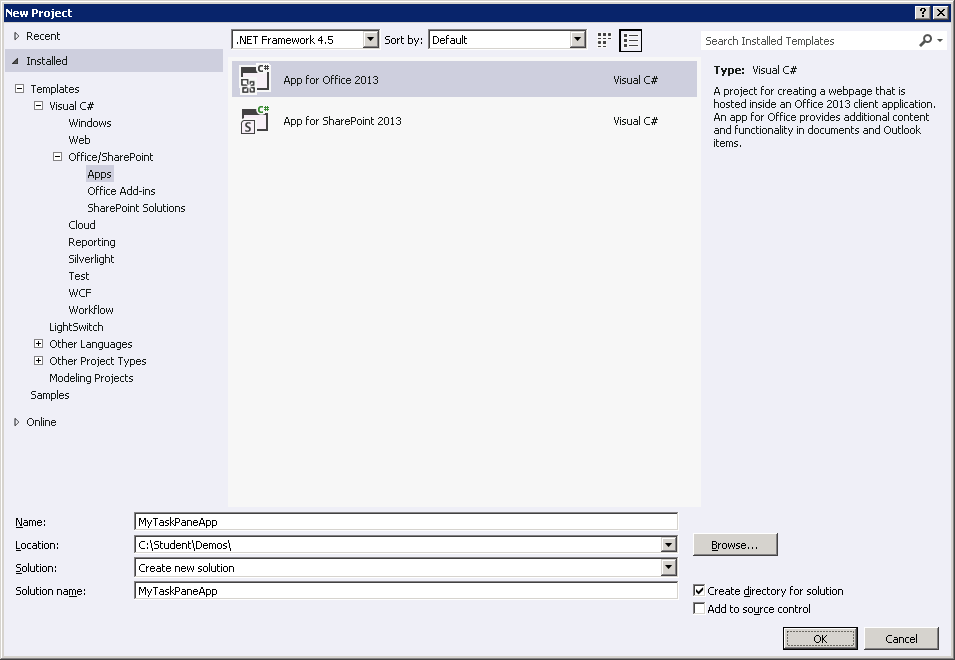
# Demo 1: Developing a Task Pane App for Office 2013

In this demo you will create and test a simple "Hello World"-style task pane app,.

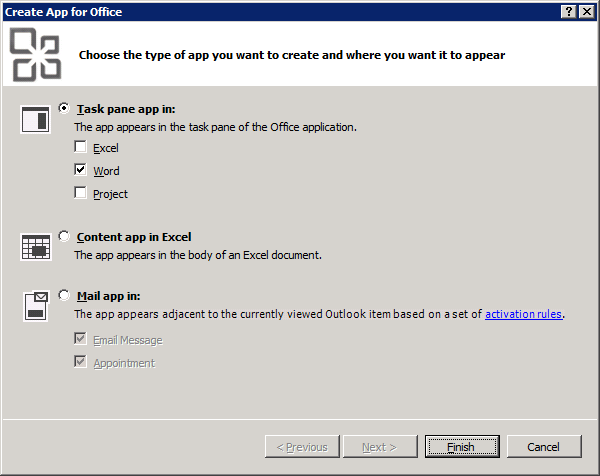
## Step 1 – Create a new app named MyTaskPaneApp

In this Step, you will develop a new task pane app in Visual Studio.

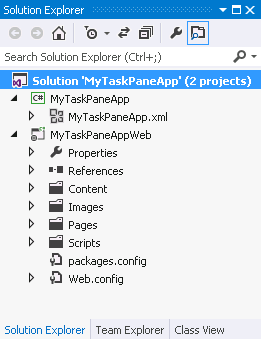
1. Open Visual Studio 2012.
2. Run the **File >** **New Project** command. When the **New Project** dialog appears, select the **Apps for Office 2013** project template from the **Office/SharePoint > Apps** template folder as shown below. Name the new project **MyTaskPaneApp** and click **OK** to create the new project.



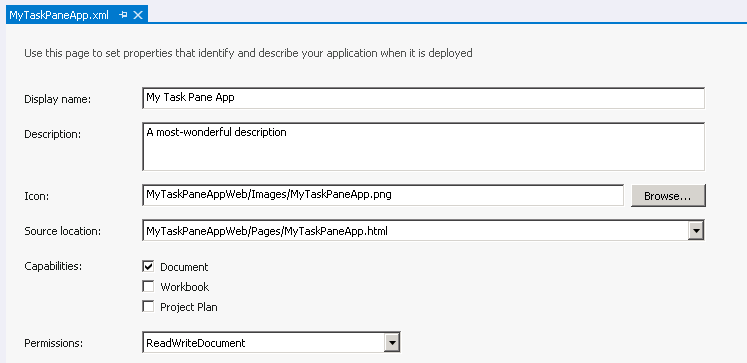
1. Fill out the next dialog as the one shown below in order to create a new task pan app that is designed to work just with Microsoft Word. Click Finish when you are done.



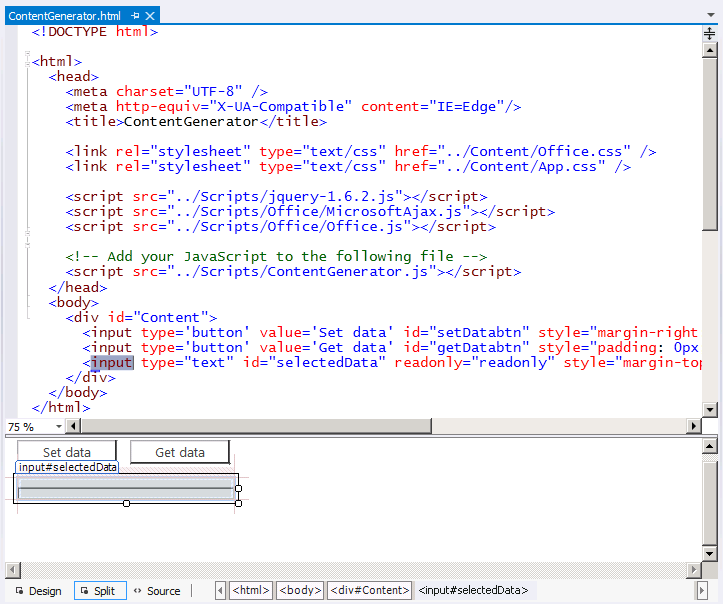
1. Once the solution has been created, take a moment to show students the two projects inside. The first project named **MyTaskPaneApp** just contains an xml file for the app manifest named **MyTaskPaneApp.xml**. The other project named **MyTaskPaneAppWeb** is the remote web project where you implement the server-side functionality for the app.



1. Walk the students through the structure of folders and files inside the remote web project. You will notice that a top-level folder structure has been created with folders named **Content**, **Pages** and **Scripts**. You poin out that the files and links have been added to support the jQuery library. That means you can begin using jQuery without any additional work.
2. Inside the top project, double-click on **MyTaskPaneApp.xml** to open the app manifest inside the special designer provided by Visual Studio 2012. Inside the app manifest designer, modify the app **Display** name and **Description** using the values shown in the following screenshot or with something else more creative or humorous if you'd like.



1. Save and close **MyTaskPaneApp.xml**.
2. As you implement this app over the next few steps, you will mainly be making modifications to three primary files that are named as follows:
   1. **MyTaskPaneApp.html**
   2. **App.css**
   3. **MyTaskPaneApp.js.**
3. Open the HTML source file that defines the user interface for the app which is named **MyTaskPane.html** located inside the **Pages** folder. Note that there is some pre-existing HTML that is supplied by Visual Studio when you create a new project. Show students how to examine this html using **Source View**, **Design View** and **Split View**.



1. Now it’s time to create a simple user interface using HTML. Start by deleting all the content inside the body section of **MyTaskPaneApp.html**. Make sure you leave everything in the head section just as it is.

<!DOCTYPE html>

<html>

**<head>**

**<!-- leave everything in head section unchanged -->** <meta charset="UTF-8" />

<meta http-equiv="X-UA-Compatible" content="IE=Edge"/>

<title>MyTaskPaneApp</title>

<link rel="stylesheet" type="text/css"

href="../Content/Office.css" />

<link rel="stylesheet" type="text/css"

href="../Content/App.css" />

<script src="../Scripts/jquery-1.6.2.js"></script>

<script src="../Scripts/Office/MicrosoftAjax.js"></script>

<script src="../Scripts/Office/Office.js"></script>

<script src="../Scripts/ContentGenerator.js"></script>

**</head>**

**<body>**  
 **<!-- delete everything from body section**

**</body>**

</html>

1. Add the following HTML code shown in the code listing below this step inside the body element. Note that you can type this HTML yourself or alternatively copy-and-paste the code from the **body\_MyTaskPaneApp.html.txt** file located in the **StarterFiles** folder for this demo. After you have added the HTML code, save your work by saving the changes to of **MyTaskPaneApp.html**.

<body>

<h1>My App Title</h1>

<div id="toolbar">

<input id="cmdUpdateApp" type="button" value="Update App" />

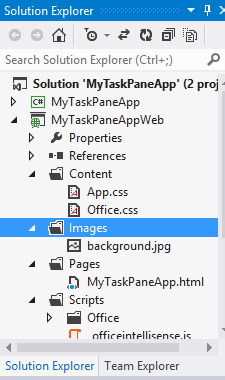
<input id="cmdUpdateDoc" type="button" value="Update Doc" />

</div>

<div id="display" />

</body>

1. Inside the Windows Explorer, look inside the folder for this lab and locate the image file named **background.jpg** in in the **StarterFiles** folder. In the next step, you will add this image file into the app's web project.
2. Inside the app's web project in Visual Studio, add a new folder **Images**. Add the image file background.jpg from the **StarterFiles** folder into your Images folder of your project.



1. Open the source CSS file named **app.css** and show student the contents. Add the following CSS rules shown below this step in **app.css**. Note that you can either type these CSS rules yourself or alternatively copy-and-paste the same code from the **App.css.txt** file located in the **StarterFiles** folder for this demo. After you have added these CSS rules, save your work by saving the changes to **app.css**.

body {

margin: 0px;

padding: 0px;

background-image: url("../Images/background.jpg");

}

h1 {

color: White;

padding: 4px;

font-size: 1.5em;

box-sizing: border-box;

height: 32px;

margin: 0px;

background-color: black;

}

#toolbar {

height: 32px;

box-sizing: border-box;

padding: 8px;

}

#toolbar input[type=button]{

width: 110px;

height:24px;

padding: 2px;

background-color:white;

border:1px solid #ccc;

cursor:pointer;

}

#display {

margin: 8px;

border: 1px solid #300;

padding: 8px;

background-color: white;

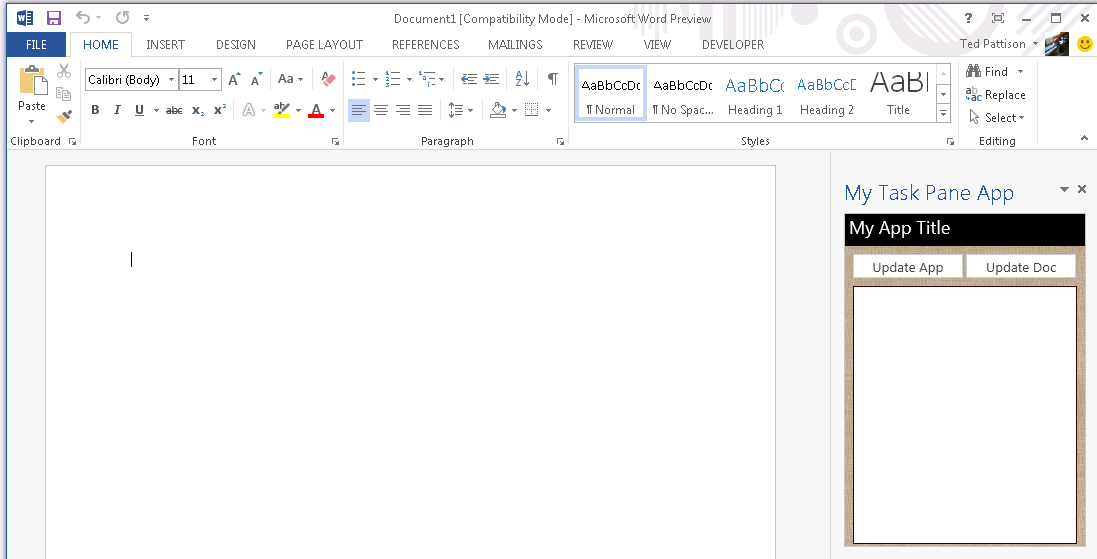
min-height: 240px;

font-size: 1.25em;

color: navy;

}

1. Now you can run the app to test it and see the user interface by pressing the {F5} key. Show the students that the user interface for the app has now be implemented. Now all that is left is to write the code to give the app its behavior.



1. Close Microsoft Word and return to Visual Studio and ensure the debugging session has stopped. If the debugging session is still running, stop it by running the **Stop Debugging** command from the **Debug** menu.
2. Open the JavaScript source file named **MyTaskPaneApp.js** and inspect the code inside. Point out that there is an **Office.initialize** function which will be executed automatically when the app is loaded and initialized. Delete all the code from **MyTaskPaneApp.js** and replace it with this generic starting point.

Office.initialize = function (reason) {

$(function () {

// intialize code goes here

});

}

1. At the bottom of **MyTaskPane.js**, add a two new JavaScript function named **onUpdateApp** and **onUpdateDoc**.

function onUpdateApp() {

}

function onUpdateDoc() {

}

1. Add initialization code using jQuery to wire these two functions up with the click event of the two command buttons that were created in HTML.

Office.initialize = function (reason) {

$(function () {

$("#cmdUpdateApp").click(onUpdateApp);

$("#cmdUpdateDoc").click(onUpdateDoc);

});

}

function onUpdateApp() {

}

function onUpdateDoc() {

}

1. Write the implementation of the **onUpdateApp** to acquire a reference to the HTML div element with the id of **display** using jQuery syntax and update its contents to **"Hello to the app!!"** using the jQuery **text** method.

function onUpdateApp () {

$("#display").text("Hello to the app!");

}

1. Implement **onUpdateDoc** to update the contents of the current Word document with a simple text-based message. Do this by creating a variable named **doc** and initialized it with **Office.context.document** and then call **setSelectedDataAsync** to write a text-based message of **"Hello to the doc!**" to the selected region in the current document.

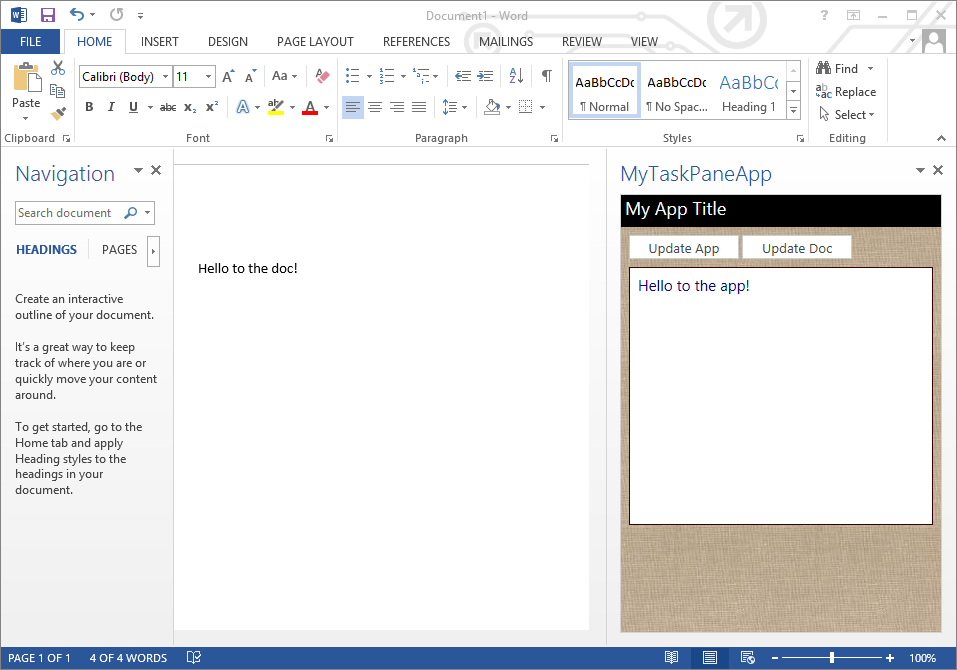
function onUpdateDoc() {

var doc = Office.context.document;

doc.setSelectedDataAsync("Hello to the doc!", function () { });

}

1. You have now added all the required functionality and it is time to test your project. Press the **{F5}** key to begin debugging the project. When the app has started, you should be able to click the **Update App** button and see your code update the user interface inside the app. When you click the **Update Doc** button your code should write its message into the active Word document.

.

1. The demo is now over. Make any closing comments and summarize the demo.
2. Return to the slides and resume the lecture.