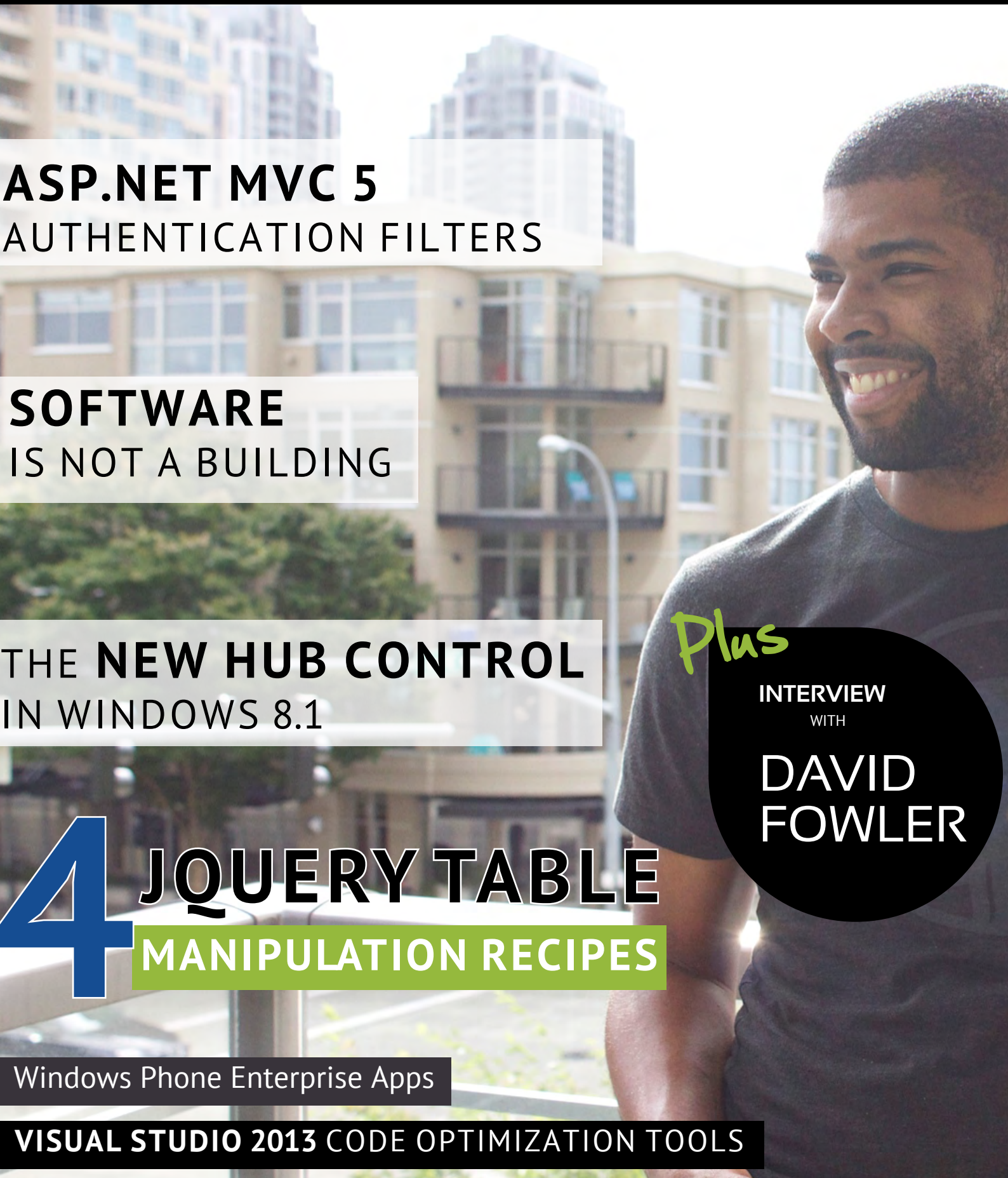
Issue 09 | NOV-DEC 2013 EDITION

DNCMagazine

www.dotnetcurry.com



ASP.NET MVC 5
AUTHENTICATION FILTERS

SOFTWARE
IS NOT A BUILDING

THE NEW HUB CONTROL
IN WINDOWS 8.1

Plus

INTERVIEW
WITH
DAVID FOWLER

4 JQUERY TABLE
MANIPULATION RECIPES

Windows Phone Enterprise Apps

VISUAL STUDIO 2013 CODE OPTIMIZATION TOOLS

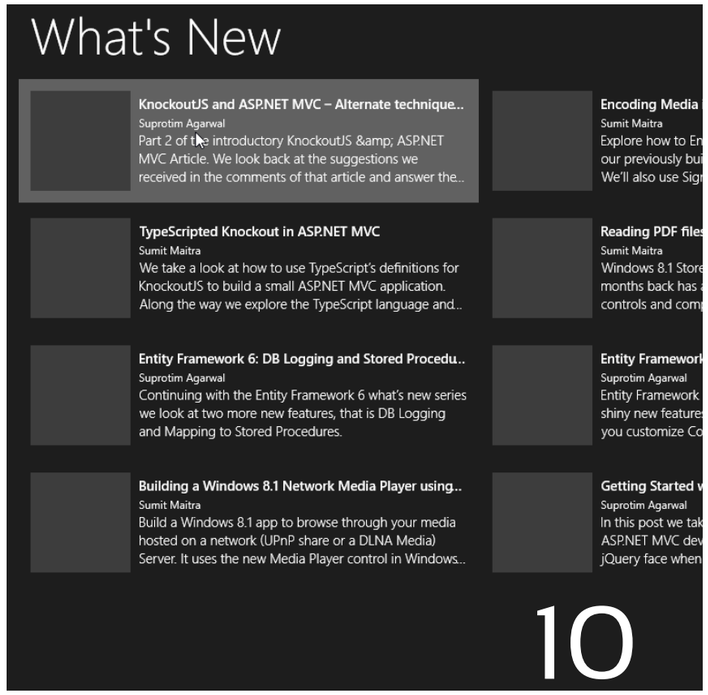
contents

FEATURES

ON THE COVER

David Fowler, creator of SignalR.

Windows, Visual Studio, ASP.NET, WinRT, Azure & other Microsoft products & technologies are trademarks of the Microsoft group of companies. 'DNC Magazine' is an independent publication and is not affiliated with, nor has it been authorized, sponsored, or otherwise approved by Microsoft Corporation.



LETTER FROM

THE EDITOR

> *Take a deep breath before getting into the groove again...*

First up, season's greetings to everyone in advance. We wish everyone a joyful Festival/Holiday Season over the next couple of months.

Lately, the last few months of every year has been full of whirlwind days for the Microsoft Dev/IT Pro community. With some major upgrades and new versions of Visual Studio and Windows being released and an abundance of Developer Conferences & Camps happening, there's hardly any time to breath!

Every new release of Visual Studio and .NET, makes developers more productive and writing code becomes more enjoyable. In this issue of *DNC Magazine*, Subodh shows how developer productivity can be increased and quality code produced, using 'Code Optimization Tools' in Visual Studio 2013. Raj demonstrates the new IAuthenticationFilter in ASP.NET MVC 5 and Sumit shows how to build a cool Feed Reader using the new Hub Control in Windows 8.1.

This month, we welcome 'Software Gardener' Craig as he shares his wisdom on Software and compares it to Gardening. Pravin introduces Windows PowerShell and showcases how to script SharePoint using it. For our client-side developers, I have written an article on jQuery Performance Tips while manipulating HTML Tables.

We talk to SignalR creator David Fowler about his journey as a developer, initial days of SignalR and factlets around it. Finally for those who think its all getting Webby here, Mayur showcases how to build Windows Mobile Apps for the Enterprise.

Take a deep breath! With the plethora of releases this quarter, the .NET Dev Community is unlikely to get any time off. We will continue our efforts via this magazine and our websites to help you sort through the noise. That's a promise!

You can have a direct impact on this magazine by sharing your thoughts with me. Your opinion matters! Email me at suprotimagarwal@dotnetcurry.com

Suprotim Agarwal

www.dotnetcurry.com

dnc mag

Editor • Suprotim Agarwal
suprotimagarwal@dotnetcurry.com

Art & Creative Director • Minal Agarwal
minalagarwal@a2zknowledgevisuals.com

Sponsorship Opportunities • Suprotim Agarwal
suprotimagarwal@dotnetcurry.com

Contributing Writers • Craig Berntson, Mayur Tendulkar, Pravin Dabade, Raj Aththanayake, Subodh Sohoni, Sumit Maitra, Suprotim Agarwal

Writing Opportunities • Carol Nadarwalla
writeforus@dotnetcurry.com

NEXT ISSUE - 1st January 2014

Copyright @A2Z Knowledge Visuals Pvt. Reproductions in whole or part prohibited except by written permission. EMail requests to "suprotimagarwal@dotnetcurry.com"

Legal Disclaimer: The information in this magazine has been reviewed for accuracy at the time of it's publication, however the information is distributed without any warranty expressed or implied.

stay **connected**

www.Facebook.com/DotNetCurry

@dotnetcurry

www.DotNetCurry.com/magazine

POWERED BY

a2z | Knowledge Visuals

THE TOP 4 HTML TABLE MANIPULATION RECIPES USING JQUERY

Microsoft MVP ***Suprotim Agarwal*** shares some performance tips while manipulating HTML Tables in jQuery

HTML Tables can be pretty boring to look at! Although you can add a dash of CSS and beautify them, users demand more interactivity by representing and manipulating tables at runtime.

Some common manipulation tasks performed with tabular data is adding and deleting rows, sorting and paginating data. JavaScript is the obvious choice to perform these operations, but many lines of code need to be written in plain JavaScript, to traverse and manipulate the DOM tree. DOM operations can get tricky at times. To add to our woes, writing code that works cross-browser is tedious, especially when performing advanced manipulations. This is where a JavaScript library like jQuery comes in handy.

jQuery makes DOM operations less scary. It provides an abstraction layer and allows you to work with the DOM, without having to know every little thing about it. One of the biggest benefits of using jQuery is that it handles a lot of cross-browser issues for you

In this article, I will share some jQuery techniques to manipulate Table Data. I will also share performance tips about jQuery selectors, caching selectors, and writing terse, efficient code.

THESE TIPS WILL HELP YOU WRITE EFFICIENT CODE

This article is based on my upcoming jQuery Book **The Absolutely Awesome jQuery CookBook** where I share similar self-contained recipes that you can easily incorporate in your websites or projects.

DEFINING THE TABLE STRUCTURE

The first step to write effective jQuery is to write well-formed HTML. Here's a subset of a well-defined HTML Table markup:

```
<table id="someTable">
<thead>
<tr>
   <th class="empid">EmpId</th>
   <th class="fname">First Name</th>
   <th class="lname">Last Name</th>
   <th class="email">Email</th>
   <th class="age">Age</th>
</tr>
</thead>
<tfoot>
<tr>
  <td colspan="5">
    <a href="http://www.jquerycookbook.com">
       The Absolutely Awesome jQuery CookBook
    </a>
  </td>
</tr>
</tfoot>
<tbody>
<tr>
  <td class="empid">E342</td>
  <td class="fname">Bill</td>
  <td class="lname">Evans</td>
  <td class="email">Bill@devcurry.com</td>
  <td class="age">35</td>
</tr>
<tr>
  <td class="empid">E343</td>
  <td class="fname">Laura</td>
  <td class="lname">Matt</td>
  <td class="email">laura@devcurry.com</td>
  <td class="age">26</td>
</tr>
...
</tbody>
</table>
```

Observe how we have declared <thead>, <tbody>, and <tfoot> tags in your table. We have also declared Id's and Classes where required, to allow jQuery selectors to have direct access to the DOM elements.

With a dash of CSS, our sample table looks like the following:

EmpId	First Name	Last Name	Email	Age
E342	Bill	Evans	Bill@devcurry.com	35
E343	Laura	Matt	laura@devcurry.com	26
E344	Ram	Kumar	ram@devcurry.com	56
E345	Yuri	Gagrin	yuri@devcurry.com	39
E340	Asha	Singh	asha@devcurry.com	42
The Absolutely Awesome jQuery CookBook				

1 INSERT A NEW ROW AS THE LAST ROW OF A TABLE

With the table in place, write the following code to insert a new row, as the last row of the table

```
$(function () {
    newRow = "<tr>" +
        "<td class='empid'>E333</td>" +
        "<td class='fname'>Fujita</td>" +
        "<td class='lname'>Makoto</td>" +
        "<td class='email'>fujita@devcurry.com</td>" +
        "<td class='age'>52</td>" +
    "</tr>";
    $('#someTable > tbody > tr:last').after(newRow);
});
```

When the page is rendered, you should see the newly added row.

EmpId	First Name	Last Name	Email	Age
E342	Bill	Evans	Bill@devcurry.com	35
E343	Laura	Matt	laura@devcurry.com	26
E344	Ram	Kumar	ram@devcurry.com	56
E345	Yuri	Gagrin	yuri@devcurry.com	39
E340	Asha	Singh	asha@devcurry.com	42
E333	Fujita	Makoto	fujita@devcurry.com	52
The Absolutely Awesome jQuery CookBook				

We are using the jQuery selector extension *:last* to select the last matched row in our table. The *after()* method inserts the new row after the set of matched elements; in our case, after the last row.

Note: *The above example works well for smaller tables, however in a large table, using the :last selector may not give you the best performance. As per the jQuery documentation, :last is a jQuery extension and not part of the CSS specs and hence*

cannot take advantage of the powerful native DOM methods like querySelectorAll, that can parse any CSS selector, .

To achieve better performance, we can rewrite our code as:

```
$('#someTable > tbody > tr').filter(":last").
after(newRow);
```

The code first selects rows using a pure CSS selector `#someTable > tbody > tr` and then uses `filter(":last")` to match the last row of the table.

There are multiple ways in jQuery to achieve a certain requirement. A point to always remember is that jQuery will always use a native method (in our case we discussed querySelectorAll) if available, as it's much quicker at getting elements and gives a notable performance with complex and large sets of data. Jsperf.com is your friend to run tests, whenever you are not sure of which selectors or methods to use in your code, for the browsers you are supporting.

2 INSERT A NEW ROW IN A TABLE AT A CERTAIN POSITION

Now let's say you want to insert a new row as the 2nd row in a table. Use the following code:

```
var index = 2;
newRow = "<tr>" +
         "<td class='empid'>E333</td>" +
         "<td class='fname'>Fujita</td>" +
         "<td class='lname'>Makoto</td>" +
         "<td class='email'>fujita@devcurry.com</td>" +
         "<td class='age'>52</td>" +
         "</tr>";

$('#someTable > tbody > tr').eq(index-1).
before(newRow);
```

Refresh the page and you will see the new row gets added as the 2nd row.

EmpId	First Name	Last Name	Email	Age
E342	Bill	Evans	Bill@devcurry.com	35
E333	Fujita	Makoto	fujita@devcurry.com	52
E343	Laura	Matt	laura@devcurry.com	26
E344	Ram	Kumar	ram@devcurry.com	56
E345	Yuri	Gagrin	yuri@devcurry.com	39
E340	Asha	Singh	asha@devcurry.com	42

Since indexes are *zero* based and we are passing index=2, *.eq(index)* would mean *.eq(2)* i.e. the 3rd row. So to add this as the 2nd row of a table, you first need to do *index-1* and then you need to go back to the 2nd row of the table and insert *.before()* that. This new row now becomes the 2nd row of the table.

Alternatively to insert a new row as the 2nd row in a table, you can also do

```
$('#someTable > tbody > tr:first').after(newRow);
```

which uses the *:first* selector to match the first row and insert a row *after()* it.

Here just like we saw previously for the *:last* selector, for large tables, you will get performance benefits by using the filter *:first*.

Similarly you can also explore other child filter selectors like *first-child, nth-child* and so on from the jQuery documentation at api.jquery.com/category/selectors/child-filter-selectors/

Note: *To become a better developer, I cannot emphasize the fact enough that you should take out some time and go through the jQuery documentation. It's probably one of the most well written documentation of any JavaScript library out there and getting familiar with difference selectors and API's, will save you tons of time and frustration in a project.*

3 REMOVE ALL ROWS EXCEPT THE HEADER ROW

If your table is well defined and contains a <thead> <tbody>, this piece of code will work to remove all the rows, except the header row

```
$('#someTable tbody tr').remove();
```

This code uses the *remove()* method to remove a set of rows inside tbody. Since we haven't supplied the *remove()* method with any selector as a parameter, the above code removes all rows, as well as all elements, events and data in the rows.

Note: *If you want to remove all rows without removing data and events, use detach() instead of remove()*

In case you do not have a <thead> defined and want to remove all rows except the first one (assuming first row is meant to be a header), use this code

```
$('#someTable tr:gt(0)').remove();
```

where the selector *gt(0)* selects all rows at an index greater than zero, within the matched rows. Similarly if you want to keep the first two rows and remove all others, change the above code to this

```
$('#someTable tr:gt(1)').remove();
```

Note: *The jQuery documentation says "Because :gt() is a jQuery extension and not part of the CSS specification, queries using :gt() cannot take advantage of the performance boost provided by the native DOM querySelectorAll() method. For better performance in modern browsers, use $("your-pure-css-selector").slice(index) instead"*

As seen and discussed earlier, in our case, a *pure CSS selector* would be `$('#someTable tr')`. All you need to do is use it with *slice()* to remove all rows except the first row.

```
$('#someTable tr').slice(1).remove();
```

Slice() is zero based and takes two arguments, start and end. Since we are supplying *1* as the first parameter to slice(), the above statement will return a new jQuery object that selects from the first row, to the end. Calling *remove()* removes this set of matched element returned by *slice* and you are left with only the first row.

4 DYNAMICALLY ADD THOUSANDS OF NEW ROWS TO A TABLE WITH PERFORMANCE

jQuery gives us great power when it comes to manipulating the DOM, and with Great Power Comes Great Responsibility!

Think about some of these functions that you can perform very easily using jQuery:

- hide/delete/insert/update elements
- resize elements/change dimensions
- move/animate elements

and so on..

All these operations cause what is known as a *reflow* operation in the browser. Google Developers documentation defines reflow as "Reflow is the name of the web browser process for re-calculating the positions and geometries of elements in the document, for the purpose of re-rendering part or all of the document".

Reflows can be very expensive if not done correctly.

Note: *If you plan on becoming a serious front end engineer, I would advise you to spend some time reading about reflow and repaint operations.*

To understand this requirement better, let's take an example where we have to dynamically insert 1000's of rows in a table. We will make use of *$.append()* to add 5000 rows using two approaches. In the first approach, we will append new rows to the table, *every time* the loop iterates. In the second approach, we will construct a string with the new rows and then append the string *only once* after the loop is completed. We will then compare the two approaches and derive our conclusion as to which one of them is better and why.

Since it is a large table, to iterate the loop, I will be using the old *for* loop instead of *$.each*.

Note: *Although a comparison of for vs $.each and performance vs readability is beyond the scope of this article, in my jQuery Book, I have included a chapter on using jsperf and demonstrated how for a large table, using our old for loop outperforms $.each.*

To keep it simple, let's proceed with the *for* loop to perform iterations.

Use the following piece of code. I have declared some additional variables like t1, t2, t1t2 to measure the time differences while using the two approaches. You also use jsperf.com to record results in multiple browsers

```
$(function () {
    var $tbl = $("#someTable");
```

```
    // Approach 1: Using $.append() inside loop
    var t1 = new Date().getTime();
    for(i=0; i < 5000; i++){
        rows = "<tr><td>" + i + "</td><td>FName
       </td><td>LName</td></tr>";
        $tbl.append(rows);
    }
    var t2 = new Date().getTime();
    var t1t2 = t2-t1;
    $('#result').append("Approach 1: Append Inside Loop
    took " + t1t2 + " milliseconds" + "</br>");


    // Approach 2: Using $.append() outside loop
    var newrows;
    var t3 =  new Date().getTime();
    for(i=0; i < 5000; i++){
        newrows += "<tr><td>" + i + "</td><td>FName
        </td><td>LName</td></tr>";
    }
    $tbl.append(newrows);
    var t4 = new Date().getTime();
    var t3t4 = t4 - t3;
    $('#result').append("Approach 2: Append Once
    Outside Loop " + t3t4 + " milliseconds" + "</br>");
});
```

As discussed, we have two sets of code. Approach 1 calls $.append on *each* iteration of the loop whereas Approach 2 constructs a string (using +=) with the new rows and calls $.append *only once after* the loop iteration.

The difference between the two approaches is considerable, especially on IE and Safari, as seen in the screenshot:

Chrome 30.0.1599.101

Approach 1: Append Inside Loop took 382 milliseconds
Approach 2: Append Once Outside Loop 86 milliseconds

Firefox 23.0.1

Approach 1: Append Inside Loop took 292 milliseconds
Approach 2: Append Once Outside Loop 51 milliseconds

Internet Explorer 10.0.10

Approach 1: Append Inside Loop took 3501 milliseconds
Approach 2: Append Once Outside Loop 533 milliseconds

Safari 5.1.7

Approach 1: Append Inside Loop took 1287 milliseconds
Approach 2: Append Once Outside Loop 51 milliseconds

Our example considered just 2 columns and some fixed length data. Imagine in a real world scenario, where there are multiple columns, with variable data; the results would be dramatic.

In Approach 1, every time you are adding a new row to the table *inside the loop,* you are causing a *reflow* and the entire page geometry gets calculated *every time*, with the new DOM change. In Approach 2, theoretically speaking, the reflow occurs only once, since the rows are constructed and added *outside the loop*. That's why it's very important for a front-end engineer to understand and evaluate the difference between the two approaches. Jsperf.com is your friend and use it whenever you can.

Note: *Off topic, in both the approaches, you could squeeze additional performance by not concatenating the string, but rather adding it as individual elements of an array and then use join() outside the loop, to construct the entire string in one go. If you know the length of the array beforehand, that will help too. Check this link to learn more about the same.*

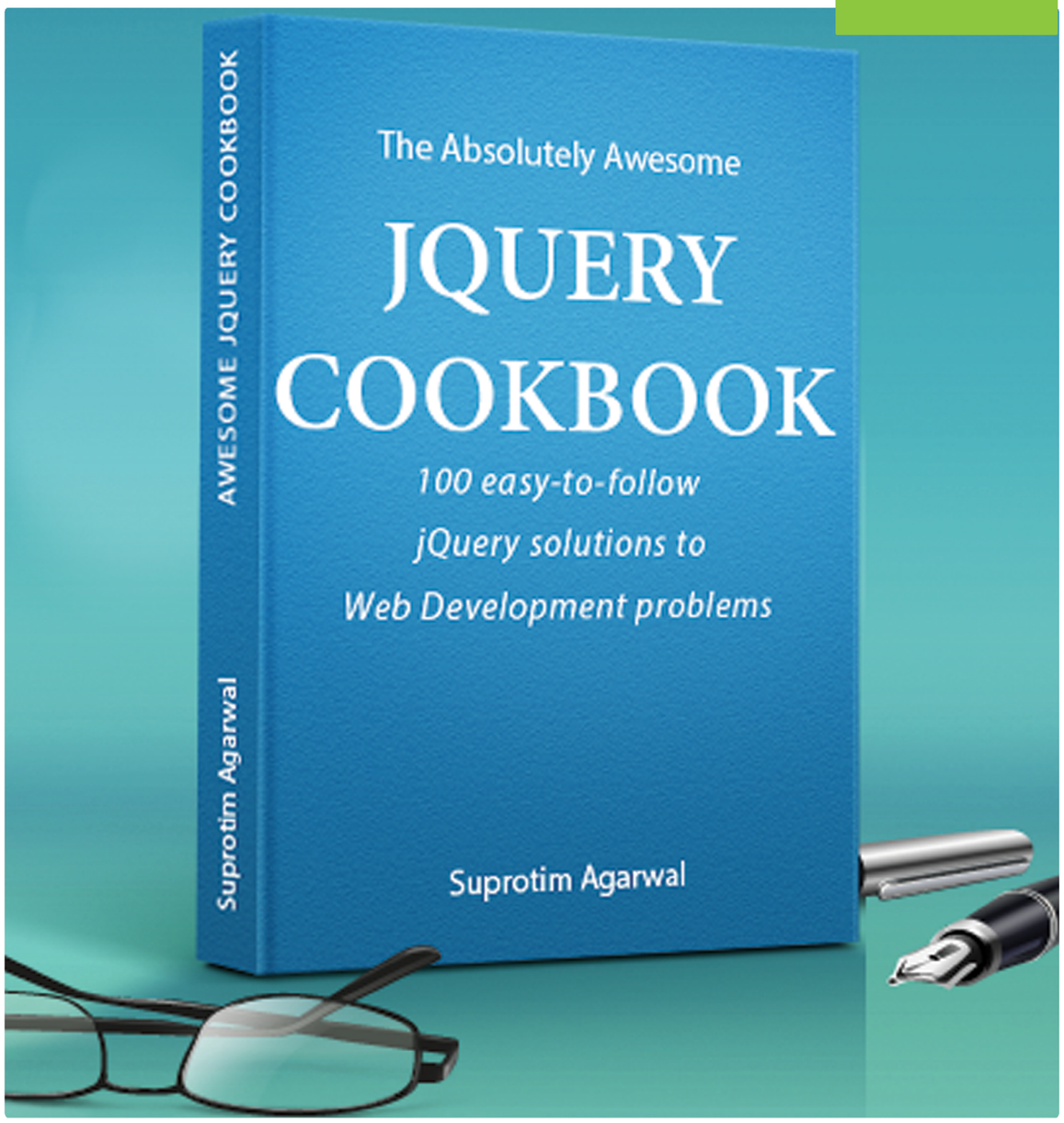
So these were some performance tips while working with HTML Tables. If you liked them, do check out my upcoming jQuery Book which is full of recipes demonstrating how to use jQuery to the best of its ability ■

Download the entire source code from our GitHub Repository at bit.ly/dncm9-jqperf

Suprotim Agarwal, ASP.NET Architecture MVP, is an author and the founder of popular .NET websites like dotnetcurry.com, devcurry.com and the DNC .NET Magazine that you are reading. You can follow him on twitter @suprotimagarwal or learn more about his new book www.jquerycookbook.com

NEW EBOOK

The Absolutely Awesome jQuery Cookbook



100 Easy-to-follow jQuery solutions

With scores of practical jQuery recipes you can use in your projects right away, this cookbook will help you gain hands-on experience with the jQuery API! Please click below to learn more.

Click Here www.jquerycookbook.com

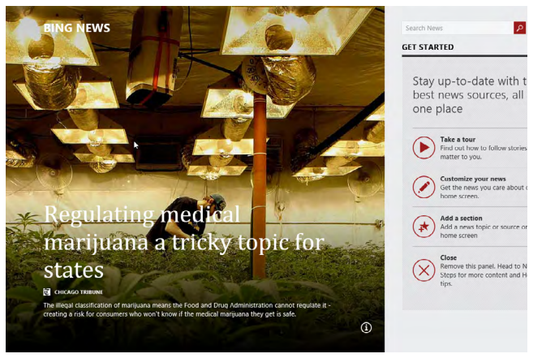
EXPLORING THE NEW HUB CONTROL IN WINDOWS 8.1

Windows 8.1 introduces a new Hub Control and a Hub style application template. ***Sumit Maitra*** digs in and shows you how to use it for a simple yet practical Feed Reader app.

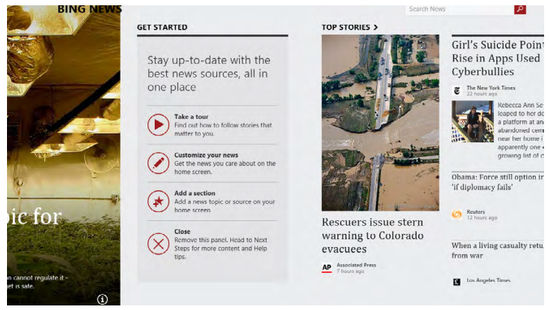
As highlighted in our previous editions, Windows 8.1 introduces a lot of new features that makes development for the platform way easier than Windows 8. Today we explore one such feature that is the *Hub Control* and a Project template that uses it, out of the box. Today we'll use the Hub Control to build an App that gives us a centralized hub for all www.dotnetcurry.com articles.

WHAT'S A HUB APP

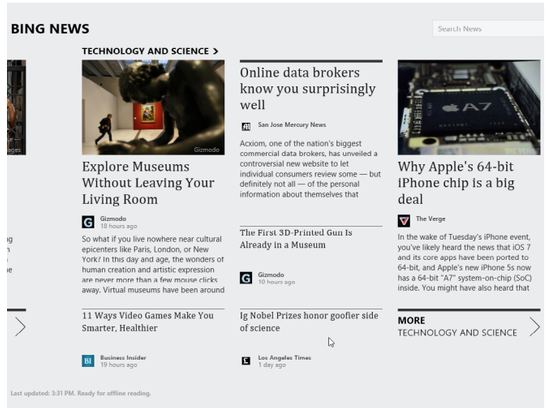
Of the bundled apps that came with Windows 8, the Bing Apps like Bing News, Bing Weather and Bing Travel were the better looking, functional and among the more popular apps. They all had a typical style starting with a big *Hero Image* that had a headline and then one could scroll across for the latest snapshots and then dig deeper for details views. For example, here is an opening page of the Bing News app (in Windows 8.1)



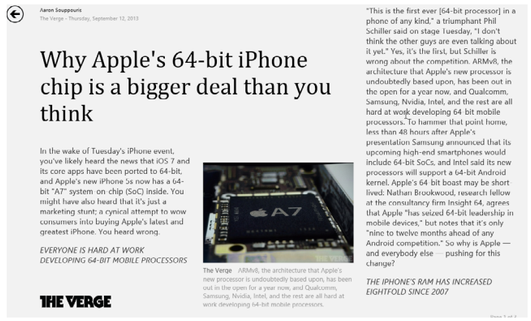
As we scroll to the right, we start with the 'Top Stories' section.



When we scroll further to the right, we start seeing more categories and their top 5-6 headlines. Each category can potentially end with a "More ..." list item that navigates the user to that particular section and lists all the possible headlines.



Tapping on a headline, gives us the complete report, which you can scroll horizontally to read it completely.



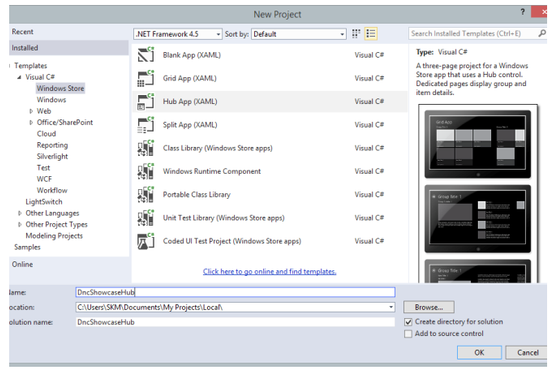
This is the typical design of a Hub App! The Hub control gives us the basic UI and navigation skeleton to build a Hub App. So lets build one.

BUILDING A HUB APP

Our Hub app will contain feeds from two sites, with Curated streams of articles being showcased in the Hub. Since our target sites are technical sites, the Images that we have are not expected to be glamorous enough to be blown up full screen, so we'll use a standard Hero Image. For rest of the layout, we'll start with the default and then tweak if required.

Off the blocks – The Hub Project Template

We start off with the new Hub App Project Template in Visual Studio 2013.



Straight off the bat, we end up with an application that looks as follows:

As we can see, a lot of groundwork with respect to the layout is already done for us. What remains is to plug in real data and add some navigation if required.

Before we get into retrieving real data, lets look at the settings of the Hub Control in HubPage.xaml

The above XAML snapshot can be interpreted as follows:

- The Hub Page has two Data Templates – Standard310x260ItemTemplate and Standard420x130Item Template.

- The Root panel is still a Grid control that contains the Hub control.

- The Hub control has *seven* subsections in it, each defining a typical layout. Let's look at these sections in details:

-- The Hub.Header – This comprises of the Application Header Text and the back button. The detailed markup for it is as follows:

```
<Hub.Header>
 <!-- Back button and page title -->
 <Grid Margin="0,20,0,0">
  <Grid.ColumnDefinitions>
   <ColumnDefinition Width="80"/>
   <ColumnDefinition Width="*"/>
  </Grid.ColumnDefinitions>
  <StackPanel Height="40">
   <AppBarButton x:Name="backButton" Icon="Back"
    Margin="-30,-14,0,0"
    Command="{Binding NavigationHelper.GoBackCommand,
    ElementName=pageRoot}"
    Visibility="{Binding IsEnabled,
    Converter={StaticResource
    BooleanToVisibilityConverter},
    RelativeSource={RelativeSource Mode=Self}}"
    AutomationProperties.Name="Back"
    AutomationProperties.AutomationId="BackButton"
    AutomationProperties.ItemType="Navigation Button"/>
  </StackPanel>
  <TextBlock x:Name="pageTitle" Text="{StaticResource
   AppName}"
   Style="{StaticResource HeaderTextBlockStyle}"
   Grid.Column="1"
   IsHitTestVisible="false" TextWrapping="NoWrap"
   VerticalAlignment="Top"/>
 </Grid>
</Hub.Header>
```

As we can see, the header uses a Grid layout with two columns. The first column is for the button and the second column is for the Application Name.

-- Next comes the Hub Section with the width of 780px. This is the Hero Image section. The markup for it is as follows:

```
<HubSection Width="780">
 <HubSection.Background>
  <ImageBrush ImageSource="Assets/MediumGray.png"
   Stretch="UniformToFill" />
 </HubSection.Background>
</HubSection>
```

The default image is set to the MediumGray.png from our app's Assets folder.

-- The third Hub Section is a single column section that shows the latest/most important item of the feed. It's markup is as follows:

```
<HubSection Width="580" Padding="120,30,40,44"
VerticalAlignment="Top" >
 <HubSection.Header>
  <TextBlock x:Uid="Section1Header"
   TextLineBounds="TrimToBaseline"
   OpticalMarginAlignment="TrimSideBearings"
   Text="Section 1"/>
 </HubSection.Header>
 <DataTemplate>
  <Grid Margin="0,10,0,0">
   <Grid.RowDefinitions>
    <RowDefinition Height="Auto" />
    <RowDefinition Height="Auto" />
    <RowDefinition Height="*" />
   </Grid.RowDefinitions>
   <Image Source="Assets/MediumGray.png" Stretch="Fill"
    Width="420" Height="280"/>
   <TextBlock Grid.Row="1"
    x:Uid="Section1Subtitle" Style="{StaticResource
    SubheaderTextBlockStyle}"
    TextWrapping="Wrap" Margin="0,10,0,0"
    Text="Lorem ipsum …"/>
   <TextBlock Grid.Row="2" x:Uid="DescriptionHeader"
    Margin="0,10,0,0"
    Style="{StaticResource TitleTextBlockStyle}"
    Text="Description text:"/>
   <TextBlock Grid.Row="3"
    x:Uid="Section1DescriptionText"
    Style="{StaticResource
    BodyTextBlockStyle}" Text="Lorem ipsum … Truncated
     Text… "/>
  </Grid>
 </DataTemplate>
</HubSection>
```

The markup defines a custom DataTemplate here but doesn't get the data from a data source; instead it has hardcoded the data. We'll have to fix this in our final application.

-- The fourth Hub Section uses the Standard420x130Template. Notice the custom HubSection Header that we can use.

```
<HubSection Padding="40,30,40,44"
DataContext="{Binding Path=[0], Source={StaticResource
groupedItemsViewSource}}" IsHeaderInteractive="True" >
 <HubSection.Header>
  <TextBlock x:Uid="Section2Header" TextLineBounds=
   "TrimToBaseline"
   OpticalMarginAlignment="TrimSideBearings"
   Text="Section 2"/>
 </HubSection.Header>
 <DataTemplate>
  <ListView
   x:Name="itemListView"
   Margin="-14,-4,0,0"
   AutomationProperties.AutomationId="ItemListView"
   AutomationProperties.Name="Grouped Items"
   ItemsSource="{Binding Items}"
   ItemTemplate="{StaticResource
   Standard420x130ItemTemplate}"
   IsSwipeEnabled="False"
   IsItemClickEnabled="True"
   ScrollViewer.VerticalScrollBarVisibility="Hidden"
   SelectionMode="None"
   ItemClick="ItemView_ItemClick">
   <ListView.ItemsPanel>
   <ItemsPanelTemplate>
    <ItemsWrapGrid />
    </ItemsPanelTemplate>
   </ListView.ItemsPanel>
  </ListView>
</DataTemplate>
</HubSection>
```

-- The fifth Hub Section again defines a custom data template for curated data. In the default app, this is hardcoded. This section also has a collage of images with hard coded links to assets in the app. Also notice the custom Hub Section Header. Mind you these are all samples of how you can represent data via the Hub Control. *None of it is mandatory* and you can have as many or as few Hub Sections as you deem necessary.

-- The sixth Hub Section maps to 'Section 4' in the hub layout and is again using custom data binding.

-- The last section uses the standard310x260Template data template and lists out the 6th group of the grouped dataset. Again index 5 here is chosen at random and hardcoded.

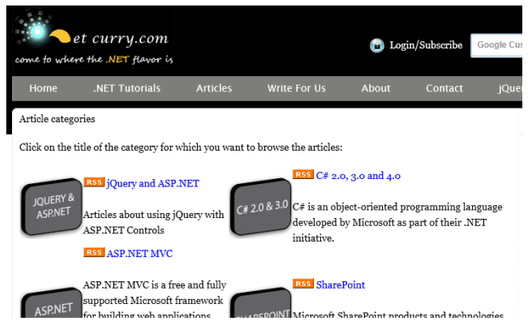
```
<HubSection DataContext="{Binding Path=[5],
Source={StaticResource
 groupedItemsViewSource}}"  Padding="40,30,150,44">
 <HubSection.Header>
  <TextBlock x:Uid="Section5Header" TextLineBounds=
   "TrimToBaseline"
   OpticalMarginAlignment="TrimSideBearings"
   Text="Section 5"/>
 </HubSection.Header>
 <DataTemplate>
  <GridView x:Name="itemGridView"
   Margin="-13,-4,0,0"
   AutomationProperties.AutomationId="ItemGridView"
   AutomationProperties.Name="Items In Group"
   ItemsSource="{Binding Items}"
   ItemTemplate="{StaticResource
   Standard310x260ItemTemplate}"
   SelectionMode="None"
   IsSwipeEnabled="false"
   IsItemClickEnabled="True"
   ItemClick="ItemView_ItemClick">
  </GridView>
 </DataTemplate>
</HubSection>
```

That was a glimpse of what we get from the Hub Control and the Hub App template out of the box! The 7 templates are there by default to showcase different possibilities. You don't have to bend your data to fit the template, rather you can easily pick and choose the template you want to use and build your own, if you want to.

Now that we've got an idea of how the Hub Template is structured, let's setup our data and then bind it to the hub.

SETTING UP YOUR DATA

www.dotnetcurry.com has different RSS feeds for each category of articles. To create a nice Feed reader that shows a glimpse of all categories, we can use these Feed URLs as our data sources.



Then we have the Home Page Feed that shows feeds of the latest articles. From this feed, we will use the first item as the Hero Item. With the sources decided, the entities to hold the Feed Data is defined as follows:

```
public class FeedDataItem
{
 public FeedDataItem(String uniqueId, String title,
   String subtitle, String
  contentPath, String description, String content)
 {
  this.UniqueId = uniqueId;
  this.Title = title;
  this.Subtitle = subtitle;
  this.Description = description;
  this.ContentPath = contentPath;
  this.Content = content;
 }

 public string UniqueId { get; private set; }
 public string Title { get; private set; }
 public string Subtitle { get; private set; }
 public string Description { get; private set; }
 public string ImagePath { get; private set; }
 public string ContentPath { get; set; }
 public string Content { get; private set; }
 public override string ToString()
 {
  return this.Title;
 }
}
```

The FeedItem class gets each result from a given RSS Feed. It contains the Title, Subtitle (in which we store Author), Description (the article Summary) and ContentPath, the actual URL of the article. Rest of the properties can be removed. I just kept them to keep the default UI Bindings working.

Next we have the FeedDataGroup class.

```
public class FeedDataGroup
{
 public FeedDataGroup(String uniqueId, String title,
  String subtitle, String feedPath, String description)
 {
  this.UniqueId = uniqueId;
  this.Title = title;
  this.Subtitle = subtitle;
  this.Description = description;
  this.FeedPath = feedPath;
  this.Items = new ObservableCollection<FeedDataItem>();
 }
 public string UniqueId { get; private set; }
 public string Title { get; private set; }
 public string Subtitle { get; private set; }
 public string Description { get; private set; }
 public string ImagePath { get; private set; }
 public string FeedPath { get; set; }
 public ObservableCollection<FeedDataItem> Items { get;
 private set; }

 public FeedDataItem HeroItem
 {
  get
  {
   if (Items.Count > 0)
   {
    return Items[0];
   }
   return null;
  }
 }
 public override string ToString()
 {
  return this.Title;
 }
}
```

This Class is modeled from the SampleDataGroup class that comes by default. It stores a UniqueId, Title, SubTitle, Description and the FeedPath or the URL to the Feed source.

The Items collection is the list of FeedItem objects based on the data returned by the Feed. Note the HeroItem property added for easy data binding. This always points to the first element in the Items array.

Next we have a helper class called FeedItemIterator. This class (inspired by the Atom Feed Sample from SDK) contains helper methods to loop through a Feed and return appropriate data once the Feed has been loaded.

```
class FeedItemIterator
{
 private SyndicationFeed feed;
 private int index;
 public FeedItemIterator()
 {
  this.feed = null;
  this.index = 0;
 }
 public void AttachFeed(SyndicationFeed feed)
 {
  this.feed = feed;
  this.index = 0;
 }
 public bool MoveNext()
 {
  if (feed != null && index < feed.Items.Count - 1)
  {
   index++;
   return true;
  }
  return false;
 }

public void MovePrevious()
{
 if (feed != null && index > 0)
 {
  index--;
 }
}

public bool HasElements()
{
 return feed != null && feed.Items.Count > 0;
}

public string GetTitle()
{
 // Nothing to return yet.
 if (!HasElements())
 {
     return "(no title)";
 }

 if (feed.Items[index].Title != null)
 {
     return WebUtility.HtmlDecode
     (feed.Items[index].Title.Text);
 }

 return "(no title)";
}

public string GetContent()
{
 // Nothing to return yet.
 if (!HasElements())
 {
    return "(no value)";
   }
   else if ((feed.Items[index].Content != null) &&
    (feed.Items[index].Content.Text != null))
   {
       return feed.Items[index].Content.Text;
   }
   else if (feed.Items[index].Summary != null &&
    !string.IsNullOrEmpty(feed.Items[index].Summary.
    Text))
   {
       return feed.Items[index].Summary.Text;
   }
 return "(no value)";
}

public string GetIndexDescription()
{
 // Nothing to return yet.
 if (!HasElements())
 {
     return "0 of 0";
 }

 return String.Format("{0} of {1}", index + 1, feed.
Items.Count);
}

public Uri GetEditUri()
{
 // Nothing to return yet.
 if (!HasElements())
 {
     return null;
 }

 return feed.Items[index].EditUri;
}

public SyndicationItem GetSyndicationItem()
{
 // Nothing to return yet.
 if (!HasElements())
 {
  return null;
```

```
 }
 return feed.Items[index];
}

internal string GetAuthor() {
 if (feed.Items[index].Authors != null &&
  feed.Items[index].Authors.Count > 0)  {
  string authors = "";
  foreach (var author in feed.Items[index].Authors) {
   authors += (author.NodeValue + ",");
  }
  return authors.TrimEnd(',');
  }
  return "";
 }

 internal string GetUrlPath()  {
  if (feed.Items[index].Links != null &&
   feed.Items[index].Links.Count > 0)   {
   return feed.Items[index].Links[0].Uri.AbsoluteUri;
  }
  return "";
 }
}
```

LOADING FEEDS

Finally we have our FeedDataSource class that encapsulates the Data loading operations. The GetFeedDataAsync class is at the heart of the class and loads the actual data. The Groups property represents the datasource.

As we can see below, we have take seven of the available Feed URLs and created a FieldGroup out of each. The First group is the Home Page feed, and the rest are assorted Feeds of various tags/categories. Next we loop through each group and load the feed items, which represent one article each:

```
private async Task GetFeedDataAsync() {
 FeedItemIterator feedIterator = new FeedItemIterator();
 if (this._groups.Count != 0)
     return;
 Groups.Add(new FeedDataGroup("whatsnew", "What's New",
 "Latest articles on DNC",
 "http://feeds.feedburner.com/netCurryRecentArticles",
 "Latest DNC articles"));

 Groups.Add(new FeedDataGroup("aspnetmvc", "ASP.NET
 MVC", "Latest ASP.NET MVC  articles","http://www.
dotnetcurry.com/GetArticlesRss.aspx?CatID=67","Latest
 ASP.NET MVC articles"));

 Groups.Add(new FeedDataGroup("winrt", "Windows 8 and
   8.1", "Articles for Windows 8 and 8.1 Store Apps",
  "http://www.dotnetcurry.com/
   GetArticlesRss.aspx?CatID=75", "Articles for Windows
   8 and 8.1 Store Apps")); //WinRT

 Groups.Add(new FeedDataGroup("azure", "Windows Azure",
  "Windows Azure Tutorials",
  "http://www.dotnetcurry.com/
   GetArticlesRss.aspx?CatID=73", "Windows Azure
  Tutorials")); //Windows Azure

 Groups.Add(new FeedDataGroup("aspnet", "ASP.NET and
  WebAPI", "Web API articles",
  "http://www.dotnetcurry.com/
   GetArticlesRss.aspx?CatID=54", "ASP.NET Web API
  Articles")); //ASP.NET

 Groups.Add(new FeedDataGroup("vstfs", "Visual Studio
  and Team Foundation Server", "Visual Studio and TFS
  Articles",
  "http://www.dotnetcurry.com/
   GetArticlesRss.aspx?CatID=60",  "Visual Studio and
  Team Foundation Server"));  //Visual Studio

Uri dataUri = new Uri("ms-appx:///DataModel/SampleData.
json");
 Uri resourceUri;
 foreach (var group in Groups)  {
  if (!Uri.TryCreate(group.FeedPath, UriKind.Absolute,
   out resourceUri))   {
    return;
  }
  try {
   feedIterator.AttachFeed(await
   FeedDataSource.GetClient().RetrieveFeedAsync(
    resourceUri));
   while (feedIterator.MoveNext())   {
    FeedDataItem item = new FeedDataItem(group.
    UniqueId, feedIterator.GetTitle(),
    feedIterator.GetAuthor(),
    feedIterator.GetUrlPath(),
    feedIterator.GetContent(),
    feedIterator.GetContent());
    group.Items.Add(item);
   }
  }
  catch (Exception ex)  {
   throw;
  }
}
```

The DataSource is created and initialized in the HubPage.xaml.cs classes' NavigationHelper_LoadState method.

```
private async void navigationHelper_LoadState(object
sender, LoadStateEventArgs e) {
 var feedDataGroups = await FeedDataSource.
GetGroupsAsync();
 this.DefaultViewModel["Groups"] = feedDataGroups;
}
```

With our custom data source setup, if we run the application as is, we'll see that Section 2 and 5 are getting Live data from the Feeds. Why? That's because these two sections were using Data binding that were bound to the Groups data. Rest of the Sections were hard coded.

To test out our HeroItem property, let's data bind the very first section to show the latest article published.

As highlighted below, we've added a DataContext to the HubSection and based on the context, we have bound the Title, Subtitle and Description properties of the HeroItem of the very first Feed Group; which in our case is the site wide feed.

```
<HubSection Width="580" Padding="120,30,40,44"
DataContext="{Binding Path=[0],
 Source={StaticResource groupedItemsViewSource}}"
 VerticalAlignment="Top" >
 <HubSection.Header>
  <TextBlock TextLineBounds="TrimToBaseline"
   OpticalMarginAlignment="TrimSideBearings"
   Text="Latest Article"/>
 </HubSection.Header>
 <DataTemplate>
  <Grid Margin="0,10,0,0">
   <Grid.RowDefinitions>
    <RowDefinition Height="Auto" />
    <RowDefinition Height="*" />
   </Grid.RowDefinitions>
   <Image Source="Assets/MediumGray.png" Stretch="Fill"
   Width="420" Height="280"/>
   <TextBlock Grid.Row="1" Style="{StaticResource
   SubheaderTextBlockStyle}"
    TextWrapping="Wrap" Margin="0,10,0,0"
    Text="{Binding Path=HeroItem.Title}"/>
   <TextBlock Grid.Row="2" Margin="0,10,0,0"
   Style="{StaticResource
    TitleTextBlockStyle}" Text="{Binding
    HeroItem.Subtitle}"/>
   <TextBlock Grid.Row="3" Style="{StaticResource
     BodyTextBlockStyle}"
    Text="{Binding HeroItem.Description}"/>
  </Grid>
 </DataTemplate>
</HubSection>
```

Next we'll bind Hub Section 2 to the same feed group but now it will show a list of latest items. The current Hub Section 3 is not very practical for us because we currently don't have images to show.

However we can replicate Section 4 to show the Hero item of each feed. Be sure to remove the `x:Uid="ItemTitle"` from each TextBlock element, else it will get the data from the Resource file and for strange reason, ignore the data-binding (I would have thought successful data-binding overrides everything else).

Once we run the application now, we'll see that it is showing us the Hero units for each Feed Group. But there is no way to navigate to the Feeds! This brings us to the next section – Navigation.

HOOKING UP NAVIGATION

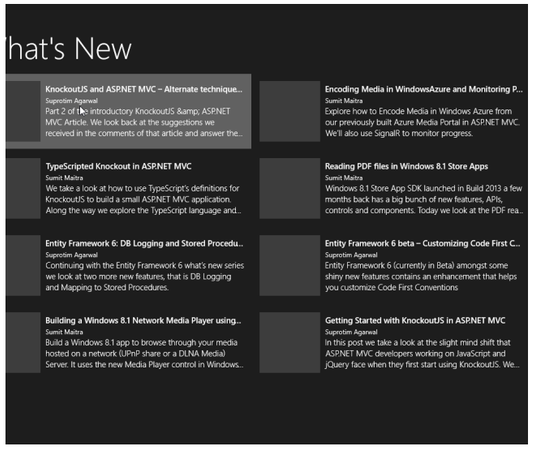
Actually hooking up Navigation is pretty easy. It is already done for us in Section 2. All you have to do is set the `IsHeaderInteractive="True"` property. This will raise the HubSectionHeader_Click event which has this default implementation

```
void Hub_SectionHeaderClick(object sender,
HubSectionHeaderClickEventArgs e)
{
   HubSection section = e.Section;
   var group = section.DataContext;
   this.Frame.Navigate(typeof(SectionPage),
((FeedDataGroup)group).UniqueId);
}
```

This simply looks up the FeedDataGroup item for the header and passes it to the SectionPage. So for all our headers, we can enable the IsHeaderInteractive flag.

If we run the App now, we can see that the "What's New" Section has a '>' indicating that it is an actionable link

Clicking on "What's New" takes us to the Sections Page with the list of Feeds.



However clicking on a FeedItem navigates to a blank page. Ideally we would like to see the article content. Let's see what we can do to show the Article content.

Adding a WebView to ItemsPage.xaml and Rendering Feed Link

If we navigate to SectionsPage.xaml.cs and check the ItemView_Click event, we'll see that it's passing the UniqueId property to the Items page. We'll take a shortcut here and instead of passing the UniqueId, we'll pass the entire FeedItem itself.

```
void ItemView_ItemClick(object sender,
ItemClickEventArgs e) {
 var item = (FeedDataItem)e.ClickedItem;
 this.Frame.Navigate(typeof(ItemPage), item);
}
```

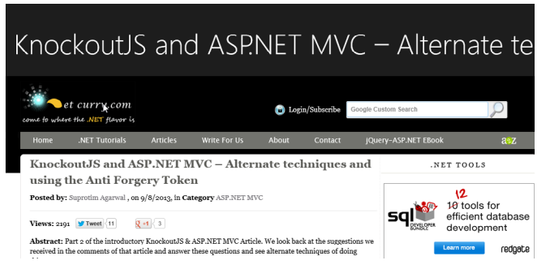
Now we move on to the SectionPage.xaml and add a WebView to the content section

```
<Grid Grid.Row="1" x:Name="contentRegion">
 <WebView Name="ContentWebView" Margin="0,3,0,0"></
WebView>
</Grid>
```

Finally in the navigationHelper_LoadState event, we update the code to load the URL into the Web View

```
private void navigationHelper_LoadState(object sender,
LoadStateEventArgs e) {
 var item = (FeedDataItem)e.NavigationParameter;
 this.DefaultViewMode["Item"] = item;
 if(!string.IsNullOfEmpty(item.ContentPath))
 {
  WebView contentWebView = this.
  FindName("ContentWebView") as WebView;
  if(contentWebView!=null)
  {
   contentWebView.Navigate(new Uri(item.ContentPath));
  }
 }
}
```

All set, if we run the app now and Navigate to an article from the ItemsPage, we'll see the following View:



Pretty neat, we have our very **own Feed Reader App** ready to go.

Conclusion

We covered a lot of ground today and it is worth doing a recap to make sure we didn't miss anything:

1. We learnt about the new Hub Control and the Hub App Template
2. We saw how to configure and layout Hub Sections, Headers, Hero Image etc.
3. We built a rudimentary Feed Reader using the built in Syndication APIs.
4. We data bound the Hub Control Sections to show up Feed Data.
5. Finally we used a WebView control to render the final Link in the Feed. Overall we got to see how we could leverage the Hub Application template to build rich, interactive, full screen Windows Store Apps ■

Download the entire source code from our GitHub Repository at bit.ly/dncm9-win81hub

Sumit is a .NET consultant and has been working on Microsoft Technologies for the past 12 years. He edits, he codes and he manages content when at work. C# is his first love, but he is often seen flirting with Java and Objective C. You can Follow him on twitter @sumitkm and read his articles at bit.ly/KZ8Zxb

PHOTO COURTESY OfficeOnline.com

SOFTWARE IS NOT A BUILDING

Johnathan had never purchased custom software in 20 years of business. But after months of searching for an off-the-shelf solution, he had given up. He was anxious to get things going and wondered how the process worked. He had decided to go with No-Glitch Software to do the work and their top programmer, Richard, would arrive in no time for their first meeting.

When Richard arrived, he was shown around the office and manufacturing floor to get a feel for how things worked, then they went to Johnathan's office to talk about the process.

"Well!", Richard said, "creating customer software is a lot like construction of a building. We have to do some planning. This is similar to the blueprints and site planning. During this phase, we'll sketch out all the screens and reports and create a prototype. When that's done, we'll start the work, which is like an actual construction process. We'll create everything, then install it so you can do some testing. That's when we enter the bug fix stage and finally, after that, you'll be up and running. The whole thing should take about six months."

The analysis process ended up taking longer than expected. Once the prototype was delivered and approved, No-Glitch Software was already a month behind schedule. Johnathan was surprised when the prototype code was thrown away and Richard started coding all over again. But the coding process seemed to be going well, at lease he assumed it was based on the bills he kept getting from No-Glitch.

Then, about a month before the test version was to be delivered, Johnathan learned that interface to the key piece of machinery had changed and that would require a change to his new program. He called Richard and he did not seem happy.

"That will require some major changes because the interface is a key part of the application and it is used in all over in the application. I estimate it will take an additional month."

Does this all sound familiar to you? Have you ever compared software development to building construction? I have, but I don't do it anymore.

There are a lot of things wrong with the above story. Let's see if we can figure out what's going on.

What is the problem?

For many years, I have heard software development compared to building a house or a building. A plan needs to be put into place, a good foundation, and solid walls that can support the roof. It will take months of work just to get a good plan ready before the real work begins.

The comparison even goes further. Software development even uses many of the same terms as Construction. Both have architects, architecture, engineers, plans, contractors, scaffolding, design, etc. That's not to say these words are bad. ***It's the comparison between construction and software development that's bad.***

You see, construction has fixed rules and regulations. The plans are set in stone and are difficult to change. The problem is, construction is with few exceptions, run using the Waterfall methodology. And this is bad for software as Waterfall does not allow change. It does not allow enhancements. Maintenance is difficult and best practices are difficult to follow.

When you use Waterfall project management in software, it leads to code bloat, rotting code, rewrites, spaghetti code, bugs, dissatisfied customers, dissatisfied managers, schedule overruns, cost overruns, and technical debt. (I'll discuss technical debt in more detail in a *future* column.) The bottom line is using Waterfall processes is like going over Niagara Falls in a barrel. Few people that did it, lived to tell about it.

Other problems with our story exists. First, the interface to the machinery was not isolated to a single area of the application. Time was also wasted on a prototype.

So, what are we to do? As a start, let's change our definition. Stop comparing software development to Construction and start comparing it to Gardening.

A new definition

I first came on this idea about a few years ago while re-reading the book Pragmatic Programmer. When talking about refactoring, the authors said:

> *Rather than construction, software is more like gardening – it is more organic than concrete.*

As I started thinking about this, I realized they were right. *Software is organic*. But the authors didn't carry this concept far enough. Gardening can be applied to many other areas of development, not just refactoring. I then started speaking about Software Gardening at conferences and code camps. The topic was hit. Many attendees came up to me afterwards and said the comparison was right. Some even said it was the best presentation of the conference.

Then earlier this year, I started reading the book Object Oriented

Software Guided by Tests. In the foreword, Kent Beck writes

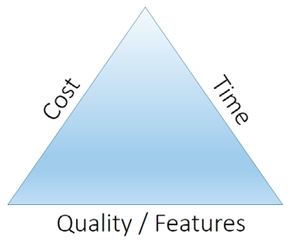
> *What if software wasn't "made", like we make a paper airplane – finish folding it and fly it away? What if, instead, we treated software more like a valuable, productive plant, to be nurtured, pruned, harvested, fertilized, and watered? Traditional farmers know how to keep plants productive for decades or even centuries. How would software development be different if we treated our programs the same way?*

After reading this, I was positive that I was onto something. After all, here is one of the great minds in software saying the same thing I have been saying for over three years. But I needed to spread the word to a larger audience. And thus, here I am, writing a regular column for the DNC Magazine and .NET Curry.

Software is organic

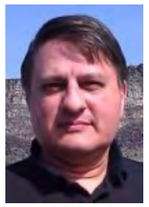
So what does it mean that software is organic? Think about a plant. It grows and changes in unpredictable ways. It may change with the seasons. It may bear fruit or shade or beauty. But we also have to care for it properly. Think about what Kent Beck said in the above quote. We need to nurture, prune, harvest, fertilize, and water our software. We also need to remove the weeds, keep the insects away, give it light, the proper soil, plant the right seeds, and use the right tools. By treating software as something organic, we expect change. We learn how to care for it so that when that change comes, we're ready for it.

You are probably familiar with the software development dilemma. If not, think of a triangle. One side is labeled time. Another is money. The third is either features or quality.



The old saying is that *you can have two of the three, so pick them.* I propose that if you treat Software as a Garden, you can have all three.

Software Gardening is about all this and more. I'm excited to be writing this new column and have it come to you every couple of months. Along the way I will talk about soil, light, water, pruning, weeding, tools, and more. Apply the concepts you will learn here. If you do, your software will be lush, green, and vibrant ■



Craig Berntson is the Chief Software Gardener at Mojo Software Worx, a consultancy that specializes in helping teams get better. He has spoken at developer events across the US, Canada, and Europe for over 20 years. He is the co-author of "Continuous Integration in .NET" available from Manning. Craig has been a Microsoft MVP since 1996. Email: craig@mojosoftwareworx.com, Blog: www.craigberntson.com/blog, Twitter: @craigber. Craig lives in Salt Lake City, Utah.

5 REASONS YOU CAN GIVE YOUR FRIENDS TO GET THEM TO SUBSCRIBE TO THE

DNC MAGAZINE

(IF YOU HAVEN'T ALREADY)

01 It's free!!! Can't get anymore economical than that!

02 Every issue has something totally new from the .NET world!

03 The magazines are really well done and the layouts are a visual treat!

04 The concepts are explained just right, neither spoon-fed nor too abstract!

05 Through the Interviews, I've learnt more about my favorite techies than by following them on Twitter

Subscribeat
www.dotnetcurry.com/magazine

INTERVIEW

DAVID FOWLER

Dear readers, in our last issue we had ASP.NET head honcho Scott Hunter, and this month we are glad to have another illustrious member of the ASP.NET team who is also the Creator and a prolific co-maintainer of the Open Source SignalR framework (which is now officially supported by Microsoft). David is here today in his personal capacity but that doesn't stop us from asking him questions, trivia about his pet project SignalR.

DNC: Hello David, welcome to DNC Magazine, thanks for taking the time out for us. To get started, tell us how long have you been with Microsoft and how did you get started?
DF: I've been with Microsoft for 5 years now. I started out as an intern on the ASP.NET team. In the first year I was an SDET (tester), then the second year I was an SDE (developer).

DNC: Okay, lets hear it from the master, for the .NET newbies in our audience - What is SignalR? And some trivia - who designed the SignalR Logo; You, Damien or was it a community contribution?
DF: When SignalR was made public, we called it a "persistent connection abstraction over http" and that was technically correct, but it didn't really sell the technology. Today we say, SignalR is incredibly simple real time web for .NET (That was Damian's idea). Damian designed the logo.

DNC: Stepping back a little, how did the idea of SignalR come about? Tell us about the initial days and some of the critical decisions that worked out and maybe a few features that you had to revisit.
DF: I had been working on a collaborative web based IDE prototype in my free time and it was doing long polling using MVC 3. I couldn't figure out why such a well-known pattern was so hard to do correctly, so I started toying with the idea of making a library that would make it all simpler. At the same time, I saw a talk that Damian had given at MIX about doing proper async in ASP.NET and one of his examples was doing long polling. The next day, I walked over to his office and we threw around a few ideas and started working nights and weekends over IM on this new "real time technology". The rest is history.

When we were trying to come up with the programming model for SignalR (we had a few in the early days), it was all based on the types of apps we were trying to enable. At the time there was a real time framework on nodejs called nowjs and it looked really cool since they had two-way RPC. I really wanted to do something similar for SignalR, so I figured out how we could abuse *dynamic* in C# to achieve something similar. That was one of the better decisions we made that hasn't changed since.

One design we had to change was how the message bus worked. In earlier versions of SignalR, we wrote a pretty naïve implementation that sent messages to all connected clients inline:

```
lock(Subscriptions)
{
        foreach(var sub in Subscriptions)
        {
                context.Response.WriteJson(sub.Messages);
        }
}
```

It worked well, but it was really slow so we wrote a completely new implementation. It introduced a message broker that decoupled publish and subscribe. We also implemented a ring buffer for storing messages that was extremely fast and lock free.

DNC: Tell us some interesting 'war stories' during the development of SignalR v1 and v2 – browser idiosyncrasies, protocol implementation gaps, the most exasperating moments and the 'hell yeah' moments!
DF: SignalR v1 was brutal. Going from 0.x to a 1.0 was hard.

> *When we became a Microsoft project, we got some developers from the ASP.NET stress team to "stress SignalR" and boy did they stress it. We ran into all sorts of problems that would cause IIS to crash. We had crazy race conditions, several memory leaks and we even exposed some bugs in the ASP.NET runtime itself "*

In SignalR, we write to the HTTP response from a background thread and we had cases where two threads would try to party on the same response object causing things to explode. We had another issue where we ended up writing to the http response after the http request had ended and this also caused IIS to crash.

Then came the security review and the amount of things we had overlooked and had to fix.

We had some really interesting issues with the .NET client that we fixed in v2. The way that HttpWebRequest and the ServicePointManager queue outgoing requests, messed up the way SignalR worked in a bunch of cases. In particular, short running hub invocations would end up waiting behind long running requests and it caused a ton of problems. Luckily,

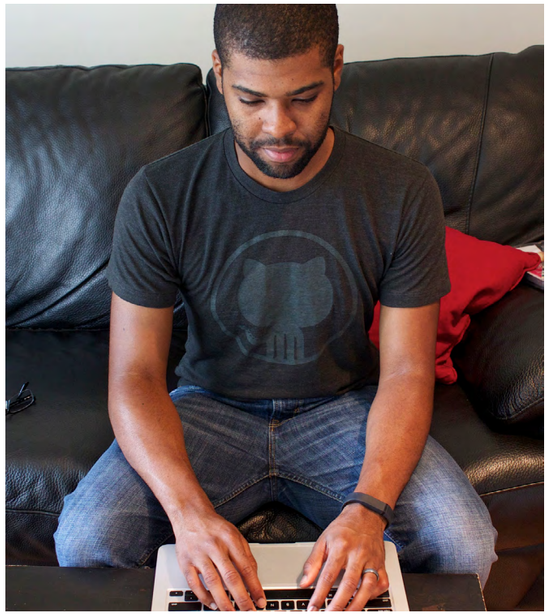
we have someone on our team who previously worked on HttpWebRequest, so we were able to resolve that problem.

Here's one from the JavaScript client: we handle document unload and we stop the connection immediately so that you can clean up any state you had associated with that connection. This failed on Firefox if you were using cross domain connections in certain situations (i.e. closing the tab, or closing the entire browser). Turns out that unload wasn't good enough so we tried beforeunload and that seemed to work… Except it only works for tab closing and we haven't figured out the magic sauce to make it work when closing the entire browser.

DNC: SignalR fills a nice void in the .NET stack and gaining official support makes it easier to adopt for a lot of people. Was the adoption of SignalR similar to adoption of NewtonSoft Json and other such libraries?
DF: Before it was adopted by Microsoft, some people were hesitant to use it because they weren't sure what kind of support they would get from a random OSS project. When we shipped item templates in Visual Studio, people began to take it more seriously as it was one of the default options from the Microsoft stack. Word spread quickly about it and the community uptake was huge.

DNC: Based on your interaction with the community on Jabbr, tell us more about some of the less talked about features in SignalR? (Tip, Tricks and Code samples welcome).
DF: The hub pipeline module is one of the lesser known features that is pretty powerful. You get full control of all of the hub invocations, new client connections, authentication, etc. At each stage of the pipeline, you are given an invocation and this gives you the ability to run code before/after it:

```
private class SamplePipelineModule : HubPipelineModule {
    public override Func<IHubOutgoingInvokerContext, Task>
    BuildOutgoing(Func<IHubOutgoingInvokerContext, Task>
    send)    {
        // Whenever there's an outgoing call to the client, log it
        return async context =>          {
            Debug.WriteLine("Before invoking {0} on {1}",
            context.Invocation.Method, context.Invocation.Hub);

            // Call the send invocation
            await send(context);

            Debug.WriteLine("After invoking {0} on {1}", context.
            Invocation.Method, context.Invocation.Hub);
        };
    }
```



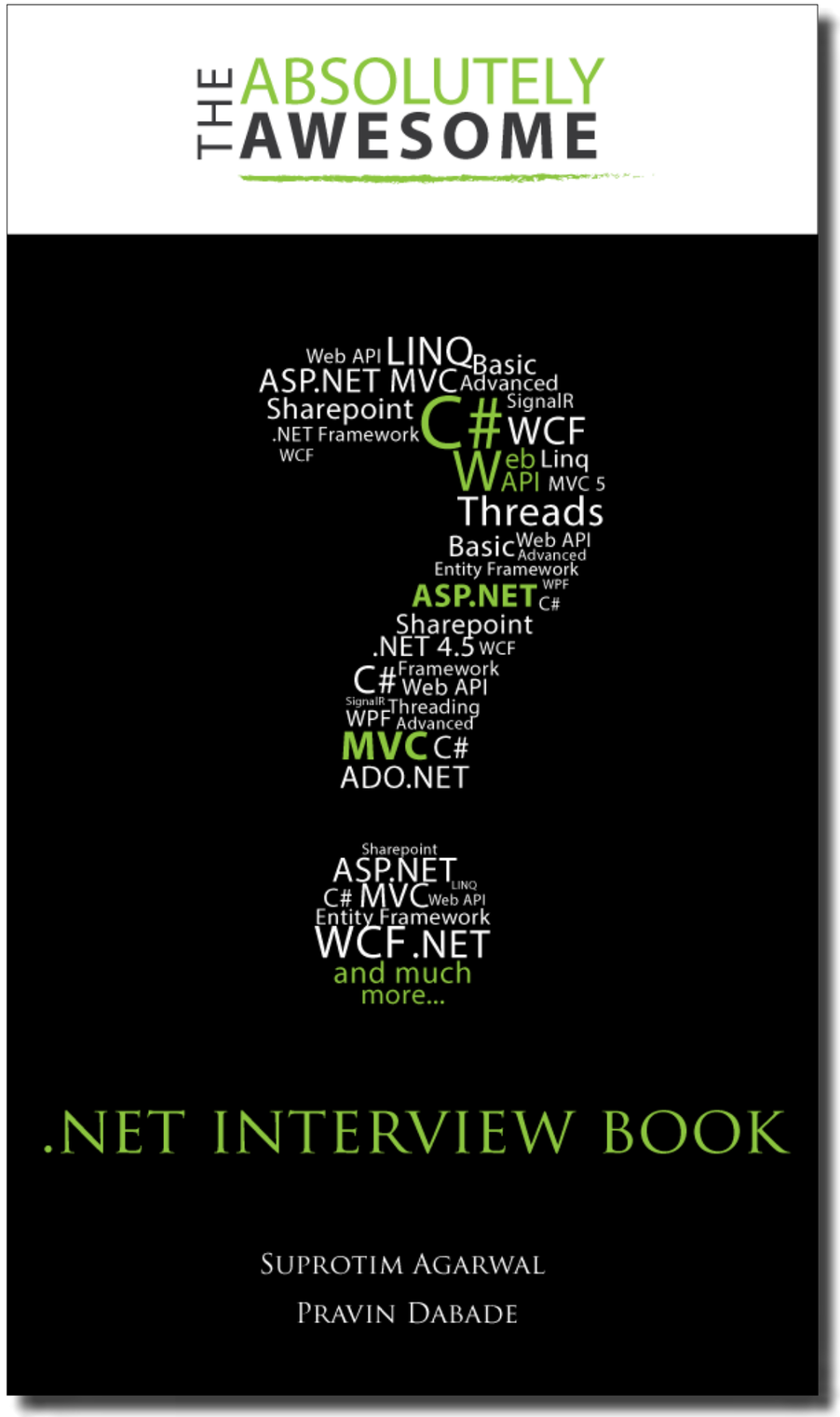
We actually had a hack week and one of the devs on the SignalR team wrote a pipeline module that compressed all payloads going over the wire.

DNC: Have things gotten easier in terms of support with official backing from MS? We know at one point you were sleeping only about four hours a day!
DF: Yep! We actually got our first support call the other day (I had a conversation with one of the customer support reps). I still give personal support to people that come to the jabbr room (for free!) whenever I'm in there, but I sleep more now.

DNC: From a comp science and engineering perspective, tell us some of your interests. SignalR covered JavaScript, C#, ASP.NET asynchrony and network protocols, that's a pretty nice smorgasbord. In an ideal world, what would you like to work on next?
DF: Probably something to expand my knowledge around *distributed* systems.

DNC: Okay, enough of work related questions, let's ease off a little, tell us how do you chill-out and relax?
DF: I play sports, Tennis, Softball, Football (soccer). I'm a bit of foodie so I like to try out new restaurants with my wife.

DNC: If David Fowler was not a Software Engineer, what would he be?
DF: A professional tennis player :) ■



CLICK HERE > www.dotnetcurry.com/interviewbook

ASP.NET MVC 5 AUTHENTICATION FILTERS

Raj Aththanayake explores the new IAuthenticationFilter which allows you to customize authentication within an ASP.NET MVC 5 application.

ASP.NET MVC 5 has some great improvements around authentication. This includes new Authentication filters, new Authentication options and ASP.NET Identity Management.

In this article, we will take a look at the new Authentication filters and how you can use these filters to make authentication decisions. At this stage, ASP.NET MVC does not provide any built-in authentication filter(s). However it provides you with the framework, so you can easily create your own custom Authentication filters.

If you have used ASP.NET MVC earlier, you probably have used AuthorizationFilters. Authorization filters allow you to perform authorization tasks for an authenticated user. A good example is Role based authorization.

ASP.NET MVC 4 also introduced a built-in AllowAnonymous attribute. This attribute allows anonymous users to access certain Controllers/Actions. This way, you can protect the entire site by using this Authorize attribute and then use the AllowAnonymous attribute, to allow anonymous users to access certain Actions and Controllers.

If you would like to read more on AllowAnonymous attribute, here's a good article http://blogs.msdn.com/b/rickandy/archive/2012/03/23/securing-your-asp-net-mvc-4-app-and-the-new-allowanonymous-attribute.aspx

New Authentication filters run *prior* to Authorization filters. It is also worth noting that these filters are the very first filters to run *before* any other filters get executed.

SO WHAT IS THE REAL REASON BEHIND AUTHENTICATION FILTERS?

Prior to Authentication filters, developers used Authorization filters to drive some of the authentication tasks for the current request. It was convenient because the Authorization filters were executed prior to any other action filters.

For example, before the request routes to action execution, we would use an Authorization filter to redirect an unauthenticated user to a login page. Another example would be to use the Authorization filter to set a new authentication principal, which is different from the application's original principal in context.

Authentication related tasks can now be separated out to a new custom Authentication filter and authorization related tasks can be performed using Authorization filters. So it is basically about *separation of concerns*, while giving developers more flexibility to drive authentication using ASP.NET MVC infrastructure.

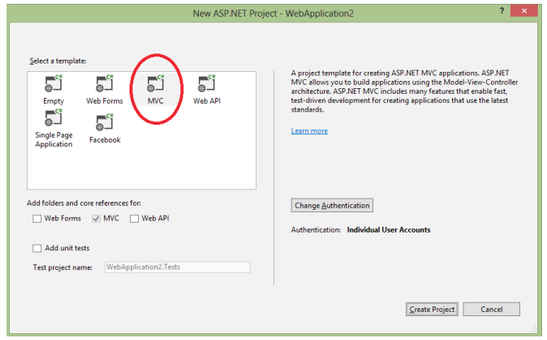
AUTHENTICATION FILTERS AS YOUR STANDARD ACTION FILTERS

As you would normally do with Action Filters, you can apply Authentication logic per Action, per Controller or Globally for all Controllers.

A practical example would be where your entire site runs on Forms based authentication using cookies, but you have a special requirement to support certain Controllers/Actions that should be transparent to a different authentication provider. This authentication provider would issue tokens along with the claim based authentication. Certain Controllers would have access to these claims principals. You can use a custom Authentication filter to set the new principal (i.e claims based), for the current request, just for the Controllers/Actions we need. The rest of the site will continue to work with the existing forms based authentication.

CREATING A NEW CUSTOM AUTHENTICATION FILTER WITH ASP.NET MVC 5

We will first create a new ASP.NET MVC 5 application. Once you have installed Visual Studio 2013, navigate to File > New Project, under templates (C#/VB) select Web, and then select ASP.NET Web application. A dialogue box will pop up. Select "MVC" from the template. Then simply click on "Create Project".



In order to create an Authentication filter, you must implement the *IAuthenticationFilter*. Below is the implementation of ASP.NET MVC's IAuthenticationFilter.

```
public interface IAuthenticationFilter {
void OnAuthentication(AuthenticationContext
filterContext);

void OnAuthenticationChallenge(
AuthenticationChallengeContext filterContext);
}
```

As I mentioned earlier, to be able to use these filters as your standard Action Filters, you also need to derive from *ActionFilterAttribute*. This allows you to decorate your Authentication filter in Controllers, Actions or use as a Global filter.

```
public class CustomAuthenticationAttribute :
ActionFilterAttribute, IAuthenticationFilter {
  public void OnAuthentication(
  AuthenticationContext filterContext)
  {  }

  public void OnAuthenticationChallenge(
  AuthenticationChallengeContext filterContext)
  { }
}
```

If you look at the above CustomAuthenticationAttribute, there are two interesting methods required to implement.

a) OnAuthentication
b) OnAuthenticationChallenge.

Let's take a look at these two methods in detail.

OnAuthentication

During Action invocation, ASP.NET MVC invokes Authentication Filters by calling the following method:

```
AuthenticationContext authenticationContext =
InvokeAuthenticationFilters(controllerContext,
filterInfo.AuthenticationFilters, actionDescriptor);
```

This method creates an AuthenticationContext using the original principal, and executes each filter's *OnAuthentication* method. Please see the following code:

```
AuthenticationContext context = new
AuthenticationContext(controllerContext,
actionDescriptor, originalPrincipal);

foreach (IAuthenticationFilter filter in filters) {
      filter.OnAuthentication(context);
}
```

AuthenticationContext provides you information for performing authentication. You can use this information to make authentication decisions based on the current context. For example, you may decide to modify the *ActionResult* to different result type based on the authentication context, or you may decide to change the current principal based on the authentication context etc.

If you change the user's current principal to use a new custom principal, the ASP.NET MVC framework will set a new principal as shown here:

```
IPrincipal newPrincipal = context.Principal;

if (newPrincipal != originalPrincipal) {
      Contract.Assert(context.HttpContext != null);
      context.HttpContext.User = newPrincipal;
      Thread.CurrentPrincipal = newPrincipal;
}
```

OnAuthenticationChallenge

OnAuthenticationChallenge method runs after the OnAuthentication method. You can use OnAuthenticationChallenge method to perform additional tasks on the request.

```
protected virtual AuthenticationChallengeContext
InvokeAuthenticationFiltersChallenge(
   ControllerContext controllerContext,
   IList<IAuthenticationFilter> filters,
   ActionDescriptor actionDescriptor, ActionResult
   result)
```

Similar to AuthenticationContext, this method also creates an AuthenticationChallengeContext as shown here. Then it invokes the OnAuthenticationChallenge method per AuthenticationFilter.

```
AuthenticationChallengeContext context = new
AuthenticationChallengeContext(controllerContext,
                actionDescriptor, result);

foreach (IAuthenticationFilter filter in filters) {
    filter.OnAuthenticationChallenge(context);
}
```

The key thing to remember is that OnAuthenticationChallenge does not necessarily run before every other Action Filter. It can run at various stages. For example, it can run after AuthorizationFilters or it can run after Action execution completed and so on. Since this is at various stages, you now have the ability to change the action result based on the authentication.

As I mentioned earlier, you can use the OnAuthenticationChallenge method to modify the ActionResult.

```
filterContext.Result = new HttpUnauthorizedResult();
```

IMPLEMENTING AND CONFIGURING YOUR CUSTOM AUTHENTICATION FILTER

We will use the *CustomAuthenticationAttribute* we just created and implement both *OnAuthentication* and *OnAuthenticationChallenge* methods. We will then configure it to execute only for HomeController's Index action. In *OnAuthentication,* we set the current principal to a custom principal with some additional properties. In *OnAuthenticationChallenge,* we check whether the user is authenticated or not and modify the ActionResult to an HttpUnAuthorizedResult.

CustomAuthenticationAttribute

```
public class CustomAuthenticationAttribute :
ActionFilterAttribute, IAuthenticationFilter {
public void OnAuthentication(
AuthenticationContext filterContext) {
//For demo purpose only.
filterContext.Principal = new MyCustomPrincipal
(filterContext.HttpContext.User.Identity, new []
{"Admin"}, "Red"); }

public void OnAuthenticationChallenge(
AuthenticationChallengeContext  filterContext) {
var color = ((MyCustomPrincipal) filterContext.
HttpContext.User).HairColor;
var user = filterContext.HttpContext.User;
if (!user.Identity.IsAuthenticated) {
filterContext.Result = new
HttpUnauthorizedResult();
}
}
```

HomeController:

Here we execute the CustomAuthenticationAttribute only for HomeController's Index action. By configuring the CustomAuthenticationAttribute this way, we have the ability to control authentication just for Index action.

```
public class HomeController : Controller {
    [CustomAuthentication]
    public ActionResult Index()      {
            return View();
    } }
```

OnAuthentication, implementation will modify the current principal to use the new MyCustomPrincipal. During the *Index* action, we will use the new MyCustomPrincipal. Any other action would use the original principal. For example, accessing the MyCustomPrincipal from any other action other than Index Action would not be possible. The following code would throw an exception:

```
public ActionResult About()       {
  var color = ((MyCustomPrincipal)
  ControllerContext.HttpContext.User).HairColor;
            return View(); }
```

OnAuthenticationChallenge method sets the ActionResult as HttpUnAuthorizedResult for un- authenticated users. This causes the user to redirect to a login page.

Additional Customizations

You probably know that the ASP.NET MVC Controller itself is also an ActionFilter.

```
public abstract class Controller : ControllerBase,
IActionFilter, IAuthenticationFilter,
IAuthorizationFilter, IExceptionFilter, IResultFilter,
IAsyncController, IAsyncManagerContainer { }
```

With the introduction of IAuthenticationFilter, you can customize your authentication within the Controller as shown here:

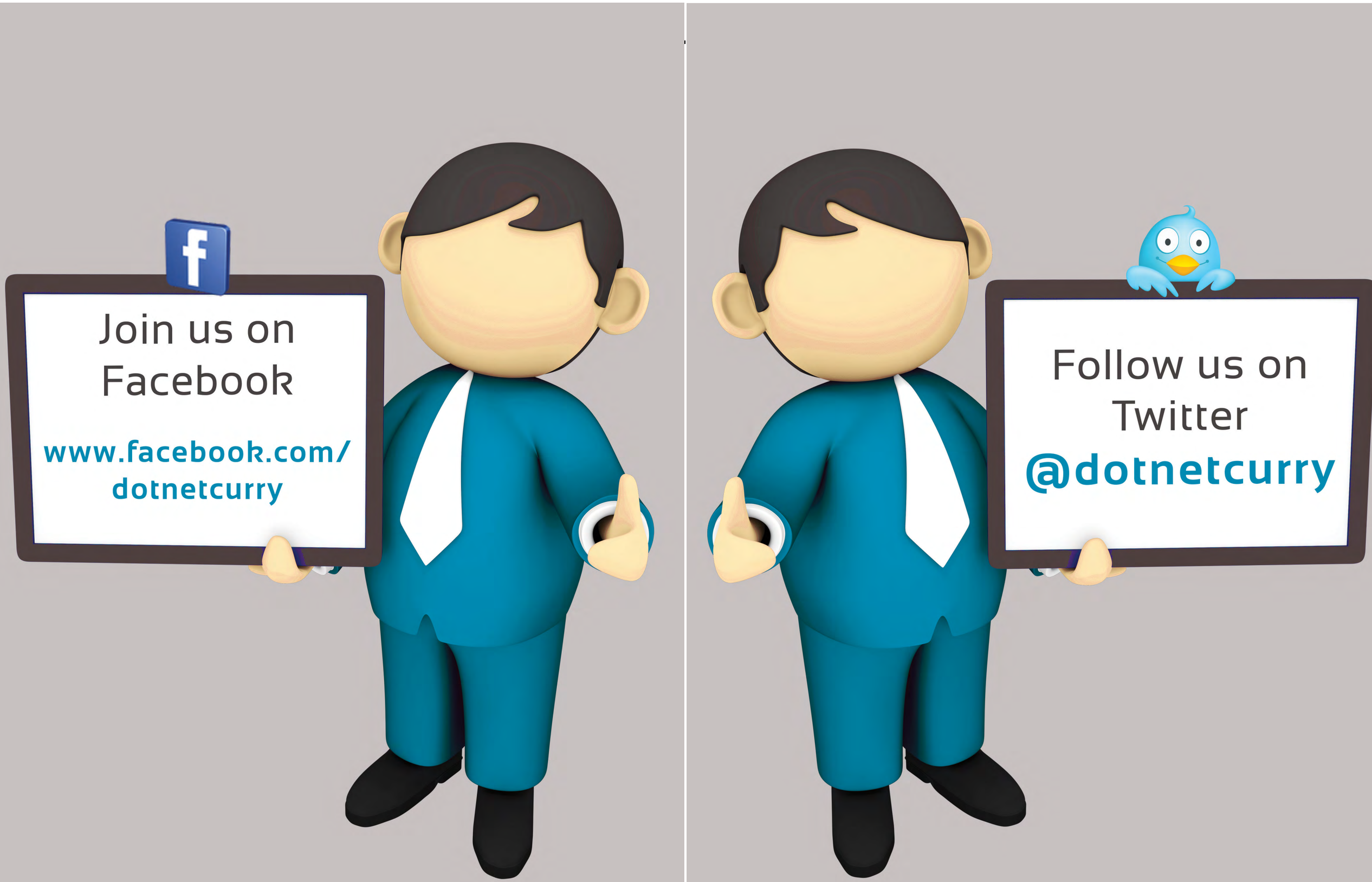
```
public class HomeController : Controller {
protected override void OnAuthentication(
AuthenticationContext filterContext) {
        //custom authentication logic
}

protected override void
OnAuthenticationChallenge(
AuthenticationChallengeContext filterContext) {
        //custom authentication challenge logic
}
}
```

Conclusion

The new IAuthenticationFilter provides an ability to customize authentication within an ASP.NET MVC 5 application. This leads to a clear separation between authentication and authorization filters. OnAuthentication and OnAuthenticationChallenge methods provide greater extensibility points to customize authentication within ASP.NET MVC framework. We also looked at a sample usage of CustomAuthentication attribute and how you can use to change the current principal and redirect unauthenticated user to a login page ■

Download the entire source code from our GitHub Repository at bit.ly/dncm9-mvc5auth

Raj Aththanayake is a Microsoft ASP.NET Web Developer specialized in Agile Development practices such Test Driven Development (TDD) and Unit Testing. He is also passionate about technologies such as ASP.NET MVC. He regularly presents at community user groups and conferences. Raj also writes articles in his blog http://blog.rajsoftware.com. You can follow Raj on twitter @raj_kba


Join us on
Facebook
www.facebook.com/
dotnetcurry
Follow us on
Twitter
@dotnetcurry

CODE OPTIMIZATION TOOLS IN VISUAL STUDIO 2013

Every developer strives to create code that is of high quality and at the same time maintain the level of productivity that is desired by the organization. Visual Studio 2013 provides a number of tools that help developers achieve these desired factors. In this article, VS ALM MVP ***Subodh Sohoni*** walks us through the details of these new tools and some tools that existed earlier, but got enhanced in Visual Studio 2013.

Development Productivity is always a matter of concern for the management of organizations that develop Software. The biggest portion of spending in development process is on code development, simply because the number of people involved in that activity are relatively very high compared to other roles. It becomes imperative for us to focus on improving productivity of code development. At the same time, it is also of utmost importance to the team, to make sure that the code developed, should be of the highest quality possible. There is a misconception in the minds of many people that productivity will suffer if we need to maintain high quality code. That is ***not true.*** If code quality is maintained right from the beginning, then the rework and bugs reduce. With the reduction of the bugs, time wasted on fixing those bugs and testing them again, also reduces. Finally the productivity improves.

IDE's like Visual Studio plays a big role in maintaining high quality of code and improving the productivity of coding at the same time. Visual Studio traditionally provides tools that are built-in to create and edit code, do compilation and debug the code that may have some bugs in it. Since Visual Studio 2005, Microsoft has provided some additional built-in tools for checking and maintaining quality of code as per policies and standards of the organization, developing code.

In Visual Studio 2013, Microsoft has included tools that provide visibility within the editor when a team is working on a project. It provides the status of the code, tests run on that code, other components that give reference to code under inspection, the relations between methods and also provides the tools to check whether the code follows certain rules or not. This visibility makes developers aware of *what they have not done* to improve the quality of code, without running time consuming manual testing.

CODE LENS

One of the most interesting features that is added in Visual Studio 2013 is called *CodeLens*. It provides the *related* status above every declaration of class and method.

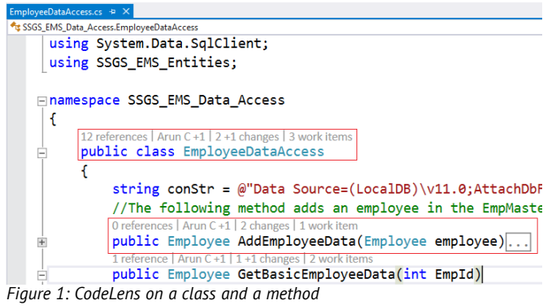


Figure 1: CodeLens on a class and a method

CodeLens contains a list of references to this code from other components. In the example shown in *Figure 1,* there are 12 references to the various methods in this class and one reference to the method *GetBasicEmpoyeeData*.

By selecting references, you can see the list of all references. When you click on a specific reference, it shows the details of the reference; like the file that has a method that *calls* the one under the lens, and some of the lines of code of that method, where the method under lens *is called*. If you double click on that reference line, it takes you to the calling code.

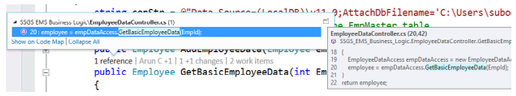


Figure 2: References details

Next detail provided by CodeLens is the history of the code. Questions answered in this history are:

• Who made the changes in the code? – Provides the persons responsible for the created and edited code. If you have *Lync* enabled, then you can even send an email, or chat with that person to know more about the code that he / she created.

• Under which changeset was that code checked in? When were the changes made?

• Which work items like tasks, bugs or requirements were associated with those changesets? – This provides the trace to know –
- Against which requirement was this code created?
- Which bug was fixed by this code?
- How much efforts were logged to create / edit this code?

Knowing answers to these questions is very important to the management and sometimes to the team members (depending upon the methodology being used).

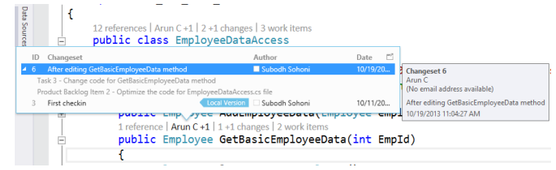


Figure 3: Code History – Who made the changes in the code under the lens?

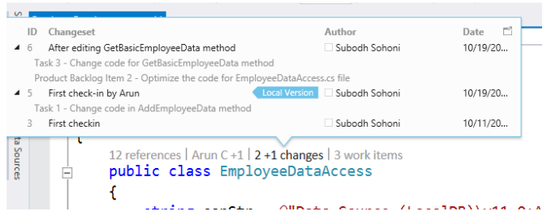


Figure 4: Code History – Under which changesets did the code under the lens get edited?

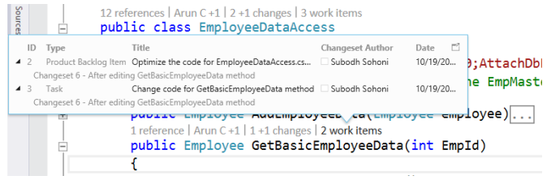


Figure 5: Code History – Which work items were associated with this code?

CodeLens also provides information about unit testing of the code under it. To view those details, let us now create a Unit Test. Unit testing itself is one of the most essential ways of maintaining quality of code. It is also called 'White Box Testing'. We write code that calls the method under test with *known output*, for *known input*. If the expected value and the actual return from the method are equal, then the test passes. Since it is a code that is used for testing, the same test can be repeated many number of times. It can also be data bound so that multiple inputs and outputs can be tested. Microsoft used to provide a Unit Test Generator up to Visual Studio 2010 when only MSTest framework was supported. In Visual Studio 2012, this generator was removed from the product as it could not support all the frameworks like NUnit, XUnit etc. that were allowed in that version of Visual Studio. There is a Unit Test Generator that is available from Visual Studio Gallery – it is available from the URL http://visualstudiogallery.msdn.microsoft.com/45208924-e7b0-45df-8cff-165b505a38d7 . Once you install this tool, a context menu (right click menu) for the method appears that allows you to generate a unit test for that method. It creates a test project, gives necessary references and also creates a class with appropriate attributes. We just need to provide the code for the test method, as follows:

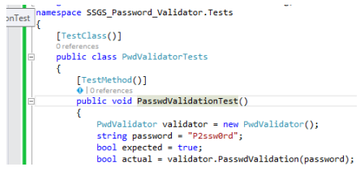
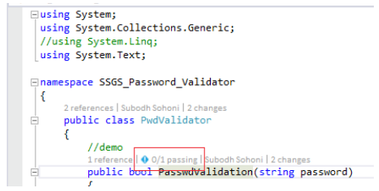


Figure 6: Unit Test Code

Let us now see what difference this has made to CodeLens.



You can see in *Figure 7* that it has added information about that test too, in CodeLens.

Figure 7: CodeLens – Unit Test Status

While we are discussing unit testing, we should also cover the topic of ***Code Coverage.*** Code Coverage is the number of lines of code from the target, tested by unit tests. Let us enable the code coverage for the assembly under test from the Test menu and run the test from Test Explorer. When the test passes, it reflects in the CodeLens, its code coverage results can be viewed and the code coverage is also shown with colors in the code editor. Blue for tested code and Pink for lines of code that were not tested.

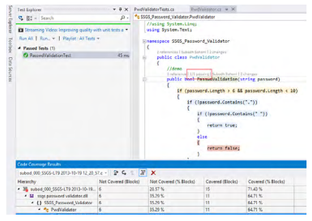


Figure 8: Effect of Unit Tests – CodeLens is updated, code coverage results and coloring

If we expand the CodeLens for the method that is tested, it shows the test details with the test run details.

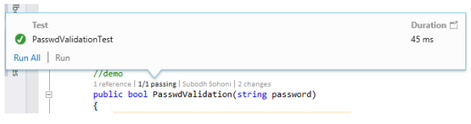


Figure 9: CodeLens after unit test run for the method that is tested.

Similar CodeLens update is also available for test method too.

CODE MAP

From CodeLens, we now will move to another tool. It is called *Code Map*. In an earlier version of Visual Studio, it was available as Directed Graph and Dependency Graph under *tools for architect*. Some similar features with more integration with code and debugger, has been introduced for developers as Code Map. From References of CodeLens, we can move to Code Map to show *relation between the referenced and referring methods*.

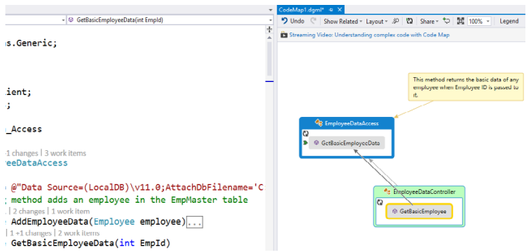


Figure 10: Code Map opened from CodeLens of a method. Shows calling and called methods with their containing classes

We can add nodes and comment nodes to the code map. We can also view the entire assembly that contains the called method and all the calling methods.

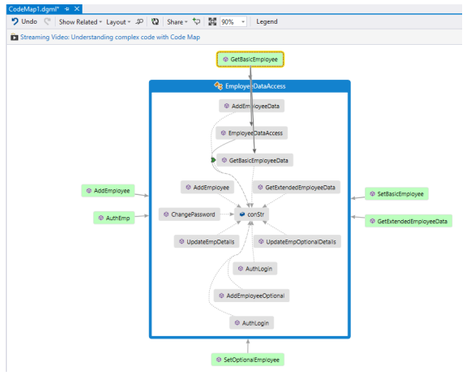


Figure 11: Code Map of Assembly and referencing methods.

We can analyze assemblies based upon various analyzers like

- Circular References Analyzer
- Find Hubs Analyzer
- Unreferenced Nodes Analyzer

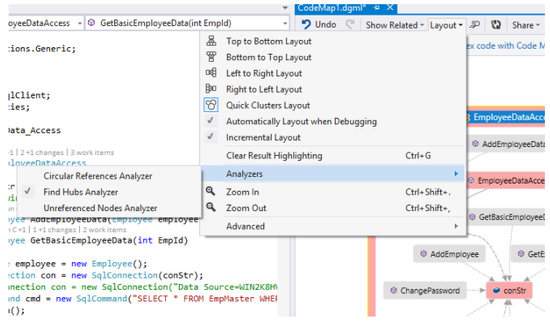


Figure 12: Code Map with analyzers

These analyzers make it possible to avoid certain errors that may have creeped in the code.

Another very important analysis that is required to be done is finding duplicate code and then removing it from wherever possible. Visual Studio 2013 provides that as one of the tools from 'Analyze' menu as 'Analyze Solution for Code Clones'.

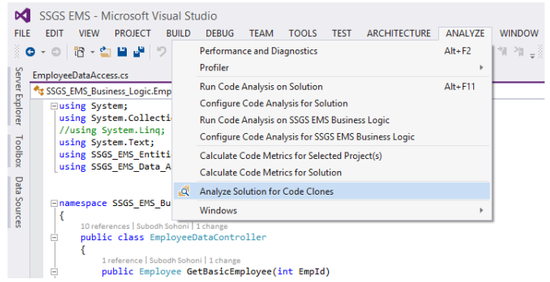


Figure 13: Code Clone Analysis

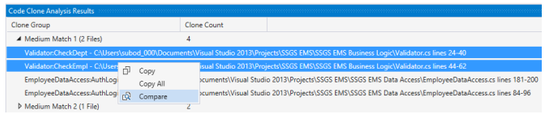


Figure 14: Code Clone Analysis Results and Context menu to compare two code snippets that are same

STATIC CODE ANALYSIS

From the same 'Analyze' menu, we can also open a more extensive tool called *Static Code Analysis*. This was present in the earlier versions of Visual Studio too, but now has been extended with many more types of rules that it evaluates. Categorization and filtering of the results is also provided in Visual Studio 2013.

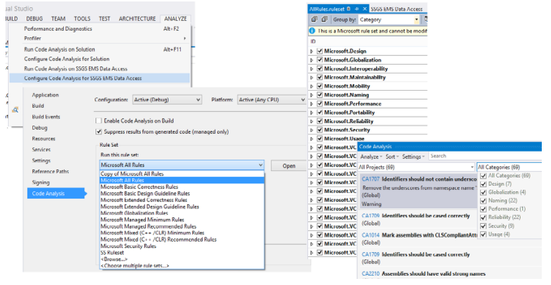


Figure 15: Static Code Analysis – Configuration and Results

Static Code Analysis can be evaluated from:

1. Context menu for any project or entire solution
2. As part of build within Visual Studio
3. As a Check-in policy for enforcement of quality check before allowing check-in
4. As part of build on the Team Foundation Server 2013

Visual Studio provides many rulesets that we can select to be evaluated in our code. We can also create a custom ruleset that represents the policies and standards that the organization has finalized. Consistency of code quality will be ensured because of evaluation of those rules, for all developers, at multiple places.

CODE METRICS

The last productivity tool that I am going to describe is Code Metrics. It provides a maintainability index to the developers. Importance of this tool will be understood better by the teams that maintain the code. If the code we develop is well maintainable, then the efforts that the maintenance team has to take, will be reduced. Code Metrics tool evaluates a maintainability index of the code based upon four parameters:

1. Length of code – Higher the length of code, lesser is it's maintainability
2. Depth of inheritance – Higher the depth of inheritance (from System.Object class) it reduces the maintainability
3. Cyclomatic Complexity – Higher the number of exit points in the code, lesser is the maintainability of code
4. Class Coupling – Tighter the coupling between two classes, more are the cascading code changes needed for one change in one class. Lesser the maintainability of code.

Considering these parameters and applying an algorithm that was originally developed by Carnegie Melon University, an Index of maintainability is worked out. Its value is between 1 to 100; *higher the better*, from the point of view of maintenance of code. When we evaluate it using the tool Code Metrics, it provides us the values for Solution, for Project, for NameSpaces, for Classes and for Methods. It provides the values of Maintainability index as well as of all those parameters. If some values are less, we can take corrective actions to make the code more maintainable.

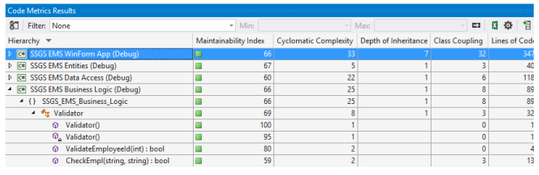


Figure 16: Results of Code Metrics

These are some of the tools available in various versions of Visual Studio 2013. The following table shows the availability of tools in the minimum version of Visual Studio 2013:

Coding Tool / Feature	Min. Version of Visual Studio 2013 in which the feature is available
CodeLens	Visual Studio 2013 Ultimate with MSDN
Code Map - Debugger Integration	Visual Studio 2013 Ultimate with MSDN
Unit Testing	Visual Studio 2013 Professional
Code Coverage	Visual Studio 2013 Premium with MSDN
Static Code Analysis	Visual Studio 2013 Professional
Code Metrics	Visual Studio 2013 Premium with MSDN

Conclusion

In this article, we took an overview of some tools that are part of Visual Studio 2013. We have seen the following new tools:

CodeLens for heads up display about code and data related to it

CodeMap to view code and relations between parts of the code graphically

Unit Testing for checking the functional correctness of code

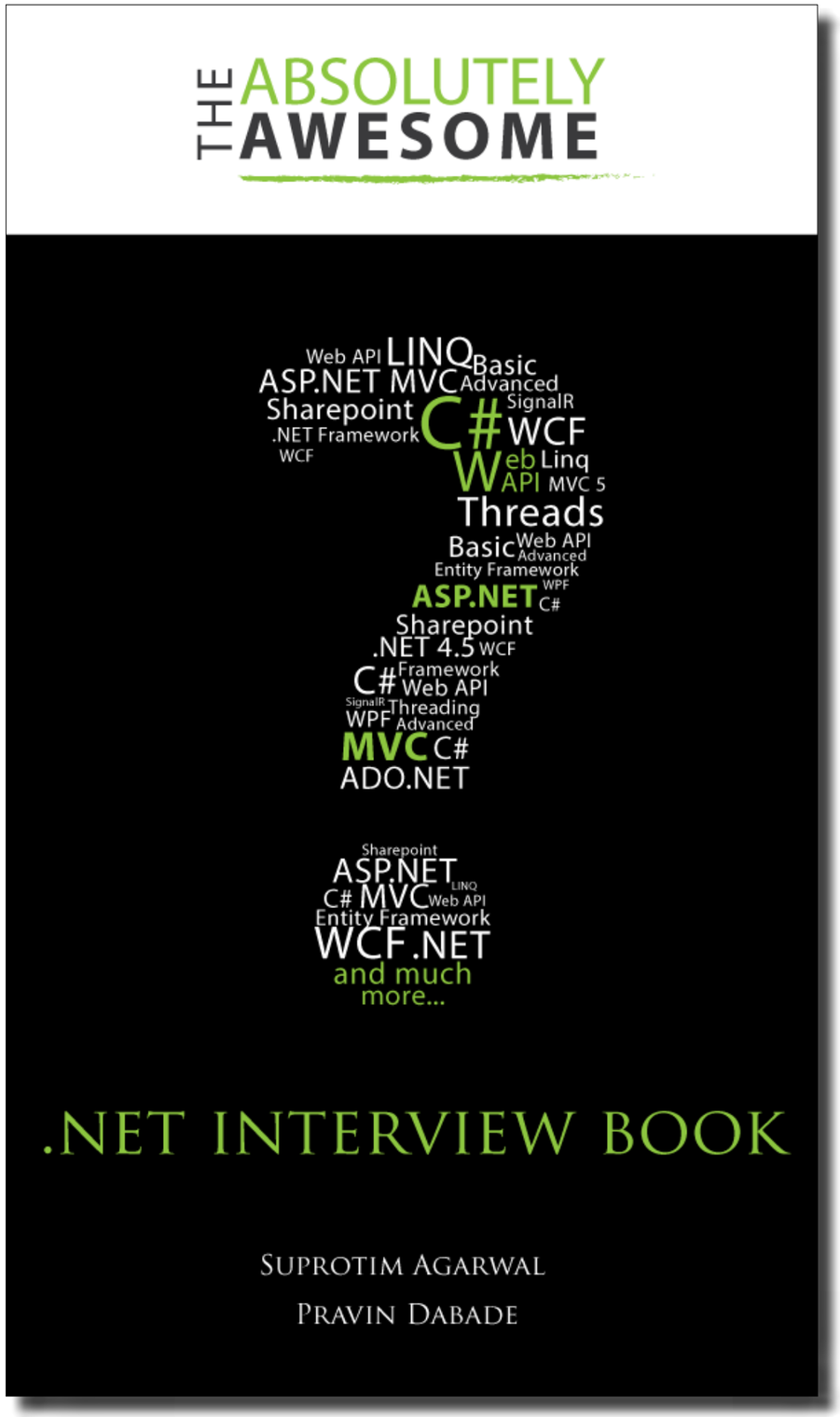
Code Coverage that provides the data about how many lines of code are tested

Code Analysis that ensures the consistency of code against organizational policies and industry best practices

Code Metrics computes the maintainability index of the code that you write

With the help of these tools, Visual Studio 2013 enables developers to write consistently high quality of code, with high productivity ■

Subodh Sohoni, is a VS ALM MVP and a Microsoft Certified Trainer since 2004. Follow him on twitter @subodhsohoni and check out his articles on TFS and VS ALM at http://bit.ly/Ns9TNU



CLICK HERE > www.dotnetcurry.com/interviewbook

USING POWERSHELL IN SHAREPOINT

In this article, *Pravin Dabade* takes a closer look at PowerShell and demonstrates some examples of using PowerShell as well as performing some basic administrative jobs in SharePoint.

Microsoft introduced PowerShell in 2003 as an advanced automation-script language which replaces legacy DOS batch files. Since PowerShell is built on top of the .NET Framework and makes use of programming concepts from C#/VB.NET, it becomes easier for a .NET professional to write scripts and manipulate .NET objects.

For the new enthusiasts who have just joined the party, let's get started with the question "Why Should I use PowerShell for performing operations in SharePoint?".

Administrators love Scripts and PowerShell allows you to script SharePoint via API's. That's a good enough reason to use PowerShell, right? With SharePoint 2010, Microsoft released several commandlets (cmdlets) to automate and administer SharePoint. Now with SharePoint 2013, PowerShell has become the de-facto standard for configuring SharePoint. In fact, there are certain operations in SharePoint that can be performed only in PowerShell, for example - Search Topology configuration on production servers.

This article is not a tutorial on PowerShell scripting. It rather gives an overview of how PowerShell commands work and then proceeds to show how these commands can be used to perform repeating administrative tasks in the SharePoint Environment.

For this article, let's consider a distinct set of audience, each having their own uses of PowerShell. For example, you have Administrators, IT Pros or Developers who would like to use PowerShell for different reasons.

The IT Pros may want to automate couple of tasks like maintenance tasks or site collection provisioning tasks. They can write PowerShell scripts and make it reusable in your organization's farm. Similarly your Administration department may request some features to be made available to Developers. This allows them to make use of PowerShell script to perform jobs.

Let's get started with some examples of using PowerShell to perform some tasks. There are around 500 commands (cmdlets) that can be used in SharePoint 2010 and about 800 commands in SharePoint 2013. Apart from the commands provided by Microsoft, there are several community command-lets which we can use. When you work with PowerShell, you must have appropriate permissions to execute the PowerShell commands.

PowerShell is included in Windows 7 & Windows Server 2008 R2 or higher machines and appears in the Windows Taskbar on Server machines. You can launch a plain vanilla PowerShell window by clicking on the PowerShell icon in the taskbar. To make this instance aware of SharePoint, you can add the SharePoint PowerShell add-in using

```
Add-PSSnapin Microsoft.SharePoint.PowerShell
```

In SharePoint 2013, a *SharePoint-branded PowerShell* is available with all cmdlets preinstalled called *SharePoint 2013 Management Shell*. You can access this shell in Windows 2008 R2 from Start > Microsoft SharePoint 2013 Products > SharePoint 2013 Management Shell. In Windows 2012, you can type SharePoint 2013 Management Shell to start the shell from Start Menu. This management shell is aware about SharePoint commands as well as general PowerShell Commands.

To see all the commands, just use the command -

```
Get-Command -PSSnapin Microsoft.SharePoint.
PowerShell
```


POWERSHELL

A COUPLE OF THINGS TO REMEMBER

- *While working with command-let, the syntax is not case sensitive.*
- *You can also make use of TAB to complete the commands in PowerShell.*
- *In case you are performing some critical operations using PowerShell, PowerShell has in-built protection for such commands which will ask you for a confirmation. But in some cases, while you are doing batch processing, you can override this behaviour by using [-Confirm:$false].*
- *You can use | [pipeline] character to pass the object to the next cmdlet.*
- *You can use aliases for a couple of commands like cls is for Clear-Host, Where or ? for Where-Object, gm for Get-Member etc.*

Now let's open PowerShell and try some commands as shown below -

Get-Command -pssnapin Microsoft.SharePoint.PowerShell	Displays all Command-lets
gcm Get-SP*	Displays all the core elements of the Farm.
Get-SPDatabase	Displays the SharePoint databases including SharePoint_Config database.
help Get-SPSite **help Get-SPSite -Full** **gcm Get-SPSite –Syntax**	To get the help for a specified command with various options.
Get-SPWebApplication http://itechy | Get-SPSite –Limit All	Displays all the site collections under a specified web application.
Get-SPSite "http://itechy/sites/biportal" | Get-SPWeb –Limit All	Display all the sites under the specified site collection.
Get-SPSite "http://itechy/sites/biportal" | Get-SPWeb – Limit All | Select Url,	Display the URL, Web Template and title of the site under the

When you work with PowerShell, most of the operations you perform are on .NET objects. In .NET, an object is described with the help of its characteristics [Properties] and the behaviours [Methods]. You can use the same when working with PowerShell. For example, you can use a *remove()* method to remove a SPSite.

DECLARE VARIABLES, FILTERS AND OPERATORS

Now let's see how to declare variables, filters and use different operators while working with PowerShell. The declaration is shown below –

Variable declaration –

- A variable declaration always starts with $.

- You can assign values to the variables like `?$projectName = "BI Portal"`

Filters –

- You can use filters while passing the object to the next cmdlet in a pipeline.

- Filters are declared using Where-Object [can be aliased "Where", "?"]

For example - `Get-SPSite "http://itechy:21237/sites/itechyprojects" | Get-SPWeb -Limit All | Where-Object {$_.WebTemplate -eq "PROJECTSITE"} |select Url, WebTemplate, Title`

Try writing some commands which will use variables, pipelines, comparison operators, the objects, properties and methods as described below –

Operators –

You can use a number of operators while working with PowerShell –

Comparison Operators –

eq – Equal to operator.

lt – Less than operator.

gt – Greater than operator.

le – Less than or equal to.

ge – Greater than or equal to.

like – pattern matching

Logical Operator –

and, or, !, -not

Let's try writing some commands which will use variables, pipelines, comparison operators, the objects, properties and

methods as described below –

Command	Purpose
$siteCollection = Get-SPSite "http://itechy/sites/biportal" $siteCollection = Select | *	Creates a variable with the name siteCollection and use that variable.
$singleSite = Get-SPWeb "http://itechy/sites/BIPortal" $singleSite.Lists | Select Title $singleSite.Lists \| where {$_.ItemCount -lt 50} | select Title, ItemCount	Declares a variable with the name singleSite. Display the lists title and use where condition which will filter the lists using a Where condition. For example – *lt* [less than condition].
$dashboard = $singleSite.Lists["Dashboards"] $dashboard | gm	Working with single list and fetching all the members. [*gm* – Get-Member]

CONDITIONS AND LOOPS IN POWERSHELL

Now we will take a look at how do we write conditions, loops in PowerShell which can be used during scripting.

If / Elseif / Else Condition

Here's how to perform tasks based on the *If* condition.

```
$conditionVariable = Get-SPSite "http://itechy/sites/
biportal" | Get-SPWeb | Select WebTemplate
if($conditionVariable.WebTemplate -eq "BICENTERSITE")
{
Write-Host "We are working on Microaoft SharePoint BI
Portal"
}
else
{
Write-Host "The Default is Team Site Template"
}
```

While Condition

Here's how to use the *while* condition while writing the scripts.

```
$services=Get-SPServiceInstance
$services=Get-SPServiceInstance
While($true)
{
if($services.TypeName -eq "PerformancePoint Service"
-and $services.Status -eq "ONLINE")      {
        Write-Host "Performance Point Service found!!"

        break
}
else    {
        Write-Host "Not found"
        break
}
}
```

Function

You can also wrap a reusable script into a function and call it as per your requirements. You can also pass parameters to your functions.

```
function CheckServiceAndStatus([string]$name)
{
$services=Get-SPServiceInstance
While($true)
{
if($services.TypeName -eq $name)
{
        Write-Host "$($name) found!!"
        break
}
else
{
        Write-Host "Not found"
        break
}
}
}
```

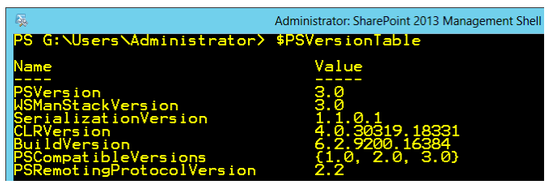
To use this command, type this:

```
CheckServiceAndStatus -name "PerformancePoint Service"
```

I have written this function in a script file.

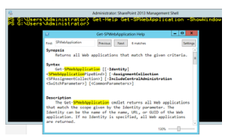
USING POWERSHELL IN SHAREPOINT 2013

Let's see some examples of PowerShell in SharePoint 2013. With SharePoint Server 2013, we get PowerShell version 3.0. To see the current PowerShell version on your machine, we can use the following command –



Note: PowerShell version 3.0 requires .NET Framework 4.0 and as you know SharePoint Server 2010, does not support .NET 4.0. Hence you can use a side-by-side approach for executing SharePoint 2010 cmdlets on PowerShell.

You can also use graphical help under PowerShell V 3.0. For example, take a look at the following command –



Similarly you can also apply filters with graphical help by clicking on "Settings" button shown on the right side of the window. You also have a search functionality in the graphical window. The screenshot we just saw shows the search result for SPWebApplication.

When you execute a PowerShell command, you can now send the output of the command to a separate, interactive table window where you can add filters and different criteria to the output, which is getting displayed.

Let's try the following command that uses a filter and sends output to a separate window –

```
$singleSite = Get-SPWeb "http://itechy/sites/BIPortal"
$singleSite.Lists | Select Title
$singleSite.Lists | where {$_.ItemCount -lt 50} |
select Title, ItemCount | Out-GridView
```

There are several enhancements which have been done in cmdlets from SharePoint 2010 to SharePoint 2013.

For example –

PowerShell V2 –

```
$myObject = New-Object -TypeName PSCustomObject
-Property @{Name="MSBI"}
$myObject | select *
```

PowerShell V3 –

```
$myNewObject = [PSCustomObject] @{Name="MSBI"}
$myNewObject | select *
```

You can also write cmdlets for Office 365. To write the script for Office 365, you will have to download SharePoint Online Management Shell. To download it, you can use the following URL – http://www.microsoft.com/en-us/download/details.aspx?id=35588

Let's quickly see some SharePoint Online cmdlets. I have already registered on the Office 365 developer site. First of all, we will connect to the SharePoint site as shown below –

To establish connection to SharePoint, we use below command

```
Connect-SPOService -Url https://itechy-admin.sharepoint.
com -Credential "demo@itechy.onsharepoint.com"
```

The following command retrieves the existing site collection –

```
Get-SPOSite
```

The –detailed parameter gets all the site collections under your subscription. When you try the `Get-SPOSite – detailed` parameter, you will see all the property values. However if you do not supply the `–detailed` parameter, all the property values will come with default values, except for a few core property values.

```
Get-SPOSite https://itechy-admin.sharepoint. com –
detailed | select *

Get-SPOSite https://itechy.sharepoint.com | select *
```

To view all the available templates, we can use the following command:

```
Get-SPOWebTemplate
```

This way you can use PowerShell to write scripts to perform various operations with our SharePoint Site, as an IT Pro or as a Developer.

Conclusion

With SharePoint 2013 embracing PowerShell as a first class citizen, PowerShell has become a standard for scripting SharePoint. This article attempted to introduce you to PowerShell in SharePoint, however being a very vast topic, we will discuss some additional commands in one of our forthcoming articles ■

Pravinkumar works as a freelance corporate trainer and consultant on Microsoft Technologies. He is also passionate about technologies like SharePoint, ASP.NET, WCF. Pravin enjoys reading and writing technical articles. You can follow Pravinkumar on Twitter @pravindotnet and read his articles at **bit.ly/pravindnc**

PHOTO COURTESY Vernon Chan

Windows Phone Enterprise Apps

Building apps for SharePoint

Windows Phone MVP **Mayur Tendulkar** gives a walkthrough of how we can build a Windows Phone application which can easily integrate with Office 365 SharePoint site and perform some common tasks

Apart from daily calls & text messaging, people use their cell-phone to check their emails, connect on social networks, listen to music, take photographs or play some games. There are many use cases for mobile devices. However for an Enterprise, mobile devices make sense when it helps to improve productivity. Microsoft being an enterprise game player and having a suite of technologies like Azure, Office 365, SharePoint, Lync etc in its stack, makes using Windows Phone in Enterprise much more easier.

Concepts like BYOD (Bring Your Own Device), MDM (Mobile Device Manager) helps us to be more productive in the organization using our own mobile phones. Developers can use Visual Studio and .NET tools they are already familiar with, to develop code that would run on PCs, tablets and smartphones.

> “
>
> Office 365 is a cloud based service from Microsoft, which is a suite of different applications like SharePoint, Exchange, Lync, etc. SharePoint is a collaboration platform, which again provides a plethora of functionalities like document management system, team-sites, etc. This is a highly extensible platform used by many enterprises across the world and using it on a mobile device, can give a boost to productivity.

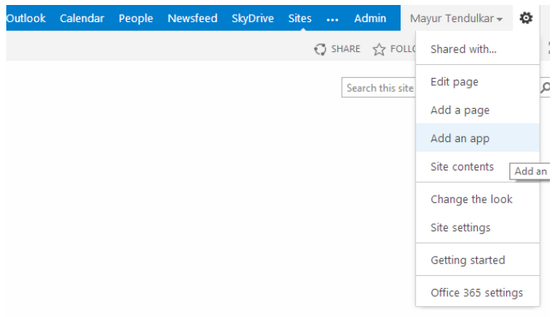
For this article, we're going to use a trial version of Office 365. If you do not have one, register for one here: bit.ly/dncm9-off365reg. Once you've created your Office 365 account, wait for a while so that it sets up collaboration tools like SharePoint (team-site), Newsfeed, SkyDrive Pro, etc...

In this article, we'll see how we can build a Windows Phone application which can easily integrate with Office 365 SharePoint site and perform some common tasks.

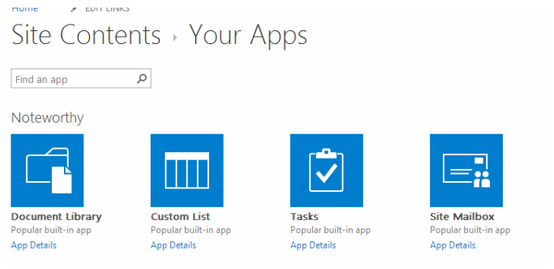
CREATING DEVELOPMENT ENVIRONMENT

To develop apps for SharePoint (Office 365), we'll create a demo site and create a list under it to store some data. Follow these steps:

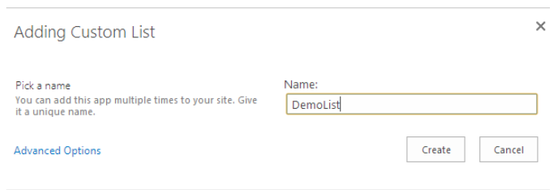
1. Locate a 'Settings' icon on right-hand top corner and click on 'Add an app' option.



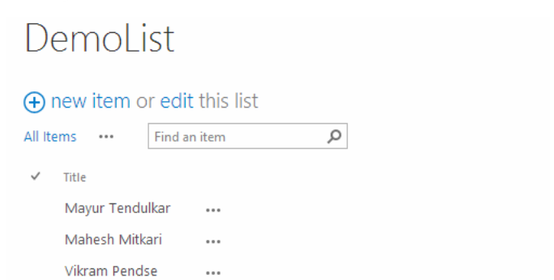
2. On the next page, click on Custom List



3. In the 'Adding Custom List' dialog box, give it a name and click on 'Create'



4. Once this list is created, click on it to add some data by further clicking on 'new item'



Note: User's are required to set up their own Office 365 environment to test the app, with 'DemoList' in it. Alternatively you can build the app from scratch, referring to this article.

BUILDING WINDOWS PHONE APPLICATION

Now that our SharePoint (Office 365) development environment is ready, we can now build Windows Phone application to communicate with it. SharePoint exposes its API as Client Side Object Model (CSOM) as well as REST services. So, while building mobile apps for SharePoint, developers can either make REST calls and manage the app cycle manually i.e. authentication, session management, sending/retrieving data etc... OR use CSOM libraries to use simple Object Model provided by SharePoint. Microsoft has released a *Microsoft SharePoint SDK for Windows Phone 8* to make life much easier. This SDK installs the CSOM libraries for Phone along with authentication libraries and project templates. This CSOM installed by the SDK is similar to CSOM available on desktop. We're going to use the SDK in this article. You can download and install the SDK from here: bit.ly/19yXi5w

CREATING WINDOWS PHONE APP FOR SHAREPOINT

When you install the SDK, it automatically adds project templates to existing Visual Studio installation. Let's go through these templates.

In the screenshot, you can see two project templates are installed during SDK installation. One is **Windows Phone Empty SharePoint Application.** You can use this template to build apps from scratch and have custom functionality around SharePoint on Windows Phone. The second one is **Windows Phone SharePoint List Application**. This project template is useful if you want to manipulate CRUD operations on a specific list. First, we'll build a blank app and then explore the List application.

CREATING WINDOWS PHONE EMPTY SHAREPOINT APPLICATION

Click File > New Project. Select 'Windows Phone Empty SharePoint Application' project template under C# and give it a name.

When you create a blank app, by default following assemblies are referenced.

```
Microsoft.SharePoint.Client.Phone
Microsoft.SharePoint.Client.Phone.Auth.UI
Microsoft.SharePoint.Client.Phone.Runtime
```

You can check these under Project > Right Click > Add Reference window.

In the design screen (MainPage.xaml), add the following code inside ContentPanel to create a list to show records.

```
<Grid x:Name="ContentPanel" Grid.Row="1"
Margin="12,0,12,0">
<ListBox x:Name="DemoList"></ListBox>
</Grid>
```

Now, in code-behind file (MainPage.xaml.cs), add an event handler for 'Loaded' event inside constructor. The code should look like the following:

```
// Constructor
public MainPage()
{
InitializeComponent();
        Loaded += MainPageLoaded;
}
void MainPageLoaded(object sender, RoutedEventArgs e)
{
}
```

To hold lists from SharePoint site and its items, we'll declare (just above the constructor) two private variables as:

```
private List _list;
private ListItemCollection _collection;
```

Creating Context for Further Requests

Just like HttpContext, SharePoint provides *ClientContext*, which maintains authentication data, sends requests on server and retrieves data to client applications. We'll create a single instance public static ClientContext which we can access from anywhere inside the app, to execute all further requests. We'll declare it in App.xaml.cs using the following code.

```
//Private instance of ClientContext
private static ClientContext _mContext;
//Public property returning client context
public static ClientContext Context
{
get
{
if (_mContext != null)
return _mContext;
//Create object of ClientContext by passing SharePoint
//Site URL
_mContext = new ClientContext("https://tendulkars.
sharepoint.com/demosite");
//Create authenticator which will hold authentication
//data for context
var at = new Authenticator {CookieCachingEnabled =
true};
//Use at.AuthenticationMode property to set auth mode.
//Nothing for default
//Set authenticator to context
_mContext.Credentials = at;
return _mContext;
}
}
```

This ClientContext can be accessed from anywhere within the application using App class.

Retrieving Data from SharePoint List

When developing SharePoint applications, you need to understand the model and how it works. For simplicity, you can think of it as a big web application; each web app having SiteCollections and each site collection (may have sub-sites) having *Lists*. These lists will hold the actual data in columns and rows format. And as a rule, you must first populate (or load) web application, then sites, etc... down the line. Following code will perform the same actions. I've added comments to each line of code to make it easier for you to understand:

```
void MainPageLoaded(object sender, RoutedEventArgs e)
{
//Get the top level Web site
var web = App.Context.Web;
//Fetch the desired list by title inside the web
selected above
_list = web.Lists.GetByTitle("DemoList");
//Create a query to load all items/fields in that list
var query = CamlQuery.CreateAllItemsQuery();
//Fetch the items from that list using above query
_collection = _list.GetItems(query);
//Load the web app
App.Context.Load(web);
//Load the collection
App.Context.Load(_collection);
//Execute the query NOW providing Success & Failure
callbacks
App.Context.ExecuteQueryAsync(SucceededCallback,
FailedCallback);
}
```

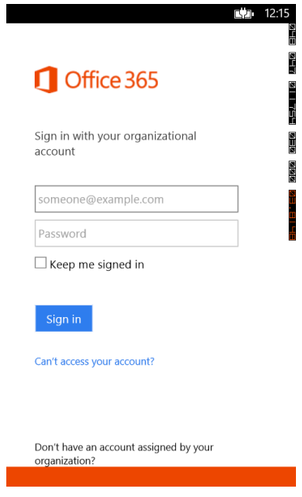
Just like LINQ queries have deferred execution, in SharePoint ClientContext, code executes when *ExecuteQuery* method is called populating items top to bottom. In this case, when ExecuteQuery is executed, first *web*, then *list* and at last list items

are populated.

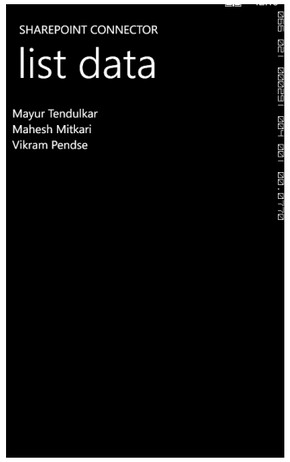
Now, depending on the execution, call-back functions will be called. These are written as follows:

```
private void FailedCallback(object sender,
ClientRequestFailedEventArgs args)
{
MessageBox.Show("FAIL");
}

private void SucceededCallback(object sender,
ClientRequestSucceededEventArgs args)
{
//Create temp list to hold items
var list = new List<string>();
//Iterate through all the items
foreach (var item in _collection)
{
//Get title and add to temp list
          list.Add(item["Title"].ToString());
}
//On UI thread bind temp list to ListBox
this.Dispatcher.BeginInvoke(() =>
{
DemoList.ItemsSource = list;
});
}
```

At this time, your application is ready to run. Just hit F5 to test it inside an emulator. The first screen you should see is Office 365 authentication window. Provide your login credentials here:



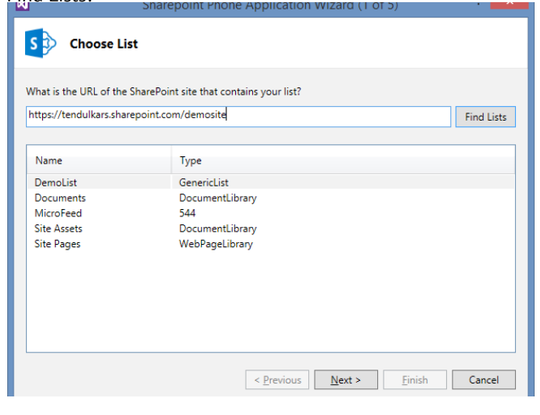
Once you provide these details, click on sign-in and you should be taken to the main page, which will be showing all the items from SharePoint list.



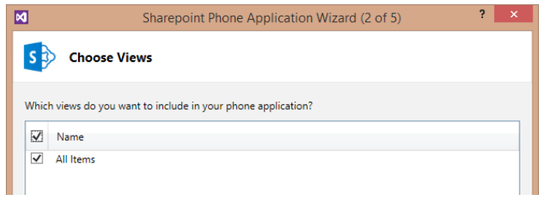
CREATING WINDOWS PHONE SHAREPOINT LIST APPLICATION

We just created a Windows Phone SharePoint Application from scratch, which retrieves data from SharePoint list and shows inside the app. But in a real-life scenario, you may want to perform CRUD operations on it. You may want to display more data, with more actions. Certainly, you can extend this app to do so. But for the same task, Microsoft provides *Windows Phone SharePoint List* application. This project templates does all the job **without writing a single line of code**. The auto-generated code follows Model-View-ViewModel (MVVM) pattern and is production ready. Let's build the same app using this template. Just create a new project with Windows Phone SharePoint List Application project template selected.

When you create SharePoint List Application, the next dialog will appear. On this screen, type-in your SharePoint site URL containing the list which you want to work on. After that click on 'Find Lists'.

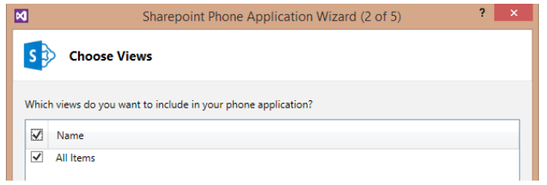


This brings up the Office 365 Authentication box. Provide your authentication details here and sign-in. In the next screen, you should see all the lists available to you.

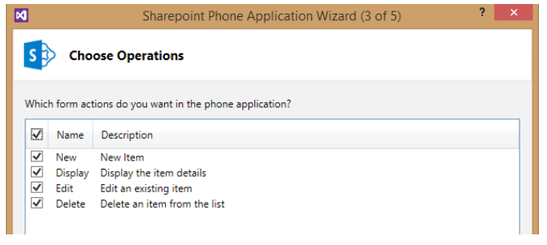


Choose the list and click on 'Next'.

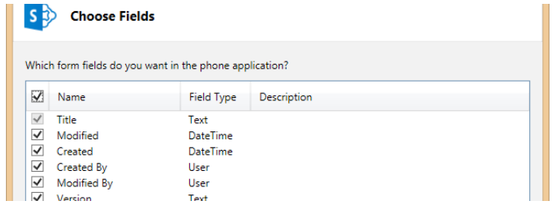
In the next dialog box, select the items that you want to be shown and processed on mobile device. This is kind of a filtering criterion i.e. top 50, all, last 10, etc... Select All Items here as we can change this value later on. Now click on 'Next'



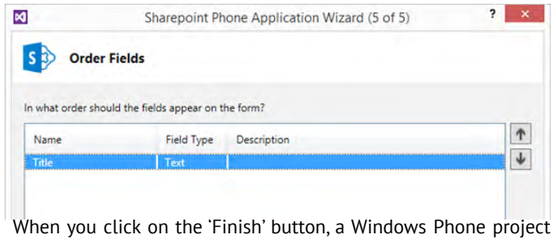
Next you should be able to select actions that you want to perform on the items you selected on the previous screen, like add, edit, delete etc...



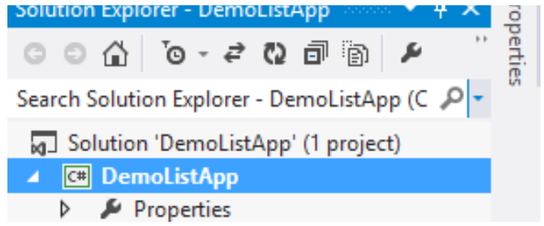
Next you can choose the 'fields' or 'columns' from list that you want to display on the screen. Select those fields and click next.



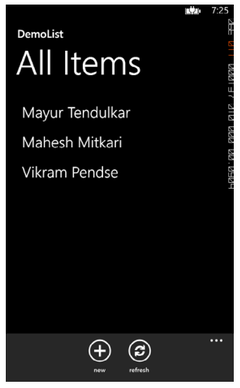
In the last screen, you can choose the display order of the fields. In this case, we've just one field, but in case of multiple fields, you can easily re-arrange the order by pressing Up-Down arrow keys.



When you click on the 'Finish' button, a Windows Phone project is created. This project is built using Model-View-ViewModel (MVVM) pattern. You can see the structure of project in Solution Explorer.



Now when you run this project, you should see the output as shown below:



Conclusion

In this article, we saw how we can build Windows Phone applications which can easily integrate with enterprise solutions like SharePoint Online (Office 365). We've used Microsoft SharePoint SDK for Windows Phone 8 to make many tasks simpler. Furthermore, you can use services like Windows Intune or Company Store to manage and publish apps to selected devices within enterprise. We'll discuss more about these features in one of our upcoming magazine editions ■

Download the entire source code from our GitHub Repository at bit.ly/dncm9-wp8sp

Mayur Tendulkar works as a Local Type Inference (i.e. Consultant) at Zevenseas. He has worked on various Microsoft technologies & is part of Pune and Mumbai User Groups. Since 2008, he has been a speaker at various Microsoft events. You can reach him at mayur.tendulkar@hotmail.com