Agenda

• Why Apps
• App Model
• App Hosting Models
• Development Strategies
Introducing SharePoint App Model

- SharePoint applications no longer live in SharePoint
- Custom code executes in the client, cloud or on-prem
- Apps are granted permissions to SharePoint via OAuth
- Apps communicate with SharePoint via REST / CSOM
- Acquire apps via centralized Marketplace
  - Corporate Marketplace
  - Public Marketplace (via submission process)
  - APIs for manual deployment
App Principles

Apps are a very different philosophical change to extending SharePoint

- More scenario-focused
  - Event Tracking
  - Ticket Management System
- Robust
  - Built-in robust semantics for install, upgrade & uninstall
- Apps are for End Users
- Cloud & Web-Oriented
• Everything in a SharePoint site is an app
  • Contact form
  • Travel request
  • Shared Documents library
  • Contacts list

• Formerly known as View All Content, the Site Contents link shows all apps in a site & links to the SharePoint Store (public) and App Catalog (internal)
• Selecting an app redirects to the app’s start page
## SharePoint Solutions & Apps

<table>
<thead>
<tr>
<th>Feature</th>
<th>Full-Trust Solutions</th>
<th>Sandboxed Solutions</th>
<th>Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Client-Side SharePoint API</td>
<td>⭐⭐</td>
<td>⭐⭐</td>
<td>⭐⭐</td>
</tr>
<tr>
<td>Use Server-Side SharePoint API</td>
<td>⭐⭐⭐</td>
<td>⭐⭐</td>
<td>⭐</td>
</tr>
<tr>
<td>Use Remote Services</td>
<td>⭐⭐⭐</td>
<td>⭐⭐</td>
<td>⭐⭐</td>
</tr>
<tr>
<td>App-based Permissions (OAuth2)</td>
<td>⭐⭐⭐</td>
<td>⭐⭐</td>
<td>⭐⭐</td>
</tr>
<tr>
<td>On-Premise Deployment Friendly</td>
<td>⭐⭐⭐</td>
<td>⭐⭐⭐</td>
<td>⭐⭐</td>
</tr>
<tr>
<td>Hosted Deployment Friendly</td>
<td>⭐⭐⭐</td>
<td>⭐⭐</td>
<td>⭐⭐</td>
</tr>
<tr>
<td>Distribution via Marketplace</td>
<td>⭐⭐⭐</td>
<td>⭐⭐</td>
<td>⭐⭐</td>
</tr>
<tr>
<td>SharePoint Provided Schematics for Install / Upgrade / Uninstall</td>
<td>⭐⭐⭐</td>
<td>⭐⭐</td>
<td>⭐⭐</td>
</tr>
</tbody>
</table>

© Copyright 2014, Neudesic. All rights reserved.
Choices when Building Apps

- Hosting
- Entry Points / Experience
- Scoping
Hosting: Choice of Three Architecture Approaches

Cloud-based Apps

Provider-Hosted App
“Bring your own server hosting infrastructure”

Autohosted App
Windows Azure + SQL
provisioned invisibly as apps are installed

SharePoint-Hosted App
Provision an isolated sub web on a parent web
• Reuse web elements (lists, files, out-of-box web parts)
• No server code allowed; use client JavaScript for logic, UX

SharePoint Web
Your Hosted Site

Windows Azure Websites

Parent Web

App Web (from WSP)
## Hosting: Cloud vs. SharePoint

<table>
<thead>
<tr>
<th>Cloud Hosted Apps</th>
<th>SharePoint Hosted Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preferred hosting model for almost all types of apps</td>
<td>Good for smaller apps &amp; resource storage</td>
</tr>
<tr>
<td>Full power of web – choose your infrastructure &amp; technology</td>
<td>SharePoint-based; no server-side code</td>
</tr>
<tr>
<td>May require your own hosting</td>
<td>Automatically hosted in SharePoint</td>
</tr>
<tr>
<td>May require you own handling of multitenancy &amp; permission management</td>
<td>Inherent multitenancy &amp; isolation</td>
</tr>
</tbody>
</table>
App User Experience Decisions

- App Shape
  - Entry Point & App Shape
- Branding
  - Custom / SharePoint UI
### Entry Point / Experience: App Shapes

<table>
<thead>
<tr>
<th>Shape</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Immersive Full Page App</td>
<td>App that implements a new scenario for customers</td>
<td>Resource Tracking, Budgeting</td>
</tr>
<tr>
<td>App Part</td>
<td>Provides new parts you can add to your sites</td>
<td>Weather, Team Mascot, News</td>
</tr>
<tr>
<td>Extension App</td>
<td>Add new actions for documents and items</td>
<td>Display Document Visualization, Print to Print Service Vendor</td>
</tr>
</tbody>
</table>
App Branding

- User Experience Guide for SharePoint 2013 Apps
- Options
  - App Template
    - ASPX page hosted in SharePoint
    - app.masterpage: includes AppWeb chrome
    - Default option when creating apps with VS
  - Chrome Control
  - Custom Branding

- Must include “Back to Site” link in upper-left
App Scoping

• **Web Scope**
  - Can register and use resources in parent site, site collection

• **Tenant Scope**
  - Can register start page, custom actions
  - Tenant Admins can filter-enable on sites
    - By Site Collection, Managed Path, Template Type
  - (SharePoint-hosted tenant-scope apps not available)
Application Isolation

- When apps are provisioned, new SPWeb (AppWeb) created within hosting SPWeb
  - Each app resides within it’s own SPWeb for isolation
  - Special DNS address configured by administrators
  - App SPWeb’s live in separate domain (DNS)

- Each App hosted on it’s own unique URL because:
  - Blocks XSS: isolation to special SPWeb under special domain blocks cross site scripting
  - Enforces App Permissions: apps communicate with sites via CSOM / API & must be granted to do so
App Model Type 1: SharePoint Hosted

- SharePoint hosted apps wholly reside in SharePoint
- Uses SharePoint artifacts (lists/libraries)
- Business logic executes or on the client
  - HTML5
  - JavaScript
App Model Type 2: Cloud Hosted

- Cloud hosted apps primarily execute outside of SharePoint
- May use SharePoint artifacts (lists/libraries)
  - Communicate via CSOM / REST
  - Granted permission to SharePoint via OAuth
- Business logic lives & executes outside of SharePoint
  - On-Premise hosted web application
  - Windows Azure
  - 3rd party host
New Context

```csharp
var spContext = SharePointContextProvider.Current.GetSharePointContext(Context);
using (var clientContext = spContext.CreateUserClientContextForSPHost())
{
    clientContext.Load(clientContext.Web, web => web.Title);
    clientContext.ExecuteQuery();
    Response.Write(clientContext.Web.Title);
}
```
Separation of Concerns
protected override void CreateChildControls()
{
    //Create the list selector
    listOfSiteLists = new DropDownList();
    ListItem newItem;
    //Get the current SPWeb and find all the lists
    using (SPWeb currentWeb = SPContext.Current.Web)
    {
        foreach (SPList currentList in currentWeb.Lists)
        {
            //Add each SharePoint list to the drop-down list
            newItem = new ListItem();
            newItem.Text = currentList.Title;
            newItem.Value = currentList.ID.ToString();
            listOfSiteLists.Items.Add(newItem);
        }
    }
    this.Controls.Add(listOfSiteLists);
}
What if we did this...

```csharp
protected override void CreateChildControls()
{
    //Create the list selector
    listOfSiteLists = new DropDownList();
    ListItem newItem;

    ISPContent MySPContent = Injector.Get("SPContent");
    List<ISPLIST> mySPListList =
        MySPContent.GetListsForCurrentWeb();

    foreach (ISPLIST list in mySPListList)
    {
        //Add each SharePoint list to the drop-down list
        newItem = new ListItem();
        newItem.Text = list.Title;
        newItem.Value = list.ID;
        listOfSiteLists.Items.Add(newItem);
    }
    this.Controls.Add(listOfSiteLists);
}

interface ISPLIST
{
    string Title { get; }
    string ID { get; }
    void Add (string title, string id);
}

public List<ISPLIST> GetListsForCurrentWeb()
{
    List<ISPLIST> mySPListList = GetSPListInstance();
    using (SPWeb currentWeb = SPContext.Current.Web)
    {
        foreach (SPList currentList in currentWeb.Lists)
        {
            mySPListList.Add(currentList.Title, currentList.ID.ToString());
        }
    }
    return mySPListList;
}
```
protected override void CreateChildControls()
{
    //Create the list selector
    listOfSiteLists = new DropDownList();
    ListItem newItem;

    ISPContent MySPContent = Injector.Get("SPContent");
    List<ISPLIST> mySPLISTList =
        MySPContent.GetListsForCurrentWeb();

    foreach (ISPLIST list in mySPLISTList)
    {
        //Add each SharePoint list to the drop-down list
        newItem = new ListItem();
        newItem.Text = list.Title;
        newItem.Value = list.ID;
        listOfSiteLists.Items.Add(newItem);
    }

    this.Controls.Add(listOfSiteLists);
}

public List<ISPLIST> GetListsForCurrentWeb()
{
    List<ISPLIST> mySPLISTList = GetSPListInstance();

    #region GetContext Token and hostWeb

    using (var context =
        TokenHelper.GetClientContextWithContextToken
        (hostWeb, contextToken, Request.Url.Authority))

    Web web = context.Web;

    IEnumerable<SPList> result =
        context.LoadQuery(web.Lists.Include(
            list => list.Title, list => list.Id));

    context.ExecuteQuery();

    foreach (List list in web.Lists)
    {
        mySPLISTList.Add(list.Title, list.Id);
    }

    return mySPLISTList;
}
Loosely Coupled Code
Resources

- Me - http://blog.randomdust.com