|  |
| --- |
| Microsoft SharePoint 2013 - Hands-on Lab |
| Introduction to Apps for Office |
| Verified Against Build 15.0.4420.1017 |

|  |
| --- |
| Microsoft  Version 1.0  August 14, 2012 |

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

The names of manufacturers, products, or URLs are provided for informational purposes only and Microsoft makes no representations and warranties, either expressed, implied, or statutory, regarding these manufacturers or the use of the products with any Microsoft technologies. The inclusion of a manufacturer or product does not imply endorsement of Microsoft of the manufacturer or product. Links are provided to third party sites. Such sites are not under the control of Microsoft and Microsoft is not responsible for the contents of any linked site or any link contained in a linked site, or any changes or updates to such sites. Microsoft is not responsible for webcasting or any other form of transmission received from any linked site. Microsoft is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement of Microsoft of the site or the products contained therein.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Copyright © 2012 Microsoft Corporation. All rights reserved.

Microsoft and SharePoint are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

# Introduction

## Estimated time to complete this lab

60 minutes

## Objectives

After completing this lab, you will be able to:

* Create Apps for Office with Visual Studio 2012
* Understand the fundamentals of App for Office Development
* Understand how to use the App for Office 2013 Project Template in Visual Studio 2012
* Understand the proper techniques to test and debug Apps for Office 2013

## Overview of Lab

In this lab you will learn how to create apps for Office with Visual Studio 2012, and you will understand the fundamentals of building an App for Office 2013.

## Virtual Machine Technology

The computers in this lab are virtual machines that are implemented using Microsoft Hyper-V. Before starting each virtual machine, ensure you apply the **Start-Lab** snapshot. When you have started a virtual machine, log on by pressing **CTRL+ALT+END** and supply the credentials listed in the lab instructions.

## Computers in this lab

This lab uses virtual machines as described in the following table. Before you begin the lab, you must start the virtual machines and then log on to the computers.

|  |  |
| --- | --- |
| Virtual Machine | Role |
| {Supplied by Instructor} | Domain Controller |
| {Supplied by Instructor} | Actual SharePoint environment with Office client and other required software. |

All user accounts in this lab use the password {Supplied by Instructor}.

# Exercise 1: Hello Apps for Office!

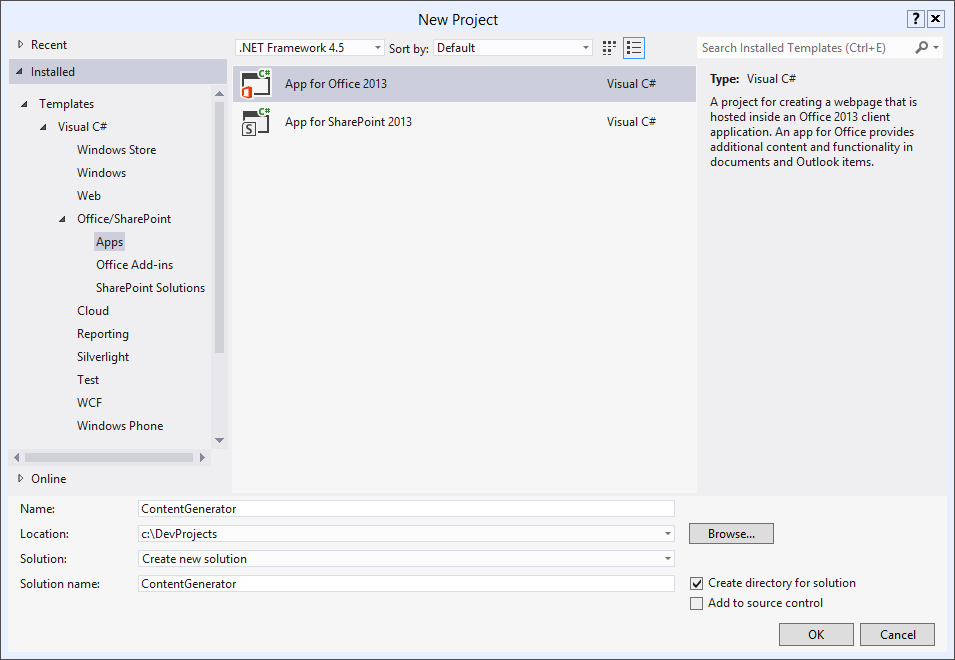
In this exercise you will create a new app for Office project in Visual Studio 2012 so that you can begin to write, test and debug your first task pane app. The user interface of the app you will create in this lab will not be very complicated. You will create a simple user interface experience using the basic building blocks of Apps for Office development which are HTML, CSS and JavaScript.

The user interface of the app you will create will include two command buttons and div elements along with the JavaScript code required to read and update content in HTML elements. Along the way you will be using the jQuery library to make your JavaScript code easier to write and maintain. At the end of the lab you will also write the JavaScript code required for your app to insert content into the current Word document.

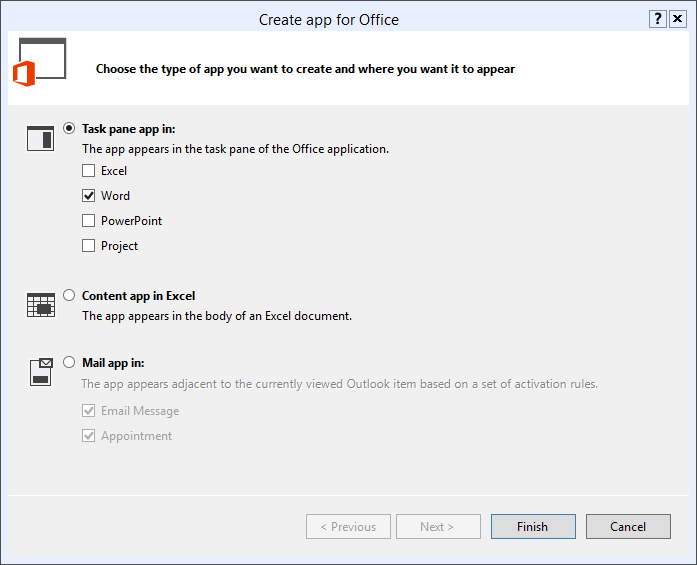
## Task 1 – Create an App for Office 2012 with Visual Studio

In this exercise you will create the classic 'Hello World" app for the purpose of testing and debugging the app using Visual Studio 2012 and Microsoft Word 2013.

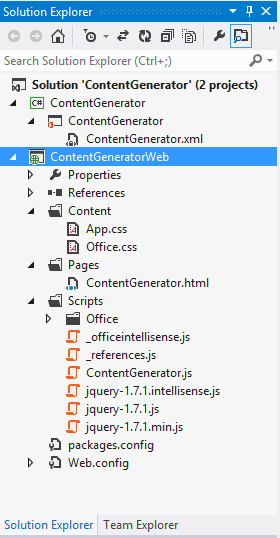
1. Open Visual Studio 2012.
2. From the **File Menu** select the **New Project** command. When the **New Project** dialog appears, select the **Apps for Office 2013** project template from the **Office/SharePoint** **> Apps** template folder as shown below. Name the new project **ContentGenerator** and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



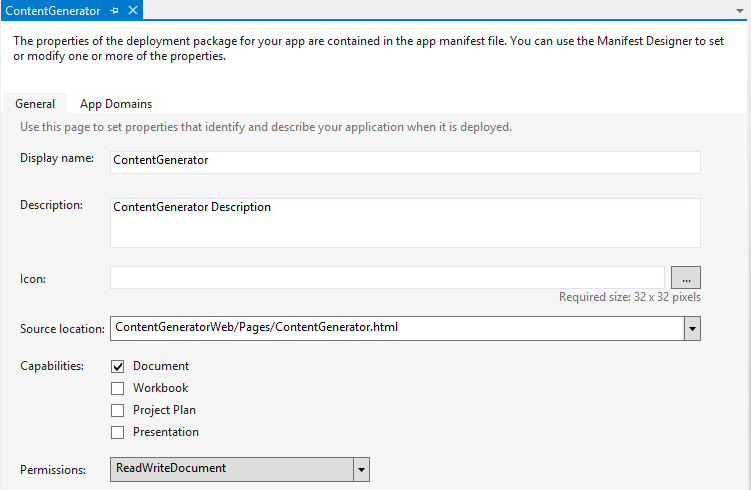
1. Fill out the next dialog as the one shown below in order to create a new task pane app that is designed to work just with Microsoft Word. Click **Finish** when you are done.



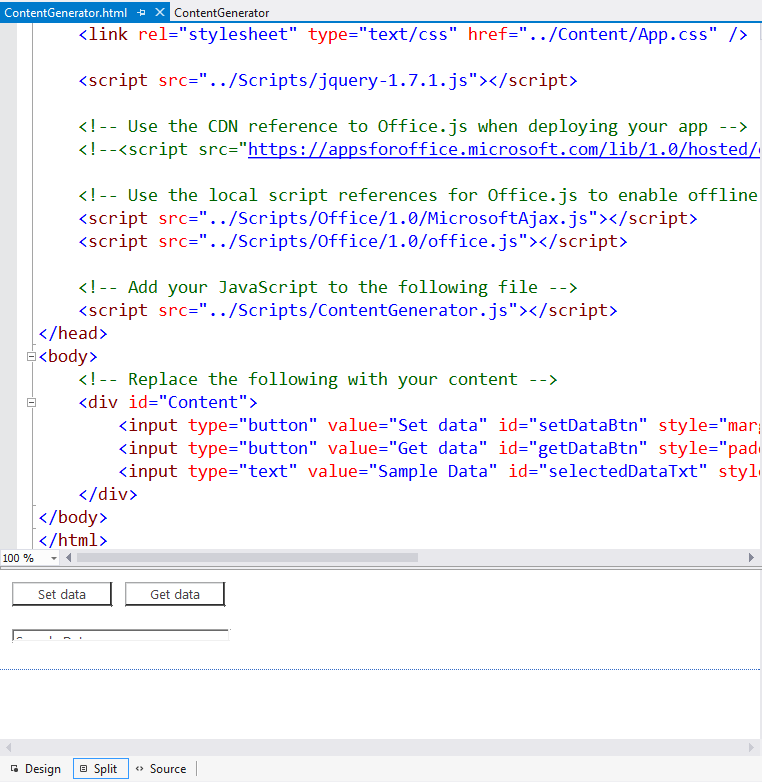
1. Once the solution has been created, take a moment and examine the two projects inside. The first project named **ContentGenerator** just contains an xml file for the app manifest named **ContentGenerator.xml**. The other project named **ContentGeneratorWeb** is the remote web project where you implement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



1. Look at the structure of folders and files inside the new project. You will notice that a top-level folder structure has been created with folders named **Content**, **Pages** and **Scripts**. You should also take note 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 **ContentGenerator.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 **ContentGenerator.xml**.
2. As you implement this first app over the next few steps, you will mainly be making modifications to three primary files that are named as follows:
3. **ContentGenerator.html**
4. **App.css**
5. **ContentGenerator.js**.
6. Open the HTML source file that defines the user interface for the app which is named **ContentGenerator.html**. Note that there is some pre-existing HTML that is supplied by Visual Studio when you create a new project. 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 **ContentGenerator.html**. Make sure you leave everything in the **head** section just as it is.

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8" />

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

<title>ContentGenerator</title>

<link rel="stylesheet" type="text/css" href="../Content/Office.css" />

<!-- Add your CSS styles to the following file -->

<link rel="stylesheet" type="text/css" href="../Content/App.css" />

<script src="../Scripts/jquery-1.7.1.js"></script>

<!-- Use the CDN reference to Office.js when deploying your app -->

<!--<script src="https://appsforoffice.microsoft.com/lib/1.0/hosted/office.js"></script>-->

<!-- Use the local script references for Office.js to enable offline debugging -->

<script src="../Scripts/Office/1.0/MicrosoftAjax.js"></script>

<script src="../Scripts/Office/1.0/office.js"></script>

<!-- Add your JavaScript to the following file -->

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

</head>

<body>

<!-- everything here has been deleted -->

</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\_ContentGenerator.html.txt** file located in the **StarterFiles** folder for this lab. After you have added the HTML code, save your work by saving the changes to of **ContentGenerator.html**.

<body>

<h1>Get a Quote</h1>

<div id="toolbar">

<input id="cmdGetContent" type="button" value="Get Content" />

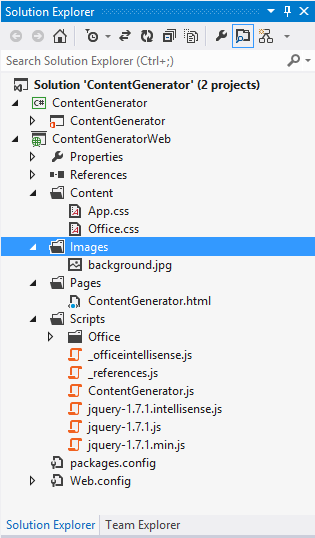
<input id="cmdInsertContent" type="button" value="Insert Content" />

</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, right-click and create a new folder **Images**. Add the image file **background.jpg** from the **StarterFiles** folder into your **Images** folder of your project.



1. Open the CSS file named **app.css**. You should see that there are no CSS rules that have been added yet. 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 lab. After you have added these CSS rules, save your changes to **app.css** and close this file.

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. Open the JavaScript source file named **ContentGenerator.js** and inspect the code inside. There is also an **Office.initialize** function which will be executed automatically when the app is loaded and initialized. Delete all the code from **ContentGenerator.js** and replace it with this following generic starting point.

Office.initialize = function (reason) {

$(function () {

// intialize code goes here

});

}

1. At the bottom of **ContentGenerator.js**, add a two new JavaScript functions named **cmdGetContent** and **cmdInsertContent**.

function cmdGetContent(){

}

function cmdInsertContent(){

}

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 () {

$("#cmdGetContent").click(cmdGetContent);

$("#cmdInsertContent").click(cmdInsertContent);

});

}

function cmdGetContent(){

}

function cmdInsertContent() {

}

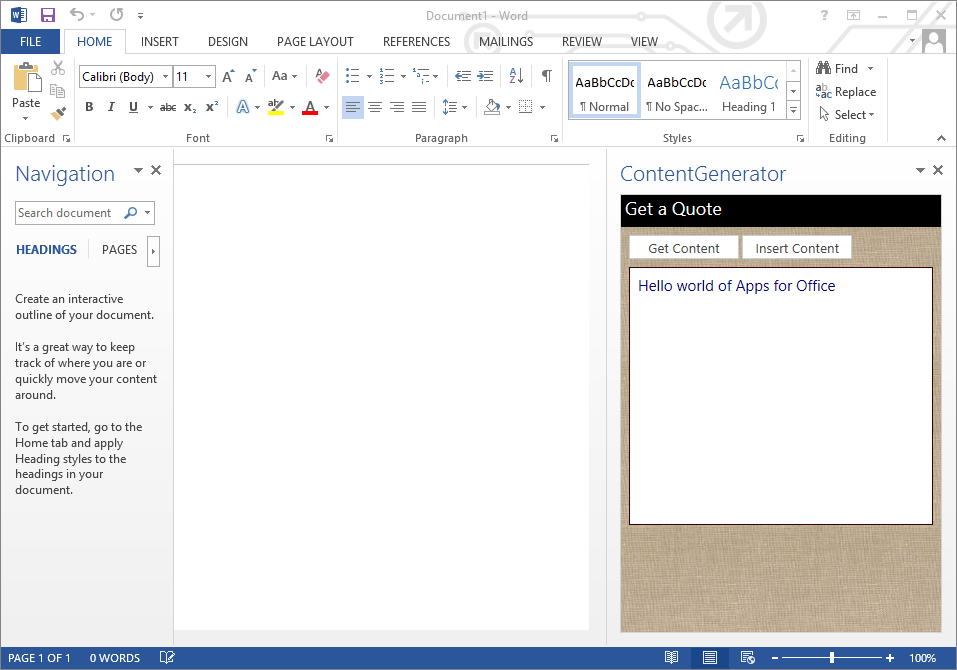
1. Write the implementation of the **cmdGetContent** function to acquire a reference to the HTML div element with the id of **display** using jQuery syntax and then to update its text to the value of **"Hello world of Apps for Office!"** or something equally as festive using the jQuery **text** method.

function cmdGetContent() {

$("#display").text("Hello world of Apps for Office");

}

1. You have now added enough functionality where you can test your project. Press the **{F5}** key in Visual Studio 2012 to begin debugging the project. When you press the **(F5)** key, Visual Studio should automatically launch Microsoft Word. You should see the app initialize inside the Word task pane. When you click the **Get Content** button, you code should run and update the user interface inside the app with your message.



1. Once you have seen how the user interface of a simple app looks and you have tested the Get Content button, quit Microsoft Word to stop the debugger and return to your project in Visual Studio 2012.

# Exercise 2: Writing Content to a Word Document

In this exercise, you will update the app to generate content and add it to the currently active document in Microsoft Word.

## Task 1 – Developing an App for Office 2012 with Visual Studio

In this task, you will using the Office JavaScript API to add content to the current document.

1. Ensure you have Visual Studio 2012 opened and that you have the **ContentGenerator** project open which you created in the previous exercise.
2. Open the main JavaScript course file named **ContentGenerator.js**.
3. Add a new JavaScript function named **getQuote**. You can either use the implementation shown below or you can copy-and-paste the function from the snippet file named **getQuote.js.txt** located in the **StarterFiles** folder for this lab.

function getQuote() {

var quotes = [

"I’d rather have an bottle in front of me than a frontal lobotomy.",

"Behind every great man is a woman rolling her eyes.",

"Between two evils, I always pick the one I never tried before."

];

var index = Math.floor(Math.random() \* quotes.length);

return quotes[index];

}

1. Update the **cmdGetContent** function to get its content using a call to **getQuote**.

function cmdGetContent() {

// display quote inside Office App

$("#display").text(getQuote());

}

1. Update the **cmdInsertContent** function to retrieve the current quote shown in the **display** div and write the quote into the select region of the currently active Word document. You can accomplish this using the following code.

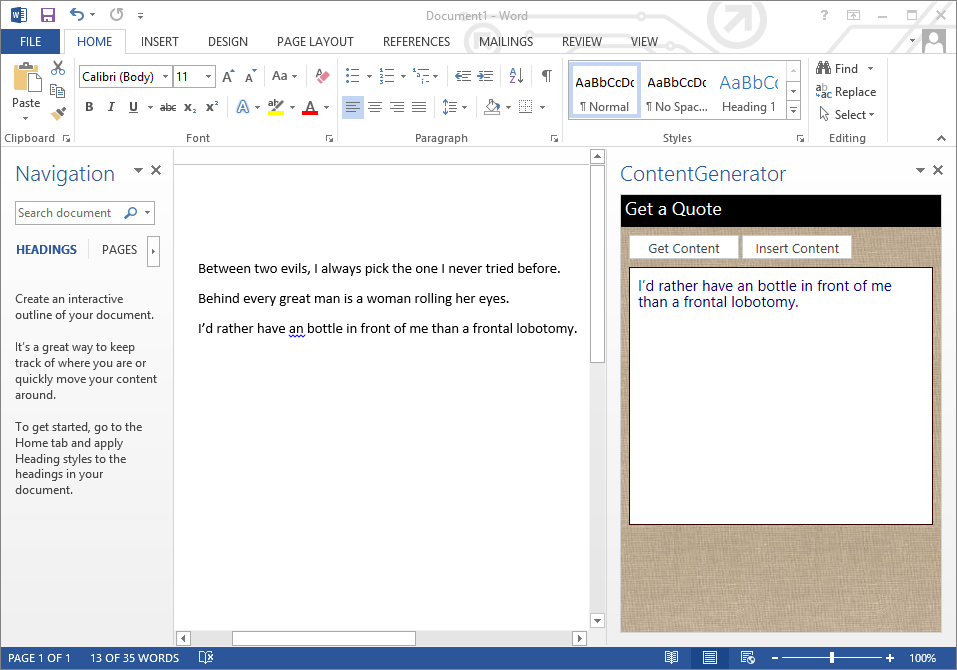
function cmdInsertContent() {

var currentQuote = $("#display").text();

Office.context.document.setSelectedDataAsync(currentQuote, {}, function () { });

}

1. Test your work by pressing the **{F5}** in Visual Studio 2012 to run the debugger and launch Microsoft Word. Now when you click the **Get Content** button, you should see it retrieves a quote and displays it in the app's display area. If you press the **Get Content** button several times in a row, it should randomly select a different quote each time and display it in the app.
2. Now click the **Insert Quote** button. It should retrieve the quote that is currently displaying in the app's **display** div and insert it into the current Word document at the location of the cursor or the current selection.
3. Press the **{F5}** the to test the app one more time. You should be able to click the **Get Content** button to see different quotes. When you see a quote you like, you should be able to click the **Insert Content** button to write that quote to the location of the current selection or the insertion point in the current Word document.



1. At this point, you have complete all the steps for this lab exercise.