# LightSwitch HTML Client Tutorial

*LightSwitch for Visual Studio 2012 HTML Client Preview*

The new HTML5 and JavaScript-based client is an important companion to our Silverlight-based desktop client that addresses the increasing need to build touch-oriented business applications that run well on modern mobile devices.

In this tutorial, we'll build an application that connects to existing data services and provides a touch-first, modern experience for mobile devices. To help ground the tutorial, we've created a fictional company scenario that has a need for such an application.

## Contents



## Helpful resources

As you walk through this tutorial, please bear in mind that there are useful resources available to help you should you get stuck or have a question [the following are external links]:

- LightSwitch HTML Client forum to post feedback and ask questions, and check back for any corrections to the walkthrough document.
- HTML Client Resources Page on the Developer Center for more learning resources.

## The MoveIt Application

MoveIt is an application that's used by *Contoso Movers, Inc* to take the inventory of customers' residences prior to moving. The data collected via the application helps *Contoso Movers* determine the resources required to move a particular client's belongings—how many trucks, people, boxes, etc. need to be allocated. The application is comprised of two clients that serve distinct business functions:

1. **Schedulers use a desktop application** to service new customer requests and create appointments. This application is a rich desktop application primarily geared towards heavy data entry with the keyboard and mouse, since Schedulers are on the phone with customers a lot and need to enter quite a bit of information into the system during the course of a day.
2. **Planning Specialists use a tablet device** to quickly take inventory—on location—of each residence on the specialist's schedule for the day. Taking inventory involves detailing each room in the residence, its size and entry requirements (if any), and listing its contents. Pictures are often taken of each room so the movers have a point of reference when they arrive. Secondarily, Planning Specialists may make notes about parking restrictions for the move team (i.e., where they can park the truck during the move).

This tutorial walks through building out the mobile client used by *Contoso Movers'* planning specialists. Before we do that, let's run through a complete version of the application that is already published on this VHD to get a better feel for what we'll build.

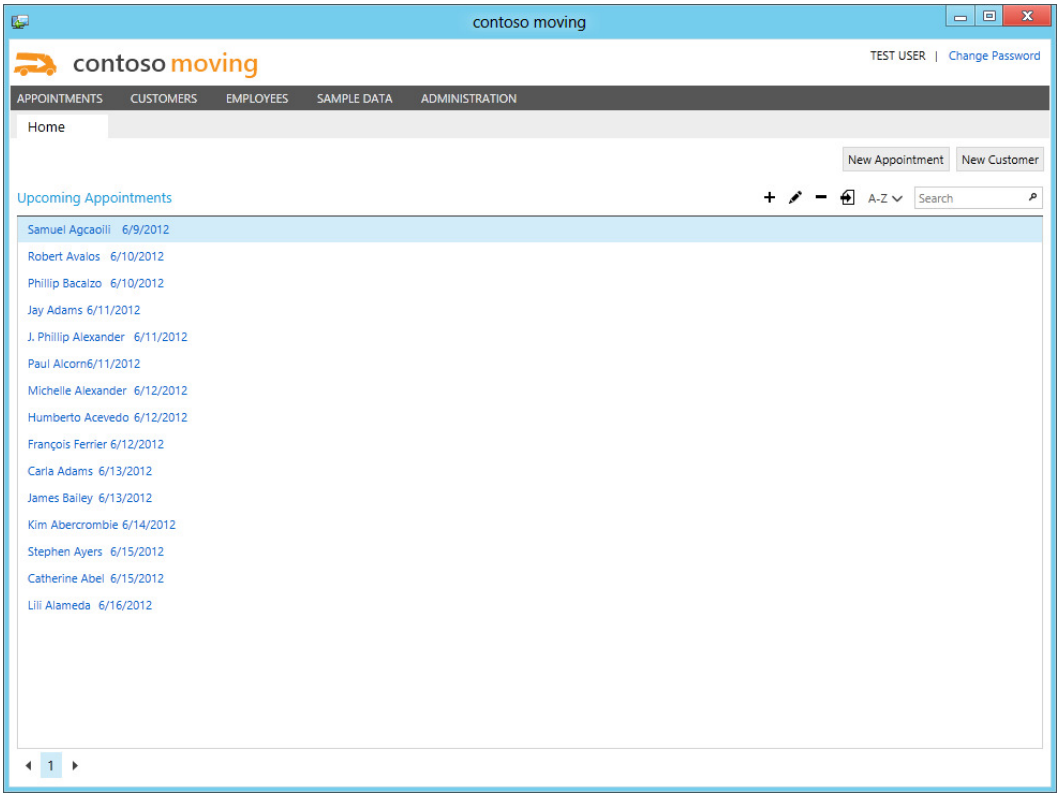
### Run the Desktop Client for Schedulers

First, let's look at the rich desktop client used by Schedulers, since that is how most of the data in the system originates. The desktop client can be run by entering the following URL in your browser: http://localhost/ContosoMovingComplete/Client. If you'd like to reach the application from another machine, the URL is http://<*machine-name*>/ContosoMovingComplete/Client, where <*machine-name*> is the machine name you specified when the VHD was first set up.

The application uses Forms Authentication. Enter the following credentials in the app's login screen, and then choose **Log In**:

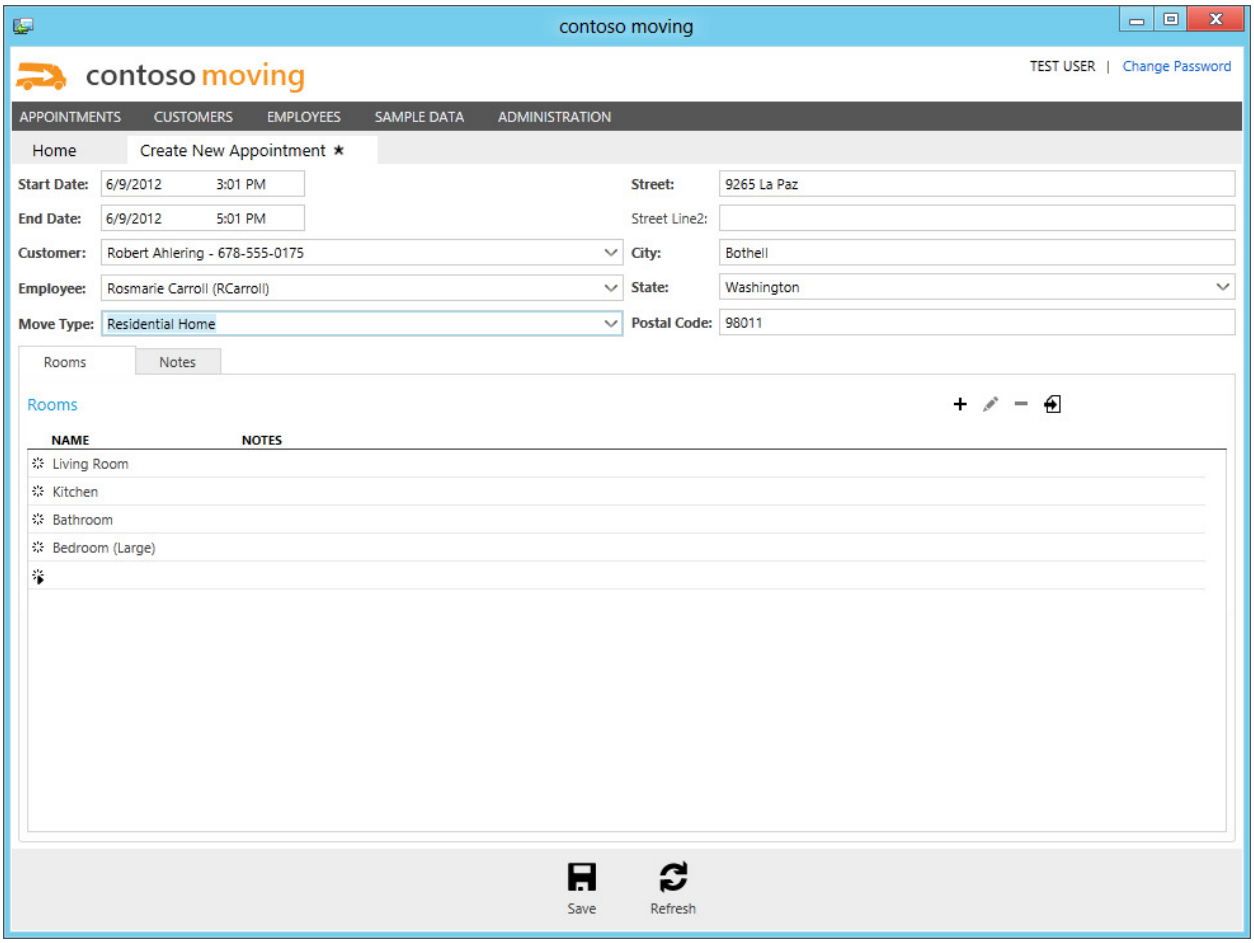
- **User Name**: admin
- **Password**: admin!!

When the application is first launched, upcoming appointments are displayed in the home screen. Two key entry points—**New Customer** and **New Appointment**—are provided for entering the details of a new customer and setting up new appointments.




Choose the New Appointment button to create a new appointment. The **Create New Appointment** screen is used to capture information about the appointment such as preferred date/time, address, and contact information. Typically, a customer will provide enough information over the phone to create a rough profile about the move; for example, is the move for a residential home, apartment, or office building, the rooms involved in the move, and any special instructions or questions that the Planning Specialist should follow up with the customer about.

Feel free to add several of your own appointments in the system, as well as explore other screens accessible from the menu bar.




## Run the Mobile Client for Planning Specialists

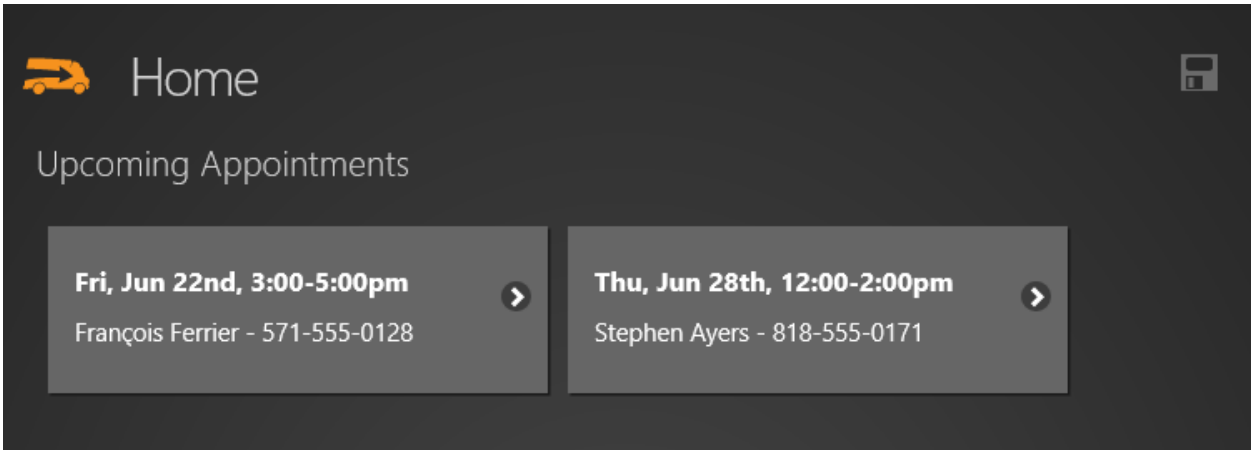
The VHD also contains a pre-built version of the mobile client that Planning Specialists use - this is the client we'll primarily focus on throughout this tutorial. You can run the pre-built mobile client by opening up a browser to http://localhost/ContosoMovingComplete/MobileClient/. If accessing the site from a mobile device, substitute <machine-name> for localhost. **Note:** If you run into problems loading the mobile client from Metro Internet Explorer, you may need to configure IE's Intranet settings – refer to the Getting Started guide for the LightSwitch HTML Client Preview VHD.

The app requires a username and password. You may use any of the following:

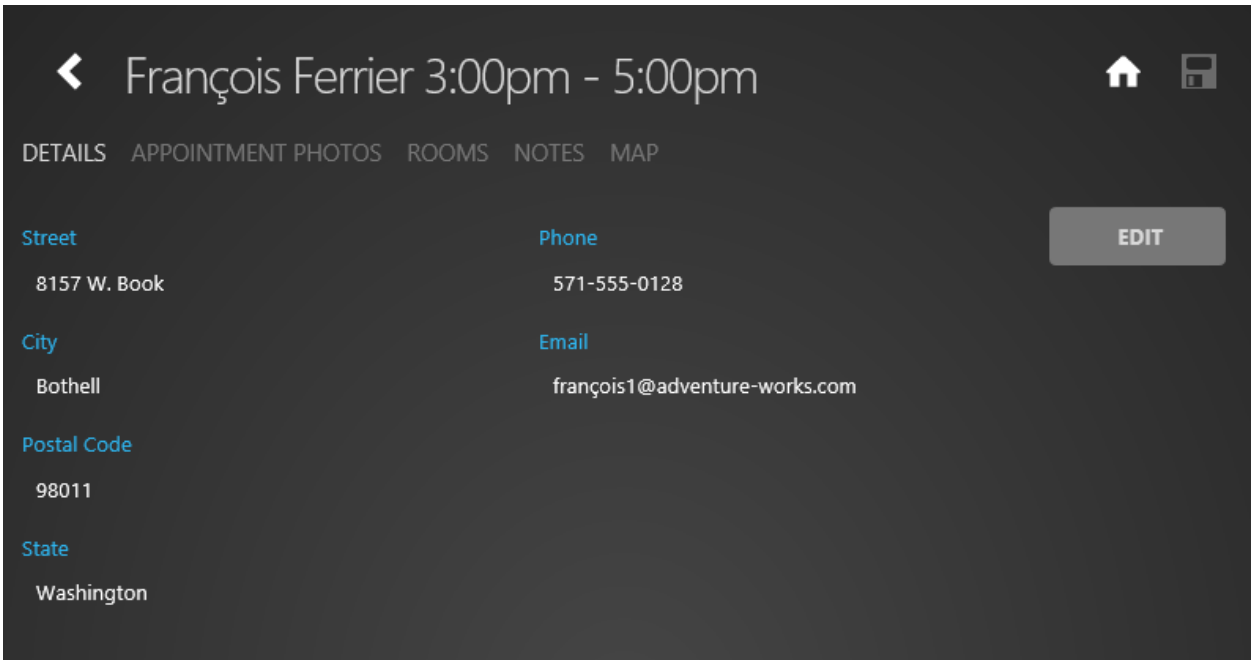
| Username | Password |
|---|---|
| ACarothers | pass@word1 |
| Dcarreras | pass@word1 |
| RCarroll | pass@word1 |
| JCastellucio | pass@word1 |

If you don't get prompted for login credentials, your browser may be caching your previous log-in to the Silverlight client. If so, please be sure to close all of your browser windows, and/or clear the cache if necessary.

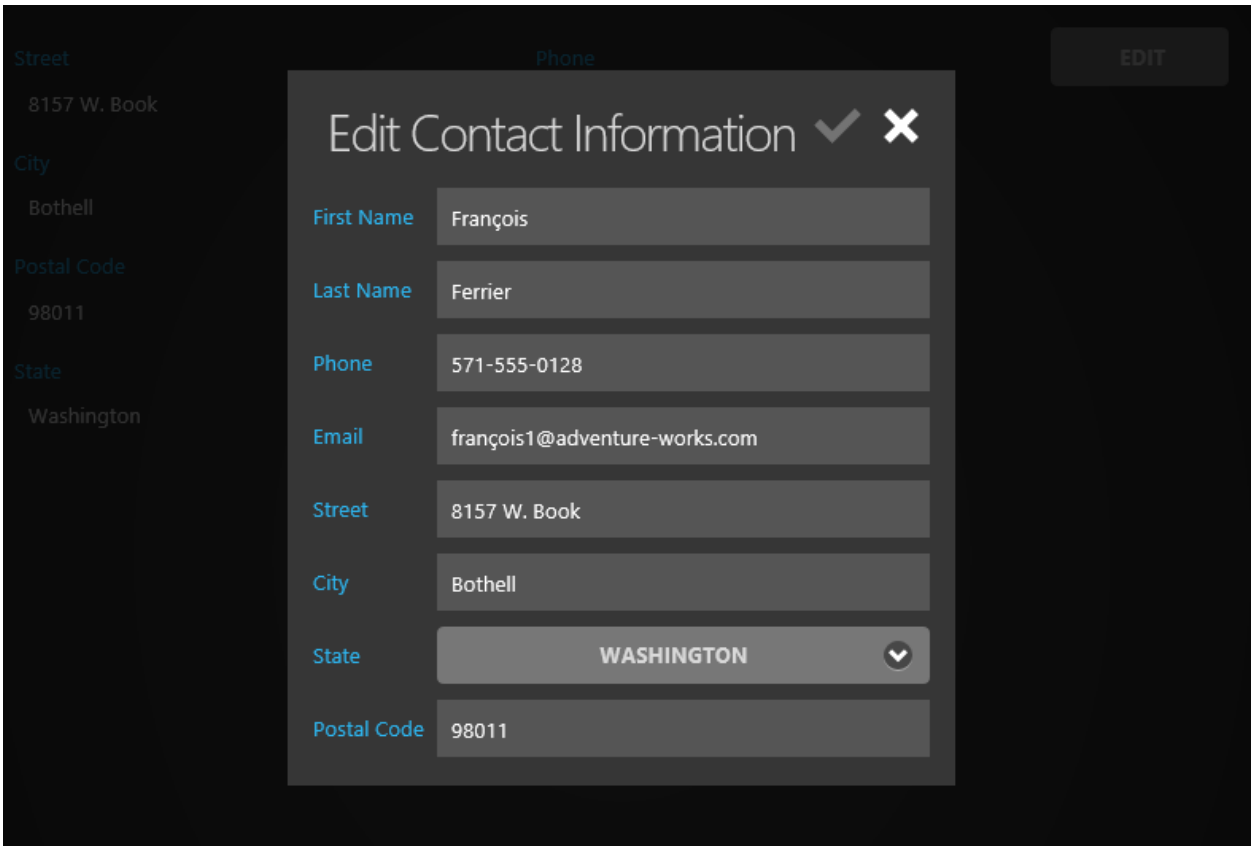
On launching this client, you'll immediately notice that the experience is personalized—only the logged-in Planning Specialist's appointments are displayed—and the UI style is optimized for navigating the application with your finger.  Also, very little chrome is displayed in order for the content to be the primary focus within the application.




Tapping an appointment tile will open the **Appointment Details** screen.  This screen is the nucleus of the application—it's where Inventory Specialists will spend most of their time. As such, the screen has been constructed to facilitate the most common tasks: view the customer information and taking inventory. You can always navigate back to the previous screen by tapping the < glyph to the left of the screen title, or by simply using the browser's back navigation button.




Editing customer information is also an infrequent but necessary task. Tapping the **Edit** button in the **Summary** section displays a modal dialog where the Planning Specialist can modify customer information.




## Implement the MoveIt App

Now that we've seen the app in action, let's build our own version. Topics we'll cover throughout the rest of the tutorial are broken into 2 main parts:

**Part 1:**

- Step 1: Add a Companion HTML Client to an Existing Application
- Step 2: Define Core Screens and App Navigation
- Step 3: Customize the UI with CSS and JavaScript
- Step 4: Test the Application on a Tablet Device

**Part 2:**

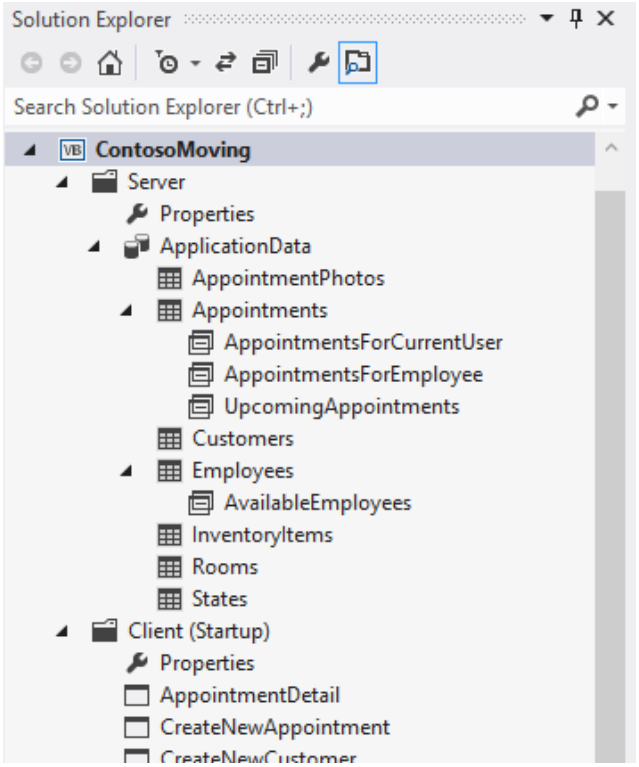
- Step 5: Create Modal Dialogs
- Step 6: Customize the Application's Theme
- Step 7: Using a Device's Built-in Camera to Shoot and Upload Photos
- Step 8: Add a Bing Map Custom Control
- Step 9: Republish the App and Test it on a Tablet Device

Let's get started!

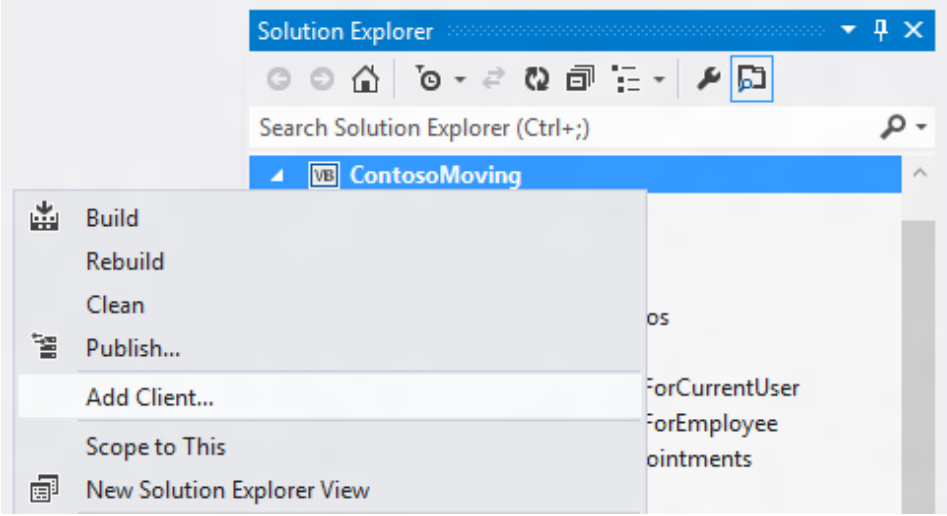
## Step 1: Add a Companion HTML Client to an Existing Application

1. Open the project located at **C:\Content\ContosoMoving-Start** on the VHD. If we look at **Solution Explorer** we see that the project already contains the core data service logic of the application (located under the **Server** node). The Silverlight desktop client used by MoveIt Schedulers has already been built (located under the **Client** node), and there is no mobile client yet.

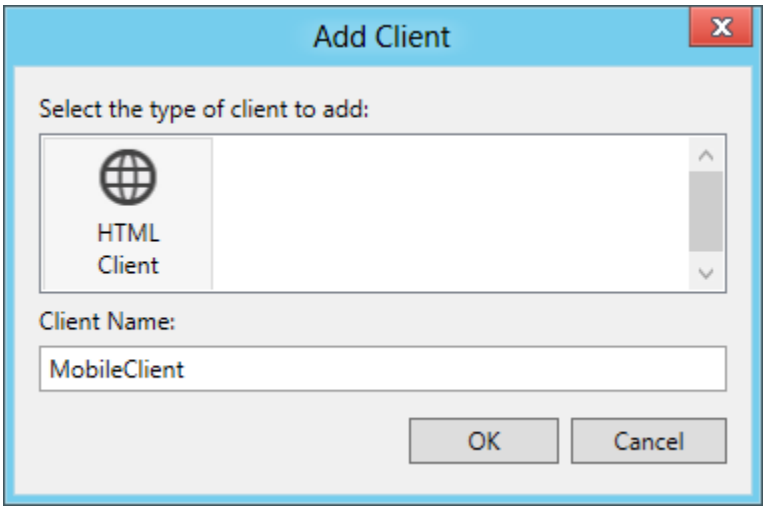
   Press F5 to run the Silverlight desktop client in debug mode.




2. Now let's add a new HTML client to the project. Exit debug mode, and in **Solution Explorer** open the shortcut menu for the **ContosoMoving** root node and choose **Add Client.**

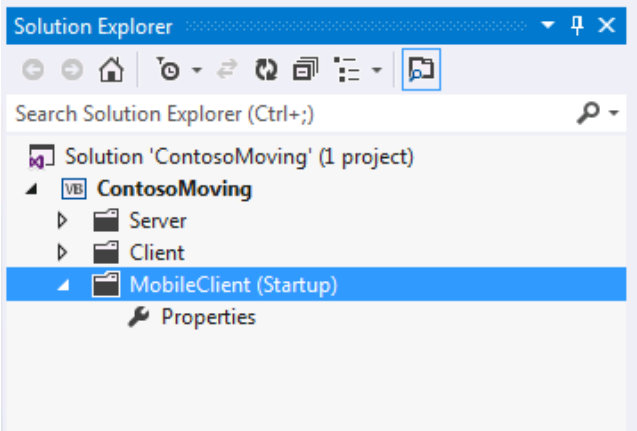



3. In the **Add Client** dialog, enter a unique name, for example “MobileClient”.




**Note:** If a dialog box pops up and prompts you to get a Developer License for Windows 8, choose **Cancel** – you don’t need it for this tutorial.

After choosing **OK** in the **Add Client** dialog box, the new client appears in **Solution Explorer** and is automatically designated to be the startup client. The startup client setting simply specifies which client launches when you press F5; this can be changed at any time by opening the shortcut menu for a client node in **Solution Explorer** and choosing **Set as Start-up Client**.



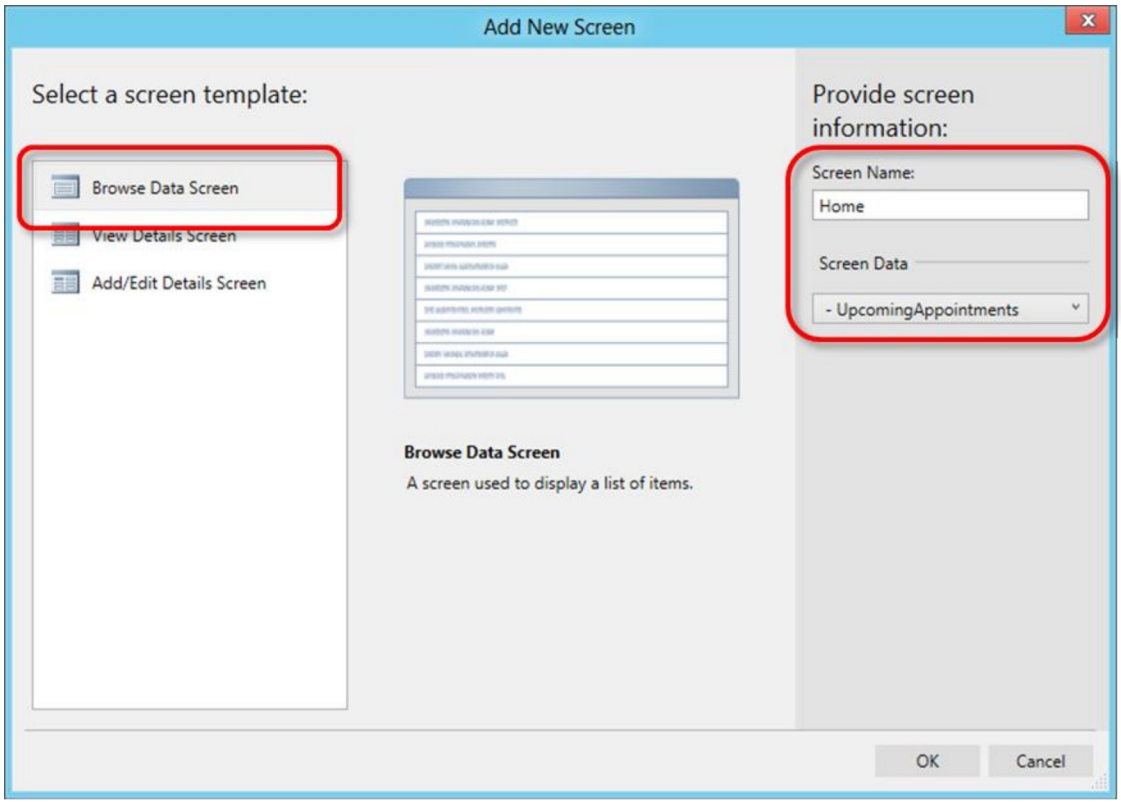

## Step 2: Define Core Screens and App Navigation

Now let's add some screens for our new mobile client. For those who have used LightSwitch before, the next set of steps should feel very familiar.

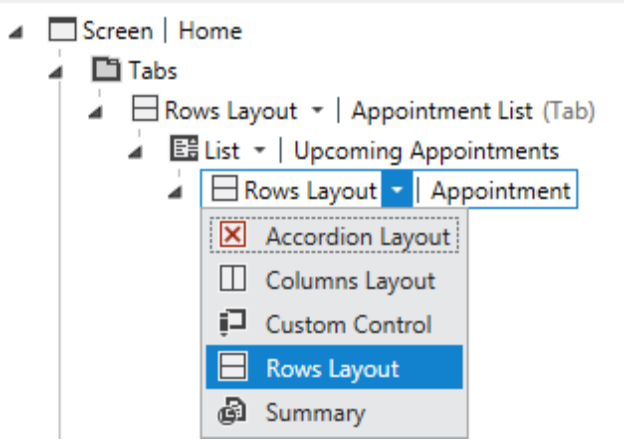
### Create the Home Screen

Let's make our home screen display a list of upcoming appointments for the currently logged-in user.

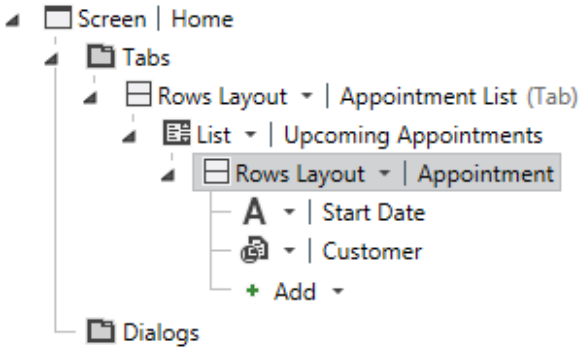
4. In **Solution Explorer**, open the shortcut menu for the **MobileClient** node and choose **Add Screen**. The **Add New Screen** dialog box opens.
5. In the **Add New Screen** dialog box, specify the following:
    - Screen Template: **Browse Data Screen**
    - Screen Name: **Home**
    - Screen Data: **UpcomingAppointments** (this query is already defined as part of the Server project – it filters for upcoming appointments based on the currently logged-in user).



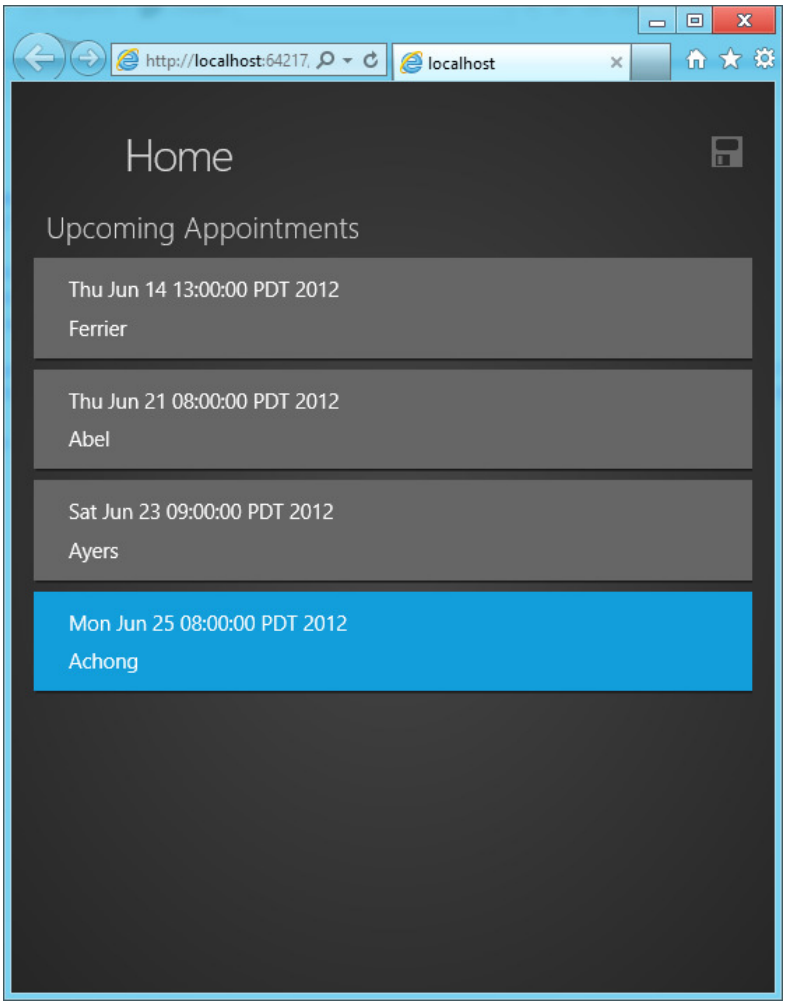

6. Let's make some simple modifications to how each Appointment is displayed in the list.
    a. First, change the **Appointment** item's control to use **Rows Layout**.




    b. Secondly, let's opt to display only the **Start Date** and **Customer** for each appointment item bydeleting the other fields. Screen Designer should now look like the following:



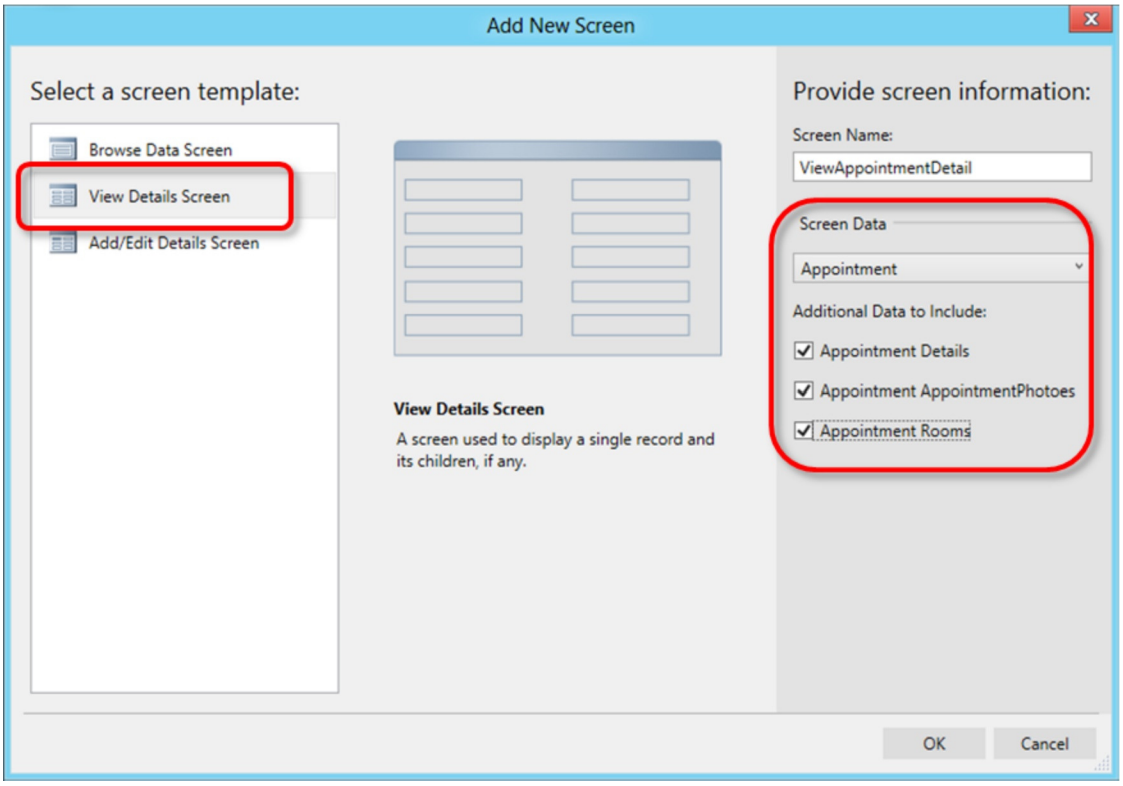

7. Press F5 to run the application. The default browser will launch and the home screen of the MobileClient will look something like the screenshot below. The date formatting isn't ideal, but this will do for now – later on, we will further customize the home screen.
8. Exit the debug session by closing the browser.



### Create the Appointment Details Screen

The appointment detail screen is the nucleus of the application—it's where the inventory specialists will spend most of their time.

9. Add a new screen with the following settings:
    - Screen Template: **View Details Screen**
    - Screen Name: **ViewAppointmentDetail**
    - Screen Data: **Appointment**. Also, select the check boxes to include all related data.

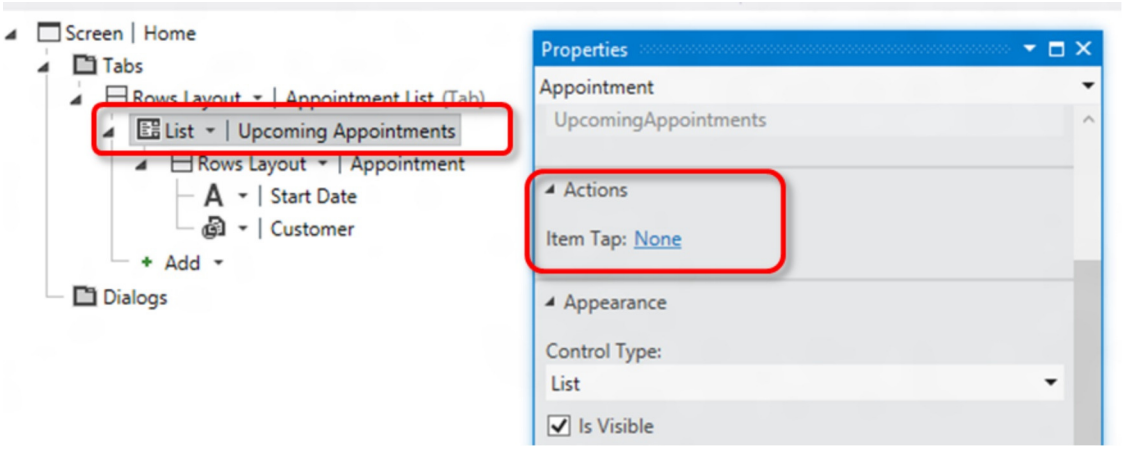



Looking at the Screen Content tree, you'll notice that several top-level screen sections have been created by default to better segment related data. Layout can easily be customized, but we'll stick with the defaults for now.

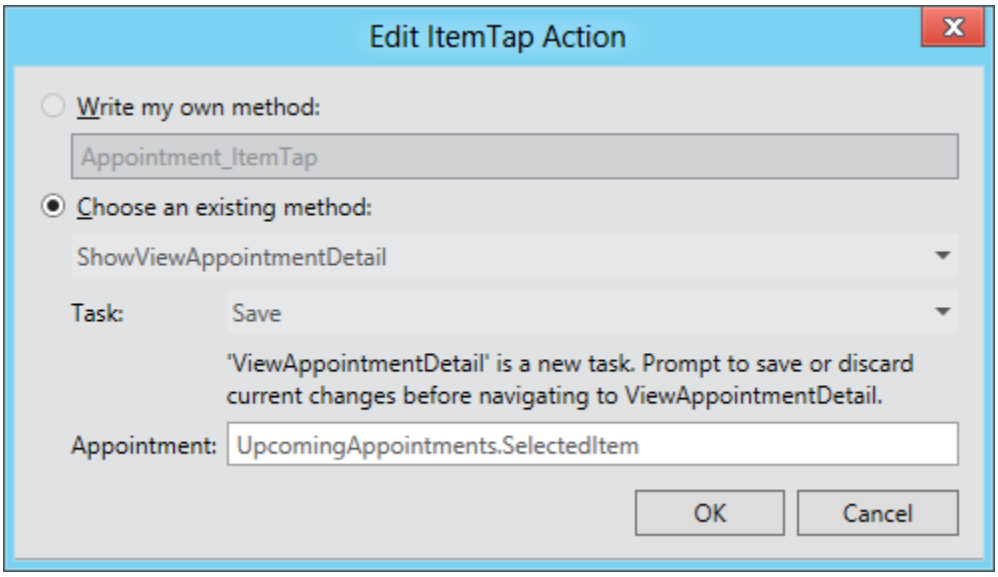
## Wire up Screen Navigation

Now all we need to do is wire up our two screens such that the application navigates to the **ViewAppointmentDetail** screen when an appointment is tapped in the **Home** screen.

10. In **Solution Explorer**, open the **Home** screen.
11. In the Screen Content tree, choose the **Upcoming Appointments** List control.
12. In the **Properties** window, choose the **Actions**, **Item Tap** hyperlink to configure what happens when a list item is tapped – the **Edit ItemTap Action** dialog box will appear.

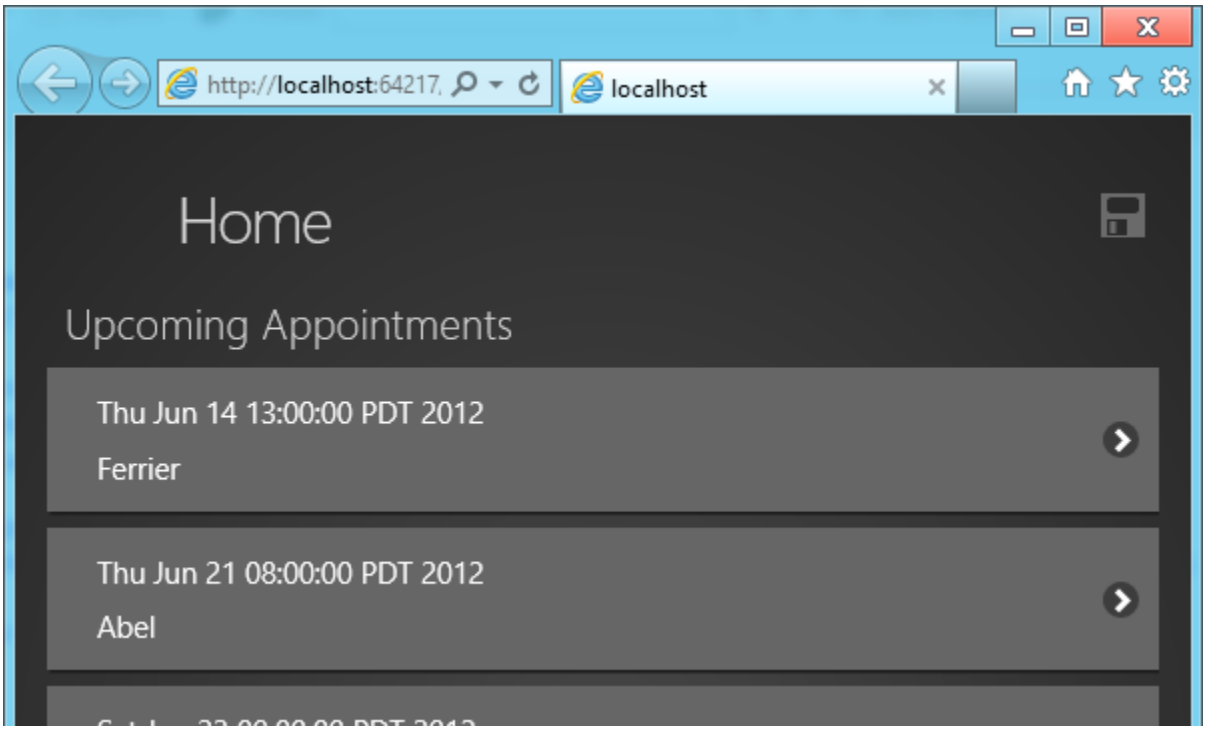



13. In the **Edit ItemTap Action** dialog box, configure the tap action as follows:

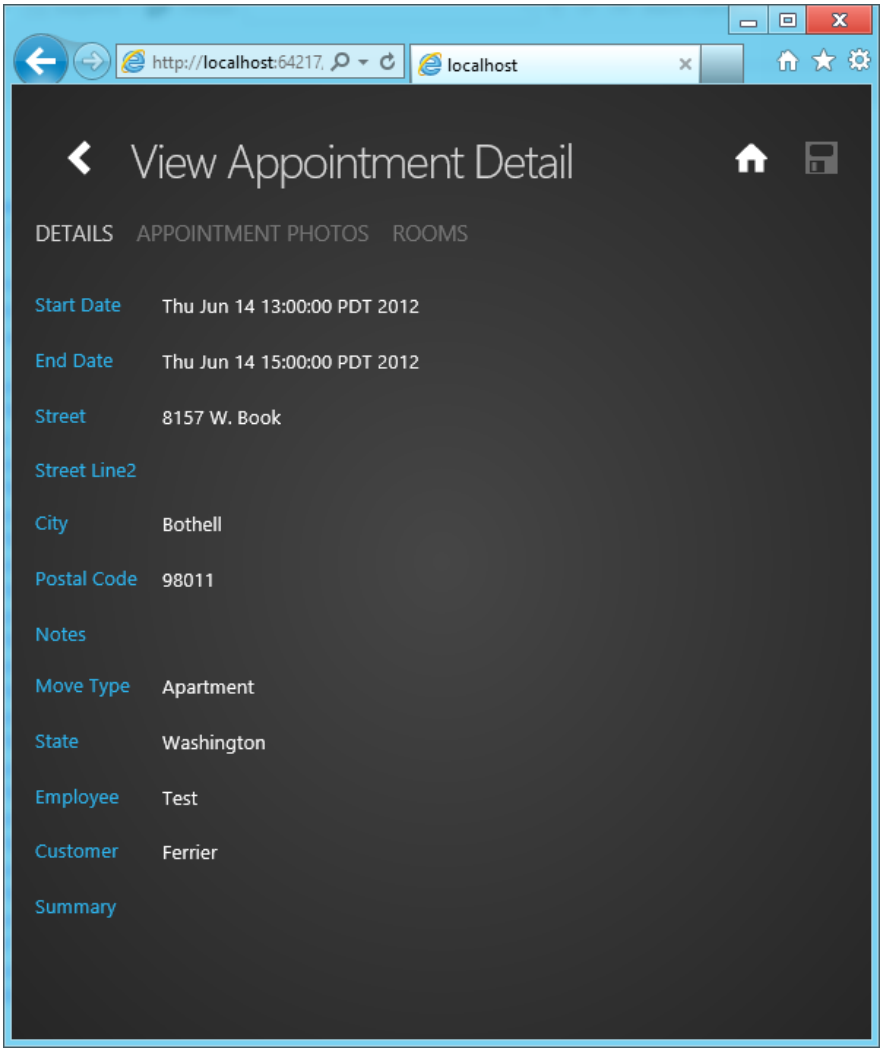



This states that we'll navigate to the **ViewAppointmentDetail** screen when a list item is tapped. Because the **ViewAppointmentDetail** screen was created using the **View Details** screen template, a screen parameter named **Appointment** was automatically defined to allow us to pass the appointment we want to view in that screen. Any other screen parameters would also be configurable here.

14. Let's hit F5 and run the application! You'll now see that each Appointment now has a glyph indicating that you can navigate to another screen.


http://localhost:64217/
localhost
Home
Upcoming Appointments
Thu Jun 14 13:00:00 PDT 2012
Ferrier
Thu Jun 21 08:00:00 PDT 2012
Abel

15. Tap an appointment to open its **Appointment Details** screen.




## Step 3: Customize the UI with CSS and JavaScript

Up until now we've mostly kept default UI and we have a functional app, but the UI can definitely be improved. Let's start addressing some of those improvements here to learn how your knowledge of HTML, CSS, and JavaScript can be applied to customize and extend your application's UI.

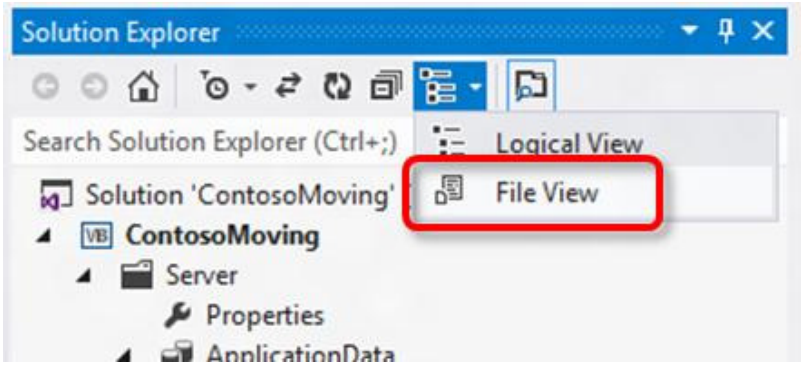
### Tile Layout using CSS

Appointments in the Home screen currently stretch out across the page. This may be ok for narrow displays, but it's a very inefficient use of screen real estate if the screen is wider (for example, consider a tablet device in landscape mode).

A common visualization for lists is to use a tile-based layout as we saw in the completed app. It turns out this is easy to do with custom CSS, which we'll now incorporate into our Home screen.

16. Add a .css file to the **MobileClient** project:

a. Switch from **Logical View** to **File View** using the **Solution Explorer** command bar.




b. Expand the **MobileClient** project, open the shortcut menu for the **Content** folder and choose **Add**, **Existing Item.**

c. Navigate to the C:\Content\Resources folder, choose the file named **tilelist.css**, and then choose **Add**. The tilelist.css file is added to the **Content** folder.

d. If you open tilelist.css you'll notice that it contains a straightforward definition of a CSS class.

```
ul.tilelist {
    width: 100%;
    padding: 10px;
    display:inline-block !important;

}

ul.tilelist li {
    list-style-type: none;
    margin: 6px;
    width: 300px;
    height: 100px;
    float: left !important;
}
```

17. Include the newly added .css file in **default.htm**.

e. Open the **default.htm** file in the **MobileClient** project.

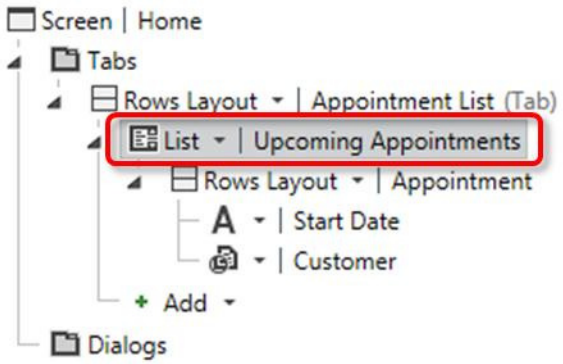
f. Insert the following statement just after the last **<link />** tag.

```
<link rel="stylesheet" type="text/css" href="Content/tilelist.css"
charset="utf-8" />
```

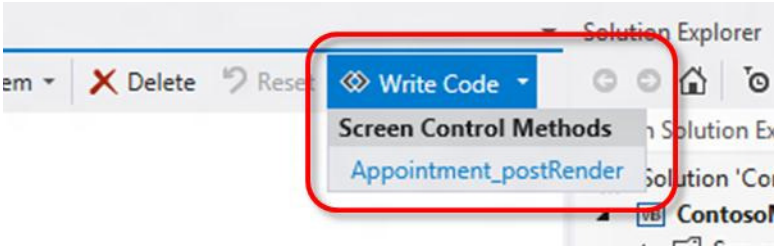
18. Customize the List control in the **Home** screen to programmatically add the tilelist CSS class in its **postRender()** event. The **postRender** event is a useful API entry point because it allows you to customize a screen element's HTML using standard JavaScript and jQuery syntax.

g. Switch back to **Logical View** and open the **Home** screen.

h. In the Screen Designer, select the **Upcoming Appointments** list control.




i. Choose the **Write Code** drop-down list located at the top-right of the Screen Designer window, and choose **Appointment_postRender**. This will open the JavaScript code editor.



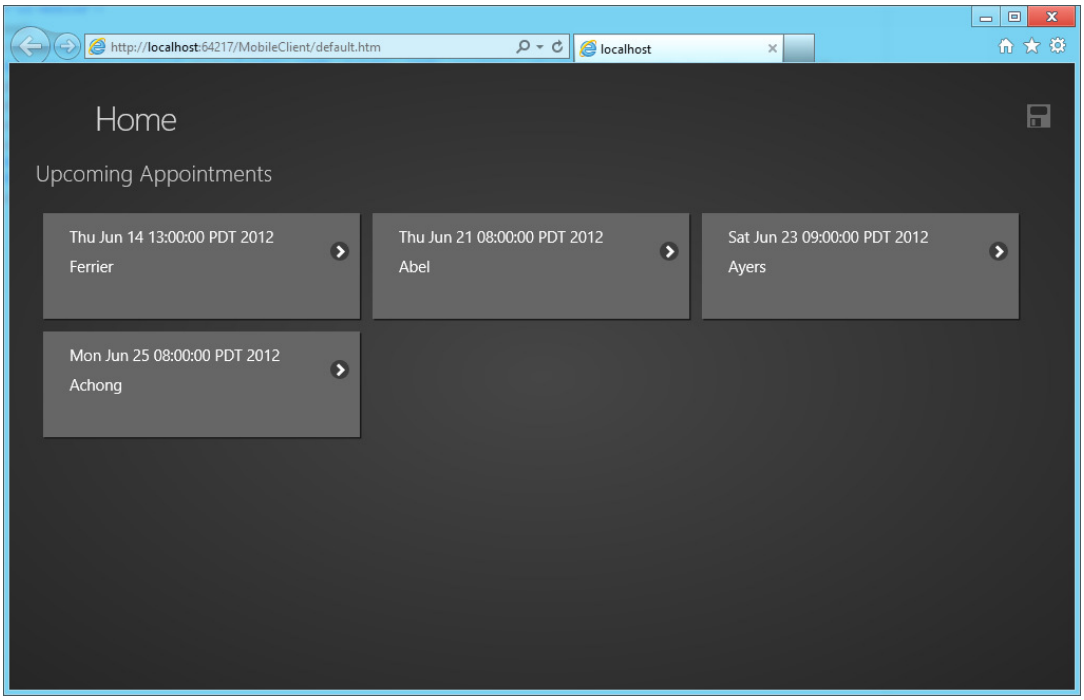

The generated **`Appointment_postRender()`** function stub has two parameters:

- **`element`**, which represents the HTML div element of the screen item;
- **`contentItem`**, which represents the actual screen's data object.

j. Insert the following line of code in the function. This is standard jQuery syntax that looks up the unordered list (ul) contained with the element, and adds the tilelist CSS class to it.

```
lightSwitchApplication.Home.prototype.Appointment_postRender = function
(element, contentItem) {
    $(element).find('ul').addClass("tilelist");
};
```

19. Press F5 – the home screen now displays upcoming appointments in a tile-based layout. Try narrowing and widening the width of the browser window – notice how the layout of the tiles dynamically adjusts to make good use of available horizontal space. This gives us an idea of how the list layout would change based on orientation of the mobile device, or how the screen would be rendered on small or large devices.

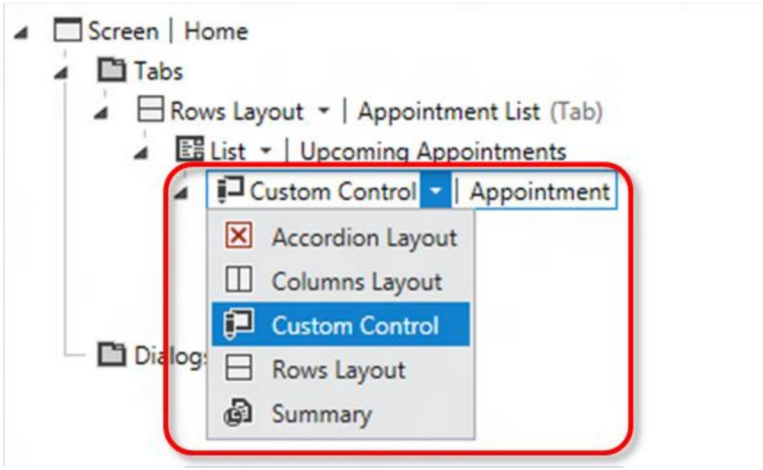



## Custom Formatting using Existing JavaScript Libraries

The Appointment items make better use of screen real estate, but the content within each tile clearly can be improved. Let's do two things: format the appointment date/time to be more legible, and include the customer's name and phone number.

We can customize how each list item is rendered by defining a custom control for the list item.

20. Exit the debug session and open the **Home** screen.
21. Select the **Appointment** item, and change its control from **Rows Layout** to **Custom Control**.




22. Choose the **Write Code** drop-down list and choose **RowTemplate_render**. This will open the JavaScript code editor.

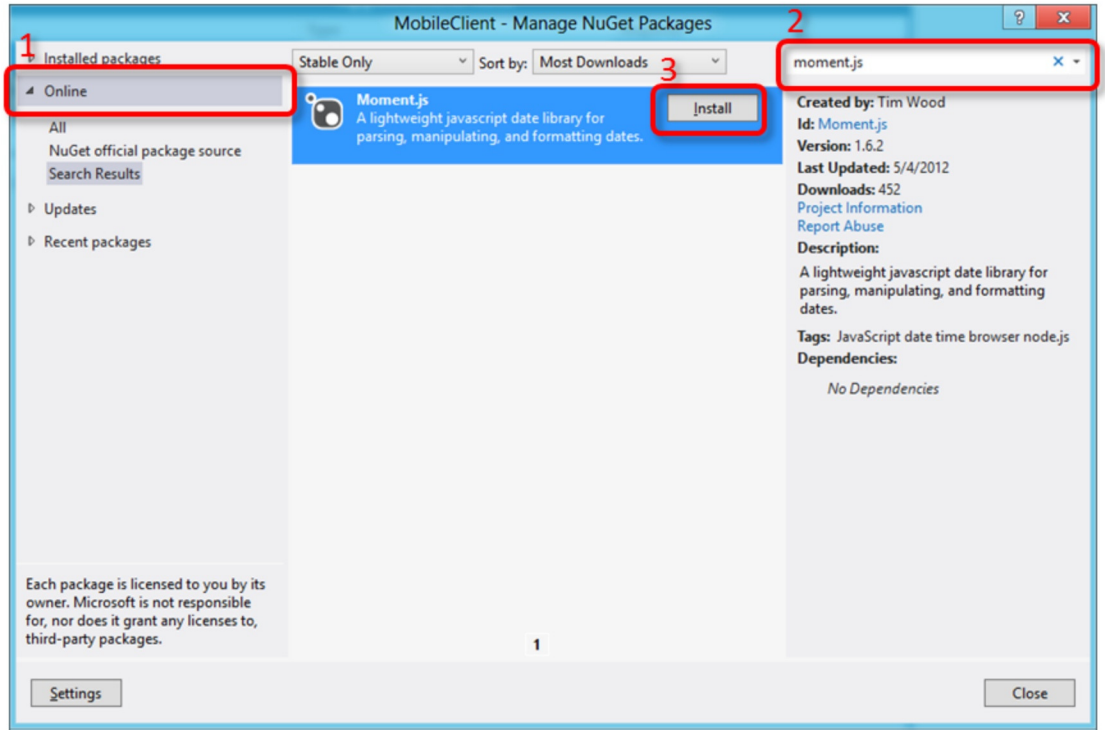
23. Insert the following code in the **RowTemplate_render** function. Again, the code in this example is standard JavaScript – it defines HTML content and adds it to an appointment's div element. In this function, **contentItem** represents the **Appointment** data item, so we can refer to the appointment's **StartDate** and **FirstName** fields by `contentItem.value.startDate` and `contentItem.value.firstName` respectively.

    **IMPORTANT:** Note the camelCasing for these fields – JavaScript is case-sensitive so it's important to keep the casing exactly as provided in the example below.

```
lightSwitchApplication.Home.prototype.RowTemplate_render = function (element,
contentItem) {
    var itemTemplate = $("<div> </div>");
    var title = $("<h3>" + moment(contentItem.value.startDate).format("ddd, MMM Do,
h:mm") + "-" + moment(contentItem.value.endDate).format("h:mma") + "</h3>");
    var subTitle = $("<span>" + contentItem.value.customer.firstName + " " +
contentItem.value.customer.lastName + " - " + contentItem.value.customer.phone +
"</span>");
    title.appendTo($(itemTemplate));
    subTitle.appendTo($(itemTemplate));
    itemTemplate.appendTo($(element));
};
```

    We need to do one more step – there are many JavaScript libraries available on the web that help parse and format dates. The code above that formats the appointment's **startDate** uses a JavaScript library called Moment.js that we'll need to include in our project. Any of the available JavaScript date libraries could be used for this example, but let's use Moment.js as it will give us the opportunity to use the NuGet management features inside Visual Studio 2012.

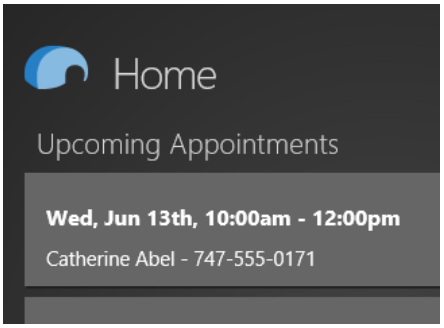
24. In **Solution Explorer**, switch to **File View**, open the shortcut menu for the **MobileClient** project and select **Manage NuGet Packages.**
25. In the **Manage NuGet Packages** dialog box, do the following:
    a. Choose the **Online** category (on the top-left).
    b. Enter "moment.js" in the Search textbox (top-right). Search results will display the NuGet package for Moment.js.
    c. Choose the **Install** button for Moment.js.




26. In **default.htm**, reference moment.js using the script tag (you can insert it as the last of the script reference tags).

```
<script type="text/javascript" src="Scripts/moment.js" charset="utf-8"></script>
```

27. Hit F5 to run the app!



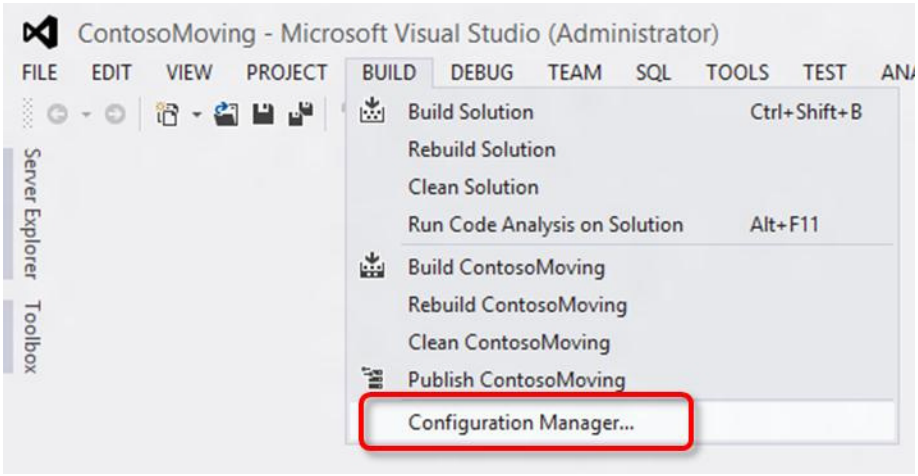

Hopefully you will have observed in these examples how standard HTML, JavaScript, and CSS – for which there is a vibrant community providing many resources – can be used to easily extend LightSwitch UI.

## Step 4: Test the Application on a Tablet Device

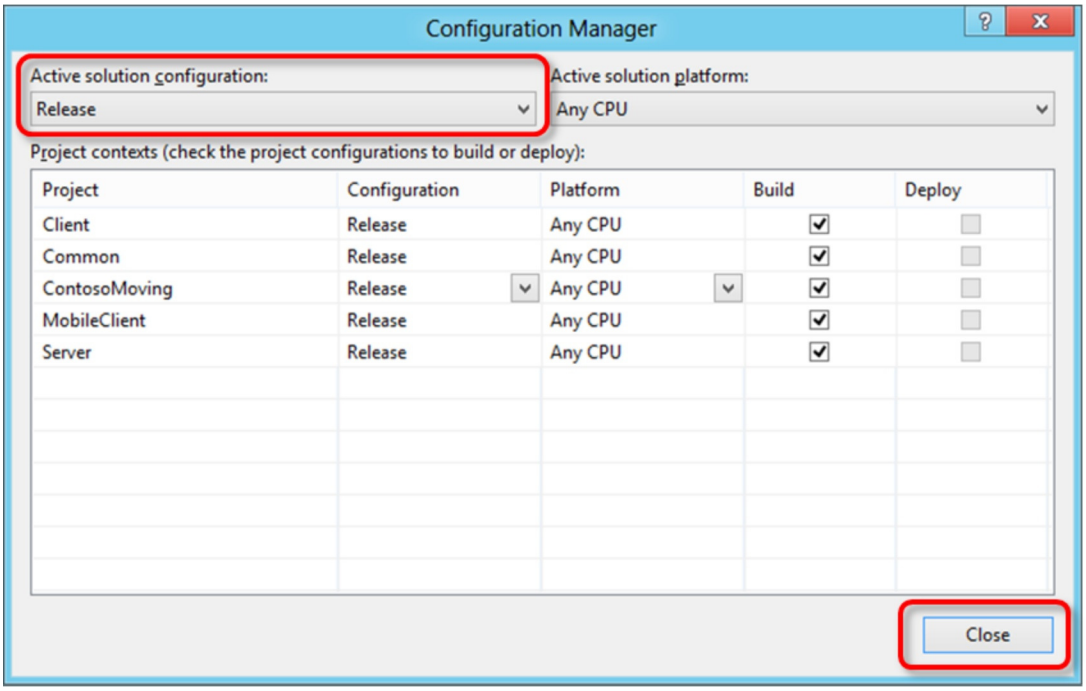
So far we have run the application in debug mode and seen it in the desktop browser. It'd be nice to publish the application and reach it from a mobile device's browser since that's what its designed to be used on.

The VHD comes preconfigured with IIS 8 and SQL Express, so we have everything we need to publish the application to the local server on the VHD – as long as your mobile device is connected to the same network you will be able to run the app from your device.

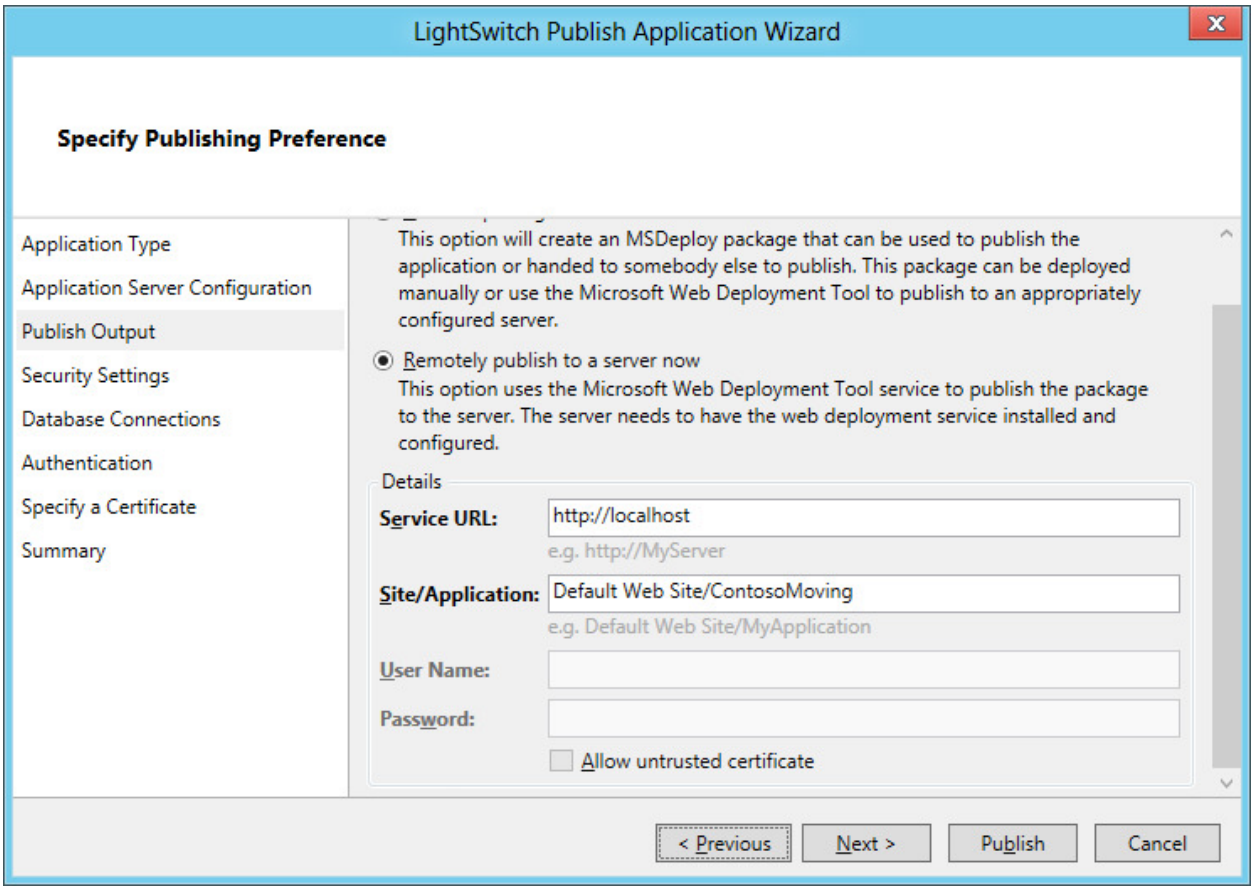
28. Before publishing, you must exit and re-open Visual Studio 2012 as an administrator. From the Windows Start screen, right-click on the Visual Studio 2012 tile, and select **Run as administrator**.
29. Once you have started Visual Studio 2012 as an administrator, load the LightSwitch project you just created by selecting it from the **Recent** menu section on the left.
30. On the menu bar, choose **Build**, **Configuration Manager**.




31. In the **Active solution configuration** drop down, choose **Release**, and then choose the **Close** button.



32. In **Solution Explorer**, open the shortcut menu for the LightSwitch project's root node and choose **Publish**.
33. In the **LightSwitch Publish Application Wizard**, set the following values:
    a. **Application Type**: Web
    b. **Application Server Configuration**: keep defaults.
    c. **Publish Output**:
        i. Choose "Remotely publish to a server now"
        ii. Service URL: http://localhost
        iii. Site/Application: Default Web Site/ContosoMoving
        iv. Since you are publishing to localhost, the username and password is unnecessary. If you see a yellow exclamation mark next to the grayed-out username textbox, you may safely ignore it.




    d. **Security Settings**: Set **Require Secure Connection (HTTPS)** to **Off**.
    e. **Database Connections**: Choose the ellipsis(**…**) button next to the **Specify the user connection** text box. In the **Connection Properties** dialog box, specify:
        i. Server name: .\SQLExpress
        Note the dot (.) before the backlash.
        ii. Use SQL Server Auth:
            - Username: sa

- Password: passw)rd

iii. Enter a database name: Choose the **ContosoMoving** database from the dropdown list. This database is already pre-populated with users and appointments, so it will save you a lot of time to use this one rather than creating a new one.

iv. Choose the **OK** button.

v. In the wizard's **Database Connections** page, make sure **Publish database schema** is checked, and then copy the connection string from **Specify the user connection** to the **Specify administrator connection** text box, and the choose the **Next** button.

f. **Authentication:** Since the existing database is already pre-populated with data and usersSelect **No, an Application Adminstrator already exists.** (The pre-published admin account is username *admin*, password *admin!!*)

g. **Specify a Certificate**: Make sure the **Specify a certificate** check box is unchecked.

h. Choose the **Publish** button.

34. After publish succeeds, the **HTML mobile client** will be available at the URL: http://*<machine-name>*/ContosoMoving/MobileClient, where *<machine-name>* is the machine name you specified when the VHD was first set up. You should be able to access it from any supported browser on your PC (including the Metro IE 10), and from any supported mobile device, provided that it is on the same network.

    a. When prompted for log-in credentials, you can use any of the following username and password combinations in the application's sample Employee data:

| Username | Password |
|---|---|
| ACarothers | pass@word1 |
| Dcarreras | pass@word1 |
| RCarroll | pass@word1 |
| JCastellucio | pass@word1 |

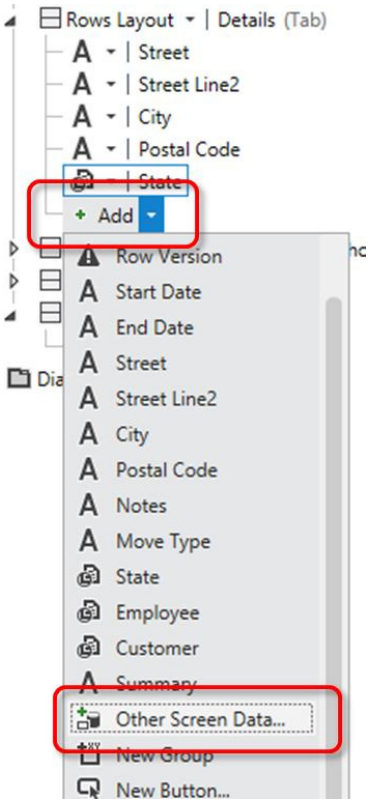
    b. Note: To log out and log in as another user, be sure to close all of your browser's windows. If necessary, you may need to clear your browser's cache.

## Step 5: Create Modal Dialogs

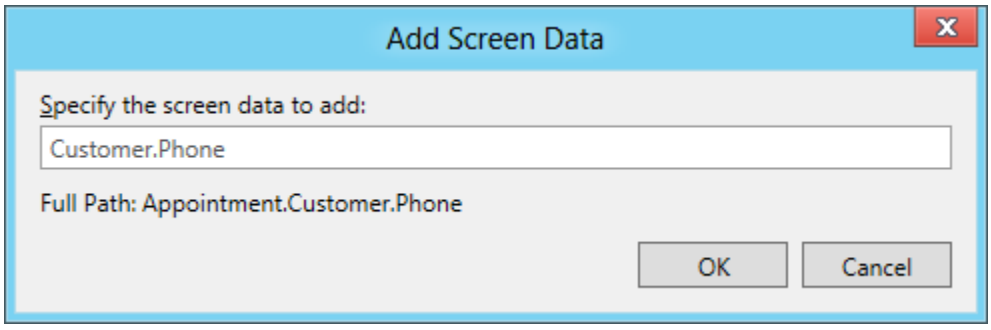
Let's now direct our attention to the **View Appointment Details** screen, where Planning Specialists spend most of their time. Because receptionists occasionally transcribe customer information incorrectly, Planning Specialists often verify contact information when they meet a customer and update it as needed. We'll add support to do this by defining a simple edit dialog.

35. Open the **View Appointment Details** screen.
36. Let's begin by cleaning up the UI in the main **Details** section (we'll later add more contextual data in the screen title, so this allows us to simplify the **Details** section):

a. Remove these fields: **Start Date**, **End Date**, **Notes**, **Move Type, Employee** (since it's always the current user in this application), **Customer**, and **Summary**.
b. Select the **Tabs** node, and then choose **Add Tab**.
c. In the **Properties** window, change the **Name** to "Notes".
d. In the **Add** dropdown for the **Notes** tab, choose **Other Screen Data**.
e. In the **Add Screen Data** dialog box, enter "Appointment.Notes", and then choose **OK**.
f. Change the control for **Notes** to be a **Text Area**. This will make it an editable field by default.
g. In the **Details** section, add the Customer's phone number and email:
    i. Choose **Other Screen Data** from the **Add** drop-down.

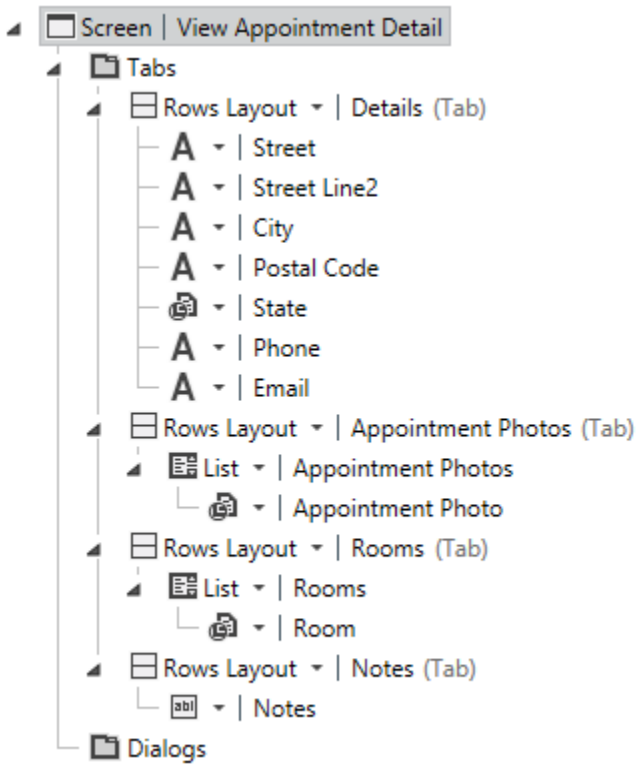



    ii. Type in the name of the field to add: **Customer.Phone.**

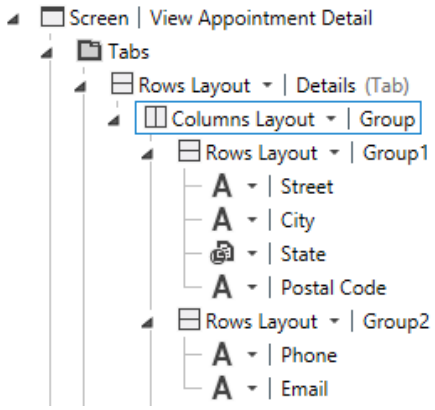



    iii. Repeat with **Customer.Email.**

The Screen Content tree should now look something like this:

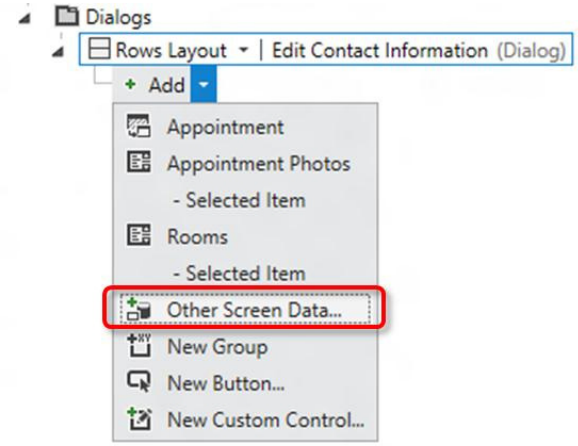


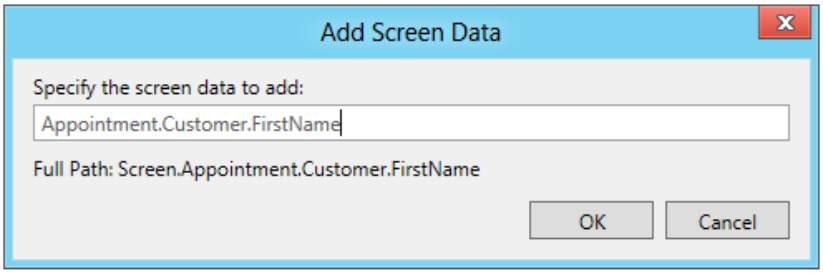

h. Bonus points: To make better use of screen space, break the data into two columns by using the **Columns Layout** control. Also, turn off attached labels for fields by selecting the root **Columns Layout** (called **Group** in the screenshot below) and setting its **Label Position** property to **Collapsed**.



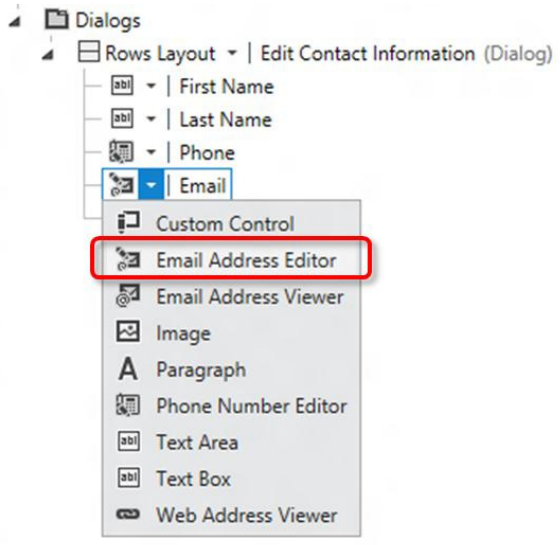

Now it's time to create our modal dialog. You may have noticed the **Dialogs** node in Screen Designer—as its name implies, modal dialogs are defined within this section. UI defined displays within pop-up visuals when invoked, but dialogs also have automatic data binding with OK and Cancel behavior built-in.

37. Under the **Dialogs** node, add a dialog, and name it **EditContactInformation**.
    a. Gesture to add **Other Screen Data**, and enter in **Appointment.Customer.FirstName** in the resulting dialog**.**

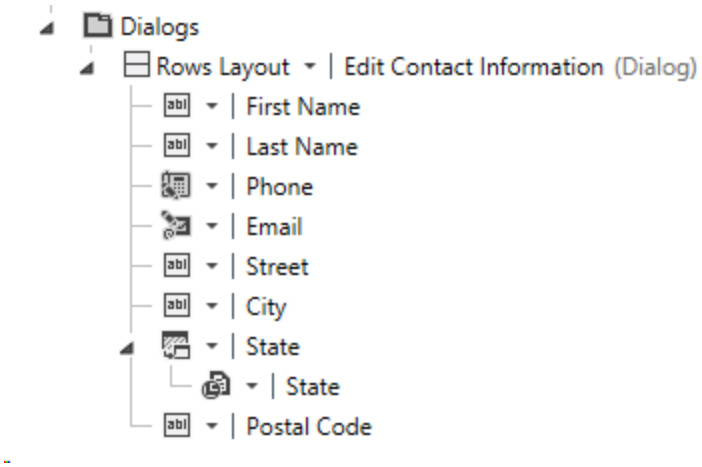



    b. Repeat the above with **Appointment.Customer.LastName**.
    c. Repeat the above with **Appointment.Customer.Phone** and **Appointment.Customer.Email**, setting the types of those fields to **Phone Number Editor** and **Email Address Editor**, respectively.

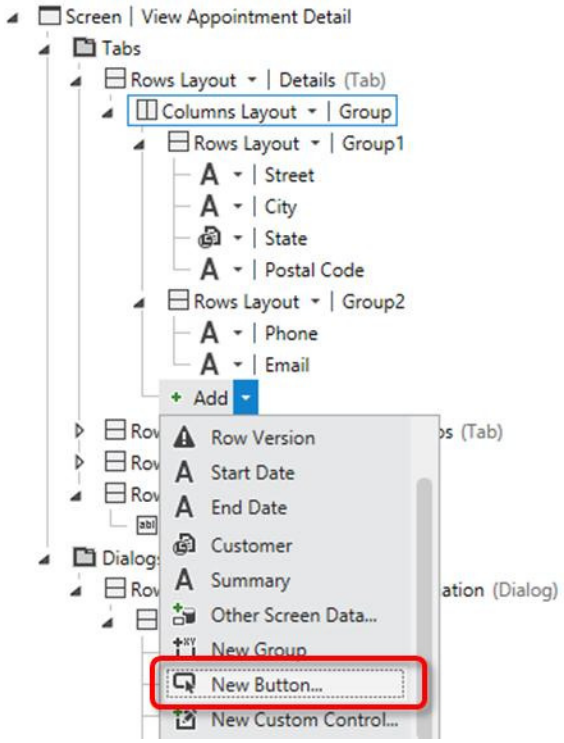



    d. Add **Street**, **City**, **State**, and **Postal Code** by dragging them from the **Appointment** sidebar into the dialog, or by adding them as above. The resulting dialog should look like this:

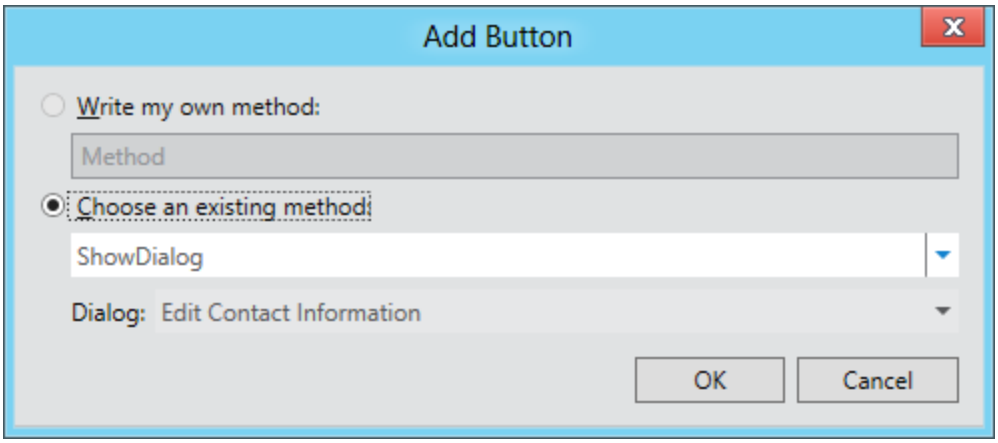



Now that we have a dialog, let's add a button that invokes the dialog.

38. Add a **New Button** in the **Details** section.




39. Configure the **Add Button** dialog as follows; this configuration shows the **ShowDialog** method referencing the **Edit Contact Information** dialog we just defined.




40. In the **Properties** window, name the button "Edit".
41. Press F5 and run the application. In the **Appointment Detail** screen, tap the **Edit** button to invoke the dialog – note how OK and Cancel behavior is provided as a built-in capability.

## Update the Appointment Screen to have a Dynamic Screen Title

Let's place contextual information in the screen title since we removed the **Start Date** and **Customer Name** fields from the **Details** tab.

42. Open the **View Appointment Detail** screen.
43. Choose the **Details** tab in the Screen Content tree.
44. Open the **Write Code** menu and choose **Details_postRender**.
45. Add the code below:

```
lightSwitchApplication.ViewAppointmentDetail.prototype.Details_postRender = function
(element, contentItem) {
    msls.bind(contentItem, "value.customer.firstName", function () {
        formatAppointmentScreenTitle(contentItem);
    });

    msls.bind(contentItem, "value.customer.lastName", function () {
        formatAppointmentScreenTitle(contentItem);
    });

    function formatAppointmentScreenTitle(contentItem) {
        contentItem.screen.details.displayName = contentItem.value.customer.firstName + "
" +
        contentItem.value.customer.lastName + " " +
        moment(contentItem.value.startDate).format("h:mma") + " - " +
        moment(contentItem.value.endDate).format("h:mma");
    };
};
```

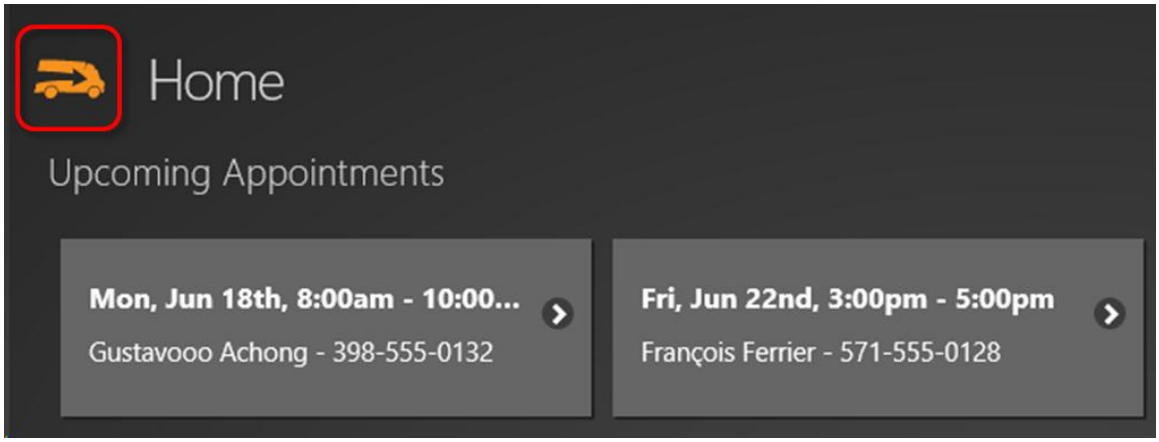
This example makes use of a LightSwitch API helper we haven't seen before: **`msls.bind()`**. This API provides a simple, yet powerful, means to data bind screen content to functions. In the code sample above, this means that whenever the data for **firstName** or **lastName** changes, the function **`formatAppointmentScreenTitle()`** will be reevaluated, thereby updating the screen title.

We can see this in action if we run the app and change the customer's name in the **Edit Appointment** dialog – notice how the screen title updates dynamically.

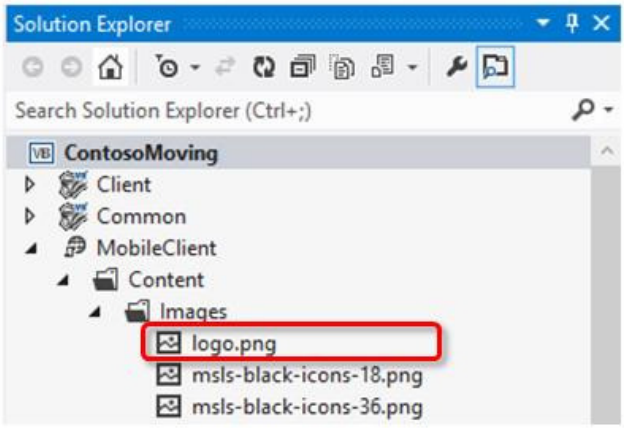
## Step 6: Customize the Application's Theme

### Add an icon in the header

One way to customize your application is to introduce a custom icon in the header:




46. In the **Solution Explorer**, switch to **File View**. Under the **MobileClient** node, expand the **Content** and **Images** nodes. You will see an default **logo.png** file.




47. Delete **logo.png**.
48. Open the shortcut menu for the **Images** folder and choose **Add, Existing Item**. Navigate to the c:\Content\Resources folder and choose **logo.png**. **Note:** You may choose any other PNG file to serve as a logo, but you will need to re-name it to **logo.png** once you add it to the project.

### Switch between "Dark" and "Light" themes

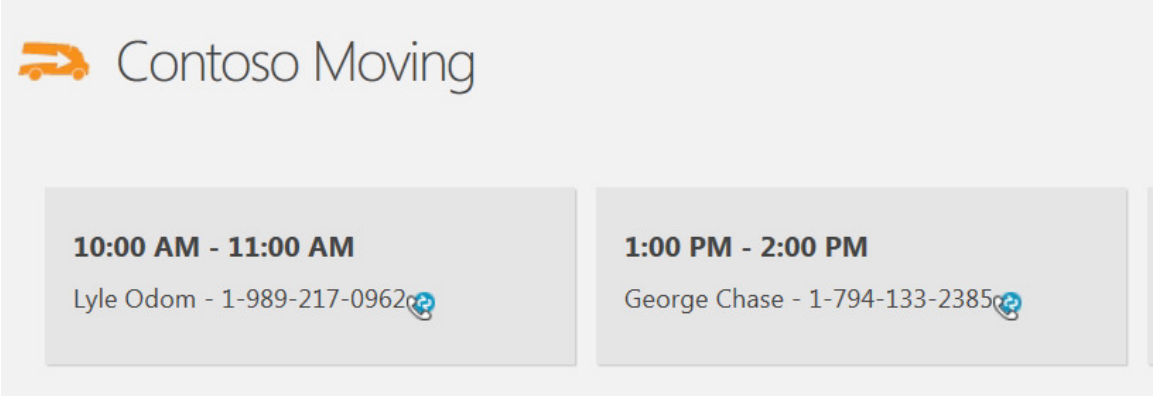
Out of the box, LightSwitch comes with two themes: a **dark** (default) and a **light** theme. To switch between the themes:

49. In the **Solution Explorer**, switch to **File View**. Under the **MobileClient** node, open **default.htm**.

50. Around line 26, you will find the following line:
    `<link rel="stylesheet" type="text/css" href="Content/msls.theme.dark.css" />`

```
12  <body>
13      <h2 id="msls-id-app-loading"
14          style="text-align: center; margin-top: 20%; width: 100%; height: 100%; z-index:
15          Loading Application ...
16      </h2>
17      <div id="msls-id-dialog-overlay" style="position: absolute; width: 100%; height: 10
18
19      <link rel="stylesheet" type="text/css" href="Content/jquery.mobile-1.0.css" charset
20      <link rel="stylesheet" type="text/css" href="Content/msls.reset.css" />
21      <link rel="stylesheet" type="text/css" href="Content/msls.global.css" />
22      <link rel="stylesheet" type="text/css" href="Content/msls.base.controls.css" />
23      <link rel="stylesheet" type="text/css" href="Content/msls.navigation.css" />
24      <link rel="stylesheet" type="text/css" href="Content/msls.dialog.css" />
25      <link rel="stylesheet" type="text/css" href="Content/msls.theme.font.css" />
26      <link rel="stylesheet" type="text/css" href="Content/msls.theme.dark.css" />
27      <script type="text/javascript" src="Scripts/winjs-1.0.RC.js"></script>
```

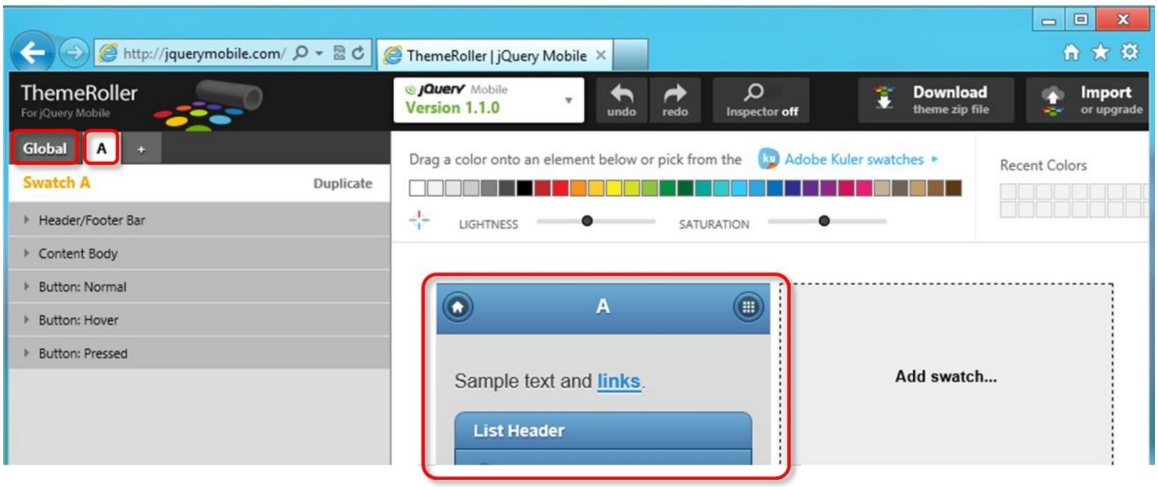
51. To switch between the dark and light themes, simply substitute **dark** or **light** into the **href** portion of the line: `href="Content/msls.theme.___.css"`.




## Integrate ThemeRoller themes

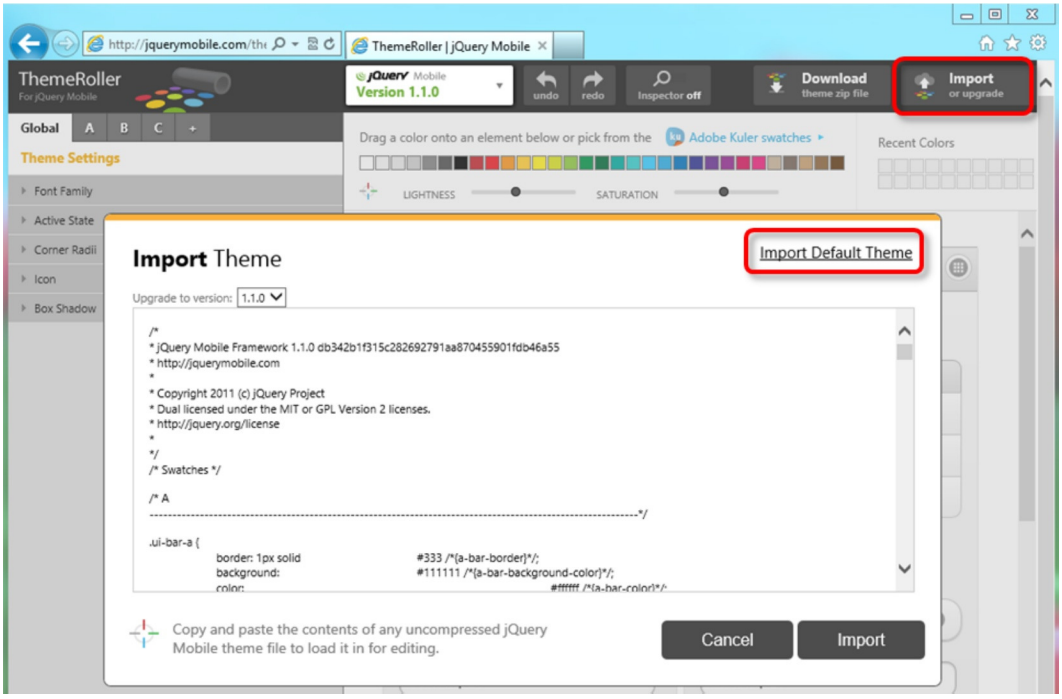
The appearance of a LightSwitch Mobile app can be customized via **ThemeRoller for JQueryMobile**. Some aspects of ThemeRoller might not integrate quite perfectly yet, but rest assured that this will be addressed in a future release.

52. Launch ThemeRoller by navigating to http://jquerymobile.com/themeroller/.
    You will be presented with a task pane on the left, and a live preview of a JQueryMobile app on the right.
53. Modify the ThemeRoller settings to suit your needs. Note that the sidebar on the left contains several tabs; the settings you will want to modify will be under **Global** and **Swatch A**. All other swatches and their respective previews will be ignored (and should probably be deleted, for the sake of keeping down the CSS size).

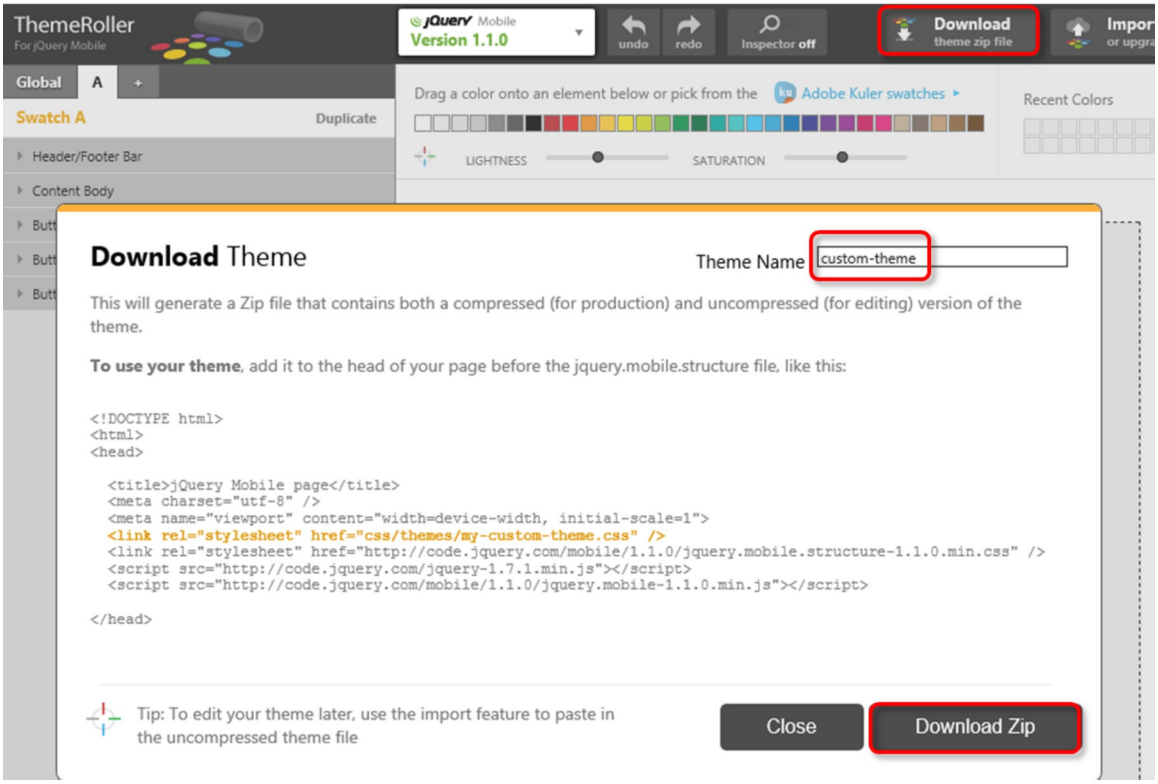



a. Note that for some settings, you can just drag colors from the top bar to the application elements. For other settings (e.g., font color), you will need to open up the respective settings in the task pane.

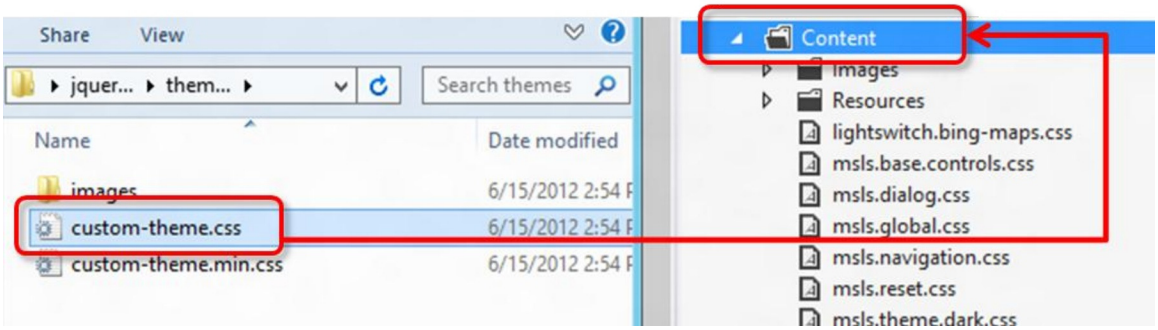
b. Tip: If you want to use a pre-built theme rather than creating your own, click **Import** and then choose **Import Default Theme** at the top of the window. This will populate 5 swatches. Choose any that you like and delete the rest, so that your chosen swatch becomes Swatch A.




54. Once you're done customizing the appearance of your theme, click **Download**. Specify a **Theme Name** (for example, "custom-theme"), and click **Download Zip**.




55. Download the zip file to a temporary location and unzip it. In the resulting folder, navigate to the **themes** folder within.
56. Back in Visual Studio, open the **Solution Explorer** to **File View**, and navigate to the **MobileClient** project. Open the shortcut menu for the **Content** folder and choose **Add, Existing Item**, and then choose the **<theme-name>.css** file in the **themes** folder inside the unzipped theme folder.



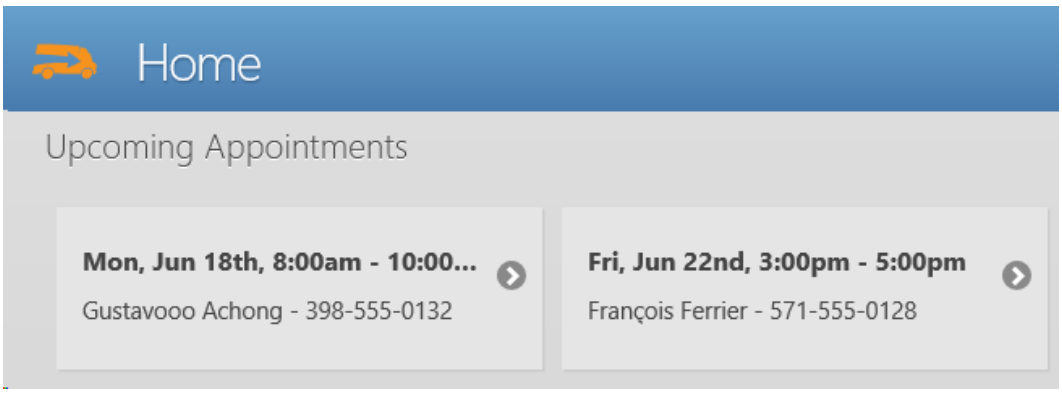

57. While still in **File View**, open **default.htm** from the root of the HTML client.
    a. Move the line with the base theme (**msls.theme.dark.css** or **msls.theme.light.css**) **just below** the start of the links section. That is, it should now be positioned right below the `Content/jquery.mobile-1.0.css` reference. See image below (whereby the red line arrow represents the move).

```
12  <body>
13      <h2 id="msls-id-app-loading"
14          style="text-align: center; margin-top: 20%; width: 100%; height: 100%; z-index: 100000; dis
15          Loading Application ...
16      </h2>
17      <div id="msls-id-dialog-overlay" style="position: absolute; width: 100%; height: 100%" />
18
19      <link rel="stylesheet" type="text/css" href="Content/jquery.mobile-1.0.css" charset="utf-8" />
20      <link rel="stylesheet" type="text/css" href="Content/msls.theme.light.css" />
21      <link rel="stylesheet" type="text/css" href="Content/custom-theme.css" charset="utf-8" />
22      <link rel="stylesheet" type="text/css" href="Content/msls.reset.css" />
23      <link rel="stylesheet" type="text/css" href="Content/msls.global.css" />
24      <link rel="stylesheet" type="text/css" href="Content/msls.base.controls.css" />
25      <link rel="stylesheet" type="text/css" href="Content/msls.navigation.css" />
26      <link rel="stylesheet" type="text/css" href="Content/msls.dialog.css" />
27      <link rel="stylesheet" type="text/css" href="Content/msls.theme.font.css" />
28      <link rel="stylesheet" type="text/css" href="Content/tilelist.css" />
29      <script type="text/javascript" src="Scripts/winjs-1.0.RC.js"></script>
```

b. Add a reference to your ThemeRoller custom theme immediately below the newly-repositioned **base theme** reference (custom theme outlined in green in the image above):
```
<link rel="stylesheet" type="text/css" href="Content/custom-theme.css"
charset="text/css" />
```

c. Note that some settings, like the color of list items and some of the headers, are determined by the base theme rather than ThemeRoller. Therefore, the choice of the dark versus light theme will probably still be relevant for you.

58. Press F5; the app should include your new style. Please beware that there are some known issues with themes, and we're working hard to resolve them by the next release.



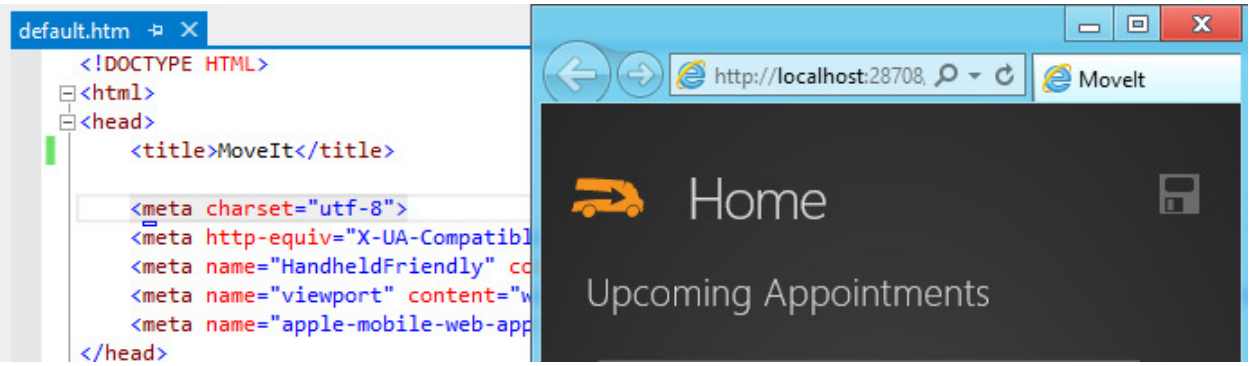

59. If, having created the ThemeRoller file, you want to modify it:
    a. Open the ThemeRoller website and click **Import** at the top of the window. Copy-paste the contents of your current custom theme and click **Import**.
    b. When you are done making changes, press **Download**, unzip the file, navigate to the **themes** folder, and open your **<theme-name>.css** file. Copy-paste the contents of that updated file into your former theme file.

c. Under some circumstances (e.g., when you replace the theme file rather than its contents), the browser might be using a cached CSS and not loading your new changes. If this is the case, and you are not seeing the changes that you'd just made, be sure to clear your browser's cache.

## Add an application title

The default title for a LightSwitch HTML client is initially blank – which means that browsers will typically render the name of the server or the web address as the title (typically shown next to the tab that represents the page). To customize it:

60. In the Solution Explorer, switch to **File View**. Under the **MobileClient** node, open **default.htm**.
61. Modify `<title></title>` near the top of the file to include a title for your app. For example, `<title>MoveIt</title>`.




# Step 7: Using a Device's Built-in Camera to Shoot and Upload Photos

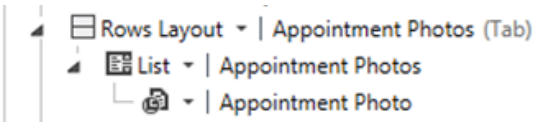
Let's give planning specialists the option to use use their mobile device to upload pictures of the home. The intent is that the inventory specialist will take a picture with the device, save it locally, and then upload it via the MoveIt app.

> **NOTE:** Due to Apple's policy of disabling the file-upload "browse" button, uploading via iOS's Safari browser is not supported at this time.

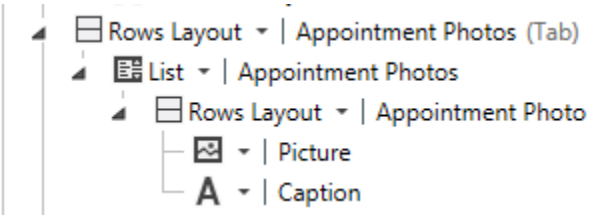
62. Open **Solution Explorer** and switch to **File View**, and then choose the **MobileClient**, **Scripts** folder.
63. Open the shortcut menu for the **Scripts** folder and choose **Add, Existing Item**.
64. From within the dialog, navigate to C:\Content\Resources, and select to add the following two file:
    a. image-uploader.js
    b. image-uploader-base64-encoder.aspx
65. Open **default**.htm to add a reference to **image-uploader.js** (the aspx page does not need a reference). The script reference should go at the end of the script references block.

```
<script type="text/javascript" src="Scripts/Generated/usercode.js"></script>
<script type="text/javascript" src="Scripts/image-uploader.js" charset="utf-8"></script>
```

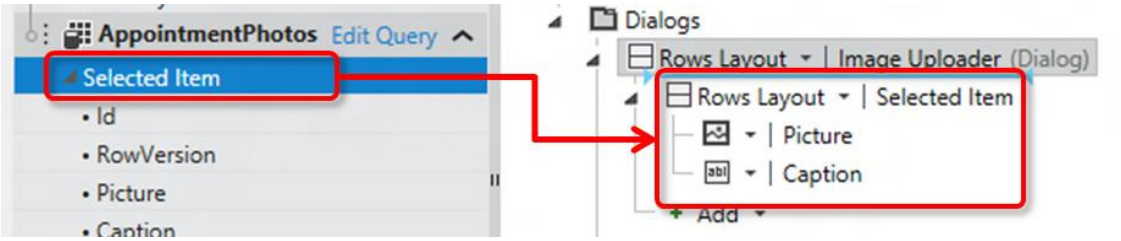
66. Switchback to **Logical View** and open the **ViewAppointmentDetail** screen. In the Screen Designer, select the **Appointment Photos** tab.




67. Change the **Appointment Photo** summary field to a **Rows Layout**.




68. Choose the **Dialogs** node, and then choose **Add Dialog**. Name the dialog "ImageUploader".
69. From the left Screen Designer pane, drag the **SelectedItem** node of **AppointmentPhotos** into the new dialog. Delete the **Appointment** item.

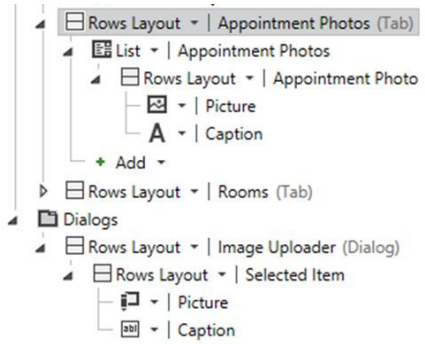


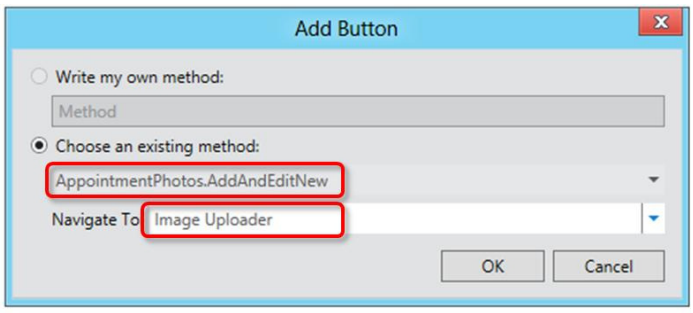

70. Select the newly-added Selected Item node, and in the **Properties** window choose **Height**. Clear the **Stretch to Container** check box.
71. Switch the dialog's **Picture** field from an **Image** to a **Custom Control**. In the **Properties** window, choose the **Edit Render Code hyperlink** and add the following code:

```
createImageUploader(element, contentItem, "max-width: 300px; max-height: 300px");
```

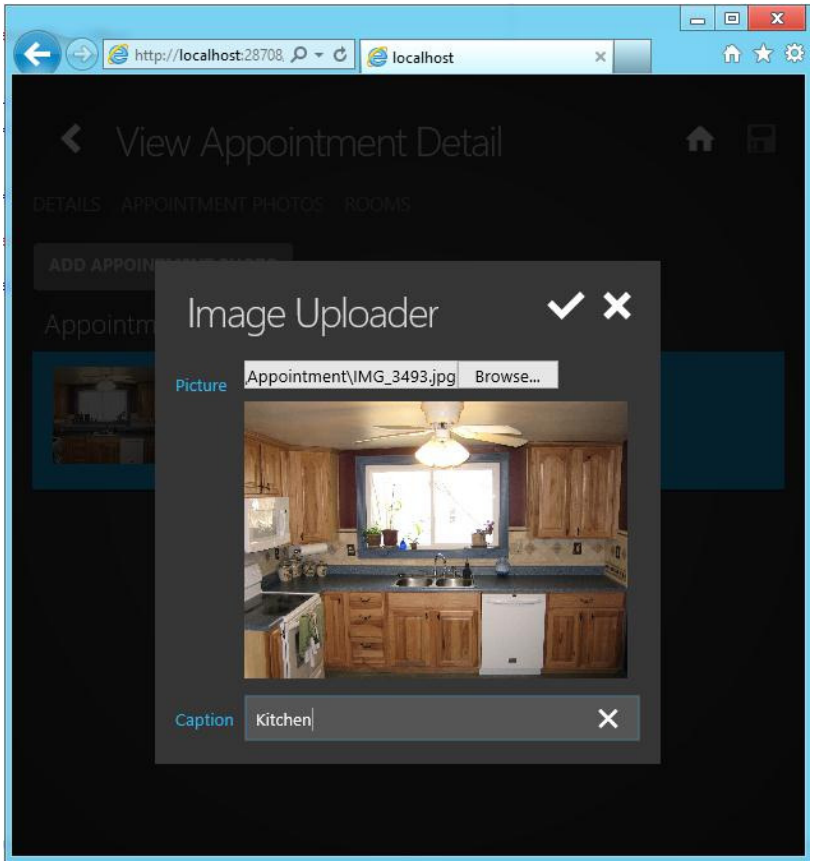
The call redirects all of the heavy-lifting to the image-uploader.js file. The customizable styling statement (`"max-width: 300px; max-height: 300px"`) specifies the image preview size.

72. In the Screen Designer, choose the **Appointment Photos** section and then choose **Add**, **New Button**. In the **Add Button** dialog box, choose the **AppointmentPhotos.AddAndEditNew** method, and **Image Uploader** as the "Navigate To" designation.



73. Once the button is created, drag it above the **Appointment Photos** list.
74. Press F5 to run the app. Select an appointment and switch to the **Appointment Photos** tab. Click **Add Appointment Photo** and see the image-uploader in action.




## Step 8: Add a Bing Map Custom Control

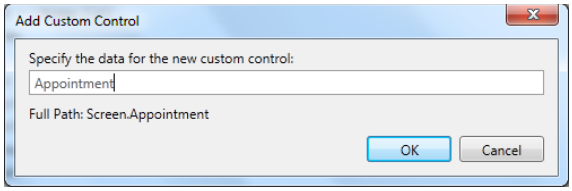
Now let's modify the **ViewAppointmentDetail** screen to display a map for the current appointment's address. We need to begin by adding some files to our project, and referencing them from the default.htm file.

75. In **Solution Explorer**, switch to **File View** and choose the **Scripts** folder of the **MobileClient** project.
76. Open the shortcut menu for the **Scripts** folder and choose **Add**, **Existing Item**.
77. Navigate to the C:\Content\Resources folder and add lightswitch.bing-maps.js.
78. Open the **default.htm** file and add script references for the **lightswitch.bing-maps.js** file, and a CDN reference to the Bing Maps control (take note of the where the file references were added in the code below because order matters):

```
<link rel="stylesheet" type="text/css" href="Content/msls.theme.font.css" />
<link rel="stylesheet" type="text/css" href="Content/msls.theme.dark.css" />
<script type="text/javascript" charset="utf-8"
        src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>
<script type="text/javascript" src="Scripts/winjS/winjs-1.0.RC.js"></script>
<script type="text/javascript" src="Scripts/JQuery/jquery-1.6.4.js" charset="utf-8"></script>
```

```
<script type="text/javascript" src="Scripts/JQuery/jquery.mobile-1.0.js" charset="utf-8"></script>
<script type="text/javascript" src="Scripts/JQuery/datajs-1.0.0.js" charset="utf-8"></script>
<script type="text/javascript" src="Content/Resources/msls.prerequisite.resources.js"></script>
<script type="text/javascript"
        src="Content/Resources/msls.defaultprerequisite.resources.js"></script>
<script type="text/javascript" src="Scripts/msls-1.0.0.js"></script>
<script type="text/javascript" src="Scripts/lightswitch.bing-maps.js" charset="utf-8"></script>
<script type="text/javascript" src="Scripts/Generated/data.js"></script>
```

79. Open the **ViewAppointmentDetail** screen in the designer.
80. In the **Details** section, click **Add Tab** after the **Rooms** tab, and name it **Map**, and then choose **Add**, **New Custom Control**.
81. In the **Add Custom Control** dialog box, specify **Appointment** as the data for the custom control.




82. Use the **Write Code** menu to edit the **Appointment_render** logic for the control.
83. Add the following utility methods above the render function:

```
var mapLastUpdated;
var mapControl;

function rebindMap(element, contentItem) {
    // Check to make sure we aren't updating the map continuously due to multiple bound values
changing.
    var now = new Date();
    if (now.getTime() - mapLastUpdated.getTime() > 15) {
        setTimeout(function () {
            updateMap(element, contentItem);
            mapLastUpdated = new Date();
        }, 20);
    }
};

function updateMap(element, contentItem) {
    var mapDiv = $("#appointmentMap");
    // If we've previously created the map, make sure to clean up the div it was contained in;
    // otherwise the bing map control fails to create properly.
    if (mapDiv.length > 0) {
        $(mapDiv).remove();
    }
    mapDiv = $("<div id='appointmentMap' class='msls-hauto msls-vauto' ></div>");

    $(mapDiv).appendTo($(element));
    mapControl = mapDiv.lightswitchBingMapsControl({
        street: contentItem.value.street,
        city: contentItem.value.city,
        state: contentItem.value.state.name,
        zipcode: contentItem.value.postalCode,
        mapTypeId: Microsoft.Maps.MapTypeId.road,
        width: "600",
        height: "400"
    });

};
```

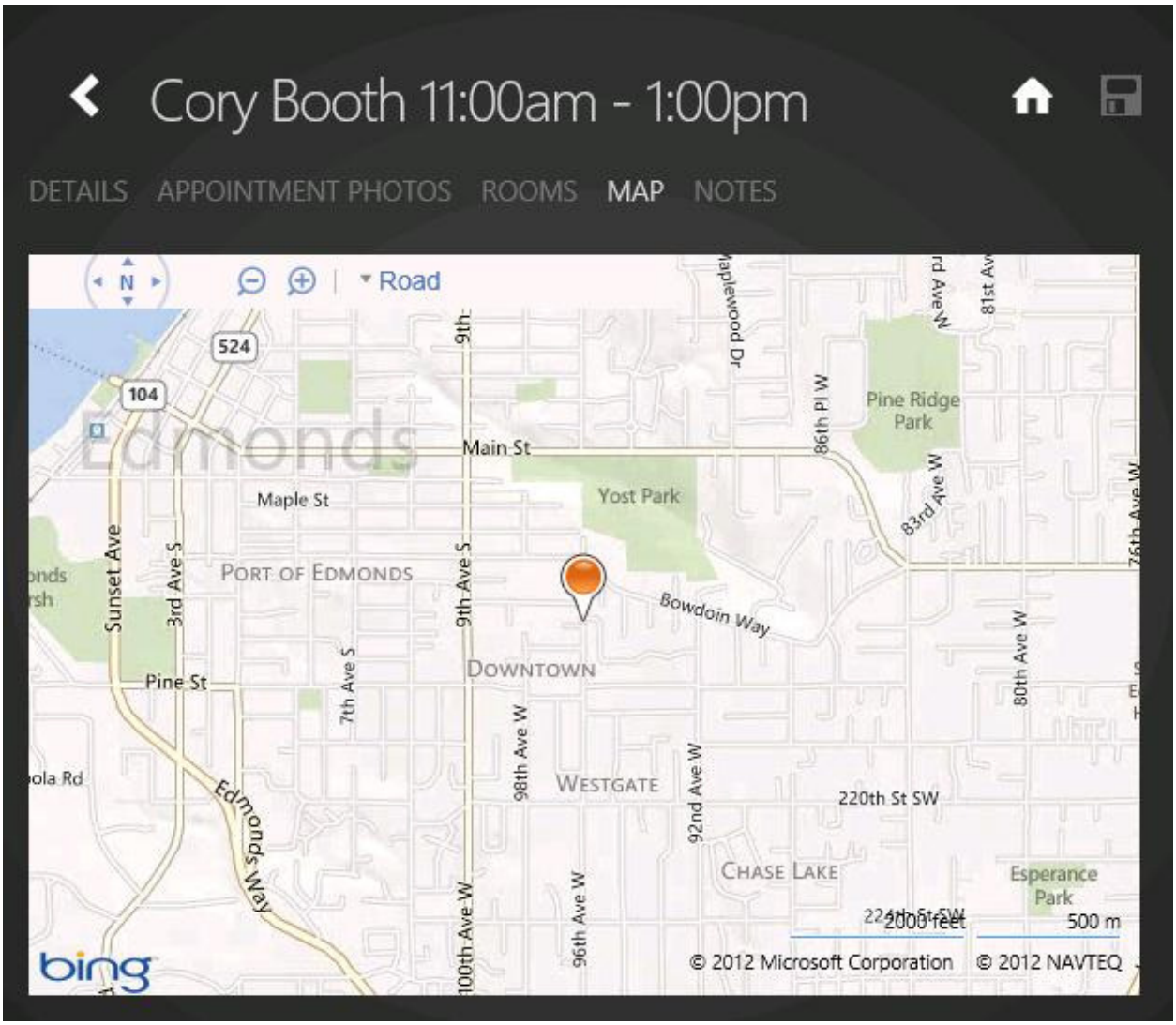
84. Now bind the map control the address fields on the appointment by adding the code to the **Appointment_render** method:

```
lightSwitchApplication.ViewAppointmentDetail.prototype.Appointment_render = function (element,
contentItem) {
    updateMap(element, contentItem);
    mapLastUpdated = new Date();
    msls.bind(contentItem, "value.street", function () { rebindMap(element, contentItem); });
    msls.bind(contentItem, "value.city", function () { rebindMap(element, contentItem); });
    msls.bind(contentItem, "value.zip", function () { rebindMap(element, contentItem); });
    msls.bind(contentItem, "value.postalCode", function () { rebindMap(element, contentItem); });
};
```

85. Press F5. After the application loads, tap on an appointment, and then tap the **Map** section.

## Step 9: Republish the App and Test it on a Tablet Device

86. Before re-publishing, if you are not already running Visual Studio as an admin, exit and re-open Visual Studio 2012 as an administrator. This is done by right-clicking on the Visual Studio 2012 tile in the Windows start screen, and selecting **Run as administrator**.
87. In **Solution Explorer**, open the shortcut menu for the **Contoso Moving** project and choose **Publish**.
88. The publish wizard remembers your existing configuration and automatically shows the Summary page. Choose the **Publish** button. As before, the **Mobile Client** app will be available at the URL: http://*<machine-name>*/ContosoMoving/MobileClient/.


Cory Booth 11:00am - 1:00pm
DETAILS
APPOINTMENT PHOTOS
ROOMS
MAP
NOTES
Road
524
104
Edmonds
Main St
Maple St
Yost Park
Pine Ridge Park
Port of Edmonds
Sunset Ave
3rd Ave S
9th Ave S
Bowdoin Way
Downtown
Pine St
7th Ave S
98th Ave W
Westgate
92nd Ave W
220th St SW
Chase Lake
Esperance Park
Edmonds Way
100th Ave W
96th Ave W
86th Pl W
83rd Ave W
80th Ave W
2000 feet
500 m
© 2012 Microsoft Corporation
© 2012 NAVTEQ
bing