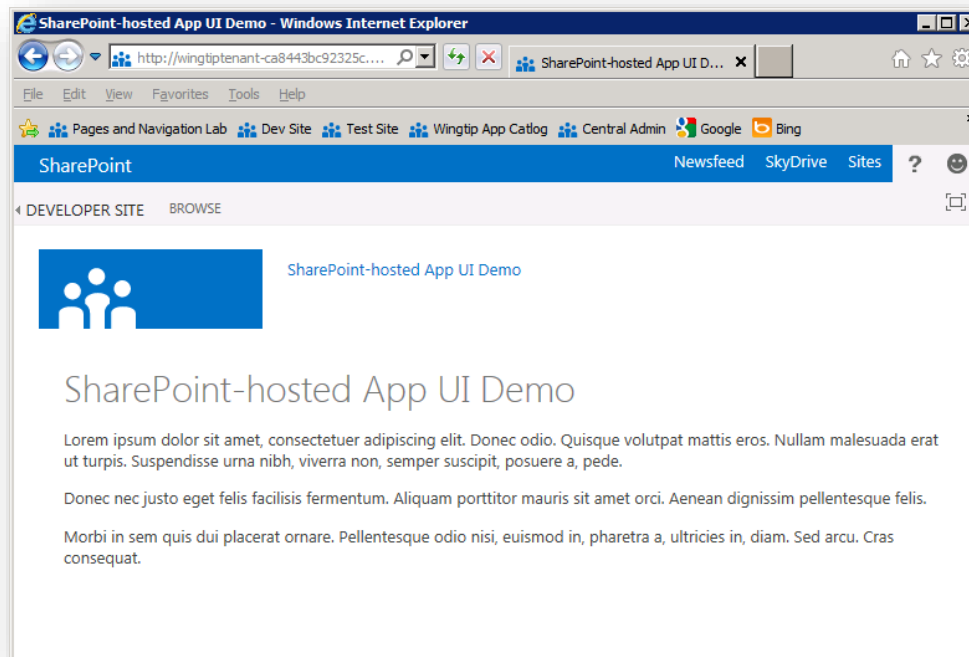
# Modules in a SharePoint-Hosted Add-in

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



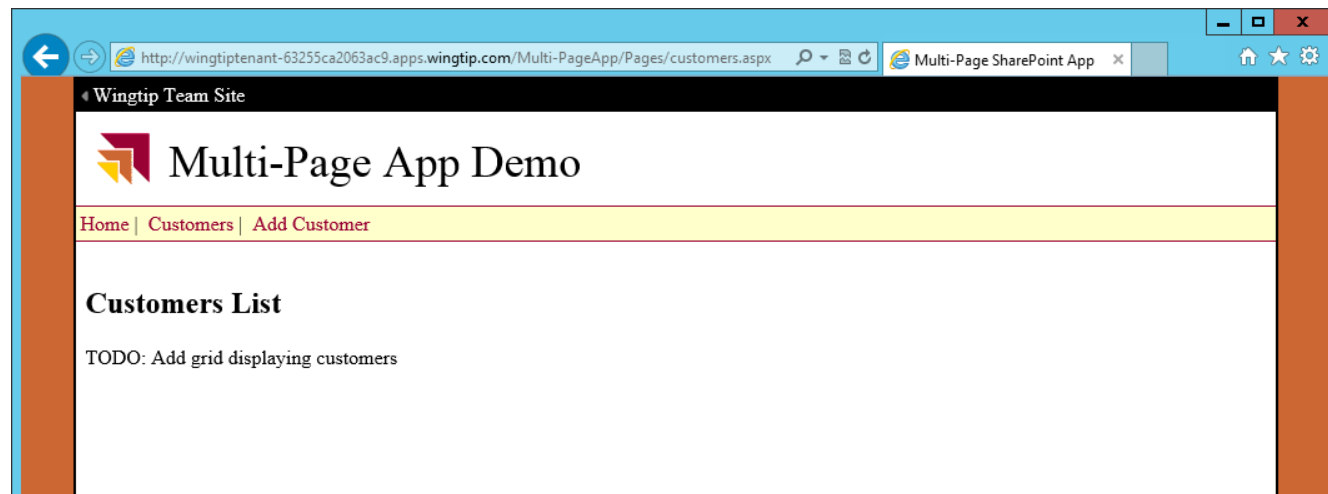
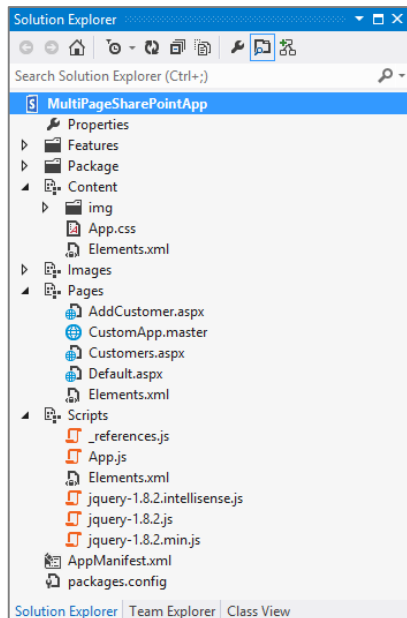
# App.master

- App web uses `app.master` by default
  - Gives Add-in SharePoint look and feel
  - Provides Add-in with required link back to host web
  - Does not have Site Actions menu or top link bar



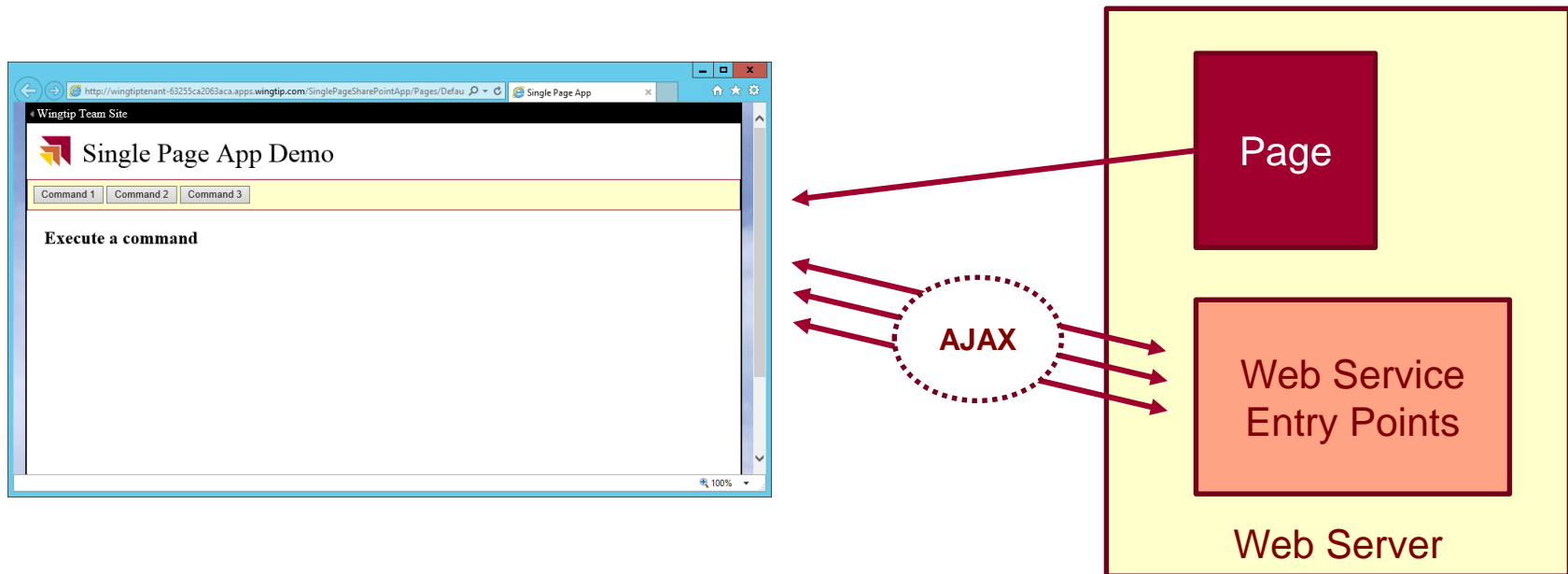
# Multi-page Add-in with Custom Master Page

- Multiple pages can use same master page
  - Link to host web can be added to master page
  - Navigation can be added to master page
  - **Issue:** query string parameters only sent to start page



# The Single Page App Model

- Benefits of single page app
  - Request data posted to start page is always there
  - JavaScript variables do not unload/reload
  - App makes AJAX calls and uses client-side JavaScript
  - Design leads to better and more fluid user experience







**DEMO**

# Creating a Single Page App

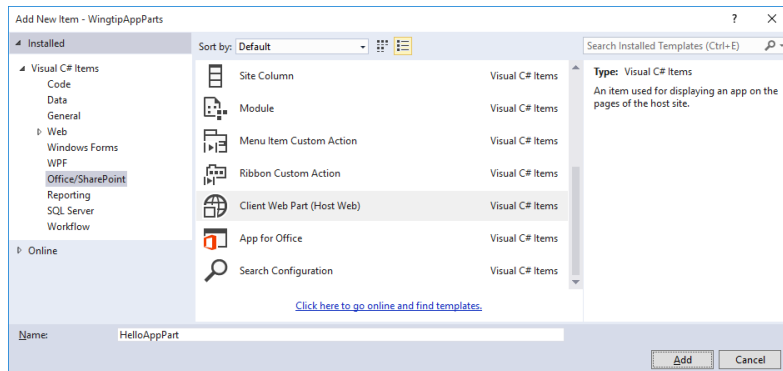
# Agenda

- ✓ SharePoint Add-in Model
- ✓ SharePoint-hosted Add-in Architecture
- ✓ User Interface Design Techniques
- Developing App Parts
  - Adding User Custom Actions



# Adding Add-in Parts to a Project

- Add new item based on Client Web Part project item



- App part requires ClientWebPart definition in element.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

```
<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_&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**



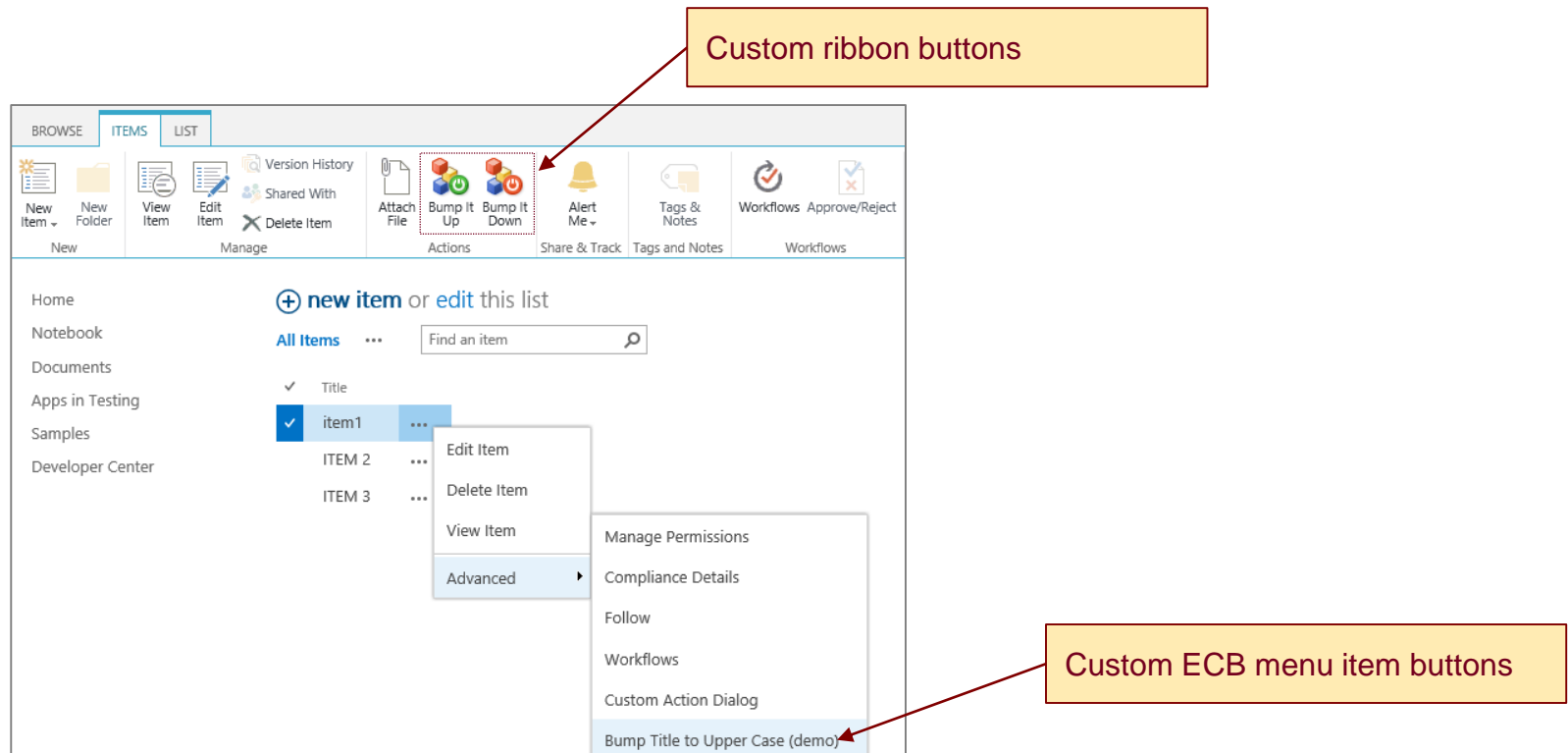
# Agenda

- ✓ SharePoint Add-in Model
- ✓ SharePoint-hosted Add-in Architecture
- ✓ User Interface Design Techniques
- ✓ Developing App Parts
- Adding User Custom Actions



# Creating User Custom Actions

- User custom actions used to add commands to host web
  - Custom action can create ECB menu items and ribbon 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 modal 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}&SPListItemId={ItemId}&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 <b>(ECB menu actions only)</b>
{SelectedItemId}	Array of item IDs in a list or library <b>(Ribbon menu actions only)</b>
{ItemUrl}	URL of target item being acted upon <b>(ECB menu actions only)</b>
{SelectedItemUrl}	URL array of target items being acted upon <b>(Ribbon menu actions only)</b>
{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







**DEMO**

## **Adding User Custom Actions**

# Summary

- ✓ SharePoint Add-in Model
- ✓ SharePoint-hosted Add-in Architecture
- ✓ User Interface Design Techniques
- ✓ Developing App Parts
- ✓ Adding User Custom Actions

