Issue 02 | **SEP - OCT 2012** EDITION

DNC Magazine

www.dotnetcurry.com

Windows 8 Style Apps

Win 8 Style App using XAML and C#

Animations using WinJS

Design and Testing Considerations

Enterprise Apps

Windows Azure Caching

Hadoop on Azure - Hive for SQL Devs

What's New in WCF 4.5

Testing for Quality and Productivity

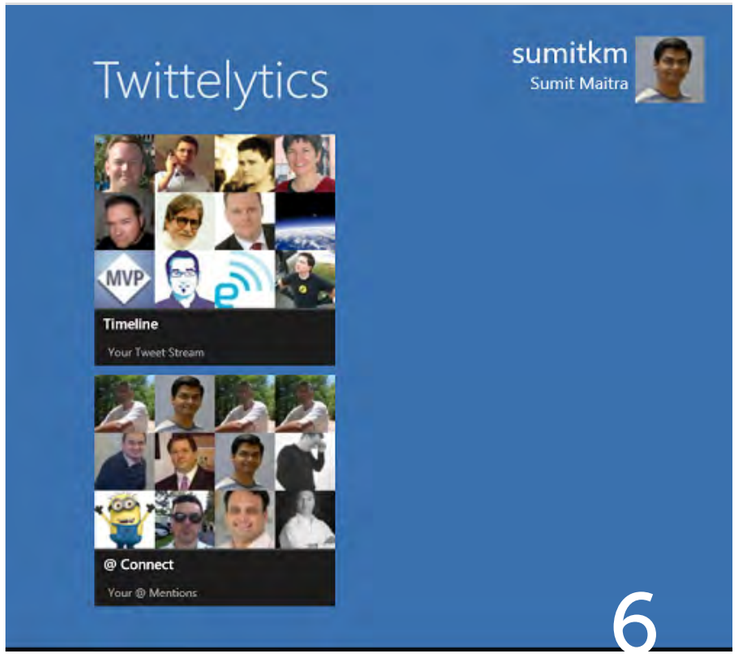
Web Apps

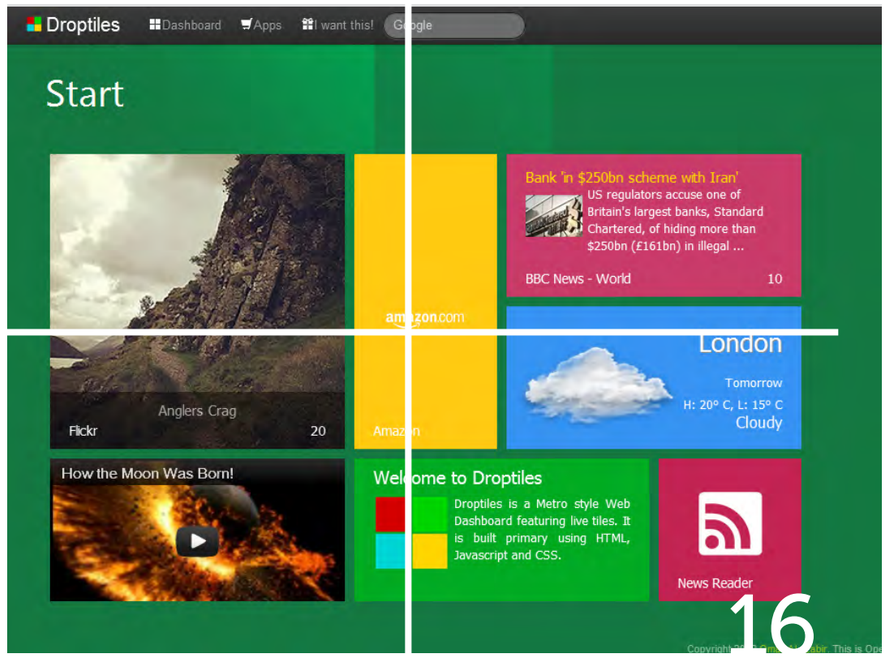
'Droptiles' Web Dashboard

MVC + Twitter BootStrap

EXCLUSIVE INTERVIEW

with Jon Skeet





web & windows 8 dev

enterprise dev

www.dotnetcurry.com
dnc mag

Editor In Chief Sumit Maitra
sumitmaitra@a2zknowledgevisuals.com

Editorial Director Suprotim Agarwal
suprotimagarwal@a2zknowledgevisuals.com

Art & Creative Director Minal Agarwal
minalagarwal@a2zknowledgevisuals.com

Advertising Director Satish Kumar
business@dotnetcurry.com

Writing Opportunities Carol Nadarwalla
writeforus@dotnetcurry.com

Contributing Writers Gouri Sohoni, Govind Kanshi, Mahesh Sabnis, Mehfuz Hossain, Omar Al- Zabir, Raj Aththanayake, Shiju Verghese, Sumit Maitra

Interview Feature Jon Skeet
Twitter @jonskeet

Next Edition 1st November, 2012
www.dncmagazine.com

POWERED BY
a2z | Knowledge Visuals

As the 2012 Holiday Season approaches, the August-October 2012 time span has just turned into the "Microsoft (product release) Season" this year. Microsoft is completely redoing their flagship product Windows in one of its most ambitious releases ever.

With Windows 8 and Visual Studio 2012 already in RTM, .NET 4.5 released, Services like Skydrive and Hotmail (now known as Outlook.com) revamped, Azure Cloud offering getting more and more features, SharePoint 2013 and Office 2013 getting primed for release, it's exciting times for people working on Microsoft Technologies.

Well we are definitely very excited too and it is showing in this episode of the DNC Magazine. We have Windows 8 development, Testing and even on the Web. We also look at Visual Studio 2012, Azure and HTML5.

stay connected

www.Facebook.com/DotNetCurry

@dotnetcurry

www.DotNetCurry.com/magazine

letter from the editor

The excitement of Windows 8 release and Microsoft's 'platform overhaul' is understandable. Many of us have been 'on the fence' about the changes that were announced first in the //Build 2011 event. But as things have matured to the release, most of the skepticism has been replaced with enthusiasm to discover and do more. Time to "Jump in"!

The Windows 8 overhaul is not just an OS upgrade. It's a paradigm shift, a realignment of an entire platform to move in a direction where computing is headed. It relies big on, touch, cloud and mobile centric devices and services; moving away from desktop and pointer centric environment, we have had so far. The change is the realization of a vision that Microsoft had way before any of the current crop of 'tablets' came into existence.

The changes in the platform are incredibly consumer friendly from a mobile platform point of view. The well tested Tile interface from Windows Phone 7 and XBox has now moved to the desktop. Not only that, now the desktop is moving to the 'couch' so to speak, with the plethora of handheld computing devices that are scheduled to arrive this holiday season.

On the developer side, the changes have meant new APIs, a new Runtime and convergence of HTML5 and JavaScript from Web only, to Web + Desktop platform. Additionally an ecosystem to sell apps for (via the Windows 8 Store) throws up a new avenue for all.

Dear Readers, change and evolution is a part of being a software engineer. Microsoft's platform overhaul is an opportunity for us to evolve and move up a notch.

The excitement on the Microsoft Software Platform revision aside, your support so far has inspired us to work harder towards the second edition. We have touched up on nearly all things Microsoft, including Development and testing of Windows 8 Style Apps, ASP.NET and Twitter Bootstrap, JavaScript, Windows Azure and for the first time in DotNetCurry history, we are foraying into Big Data too. Just like the last time, we have a special guest. This time we have **Jon Skeet** in the interview chair. Don't forget to attempt the trivia questions Jon has for you and no taking help of StackOverflow, we are watching :).

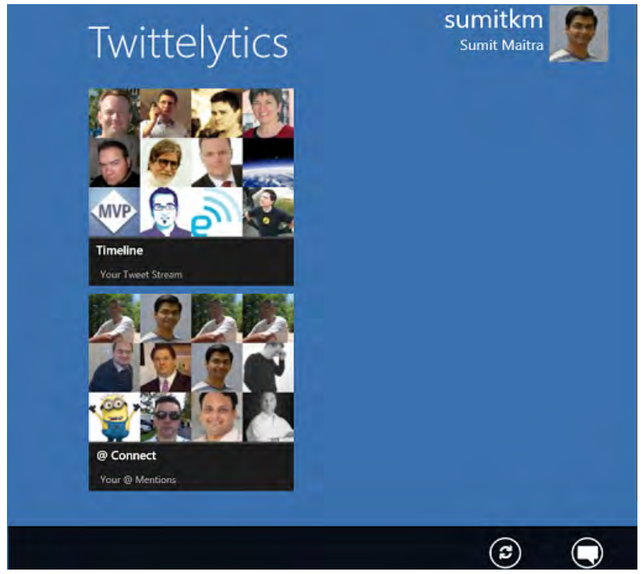
One noticeable change in this edition that we are proud to announce is, better support for screen readers. We are happy to say that now none of our code is captured in screenshots instead its plain text, making the entire code available to screen readers. Keep your suggestions coming, enjoy the magazine and spread the joy!

Sumit K. Maitra

BUILDING A WINDOWS 8 STYLE APP USING XAML & C#

DOWNLOAD FILES >
bit.ly/dncmag-twtly

Sumit Maitra *explains the nuances of Modern UI Style Application Development by building a 'non-default' looking application that follows the Windows 8 Design Aesthetic.*



INTRODUCTION

The Windows 8 RTM and impending customer release has created a lot of buzz in the Microsoft developer community. Windows 8's distinct UI style and new runtime, the WinRT, has gathered a lot of attention both deserved and un-deserved. WinRT is essentially a 'closer to metal' runtime that you can program against, using languages like C#, VB.NET, C++, JavaScript etc. For JavaScript apps, HTML5 is the presentation technology, whereas for C#, VB.NET or C++, it's XAML. However XAML for WinRT is slightly different from XAML in Silverlight.

Though I have been a 'Web Developer convert' for the better part of last 6 years, I finally gave in to the temptation and took a plunge into building a XAML + C# based Windows 8 Style Application. Along the way, I found that there were lots of fine nuances to building a good quality Windows 8 app. I will try and share some of those things here.

In this article, we will develop a Twitter client using the excellent Linq2Twitter library by Joe Mayo. I will focus on the Modern UI Concepts more than the nitty-gritty of the application. However in the end, the code for a functional Twitter Client is available. Just add your Twitter Keys.

Target Audience for the article are people like me sitting on the fence with regards to WinRT, as well as Silverlight WPF developers looking to port their knowledge over to WinRT.

MODERN UI DESIGN AESTHETIC DOCUMENTATION

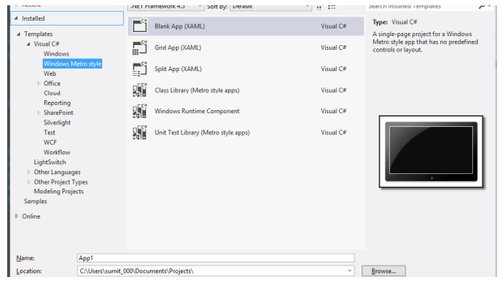
Microsoft has detailed documentation explaining the entire Modern UI design aesthetic at http://msdn.microsoft.com/en-US/library/windows/apps/hh465424. The aim of this article is not to list all of them again, instead we will take some (the entire document is 300+ pages) of these and see how they apply to a real life application. Along the way we will encounter some quirks and hack around them.

Exploring the Templates and picking one for your application

Microsoft Visual Studio 2012 provides a set of Project templates that help us get started with the Modern UI design aesthetic. If you are using C# or VB.NET as your development language, *there are three templates to help you get started with.*

The Blank Template

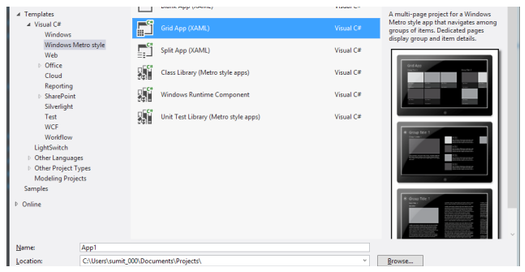
This is an empty canvas allowing you to start fresh and design your application, ground up. It does not have a sample view model associated with it. Pick this one for maximum freedom and improvisation.



However it also means you have to do all the plumbing for the visual elements you use

The Grid Template

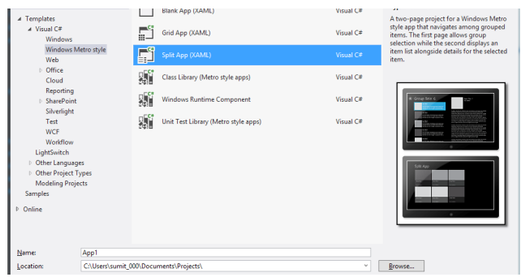
The Grid template is actually the most extensive template with a two-level ViewModel hierarchy



The Landing page is 'Grouped Tile' (my term not MS') layout where groups of large tiles are used to present a snapshot of information. An example use case for this template would be an RSS Feed Reader that allows you to group your Feeds. These grouped Feeds would come up on the landing page with a small preview image and text for each Feed representing a 'tile'. The Feed Groups would be separated by whitespace. On tapping a 'tile', we can see the preview of the first item in the feed and a list of older items. Tapping or clicking on a particular feed, would navigate you to the details of the feed.

The Split Template

The Split template is a two page layout with the same ViewModel hierarchy as the Grid Template, but does not have Tile grouping on the landing page. It also does not have a third page for details, instead the second page is a multi-column page that expands horizontally to fill all the details of the selected item.



Our application will use this layout, hence the Split Template is going to be the starting point of our application.

> *We'll call our application, 'Twittelytics'.*

The application's layout will be as follows:

Landing Page

The landing page is the page that contains the large tiles and will contain a collage of the Twitter avatar for the last 12 tweet authors. By default, there will be two tiles - Timeline and @ Mentions. If users desire, they can add more tiles like Direct Messages and Saved Searches and Lists. Thus the landing page is a graphical snapshot of what's happening on Twitter for the user.

The Details Page

On clicking on a particular tile, the App should navigate to a particular stream and show tweets from the stream. The left hand side list is thus treated as the stream of tweets from twitter and the right hand side is used for the tweet details.

Templates in other Languages

Visual Studio has a bunch of other templates should you choose to build your Modern UI Style application in JavaScript/ HTML5 or C++. Some of these templates are

- For C++ Development we have several other templates viz
  - o Direct2D (for 2D DirectX Game development)
  - o Direct3D (for 3D Game development)

- JavaScript also has two additional templates viz.
  - o The Fixed Layout template, which basically is a fixed aspect ratio application
  - o Navigation App template, which has built in forward/ backward navigation buttons to get started with a Picture Gallery type of apps.

Setting up the dependencies

Building a Twitter Client is a non-trivial activity. One, you have to use OAuth for authentication and second, you have to map the HTTP service requests into C# method calls by using an HttpClient. Twitter, being the popular service it is, already has a bunch of library that does the above using C#. You can see a sample of available libraries on Twitter's Developer site. After a little searching, I zeroed in on Linq2Tweeter.

Linq2Tweeter

Linq2tweeter is an excellent twitter library with the code available on CodePlex (bit.ly/dncmag-l2t). It makes Twitter data available via LINQ queries and is one of the most extensive C# Twitter libraries I have seen. Added bonus, it has WinRT Samples in C# + XAML. I used Linq2Tweeter dependencies from Nuget. To include it in your project simply use the following command on the Package Manager Console and it automatically detects the WinRT runtime and downloads the correct version for you.

```
PM> install-package linqtotwitter
```

WritableBitmapEx.WinRT

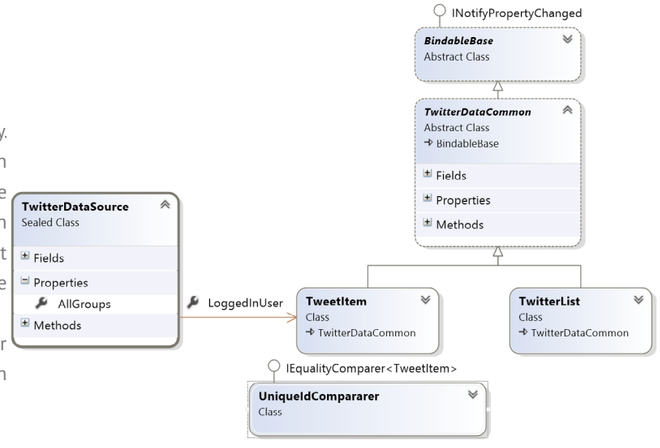
WritableBitmap is an object that allows in-memory management of Graphics. The WritableBitmapEx.WinRT package adds additional Extension methods to the WritableBitmap class. These come in handy for creating a collage from the twitter stream. We can install it again from Nuget using

```
PM> install-package WritableBitmapEx.WinRT
```

Apart from these two, no other external dependencies are required.

ViewModel, Databinding and MVVM

WPF/Silverlight developers are very familiar with the Model View View Model (MVVM) pattern of development for Rich



client applications. Our Split Template sample comes with a default ViewModel that surprisingly takes care of a lot of the requirements for a Twitter client. However to keep things comprehensible, instead of using the name SampleDataSource all over the place, we will rename the ViewModel entities (see the image on the previous page)

TwitterDataSource is the static DataSource class that has an instance of an ObservableCollection<TwitterList> called AllGroups. The Tiles on the homepage are bound to items in this list. As User Navigates by clicking on a tile, the child items are inferred from the selected item.

TwitterDataCommon is an abstract base class that has a lot of common properties like UniqueId, Title, SubTitle, Image and Description. It inherits from the BindableBase class in the Common namespace of our project. The BindableBase implements INotifyPropertyChanged thus enabling Databound controls to refresh themselves when the property value changes. We will see a little later how we leverage this for our home page. TwitterList has an observable collection of items. These are of type TweetItem. The TweetItem class encapsulates the actual Tweet.

With our ViewModels set, we can start with the application layout and make it more vibrant and cheery!

Changes to Default Layout and New Custom Styles

Our Application has two pages to start off with, the ItemsPage.xaml and then we will see the SplitPage.xaml. We will also need a couple of new pages that we will add a little later.

ItemsPage.xaml

1. First up, we change the background color of the page to make it a little more vibrant than the default Black and Greys (colors are subjective and if you want a black and grey theme, feel free to ignore the suggestion).

2. Next we change the Title to Twittelytics and add two Labels and an Image on the Top Right. This will display the logged in users' credentials and their Twitter Avatar.

3. For the controls we placed, we must ensure that their

position is constant irrespective of the three modes that our application may be, Landscape, Portrait and Snapped. To ensure this, we have to do the following:

- Manage Position: Notice the link on the top, right and bottom of the image above. They are each 34px and the 'unbroken" links imply that the 'image' control will at all times be 34px from the top, right and bottom of first row. If you look at the link on the right (near the T of Twittelytics), its broken indicating it is flexible and can change as required. The corresponding XAML for the image is as follows:

```
<Image x:Name="userImage"
Grid.Column="1" Margin="0,34,34,34"
Tapped="Image_Tapped_1"
HorizontalAlignment="Right"
Width="72"
Height="72"/>
```

- Hide extra components when not required: The above XAML is fine for fullscreen portrait or landscape mode. However in the 'snapped' mode, where the app resides next to another app, the twitter handle and User Name overlap with the application title. In Snapped mode we would rather hide them. To do this, we use the VisualStateManager. You can configure the VisualStateManager in XAML itself. Locate the VisualState with x:Name="Snapped" and then add the following XAML to hide the two labels in Snapped state.

```
<ObjectAnimationUsingKeyFrames
Storyboard.TargetName="userIdText"
Storyboard.TargetProperty="Visibility">
<DiscreteObjectKeyFrame KeyTime="0" Value="Collapsed"/>
</ObjectAnimationUsingKeyFrames>
<ObjectAnimationUsingKeyFrames
Storyboard.TargetName="userNameText"
Storyboard.TargetProperty="Visibility">
<DiscreteObjectKeyFrame KeyTime="0" Value="Collapsed"/>
</ObjectAnimationUsingKeyFrames>
```

- Show different controls as per view: The SplitView project template already does this for us, but it is an important thing to keep in mind that between Snapped View and the (Portrait View or Landscape View), the VisualStateManager is configured to use different controls for the ItemsPage. If the current state of the application is snapped as per the following XAML, the elaborate GridView will be hidden and instead a ListView will be shown.


```
<ObjectAnimationUsingKeyFrames
Storyboard.TargetName="itemListView"
Storyboard.TargetProperty="Visibility">
<DiscreteObjectKeyFrame KeyTime="0"
Value="Visible"/>
</ObjectAnimationUsingKeyFrames>
<ObjectAnimationUsingKeyFrames
Storyboard.TargetName="itemGridView"
Storyboard.TargetProperty="Visibility">
   <DiscreteObjectKeyFrame KeyTime="0"  Value="Collapsed"/>
</ObjectAnimationUsingKeyFrames>
```

SplitPage.xaml

The SplitPage has a selected stream on the left in a list. On selection of an item, shows the details of the 'tweet' on the right. Ideally this could be expanded to show a threaded conversation or preview of images or web pages that the tweet might link to. However for now, we will simply show the detailed tweet. The selected tweet can then be replied to, re-tweeted or favorite. Each of these functions have a dedicated button. There is also a 'New Tweet' button that's active irrespective of selection. Overall the UI in the Design Mode looks as shown in the next section.

It's worth mentioning that the auto-generated template already has VisualStateManager setup so that it hides the right hand side automatically in case of Snapped or Portrait view. We will simply add the following XAML to make sure that the "New Tweet" button gets hidden in the Snapped and Portrait mode. As we will see later, the user can access the App Bar in these modes to "Send" new Tweets.

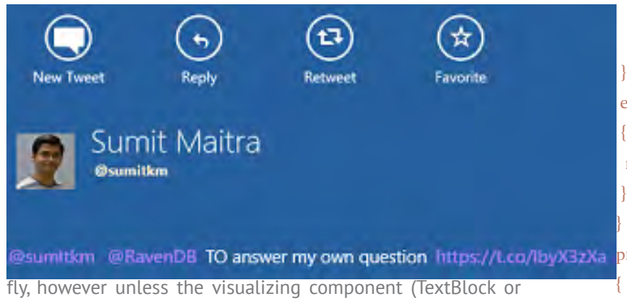
```
<ObjectAnimationUsingKeyFrames Storyboard.
TargetName="sendTweet"
Storyboard.TargetProperty="Visibility">
<DiscreteObjectKeyFrame KeyTime="0" Value="Collapsed"/>
<!--Hide the Send Tweet Button -->
</ObjectAnimationUsingKeyFrames>
<ObjectAnimationUsingKeyFrames Storyboard.
TargetName="replyTweet" Storyboard.TargetProperty="Width">
<DiscreteObjectKeyFrame KeyTime="0" Value="80"/>
<!--Reduce width of the Reply Button -->

</ObjectAnimationUsingKeyFrames>
<ObjectAnimationUsingKeyFrames Storyboard.
TargetName="reTweet" \
            Storyboard.TargetProperty="Width">
<DiscreteObjectKeyFrame KeyTime="0" Value="80"/>
<!--Reduce width of the Retweet Button -->
</ObjectAnimationUsingKeyFrames>
<ObjectAnimationUsingKeyFrames Storyboard.
TargetName="favorite"
            Storyboard.TargetProperty="Width">
<DiscreteObjectKeyFrame KeyTime="0" Value="80"/>
<!-- Reduce width of the Favorite Button -->
</ObjectAnimationUsingKeyFrames>
```

A Custom DataTemplateAdapter

On the right hand side of the SplitView is a Grid Layout that shows the details of the Tweet. Twitter mandates that all @ mentions should link to their profile on the web. Similarly HashTags should be linked to the search page on Twitter.com.

The Tweet text on data bind, by default, comes up as plain text

instead of Rich Text or HTML. As a result none of the links in a tweet are clickable. We can generate these links on the fly, however unless the visualizing component (TextBlock or RichTextBlock) makes them clickable, we will not be compliant with Twitter's recommendations.



What we need is a DataTemplateAdapter that will generate the visualization pieces dynamically. As a result, on the DataBind operation, the entire visualization will be created by the adapter and injected into the view. We create the 'ParagraphTemplateSelector' for this. We will take it up piecemeal.

The DefaultTemplate: A default DataTemplate if none were provided.

```
public DataTemplate DefaultTemplate { get; set; }
```

The SelectTemplateCore override: This is the method where we generate and inject the entire XAML required. As we can see, we start off with a DataTemplate definition that wraps a <Paragraph … /> element. Contents of the paragraph are generated by subsequently Tokenizing the tweet and handing @ mentions, # tags and http://t.co links specially. In the end, we have well formatted XAML

```
protected override DataTemplate SelectTemplateCore(
          object item, DependencyObject container)
{
 if (((ListViewItem)container).ContentTemplate == null)
 {
  string template = String.Format(@"
   <DataTemplate xmlns='http://schemas.microsoft.com/winfx/2006/
        xaml/presentation'>
    <RichTextBlock>
     <Paragraph>
      {0}
     </Paragraph>
    </RichTextBlock>
   </DataTemplate>", TokenizeTweet(item.ToString()));
      return XamlReader.Load(template) as DataTemplate;
 }
 else
 {
  return ((ListViewItem)container).ContentTemplate;
 }
}
private string TokenizeTweet(string currentValue)
{ // check the code
}

   private string AddHyperLink(string token)
   {
    …
   }

   private string AddHashLink(string token)
   {
    …
   }

   private string AddAtLink(string token)
   {}
```

The AddHyperlink, AddHashLInk and AddAtLink generate the required XAML for each element as required.

So the DataTemplateAdapter is simply injecting a RichTextBlock, XAML for which is generated at run-time.

Using a Custom DataTemplateAdapter

A DataTemplateAdapter can be used with a List or other components that iterate over a collection of objects. For us however, though paragraph is a collection of tokens (known as 'Run') however in our ViewModel the tweet is still a single text field. We hack around this by doing the following:

a. Creating a collection property called 'Contents' in the TweetItem ViewModel. The only entry in the 'Contents' is the string in the 'Content' view property.
b. Add a wrapper ListView that will hold the component returned by our DataTemplateAdapter
c. Bind the ListView to the 'Contents' collection in the ViewModel

d. The Final markup in SplitPage.xaml is as follows

```
<StackPanel>
   <StackPanel.Resources>
   </StackPanel.Resources>
   <ListView CanDragItems="False"
          CanReorderItems="False"
          Height="Auto"
          SelectionMode="None"
          AllowDrop="False"
          IsActiveView="False"
          IsTabStop="True"
          ItemsSource="{Binding Contents}"
          ItemTemplateSelector="{StaticResource
              ParagraphTemplateSelector}" />
</StackPanel>
```

e. The ParagraphTemplateSelector must be declared as a resource in the <Page.Resources> section as follows

```
<Page.Resources>
...
 <common:ParagraphTemplateSelector
  x:Key="ParagraphTemplateSelector"
   DefaultTemplate="{StaticResource
    DefaultParagraphTemplate}" />
</Page.Resources>
```

The Custom Styles

We create a ResourceDictonary of custom styles to save styles that differ from the default ones provided. Our custom styles are saved in CustomStyles.xaml. It refers to the default StandardStyles.xaml hence styles in CustomStyles.xaml can be 'based-on' styles in StandardStyles.xaml.

In CustomStyles we have redefined the look and feel of some of the existing styles and also added the styles for our specific functions like 'New Tweet', 'Retweet', 'Reply' and 'Favorite'. The XAML file is in the Common folder of the project.

Integrating Linq2Tweeter

Earlier we saw how to pull in the Linq2Tweeter dependency from Nuget. The complete process of setting up the client is rather involved. Since we are focusing on the Design philosophy more than the implementation details, we will skip through the internals. Conceptually here is what we do:

1. First up, we implement the Login functionality. This requires having a XAML page with an embedded browser control. In the browser control, we direct our application user to Twitter's OAuth page and request them to authorize Twittelytics. Twitter returns an Authorization Key and a Pin. We save the pin and the Authorization key locally and going forth use them for authentication and posting to Twitter.

2. Next we build the timeline functionality. This is relatively easy thanks to the Linq interfaces. The following query fetches the User's timeline.

```
using (var twitterCtx = new TwitterContext(_auth))
{
   var timelineResponse =
      (from tweet in twitterCtx.Status
       where tweet.Type == type &&
       tweet.ScreenName == LoggedInUser.Title
       select tweet)
      .ToList();

IEnumerable<TweetItem> tweets =
   (from tweet in timelineResponse
    select new TweetItem(tweet.StatusID,
       tweet.User.Name,
       tweet.User.Identifier.ScreenName,
       tweet.User.ProfileImageUrl,
       tweet.Text,
       tweet.Text,
       matches)
   );
if (tweets.Count<TweetItem>() > 0)
{
   IEnumerable<TweetItem> results =
      matches.Items.Union<TweetItem>(tweets,
        new UniqueIdComparerer()).OrderByDescending
          <TweetItem, string>
            (k => k.UniqueId);
   matches.Items.Clear();
   foreach (var item in results)
   {
      matches.Items.Add(item);
   }
}
}
await CreateCollage();
```

Here _auth is the authentication object created using the Twitter credentials. TwitterContext is like our DBContext for databases. The variable 'type' is an enum called StatusType with the values – Public, User, Show, Mentions, Home, Retweets, RetweetedByMe, RetweetedToMe, RetweetedToUser, RetweetedByUser, RetweetsOfMe, RetweetedBy. As we can see, each of these imply a particular filter on Twitter's timeline. So if required, we can provide users with these predefined lists. Currently we use only Home and Mentions to show the Home Timeline and the @Mentions. The last line of code is method call that refresh the Image for the respective timeline (e.g. Home or @Mentions). We will see it in details a little later.

3. Finally we build the Retweet, Favorite and Reply functionality. Each of these functions are a single statement implementation on the TwitterContext object.

Retweet

```
Status response = twitterCtx.Retweet(selectedItem.UniqueId);
```

Here UniqueId is the Id of the Tweet that we are re-tweeting.

Favorite

```
Status response = twitterCtx.CreateFavorite(selectedItem.UniqueId);
```

Again here UniqueId is the Id of the Tweet we are marking as favorite

New Status and Reply

The Status update and Reply method takes in two parameters. First one is the Tweet String and the second one is the tweet to which the current update is replying to. In case of new Tweet, the inReplyTo parameter is null.

```
internal static Status SendUpdate(string updateText, TweetItem
inRepyTo)
{
   if (_sampleDataSource._auth == null)
   {
      _sampleDataSource.InitGroups();
   }
using (var twitterCtx = new
        TwitterContext(_sampleDataSource._ auth))
{
   Status tweet = null;
   if (inRepyTo != null)
   {
      tweet = twitterCtx.UpdateStatus(updateText, true, inRepyTo.
      UniqueId);
   }
   else
   {
      tweet = twitterCtx.UpdateStatus(updateText, true);
   }

   return tweet;
}
}
```

This covers the basic functionality for the Twitter client. All of it is neatly encapsulated in the ViewModel class TwitterDatasource. However one thing is not done yet! Refreshing the timeline periodically. Let's see how we can do that.

Refreshing the Timeline – Timers and Threads

Any self-respecting Twitter client needs to be able to pull in latest tweets without user intervention. The best way to go about it is query twitter periodically about the users' current rate limits and adjust the refresh interval. Refresh too fast and you are likely to overrun the 350 tweets per hour limit. However to keep things simple we put a 30 second timer interval and poll periodically. Twice every minute implies 120 hits per hour, should be well below Twitter rate limits.

There are two types of timers in WinRT - DispatchTimer and the ThreadPoolTimer. The DispatchTimer works on the UI Thread whereas the ThreadPoolTimer works on the background thread. We use the DispatchTimer to update our DataSource.

We setup the timer in our TwitterDataSource and start it up on first data access. On timer Tick (every 30 seconds), we call the Refresh method. The refresh method is implemented as follows

```
internal void Refresh()
```

```
{
  CoreDispatcher dispatcher = Window.Current.Dispatcher;
  Parallel.ForEach(_sampleDataSource.AllGroups, item =>
  {
    dispatcher.RunAsync(Windows.UI.Core.CoreDispatcherPriority.
          Low, () =>
    {
      var matches = _sampleDataSource.AllGroups.Where(
        (group) =>
          group.UniqueId.Equals(item.UniqueId)).
                ToList<TwitterList>();
      if (matches.Count > 0)
      {
        if (item.UniqueId == "timeline")
        {
          _sampleDataSource.RefreshTimeLine(matches.First(),
            StatusType.Home);
        }
        else if (item.UniqueId == "atMentions")
        {
          _sampleDataSource.RefreshTimeLine(matches.First(),
            StatusType.Mentions);
        }
      }
    });
  });
  if (!_timer.IsEnabled)  {  _timer.Start();  }
}
```

We are using the Task Parallel library's Parallel Foreach to split up each timeline processing in its own thread, if possible. Also to keep UI responsive, we start off the TimeLine Refresh asynchronously in a low prio thread. With these two setting, we call the RefreshTimeLine method that does a union of the old results with the new results to update the list of tweets.

Point to Note: Using a DispatchTimer for actions that depend on web results ,may not result in good UX. If network is slow, it might make the UI unresponsive.

With the timeline refresh done, we now move on to make our landing page more appealing.

Collage on Home Page

The home page of our application has a big tile for each list we save. By default we have two lists Home and @Mentions. As we see in the snapshot, each large tile consists of two sections, the bottom has the name and the top section is a collage of Twitter Avatars. The Avatars are picked up from the last 12 tweets in the timeline.

Needless to say we have to build this collage on our own and assign the resultant bitmap to the Image property of the respective TwitterList object.

For sake of brevity, I will not reproduce the code here. Checkout the CreateCollage() method in the DataSource class.

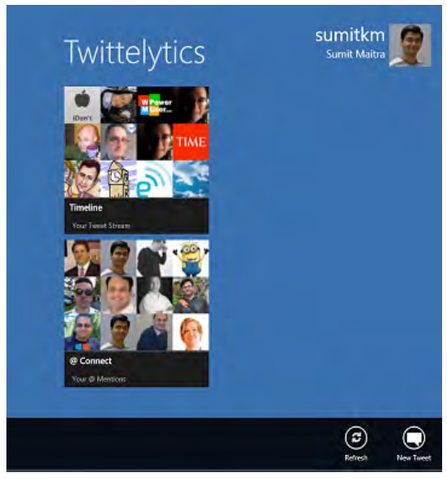
The idea is as follows:

- Create a 'destination' WriteableBitmap object of the size (48x4) x (48x3). We are assuming the size 48 because twitter sends us Avatars in 48x48.
- Next we start looping through our source images. For each image:
  - We convert it into a stream and build a WritableBitmap out of it.
  - Use this source bitmap and use the Blit function provided by WriteableBitmapEx extension we got from Nuget, to 'paste' it into the destination.
  - As we move through each image, we keep updating the last position where the image was painted.
- At the end of the loop we have the entire Collage painted out and we assign it to the Image property. Since the property is Databound to the Image UI element, the UI will get refreshed too.

Now if you remember from earlier, the refresh timeline method calls the CreateCollage() method and the refresh timeline is getting called every 30 seconds. So essentially we have a 'dynamic' collage that will change every 30 seconds if you get new tweets in that period.

Introducing the Bar – The App Bar

As we know, the full screen apps are tuned for touch operation and fingers have no good way of conveying a 'right click' :) . MS has found a nice workaround. If you swipe up from the bottom bevel of a touch device, it brings up what is referred to as the App Bar.



App Bar is the place of shortcuts to context sensitive or universal actions in the app. In our application, one universal action we can have is sending a new tweet and another one to take a chance against the Twitter Rate Limit and hit refresh to reload all the lists.

A Twitter client should be able to send a tweet from anywhere right? Another shortcut could be settings, but at this moment we don't have any worthwhile settings that need a dedicated UI. AppBars can be associated to the Top or the Bottom of the page.

```
<Page.BottomAppBar>
<AppBar x:Name="BottomAppBar1"
Padding="10,0,10,0" AutomationProperties.Name="Bottom App Bar">
<Grid>
<Grid.ColumnDefinitions>  <ColumnDefinition />
</Grid.ColumnDefinitions>
  <StackPanel x:Name="RightPanel" Orientation="Horizontal"
   Grid.Column="1" HorizontalAlignment="Right">
    <Button x:Name="Refresh"
          Style="{StaticResource RefreshAppBarButtonStyle}"
                Tag="Refresh" Tapped="Refresh_Tapped"/>
     <Button x:Name="NewTweet"
          Style="{StaticResource NewTweetButtonStyle}"
                Tag="Previous" Tapped="NewTweet_Tapped"/>
</StackPanel>
</Grid></AppBar></Page.BottomAppBar>
```

Page has a BottomAppBar property where we have placed an Appbar with a Grid layout. The grid has one column and one row that has a stack panel of two buttons on for Refresh and one for New Tweet. In the Tapped event handlers, we do the respective action of either navigating to the Send Update page or Refreshing all the panels on the home page.

Signing off

With that we sign off this article. There are a few more things that are new to WinRT and Windows 8, like Charms and LiveTile. We have to leave these for another day. We are also missing test cases and error handling for 'non-happy-path' flow of the application.

To sum up, we saw how to build a 'non-default' looking application that follows the Windows 8 Design Aesthetic. Fact is, the default boilerplate code can be leveraged well enough to make engaging applications.

Windows 8 is a bold new interface and Modern UI apps represent a shift from the traditional WinForms/WPF/Silverlight approach. So far the default applications that come with Win 8 have only shown a glimpse of the capabilities. Overall WinRT is like a fresh, blank Canvas. It's up to us, how well we paint it by building Vibrant and engaging applications. ■

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

DROPTILES

"Omar demonstrates a Windows 8 like Metro Dashboard in ASP.NET"

Droptiles is an Open Source project that mimics the Windows 8 Start-like experience on the Web, featuring Live Tiles. Tiles are widgets that can fetch data from external sources and display on the Dashboard. Clicking on a Tile launches the full app. Apps can be from any existing website to interactive HTML 5 apps specifically built to fit the Dashboard experience. Droptiles is built using HTML 4, JavaScript and CSS 2, thus highly portable to any platform. The default implementation uses ASP.NET to demonstrate some server side integration, but you can easily port the server side to PHP, JSP, Ruby and so on. It is the sequel of Dropthings, which was my first Open Source project to offer a Web 2.0 Dashboard comprised of dynamic Widgets. See it live!, Go to Droptiles.com. Get the code from: https://github.com/oazabir/Droptiles

TECHNOLOGIES

Droptiles is built using:

- HTML 4, CSS 2
- jQuery
- KnockoutJS
- UnderscoreJS

It does not use any special HTML 5 or CSS 3 technique although in HTML 5 and CSS 3 compatible browsers, certain animations, transitions can be turned on for a faster, smoother experience. It heavily uses jQuery to offer client side interactivity. KnockoutJS is used for the client side MVVM implementation. Finally UnderscoreJS is the awesome utility library to save thousands of lines of repeated JavaScript code, that you always wished someone should have done already.

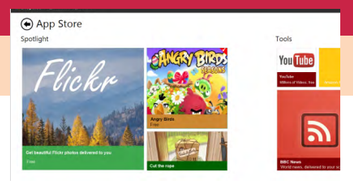
FEATURES

- **Metro style** user interface. CSS framework to build metro style websites, inspired by metroui.org.ua.
- **Full screen apps** built using HTML 5.
- **Drag & Drop** Tiles to personalize the experience.
- Client side object model and data binding for easy MVVM implementation.
- Server side platform neutral implementation. Can be ported to PHP, JSP easily.
- **Live tiles.** Tiles are mini-apps, loading data from variety of sources.
- IE 7+, Chrome, Firefox, Safari Compatible. .

There are four main parts in Droptiles: App Store, Full screen apps, Dashboard and Login, Signup, Settings.

DOWNLOAD FILES >

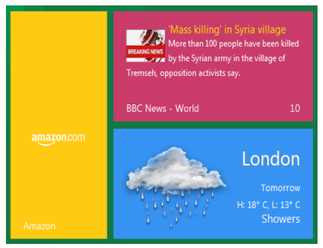
bit.ly/dncmag-dtil



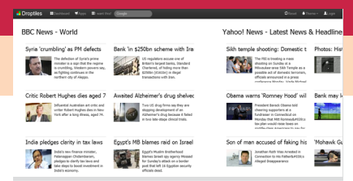
DASHBOARD

Dashboard comprises of Sections. Each section contains a collection of *tiles*. Each box you see is a Tile. Tiles are mini apps. A tile can be of the following type as shown here:

- Simple html pages.
- A dynamic JavaScript mini-App.
- Dynamic page.



The Amazon.com tile is just a static tile with no dynamic behavior. The other two tiles – News and Weather tiles are fully dynamic mini-apps, offering interactivity on the Dashboard.

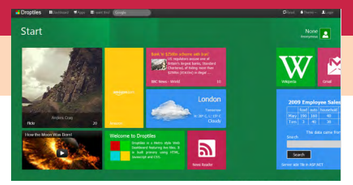


FULL SCREEN APPS

Full screen apps are apps that are launched inside the Dashboard when you click on the respective Tiles. There are 2 example apps – Flickr and News Reader.

The News Reader shows another reuse of the Droptiles framework. In less than 80 lines of JavaScript, fully reusing the Droptiles framework, you can create a News Reader experience similar to the Windows 8 News Reader.

Moreover, it demonstrates how you can produce Tiles entirely from server side on the fly, unlike Dashboard and App Store where the Tiles are pre-defined in a configuration file.



APP STORE

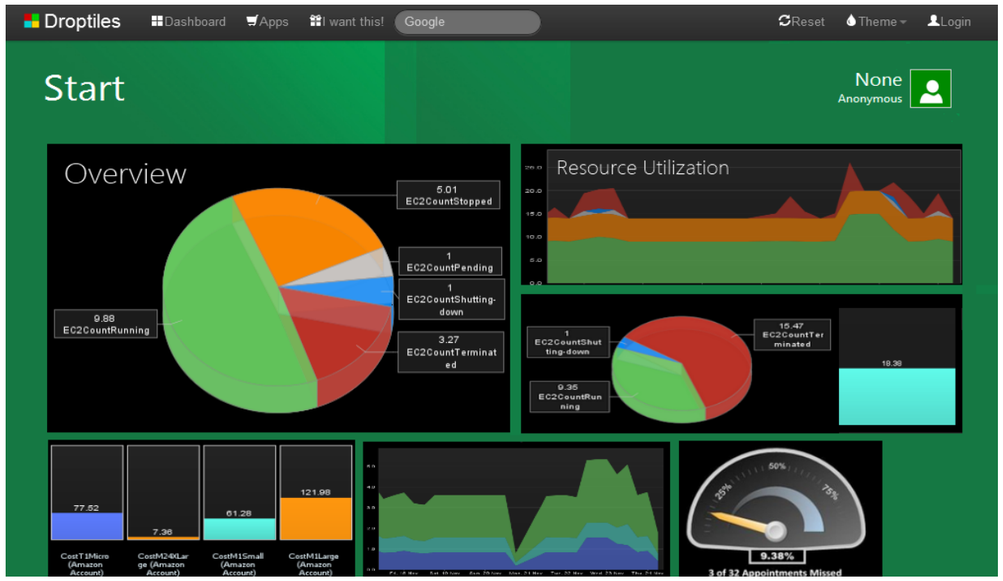
Droptiles has its own App Store where you can showcase additional apps. User can add these apps on the Dashboard. The App Store is built using the same common framework that the Dashboard uses, thus showing how reusable the Droptiles framework is. In less than 150 lines of JavaScript, you can create the App Store landing page reusing the Droptiles framework.

App Store shows a collection of apps available for users to add on the Dashboard.

USE CASE FOR DROPTILES

Enterprise Dashboard aggregating data from various systems and offering a launch pad for intranet/internet applications.

For example, you could create a Dashboard like this:



Web 2.0 Portal offering Portlets in the form of Tiles, aggregating data from various services and as a launch pad for different services.

Touch enabled Kiosk front-end. Great for Hotels, Restaurants, Banks self-service Kiosks.

Content aggregator for News and Research purpose.

GETTING HANDS ON WITH DROP TILES

CODING A TILE

Let's look at the **Flickr** tile. The appearance of the Tile is defined in the Tiles.js file, which contains the metadata for all the tiles. For example, the Flickr tile metadata is defined as following:

```
flickr: function (uniqueId) {
return {
        uniqueId: uniqueId,
        name: "flickr",
        iconSrc: "img/Flickr alt 1.png",
        label: "Flickr",
        size: "tile-double tile-double-vertical",
        color: "bg-color-darken",
        appUrl: "http://www.flickr.com/",
        cssSrc: ["tiles/flickr/flickr.css"],
        scriptSrc: ["tiles/flickr/flickr.js"],
        initFunc: "flickr_load"
    };  }
```

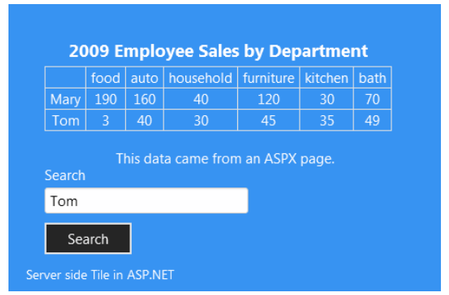
The icon is the default icon shown on the Tile while the Dashboard loads the JavaScript, CSS and HTML dynamically. The horizontal and vertical size of the tile is defined in the size attribute. When you click on a Tile, the URL to host in the full screen view is defined in appUrl. Additional CSS, JavaScript to load are defined in cssSrc and scriptSrc. They are used when the Tile is a dynamic tile and needs its own style sheet and JavaScript to offer the interactivity inside the Tile. Finally the initFunc tells what function to invoke once the scripts are loaded. You can define a function in your own Tile specific JavaScript and once Dashboard has finished loading the JavaScript, it will invoke that function so that the script can start offering its interactivity inside the Tile.

The metadata defines how the tile is displayed on the Dashboard. The behavior to load data from Flickr comes from the Flickr.js file, defined as follows:

```
function flickr_load(tile, div) {
 var url = "http://api.flickr.com/services/feeds/photos_public
.gne?lang=en-us&format=json&tags=nature&jsoncallback=?";
 $.getJSON(url, function (data) {
 var ctr = 0;
 $.each(data.items.reverse(), function (i, item) {
  if (item.tags.length < 150) {
   var sourceSquare = item.media.m;
   var sourceOrig = (item.media.m).replace("_m.jpg", ".jpg");
   var htmlString = '<div class="flickr_item"> <a target="_blank"
    href="' + sourceOrig +
    '" class="link" title="' + item.title + '">';
   htmlString += '<img title="' + item.title +
    '" src="' + sourceSquare + '" ';
   htmlString += 'alt="' + item.title +
    '" />';
   htmlString += '</a><div class="flickr_title">' + item.title + '</div>' +
    '</div>';
    tile.slides.push(htmlString);
    ctr = ctr + 1;
   }
  });
  tile.counter(ctr);
 }); }
```

That's it! In less than 35 lines of code, you have a fully dynamic Flickr Tile loading Photos from Flickr and showing inside the Tile.

Besides static html tiles, you can make Tiles that are loaded from dynamic pages. For e.g. the Dynamic Tile shown on the second section is loaded from an ASPX file. Thus every time the Tile loads, the dynamic page can deliver dynamic content from the server.



The way Dashboard loads and initialize the Tiles is:

- First get the list of Sections and the tiles inside each section.
- Create Tile boxes as per the definition stored in Tiles.js file.
- For each Tile, see if the Tile has any external JavaScript, CSS and html files to load. If yes, then load them.
- Execute the function defined in the initFunc. Pass the tile object, the Tile div reference and the initParams to it.

DEFAULT TILES TO SHOW ON DASHBOARD

The default tiles shown on Dashboard are defined in the same Tiles.js file as following:

```
window.DefaultTiles = [
{
 name :"Section1",
 tiles: [
  { id: "flickr1", name:"flickr" },
{ id: "amazon1", name:"amazon" },
...
]
```

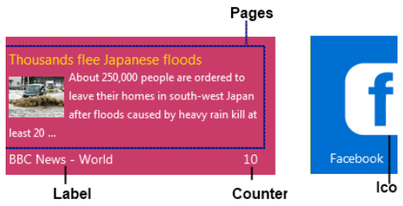
```
},
{
 name: “Section2”,
 tiles: [
  { id: “wikipedia1”, name: “wikipedia” },
  { id: “email1”, name: “email” },
...
},
{
 name: “Section3”,
 tiles: [
 { id: “youtube1”, name: “youtube” },
 { id: “ie1”, name: “ie” },
 ...
 ]
}];
```

Tiles are shown in the exact order they are defined. The name has to be the same name used to define the Tile metadata.

HTML MARKUP FOR TILES

Tile comprises of four parts:

- Slides
- Label
- Counter
- Icon



The following HTML defines the Tile markup:

```
<div class=”metro-sections” data-bind=”foreach: sections”>
  <div  class=”metro-section”  data-bind=”attr:  {id  :  uniqueId},
foreach: sortedTiles”>
 <div data-bind=”attr: { id: uniqueId, ‘class’: tileClasses }”>
 <!-- ko if: tileImage -->
      <div class=”tile-image”>
      <img data-bind=”attr: { src: tileImage }” src=”img/
      Internet%20Explorer.png” />
     </div>
 <!-- /ko -->
  <!-- ko if: iconSrc -->
  <!-- ko if: slides().length == 0 -->
  <div data-bind=”attr: { ‘class’: iconClasses }”>
  <img data-bind=”attr: { src: iconSrc }” src=”img/Internet%20
     Explorer.png” />
  </div>
  <!-- /ko -->
 <!-- /ko -->
 <div data-bind=”foreach: slides”>
  <div class=”tile-content-main”>
   <div data-bind=”html: $data”>
   </div>
  </div>
  </div>
  <!-- ko if: label -->
   <span class=”tile-label” data-bind=”html: label”>Label
   </span>
  <!-- /ko -->
  <!-- ko if: counter -->
   <span class=”tile-counter” data-bind=”html: counter”>10
    </span>
  <!-- /ko -->
   <!-- ko if: subContent -->
   <div data-bind=”attr: { ‘class’: subContentClasses }, html:
      subContent”>
    subContent
   </div>
  <!-- /ko -->
  </div>
 </div>
</div>
```

The markup is defined using KnockoutJS markups, which is used to bind the Tile object model to the html markup.

THE VIEW MODEL

Client side View Model is defined using KnockoutJS. There are three entities:

- Dashboard Model
  - Section
    - Tile

Dashboard Model

Dashboard Model is the root view model. It contains a collection of Sections.

```
/*
 The root Model class holds all the sections and tiles inside the sec-
tions.
 Params:
  title - Title for the Dashboard e.g. “Start”
  sections - An array of section models.
  user - Currently logged in user details, or anonymous.
  ui - UI configuration, defaults.
*/

var DashboardModel =
 function (title, sections, user, ui)
 {
  var self = this;
  this.appRunning = false;
  this.currentApp = “”;
  this.user = ko.observable(user);
  this.title = ko.observable(title);
  this.sections = ko.observableArray(sections);
  ...
 }
```

There are some handy functions in DashboardModel to get a Tile using its id, no matter which section contains it. Similarly removing a tile by id, subscribing to change notifications so that whenever a tile is added or removed, subscribers can be notified. This notification mechanism is used by the Dashboard to save the tiles in a cookie whenever user adds or removes tiles.

```
/*
 The root Model class holds all the sections and tiles inside the sec-
tions.
 Params:
  title - Title for the Dashboard e.g. “Start”
  sections - An array of section models.
  user - Currently logged in user details, or anonymous.
  ui - UI configuration, defaults.
*/

var DashboardModel =
 function (title, sections, user, ui)  {
  var self = this;
  this.appRunning = false;
  this.currentApp = “”;
  this.user = ko.observable(user);
  this.title = ko.observable(title);
  this.sections = ko.observableArray(sections);
  ... }
```

When the tiles are reordered or a new tile is added or a tile is removed, the final configuration of the tiles are stored in a cookie in a serialized form. First the serialized string is prepared by calling the toSectionString() function. Then the Dashboard creates a cookie to save the setting. If a user is logged in, it informs the server to save the cookie in a database.

Section Model

Section model does not have any sophisticated functionality. It is more or less a collection of Tiles. The section model is defined as following:

```
/*
 Section holds a collection of tiles.  Each group of tiles you see
 huddled together on screen, are sections.
*/
var Section = function (section) {
 var self = this;

// Name of a section. Can be used to show
// some title over section.
 this.name = ko.observable(section.name);
// Unique ID generated at runtime and
// stored on the section Div.
 this.uniqueId = _.uniqueId(‘section_’);
// Returns tiles sorted by index
this.tiles =
  ko.observableArray(section.tiles);

this.getTilesSorted = function () {
  return self.tiles().sort(function (left,
   right) {
    return left.index == right.index ? 0 :
     (left.index < right.index ? -1 : 1)
   });
  }
 // Computed function to data-bind
 this.sortedTiles = ko.computed(this.getTilesSorted, this);
```

Tile Model

The most important model is the Tile model. It represents a single tile object and it encapsulates the following data:

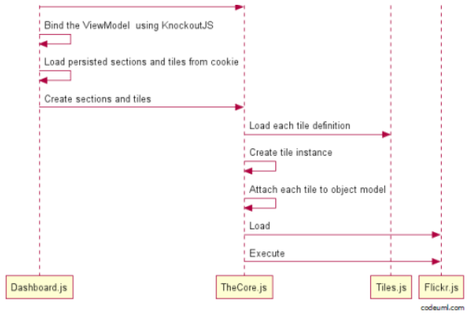
uniqueId: ID of a tile
name: Title of the tile
label: The bottom left label
counter: The bottom right counter
index: Order on screen that is calculated at runtime.
size: Size of the tile with values like tile-double or tile-double-vertical.
color: The background color of the tile
additionalClass: Additional CSS classes if required for the tile
timeImage: An image that can fit the tile's background if provided.
cssSrc: Additional CSS files that may be required at runtime
scriptSrc: Additional JavaScript files that need to be loaded at runtime
initFunc: A (optional) JavaScript function name that is called on document load.
initParams: An (optional) object to be passed to the initFunction.
slidesFrom: Array of HTML Pages to load and inject as slides inside the tile. The pages are rotated in the tile.
appTitle: Title of the application when launched by clicking on tile.
appUrl: URL of the application to launch.
appInNewWindow: To load the app in new browser window outside the Dashboard. Defaults to false.
iconStyle: Tile icon size
iconAdditionalClass = Additional CSS class for the tile icon.
iconSrc: Icon url
appIcon: Icon to show when full screen app being launched.

After a Tile is created, it needs to load the associated JavaScript, CSS and content for the tile. That is done in the init function of Tile.

All of this is inside the TheCore.js file.

DASHBOARD EXECUTION

Here's a sequence diagram that shows how the Dashboard loads

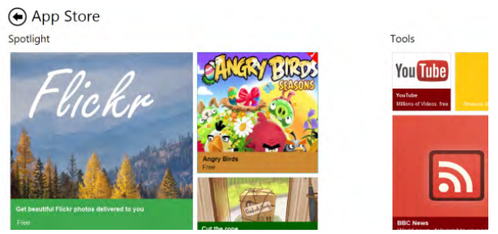


Code in the Dashboard.js is very straightforward and it does all the magic!

```
// This is the the starting point. for the entire Dashboard.
// It takes the currentUser (defined in the Droptiles.master),
the UI config (as above) and the TileBuilders that comes from
Tiles.js.

var viewModel =
 new DashboardModel("Start", [],
  window.currentUser, ui, TileBuilders);

$(document).ready(function () {
 // Hide the body area until it is
 //fully loaded in order to prevent flickrs
 $('#content').css('visibility', 'visible');
 // Initiate KnockoutJS binding which creates
 // all the tiles and binds the whole
 // UI to viewModel.
 ko.applyBindings(viewModel);

 ui.hideMetroSections();

 // See if user has a previous session
 // where page setup was stored
 var cookie = readCookie("p");
 if (cookie != null && cookie.length > 0) {
  try {
   viewModel.loadSectionsFromString(
    cookie);
  } catch (e) {
  // Failed to load saved tiles.
  // Load the default tiles.
   viewModel.loadSectionsFromString(DefaultTiles);
  }
 }
 else {
  // No cookie, load default tiles.
  // Defined in Tiles.js
  viewModel.loadSectionsFromString(
   DefaultTiles);
 }
 ui.showMetroSections(function () {
  ui.attachTiles();
  //ui.reflow();
  ui.makeSortable();
  ui.animateTiles();
 });
```

First it tries to read the Sections and Tiles setup from the cookie, if it is saved. If not found in cookie, it loads the default Tiles. Next the framework in TheCore.js kicks in and takes care of creating the Tiles. Finally, if user has configured additional dynamic behavior via tile specific scripts, they are loaded via the ui.attachTiles() and ui.animateTiles() methods.

APP STORE SAMPLE



The App Store experience is built the same way as the Dashboard. The App Store shows how reusable the Droptiles framework is. It uses the same TheCore.js to provide the experience. The only difference is instead of Tiles.js which defines the tile metadata and the default tiles, it has its own AppStoreTiles.js that defines the Tile metadata and the default tiles to show on the App Store. That's all that differs. Here's the code in AppStore.js:

```
// This is the viewModel for the App Store.
var viewModel = new DashboardModel
 ("App Store", [], window.currentUser,
  ui, TileBuilders);

$(document).ready(function () {
    // Hide the body area until it is fully
 // loaded in order to prevent flickrs
$('#content').css('visibility', 'visible');
 // Initiate KnockoutJS binding which
 // creates all the tiles and binds the
 // whole UI to viewModel.
 ko.applyBindings(viewModel);
 viewModel.loadSectionsFromString(
  window.AppStoreTiles);
}
```

That's it. The loadSectionsFromString function takes care of creating the AppStore Tiles. It takes the Section and Tile configuration in a string serialized format, like this:

```
Section1~flickr1,flickr.news1,news|Section2~angrybirds1,angrybirds
```

This is how it is stored in the cookie as well so that next time you visit the Dashboard, it restores the Tiles as you how you have left it.

CONCLUSION

Droptiles replicates the awesome Metro style experience that Windows 8 has brought to the world. It shows how you can use plain HTML, JS and CSS to replicate the Metro look and feel on the web. It can be adopted to build various web applications, especially touch enabled applications. The modern Metro look & feel makes it quite attractive to use as starting point for your Enterprise web apps. ■

Omar is the Chief Architect of SaaS Platform at BT in London, UK. He is a Microsoft MVP, CodeProject MVP and author of an O'Reilly title - "Building Web 2.0 Portal with ASP.NET 3.5". Omar has several popular Open Source projects including Dropthings, CodeUML, PlantUMLEditor. Omar's specialization on Performance and Scalability techniques can be found in his blog – http://omaralzabir.com

The Windows 8 ListView control already comes with built in list animation.

ANIMATIONS & GESTURES IN A WINDOWS 8 APP

***Mehfuz Hossain** demonstrates gesture recognition and element animations in a Windows 8 application using WinJS, making it intuitive and fun to use.*

DOWNLOAD FILES >
bit.ly/dncmag-aniwinjs

Windows 8 JavaScript library or WinJS comes with built in support for various gesture effects and element animations that can make your app intuitive and fun to use. In this article, we will create a simple ToDo list (I call it the "Tada list") app and change the way items are added and removed using element list and gesture animation.

GETTING READY

We will skip over the details of data initialization and layout design to look at some quick pointers on wiring things up. The UI is defined in *todo.html* and the data initialization and actions are implemented in the *data.js* and *todo.js*. We'll go down to the details a little later in the article.

We start at data.js by declaring the data variable that will be used to pull things from local settings. This is the current instance of *ApplicationData* class that is associated with the app's app package:

```
var appData = Windows.Storage.ApplicationData.current
```

Next, we create the array and add some default todo items. The "data" namespace is declared using *WinJS.Namespace.define*, which has a binding list and an action method to add new item. This is exposed to the UI.

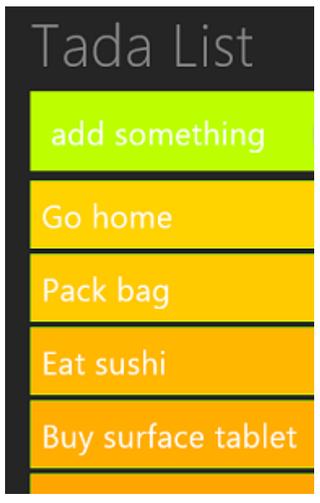
Objects are serialized using *JSON.Stringify* and de-serialized using *JSON.Parse* to save and retrieve data from local settings as saving JSON objects directly to roaming / local settings is not supported.

Full code snippet for what we have described so far is shown below:

```
var todos = [];

todos.push({ title: "Buy surface tablet" });
todos.push({ title: "Buy milk" });

if (!appData.localSettings.values.todos) {
  appData.localSettings.values.todos = JSON.stringify(todos);
}

todos = JSON.parse(appData.localSettings.values.todos);

function insertAtTop(item) {
  if (item.title != null) {
    todos = JSON.parse(appData.localSettings.values.todos);
    todos.splice(0, 0, item);

    appData.localSettings.values.todos = JSON.stringify(todos);

    data.items.splice(0, 0, item);
  }
}

WinJS.Namespace.define("data", {
  items: new WinJS.Binding.List(todos), insertAtTop: insertAtTop
});
```

LIST ANIMATION

Before digging deep into the animation, let's first see how the final UI will look. It is defined in the 'todo.html'.



The definition consists of a simple container div (id ="todoList"). The Title container is defined using the HTML5,'header' tag. We also have an input box for entering new ToDo items.

```
<header role="main">
    <h1>
      Tada List
    </h1>
</header>

<input class="todo-text" type="text"
placeholder="add something" />

<div id="todoList"> </div>
```

As the data is being inserted, we create the list animation instance using WinJS.UI.Animation.createAddToListAnimation with affected items list and then dynamically create an element in the DOM, insert the new item to the main container (id="todoList") and finally execute the animation.

```
function addNewTodo(todo, prepend) {
 var affectedItems = document.querySelectorAll(".item-container");

  var newItem = document.createElement("div");

  newItem.className = "item-container";

  var titleElement = document.createElement("h4");

  titleElement.className = "item-title";
  titleElement.innerHTML = todo;

  newItem.appendChild(titleElement);
  newItem.style.background = listColor();

  ++currentIndex;

  prepareElementForGesture(newItem);

  var addToList = WinJS.UI.Animation.createAddToListAnimation
 (newItem, affectedItems);
```

```
    if (todoList.childElementCount > 0) {

        if (prepend)
            todoList.insertBefore(newItem, todoList.childNodes[0]);
        else
            todoList.appendChild(newItem);
    }
    else {
        todoList.appendChild(newItem);
    }

    addToList.execute();
}
```

Moreover, we have initialized each element to recognize swipe gesture (which we will cover in the next section).

Note that items are inserted from the top, which will create the effect of pushing items down as new ones are inserted.

Similarly, for removing items, we have implemented delete list animation:

```
function deleteTodo(deletedItem) {
        var allItems = document.querySelectorAll
(".item-container:not([deleting])");

deletedItem.setAttribute("deleting", true);

--currentIndex;

var affectedItems = document.querySelectorAll
                    ".listItem:not([deleting])");

// Create deleteFromList animation.
var deleteFromList =
WinJS.UI.Animation.createDeleteFromListAnimation
(deletedItem, affectedItems);

// Take deletedItem out of the
//regular document layout flow
//so remaining list items will
//change position in response.
deletedItem.style.position = "fixed";
deletedItem.style.background = "transparent";
deletedItem.style.transform =
                    "translate(100px, 0px)";
deletedItem.style.opacity = 0;
deleteFromList.execute();
}
```

GESTURE ANIMATION

We want to delete an item using a left-to-right swipe gesture. So we have to capture and act upon the gesture. Once identified, we have to remove item from the current view and hide it. We have also added a CSS transform animation to move the element to right, which is the effect after someone does a swipe gesture.

After we have the move-to-right effect in place with delete list animation, we also need to make elements (class ="item-container") respond to gestures.

To implement this, first we need to capture mouse pointer events. More specifically, the following ones:

```
function prepareElementForGesture(element) {
    element.addEventListener("MSPointerDown",
        processDownEvent, false);
    element.addEventListener("MSPointerMove",
        processMoveEvent, false);
    element.addEventListener("MSPointerUp",
        processUpEvent, false);
    element.addEventListener("MSPointerCancel",
        processDownEvent, false);
}
```

Each of these will call the corresponding gesture event. For mouse pointer down / up, it will just be processing the up/ down event of gesture recognizer.

```
function processUpEvent(e) {
    lastElement = e.currentTarget;
    gestureRecognizer.processUpEvent(e.currentPoint);
}
```

For the move event, it is the same. However, instead of single point, we need to capture intermediate points collection for the target.

```
function processMoveEvent(e) {
        astElement = e.currentTarget;
        gestureRecognizer.processMoveEvents
        (e.getIntermediatePoints(e.currentTarget));
}
```

Next step is to bind mouse events to gesture recognizer (prepareElementForGesture):

```
gestureRecognizer.gestureSettings =
        Windows.UI.Input.GestureSettings.crossSlide;
        gestureRecognizer.crossSlideHorizontally = true;

var manupulationStarted = false;

gestureRecognizer.oncrossslіding = function (e) {
        deleteTodo(lastElement);
};

var itemsToGestureTrack =
        document.querySelectorAll(".item-container");

for (var index = 0; index < itemsToGestureTrack.length;
    index++) {
        prepareElementForGesture(
                itemsToGestureTrack[index]);
}
```

Since our use-case was to remove an item if user does a swipe gesture, therefore we have specified a gesture recognizer that will capture only the horizontal cross slide gestures.

Finally, we implement it for all the existing elements (todo items) as well as newly added elements. This is shown in the addNewTodo method above. Now we are all set!

When we run the application (need Win8 RC minimum), it will initially come up with the two default elements. We can add ToDo items but simply typing in the text and hitting enter.

To delete an item, on a touch-enabled device, we swipe left to right on the list item we want to delete. For a pointer driven device like the mouse, we do a left mouse button down and move horizontally before mouse up to delete the item.

CONCLUSION

Creating a todo list and adding custom animation or gesture recognition didn't require us to call native or COM API. We were easily able to add create / delete list and gesture animation through built in animation library provided by WinJS. These easy to use effects add to the default behavior and help make the app fluid and intuitive to use.

The Windows 8 *ListView* control already comes with built in list animation. However, if there are cases where you want to implement a custom list view with specific gesture support, the default selection model may not work. In such cases, this article will give you the pointers needed for implementing features as and when you need more control over the experience. ■

Mehfuz works as the Team Lead at Telerik focusing on JustMock. He is passionate playing around with the latest bits. He has been a Microsoft MVP, author of OS projects like LinqExtender and LINQ to flickr. Prior to working at Telerik , Mehfuz worked as a core member in many high volume web applications including Pageflakes that is acquired by Live Universe in 2008. He is a frequent blogger and was also a contributor and site developer at dotnetslackers.com. Follow him at @mehfuzh

REFACTORING AN EXISTING WINDOWS 8 STYLE APP FOR TESTABILITY

Raj Aththanayake explores some key concepts that involves making a testable Windows 8 style app (XAML/C#)

DOWNLOAD FILES >

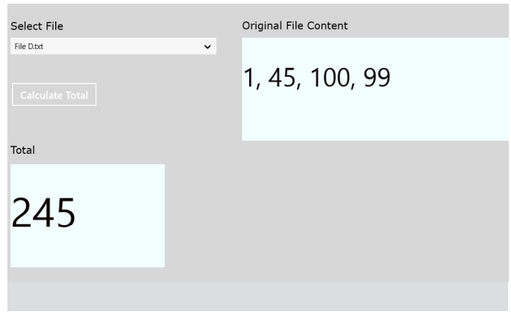
bit.ly/dncmag-metrotest

Windows 8 style app is a new type of application that runs on the WinRT runtime.

Microsoft has heavily invested in this exciting new platform for developing Windows 8 style applications. If you are new to Windows 8 style application development, I suggest you refer to *Develop great Metro style apps for Windows 8 article* (bit.ly/dncmag-msmetro). Since Windows 8 style application development is becoming popular, there is also a strong emphasis on the testability of these types of applications. In this article, we will look at how to refactor an existing Windows 8 style application for better testability. We will also cover some of the key challenges we face, and how to overcome those challenges so we can make a testable Windows 8 style app. We will be using the new VS2012 for our demonstration purposes.

FILE ACCESS DEMO – WINDOWS 8 STYLE APP

Shown here is a simple File Access application using WinRT, which allows us to select files from the user's document folder and display the content of the file within the "Original File Content" text block. We assume the file contains only numbers (comma delimited) and we use the "Calculate Total" button to calculate the sum of the numbers.



Let's look at a portion of the XAML mark-up that renders the UI

```
<!-- Select File -->
<TextBlock Text="Select File" Foreground="Black"
FontFamily ="Verdana" FontSize="20" Margin="0,90,0,0" />

<ComboBox x:Name="cboSelectFile"
SelectedItem="Name" DisplayMemberPath="Name"
Width="400" HorizontalAlignment="Left"
Margin="0,10,0,0" SelectionChanged="cboSelectFile_
SelectionChanged" />

<!-- Display Original File Content -->
<TextBlock Text="Original File Content" Foreground="Black"
FontFamily ="Verdana" FontSize="20" Margin="450,-68,0,0" />

<Grid Background="Azure" Height="200" Width="800"
HorizontalAlignment="Right" Margin="0, -32 630, 55" >
  <TextBlock x:Name="LblReadOriginalFile"
    TextWrapping="Wrap" Style="{StaticResource H5Style}"
    Width="2000" Height="100" HorizontalAlignment="Left" />
</Grid>

<!-- Calculate Total Button -->
<Button x:Name="btnCalculateTotal"
Content="Calculate Total" Height="50" Width="170"
FontSize="20" Margin="0,-290,0,0" Click="btnCalculateTotal_Click"
/>

<!-- Display Total Value -->
<TextBlock Text="Total" Foreground="Black" FontFamily ="Verdana"
FontSize="20" Margin="0,-50, 130, 55" />

<Grid Background="Azure" Height="200" Width="300"
HorizontalAlignment="Left" Margin="0,-40,130, 55" >
   <TextBlock x:Name="LblReadFile" TextWrapping="Wrap"
     Style="{StaticResource H3Style}" Width="991" Height="100"
     HorizontalAlignment="Left" />
</Grid>
```

Here is the code behind file:

```
public sealed partial class MainPage : Page    {
    private readonly StorageFolder storageFolder;

    public MainPage()       {
      this.InitializeComponent();
      storageFolder = KnownFolders.DocumentsLibrary;
      AddFiles();
    }

private async void AddFiles() {
   var filesList = await storageFolder.GetFilesAsync();
    cboSelectFile.ItemsSource = filesList;
}

 private async void cboSelectFile_SelectionChanged(object sender,
SelectionChangedEventArgs e)   {
        var selectedItem = ((ComboBox)sender).SelectedItem;
          LblReadOriginalFile.Text = await FileIO.
          ReadTextAsync((IStorageFile)selectedItem);
    }

private void btnCalculateTotal_Click(object sender, Windows.
UI.Xaml.RoutedEventArgs e)      {
        var fileContent = LblReadOriginalFile.Text;
        if (!string.IsNullOrEmpty(fileContent))   {
           var values = fileContent.Split(',');
         LblReadFile.Text = values.Sum(x => int.Parse(x)).ToString();
        }
    }

protected override void OnNavigatedTo(NavigationEventArgs e)   {
}
```

You might have already noticed that we have new Windows API references, which access the File System within the code behind. For example:

```
KnownFolders.DocumentsLibrary
storageFolder.GetFilesAsync();
FileIO.ReadTextAsync(..}
```

There is also some logic, for example splitting up the content and summing up within the btnCalculateTotal_Click event.

Now let's say, we want to write a Unit Test to verify the logic where we calculate the total. We can spin-up a new instance of the "MainPage" object, and try to Unit Test the private btnCalculateTotal_Click event. Even before I go any further, you can understand that this is not going to be a good Unit Testing experience. We feel like it is just wrong. And of course, we are not really writing a Unit Test, but we are testing the User Interface. Since there is no clear separation of concern between the UI and the business logic, it is almost impossible to test the total calculation logic.

MAKE FILE ACCESS DEMO APP TESTABLE

MVVM (Model-View-ViewModel) is a very well known pattern that has been around for a while. It has been widely used by WPF/Silverlight developers. This is the same for new Windows 8 style apps. In other words, you can apply the same MVVM concepts and patterns to Windows 8 style apps development. Since there are plenty of resources on MVVM pattern, I will only briefly describes the usage of MVVM and put more emphasize on the testing of Windows 8 style apps.

First let's briefly look at how to refactor this Windows 8 style app so we can test the calculate logic in isolation.

The View Model

By introducing a ViewMode, we can provide a clean separation between the UI elements and the presentation logic. The ViewModel is an abstraction of the XAML View, and plays a key role in Data Binding. ViewModel implements the INotifyPropertyChanged interface (please see the code). By implementing this special interface, we can take advantage of the XAML data binding capabilities. It keeps a track of the state of the View such as, loading files, changing the drop down list, button click event etc. If you are a WPF/Silverlight Developer, there is absolutely nothing new here. Time to get excited, as you can apply your existing skills to develop a testable Windows 8 style app.

```
public class FileDataViewModel : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler PropertyChanged;

    private string fileContent;
    public string FileContent
    {
        get { return fileContent; }
        set
        {
            fileContent = value;
            if (PropertyChanged != null)
            {
                PropertyChanged(this, new
                    PropertyChangedEventArgs("FileContent"));
            }
        }
    }

    private ICommand calculateCommand;
    public ICommand CalculateCommand
    {
        get
        {
            if (calculateCommand == null)
            {
                calculateCommand = new CalculateCommand();
            }
            return calculateCommand;
        }
    }

    //more property bindings (see the complete code sample)..
}
```

The above is a portion of the FileViewModel class. There are other properties which I can bind to, but for the purpose of the demo, I only consider the above 2 properties. In order to make the data binding work, the existing XAML elements require some changes as shown here:

```
<TextBlock x:Name="LblReadOriginalFile" Text="{Binding
FileContent}" TextWrapping="Wrap" Style="{StaticResource
H5Style}" Width="2000" Height="100" HorizontalAlignment="Left"
/>
<Button x:Name="btnCalculateTotal" Command="{Binding
Path=CalculateCommand}" CommandParameter="{Binding}"
Content="Calculate Total" Height="50" Width="170" FontSize="20"
Margin="0,-290,0,0" />
```

These declarative bindings will ensure that these properties can bind to the ViewModel properties when the data binding occurs. Also whenever a View Model's property value has been changed, it raises property change event, which we have subscribed to. This event can be used to notify the new value to the XAML data binding system.

CalculateCommand

The above ViewModel's CalculateCommand returns an instance of the ICommand interface. This command also binds to the button. The CalculateCommand class implements behavior of the click event.

```
public class CalculateCommand : ICommand
{

    public void Execute(object parameter)
    {
        var model = (FileDataViewModel)parameter;

        if (!string.IsNullOrEmpty(model.FileContent))
            model.TotalValue = model.FileContent.Split(',').Sum(num
             => int.Parse(num)).ToString();
    }

    public bool CanExecute(object parameter)
    {
        return true;
    }

    public event EventHandler CanExecuteChanged;
}
```

There are few ways of achieving the same result that I have implemented above - for example taking a service/model reference to the command class or passing as a delegate which contains the logic to execute etc. However to make the code simple, I have the logic implemented within the Execute method. What's important to understand here is that we do not have calls to private methods, or UI elements, which we saw before with the UI code behind file. Therefore we can test this logic in isolation. Execute method accepts an object parameter which is the ViewModel.

Unit Testing the calculate logic

Now we can easily test the logic that requires summing up the file content as shown below:

```
[TestMethod]
public void CalculateCommand_WhenValidDataValueExists_
ReturnsExpectedTotal()
{
        //Arrange
        string validFileContent = "3, 5";
        string expectedTotal = "8";

        var model = new FileDataViewModel() { FileContent =
            fileContent };
        var sut = new CalculateCommand();

        //Act
        sut.Execute(model);

        //Assert
        Assert.AreEqual(expectedTotal, model.TotalValue);
}
```

So far we saw pretty familiar stuff if you are a WPF/Silverlight Developer. But the key take away is you can utilize your existing MVVM skills with Windows 8 style apps. Let's look at some of the WinRT (metro) specific testing.

Defining a file repository and testing Windows API

We introduce a repository class, so we can abstract the direct access to file system via Windows API. This makes the testing of the domain model much easier, as we can stub out the repository during the Unit Testing.

If you take a look at the XAML code behind file, which I demonstrated earlier, there are some asynchronous calls that have been made to make the app responsive. In the new Windows API, there are over 80% of WinRT calls, designed to call

asynchronously. This also means that we can take the advantage of the new async, await keywords (in .NET 4.5). New repository methods can also be made async aware and the implementation can be simplified as shown below.

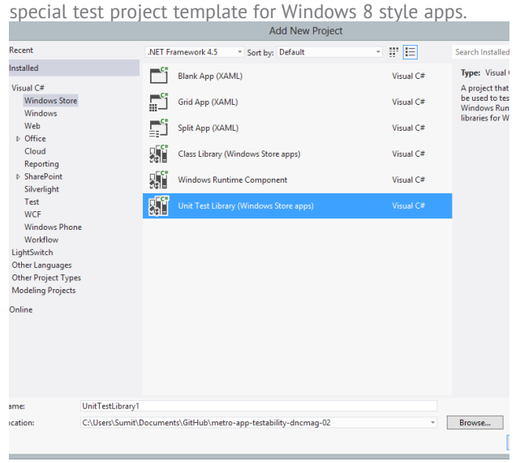
```
public interface IFileRepository
{
    Task<IEnumerable<FileItem>> GetFilesAsync();
    Task<string> GetFileContentAsync(FileItem storageFile);
}

public class FileRepository: IFileRepository
{
    private readonly StorageFolder storageFolder;

    public FileService()
    {
        this.storageFolder = KnownFolders.DocumentsLibrary;
    }

    public async Task<IEnumerable<FileItem>> GetFilesAsync()
    {
            var files = await storageFolder.GetFilesAsync();
            return files.Select (x => new FileItem(x.Name,
            x.DisplayName));
    }

     public async Task<string> GetFileContentAsync(FileItem
     fileItem)
     {
            varstorageFile=awaitstorageFolder.GetFileAsync(fileItem.
                Code);
             return await FileIO.ReadTextAsync(storageFile);
     }
}
```

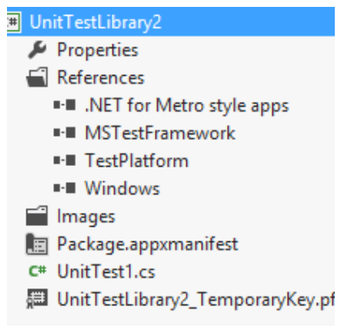
What I'm about to show you is an integration type test. This is because the test is going to touch the actual file system (via Windows API) and therefore it is not a Unit Test. Our intention of the test is to verify whether the GetFilesAsync method returns a list of file items/IEnumerable<FileItems>.

The first step is to create a Test project. However we cannot use the standard .NET MS Test project. **Windows 8 style apps required a special type of Unit Test project**. This is important to know because of couple of reasons.

a. The Windows 8 style apps have limited access to the standard .NET API. It only uses a subset of .NET API.
b. The Windows 8 style apps run in an AppContainer Sandbox. The app runs on its own container/app domain and does not have access to the outside world.

If you look at the new VS2012 project templates, there is a special test project template for Windows 8 style apps.





You notice that there is a .NET API for Windows 8 style apps, Testing components, and the Windows Runtime. There is also a new file Package.appxmanifest file. We will look at this file in more details bit later.

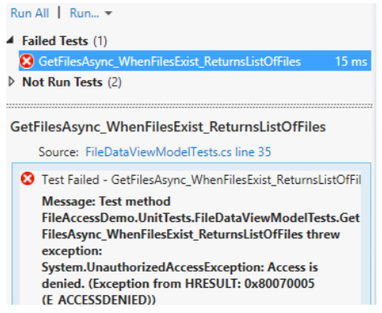
First I will write the test as shown below.

```
[TestMethod]
public void GetFilesAsync_WhenFilesExist_ReturnsListOfFiles()
{
        //Arrange
        var sut = new FileRepository();

        //Act
        var result = sut.GetFilesAsync();

        //Assert
        Assert.IsTrue(result.Any());
}
```

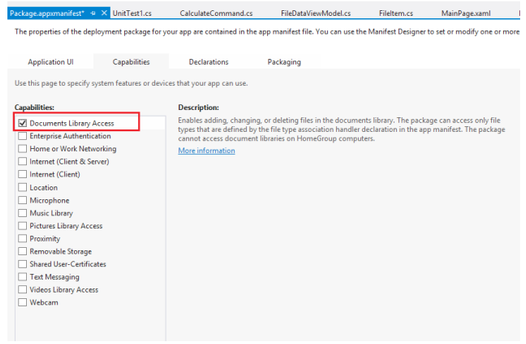
Obviously this code would not compile because GetFilesAsync returns an awaitable Task<T>. Writing tests for asynchronous method has never been easy before. It makes the tests harder to read and maintain. It can also introduce bugs within the test because of the complexity. Fortunately the new MS Unit Testing framework and also xUnit.NET have simplified asynchronous testing a lot. The same keywords, async and await can be used within your test to test the asynchronous code. Please see the code below.

```
[TestMethod]
public async Task GetFilesAsync_WhenFilesExist_
ReturnsListOfFiles()
{
        //Arrange
        var sut = new FileRepository();

        //Act
        var result = await sut.GetFilesAsync();

        //Assert
        Assert.IsTrue(result.Any());
}
```

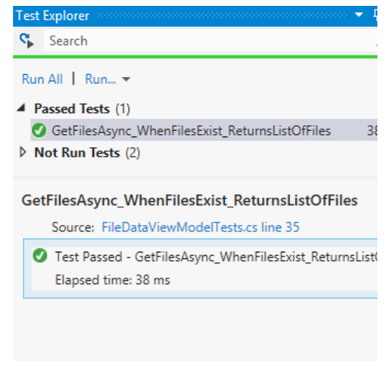
Let's run the above Test method assuming it returns a list of files from the file system. However as you can see here, we get an "UnAuthorizedAccessException".



The reason for this exception is that the test runs in its own test context. It does not have the visibility of the metro configuration that runs under the production code. This is why we need a special type of Unit Test project for metro apps so we can configure the Package.appxmanifest file and our test can access the File System. Let's open the Package.appxmanifest file and configure the Capabilities settings as below.



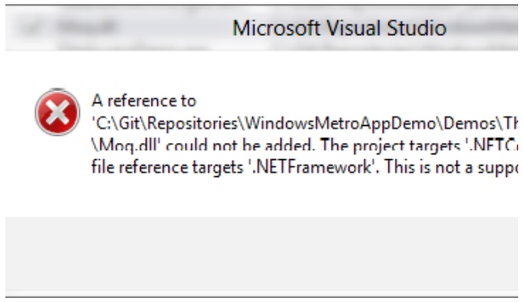
Now if we run the Unit Test, the test passes as shown below.



MOCKING/STUBBING IN WINDOWS 8 STYLE APPS

In the WinRT environment, isolating your dependencies is not the same as the legacy Desktop environment. The popular isolation/mock object frameworks (i.e. Moq, RhinoMock, NMock etc.) currently are not compatible with Windows 8 style apps testing. One of the main reasons for the incompatibility is that there is limited access to the .NET BCL. The methods that require generating dynamic proxy instances on the fly are not available in the WinRT environment.

For example, if we attempt to add Moq to the Windows 8 style Unit Test project,, we get the following error.



You might be already familiar with the new Visual Studio Fakes framework. This works great with non-metro apps. However currently Visual Studio Fakes do not have any support for Windows 8 style testing.

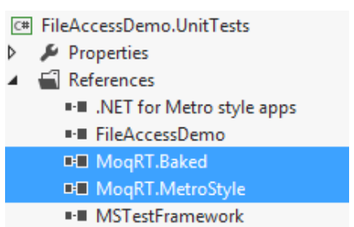
Below is a work around for your Windows 8 style applications using Moq and MoqRT

MoqRT

MoqRT is an open source library that was developed by Matthew Baxter. It is exactly the same as the Moq isolation framework, but it has been compiled against the WinRT library. The key difference is that the MoqRT does not generate the mock objects on the fly. It uses .NET Remoting to listen to the new build within the test assembly and generates dynamic proxy assemblies (a process called baking). It is still fairly new (alpha release) so there may be some bugs you might come across.

I personally use this for Windows 8 style apps testing isolation needs and I find it quite useful. It is bit trickier to set it up, but once you have configured it, you can Moq APIs as you would normally do with the standard Moq. If you are interested setting up MoqRT, I encourage you to read the "Getting Started" section.

Baking Assembly and MoqRT



Now assume if the FileRepository requires an ILoggerService and we want to verify whether the Log method has been called only once.

The system under test

```
public FileRepository(ILoggerService loggerService)
{
        this.loggerService = loggerService;
}

public async Task<IEnumerable<FileItem>> GetFilesAsync()
{
    loggerService.Log("retrieving files");

    var files = await storageFolder.GetFilesAsync();

     return files.Select(x => new FileItem(x.Name, x.DisplayName));
}
```

As you see, we can use the Moq API as we would normally use in non-metro testing.

SUMMARY

We looked at some of the key concepts that involved making a testable Windows 8 style app. We briefly utilised the MVVM pattern and make the Windows 8 style app testable. We also looked at areas such as testing asynchronous API, and Windows 8 style integration testing. Finally we briefly looked at the options we have with stubbing and mocking with Windows 8 style app development. ■

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

INTERVIEW WITH JON SKEET

UP, CLOSE & PERSONAL WITH JON SKEET

In the second edition of this Magazine, we are happy to have Jon Skeet in our 'virtual' interview chair. Jon is a Prolific community contributor (checkout his Stack Overflow badges on the next page), a leading authority on C# and author of multiple books. Jon has been a Microsoft MVP since 2003. Currently he is working at Google.

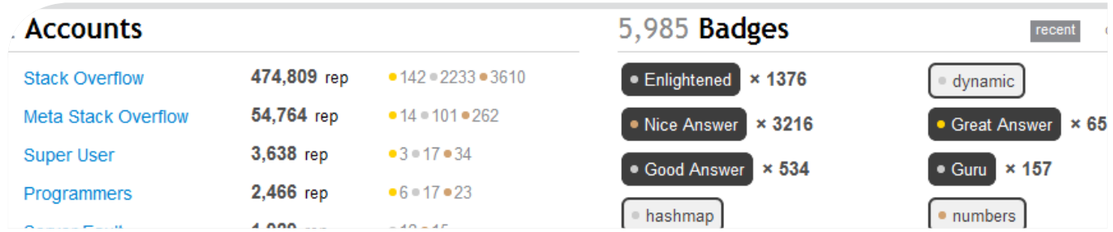
Ladies and Gentleman, without further ado, presenting Jon Skeet – A Software Engineer and a Gentleman.

DNC: Hello Jon, we are really glad you could take time off your schedule for this interview. To start off with, we would all like to know more about 'Jon Skeet', please tell us how it all began? How did Jon get started with Computers?

JS: My first computer – shared by the whole family – was a Sinclair ZX Spectrum 48K, which we bought when I was 8. Over time I bought other models of the Spectrum over time, but then gradually moved into using a PC. For a long time I spent most of my time on computers just playing games, but programming has always been in the mix too.

DNC: What were some of the challenges that you took up on the Sinclair? Challenges that kept getting you deeper into Computer Science?

JS: One of my first "big" projects on the Spectrum was writing a version of Logo. At school, we had BBC Micros and Logo was used as an introduction to computing; I thoroughly enjoyed it and wanted to use it at home, but we didn't have a Logo interpreter. I didn't know any trigonometry and had no sense of well-structured programming, but I persevered and ended up with a fairly reasonable implementation. The manual that came with the Spectrum was wonderful – I literally learned simple trigonometry from that, long before we covered it in school.



I still remember one subroutine, to either draw or remove the triangle representing the turtle – it was just a case of doing the same thing with an XOR mask. The subroutine started at line 7000, and if things didn't look right, I just added calls to "GOSUB 7000" until it worked. I realized that as soon as you'd got more than one call in a particular place, something was wrong… but I didn't always know exactly what.

DNC: Cool, we wanted to save this question for later, but now that you have mentioned building Logo, when you were… 8(?)… Did you ever consider writing a programming language of your own?

JS: Occasionally… but I don't think I have a strong enough imagination to really create something revolutionary. (I certainly don't have the *time* to do so.) I think I could perhaps provide some useful help in a team designing a new language, but it's much easier to be an armchair language designer – commenting on mistakes made in existing languages – than it is to do the real thing

DNC: Well, there are lots of armchair specialists and then we have Jon Skeet – a league apart. We all know about your love of and command over C#. A lot of your work is in Java. Which other programming languages do you use at work and play? Any hobby languages you are trying to pick up or would like to pick up? Why?

JS: My guilty secret is that I really *don't* know any other languages well. I know enough VB to answer a few Stack Overflow questions, and I can read a certain amount of F#, but I'm shockingly un-cosmopolitan in that sense. I definitely *want* to learn more F#, and I'm hoping to learn Go at some point, as colleagues have raved about it.

The tricky thing is finding an actual use for things – most of my hobby programming is either building .NET libraries such as Noda Time or is related to writing about C#. I'm not much of an *application* developer, which is where I think it tends to be easier to give another language a try. I now have a Raspberry Pi though, which I'm hoping will fire my imagination somewhat. Although I have Mono working on that, I'm going to try using it as an excuse to use Go.

> *"My guilty secret is that I really don't know any other languages well. know enough VB to answer a few Stack Overflow questions, and I can read a certain amount of F#, but I'm shockingly un-cosmopolitan in that sense."*

DNC: Very nice and we will now start following the Go tag on Stack Overflow. As a Polyglot Programmer, how often do you get the, "umm... that's interesting, it works differently in [the other language]"? Anything specific that comes to mind?

JS: Closures spring to mind. When you refer to an "outer" variable within a closure, what does that really mean? Different languages have different answers... and within C#, the answer is even changing over time. (If you capture the iteration variable in a foreach loop, in C# 2.0-4.0 you'll end up capturing the same variable every time which takes on each value of the sequence; in C# 5 it will capture a new variable in each iteration.)

DNC: Coming back to your love for C#, what is the one feature in C# you always wanted but never got implemented OR If you had to add 'the one' feature in C#, what would it be?

JS: I'd like to see more support for immutability in one form or other. There are lots of degrees to which support could be added. For example, I love the behavior of anonymous types – simple shallowly-immutable classes which override Equals, GetHashCode and ToString in useful ways, and allowing you to give meaningful names to the properties they expose. If only we could do the same for named classes with the same ease! Creating a "normal" class which exposes a few properties, has a constructor taking those values, and overrides the various methods takes quite a bit of boiler-plate code, particularly if you want to create genuine read-only variables rather than having properties with private setters.

Similarly, I'd love more support for *constructing* immutable objects – object and collection initializers are great, but only work for mutable types. You can use the builder pattern to get around this, but it's a bit ugly. I wonder whether language support for the builder pattern would be an interesting addition to the language. At least named arguments and optional parameters give *an* alternative.

Of course, these ideas are only thinking about immutability at a shallow level. Deep immutability is harder, but gives bigger advantages. I know the C# team are aware of these benefits – if they can come up with a thorough and efficient solution, that could be amazing.

DNC: C# started off as a statically typed language but has increasingly adopted a lot of dynamic language traits. Is there anything that can be added to C# that will be ground breaking, or is it going to be evolutionary from here on?

JS: The interesting thing about this is that a lot of the features which appeared earlier in dynamic languages actually aren't particularly dynamic. LINQ is statically typed, but list comprehensions have appeared in Python (and of course other languages) before. Likewise "var" makes C# *feel* a bit more like a dynamic language – but without losing any type safety. Of course the "dynamic" type in C# 4 is a different matter.

> *I think the new async/await feature of C# 5 (and VB, of course) is a game-changer.*

I'm aware that similar features exist in some other – less mainstream – languages, but I think it's going to be a huge deal for regular developers. The ability to write asynchronous code without turning your codebase into spaghetti is astounding. I'm really excited about it as a feature, and I'm hoping it will transform how we all think about writing code, just as I believe LINQ changed how many people think about data manipulation.

DNC: Apart from Immutability, which seems to have only changed with respect to support for ReadOnlyDictionaries in 5.0, what's on your wish-list for C# 6.0?

JS: Whatever I could propose, I'm sure the C# team have something far more awesome in mind. I've mentioned better support for immutability, and I have a few other minor language features I'd be interested to see – good tuple support could make some code rather simpler – but I also think **it's important to avoid C# becoming a kitchen sink language**. The team have to keep the bar for new features really high; there's only so much they can expect developers to learn over time. I was lucky enough to learn C# right from v1.0, but can you imagine *starting* with C# 5? How long would it take you to learn about the whole language, including the more recent features? (I've deliberately kept away from unsafe code, for the most part – I've cut out a whole section of language knowledge, hoping I'll never need it.)

DNC: Tell our readers more about the Protobuf (.net port) project. Apart from Protobuf is there a 'top secret' personal project that you are working on?

JS: "Protocol Buffers" are used to represent structured data in Google. There's a reasonably simple language to describe the structure, and then the wire representation is efficient enough to be used for storage as well as transmission. It's a platform-independent format, which is obviously useful – so long as you've got a supporting library for the platform you want to use. Google turned Protocol Buffers into an open source project fairly soon after I joined, and I decided to write a .NET port. My version is reasonably close to the Java and C++ code provided by Google, but there are other ports available. In particular, I know Marc Gravell has put a lot of interesting work into his protobuf-net version, which takes a different approach.

I don't do much work on Protocol Buffers now – most of my open source coding time is spent on Noda Time, which started off as a port of the Joda Time date/time library for Java, but has ended up being a port of the "engine" of Joda Time, but with a fairly different API on top of it. It's been a really good project for learning about API design – as well as for finding out all kinds of interesting corner cases in human representations of time.

DNC: Please give us three interesting trivia (Jon Skeet special) questions on C# for our readers?

JS: Okay, a few bits of trivia of varying levels of obscurity:

Q1. What constructor call can you write such that this prints True (at least on the Microsoft .NET implementation)?

```
object x = new /* fill in code here */;
object y = new /* fill in code here */;
Console.WriteLine(x == y);
```

Note that it's just a constructor call, and you can't change the type of the variables...

Q2. How can you make this code compile such that it calls three different method overloads?

```
void Foo()
{
   EvilMethod<string>();
   EvilMethod<int>();
   EvilMethod<int?>();
}
```

Q3. With a *local* variable (so no changing the variable value cunningly), how can you make this code fail on the second line?

```
string text = x.ToString(); // No exception
Type type = x.GetType(); // Bang!
```

DNC: Readers please submit your responses to bit.ly/dncmag-js-quiz. Let us see how long, till these questions come up on StackOverflow ;)

DNC: Coming to your journey so far at StackOverfow, can you pick one or two interesting conversations? (Personally I love the Meta Post on Jon Skeet :))

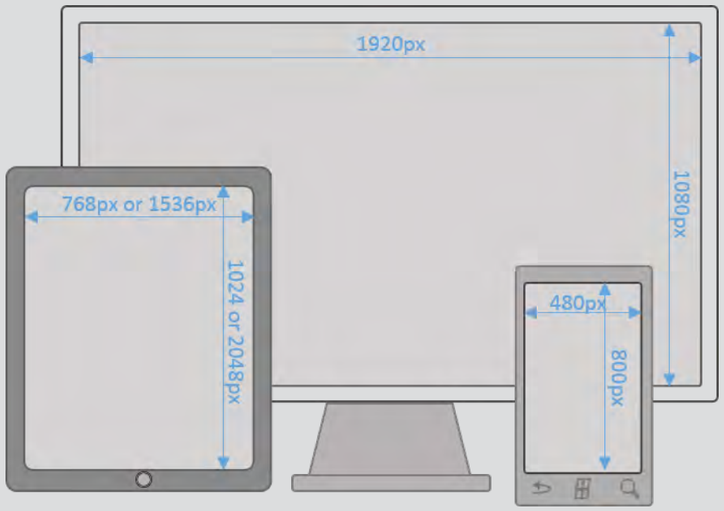
JS: : Yeah, I like the Jon Skeet Facts page too. As I've said elsewhere, it's not *really* about me at all – I'm just a convenient name to hang on the joke.

The posts I've enjoyed most are the ones where either I've learned more about C# (often through deep spec-diving) or ones where I've had a good excuse to write some really evil code – usually copiously commented with "This is horrible. Do not use in real life under any circumstances." I was immensely gratified when Eric Lippert added a comment to one such post that it was "the best abuse of C# I've seen in a while." One of the nice things about being an *amateur* C# developer is that I get to write code like that without it causing any pain.

DNC: If budding/young developers aspire to be Jon Skeet - Programmer Extraordinaire, where do they start? Is it possible to 'deep copy' Jon Skeet's command over programming?

JS: To start with, set the bar much higher! Rumors of my competence are greatly exaggerated, as you'd no doubt hear if you spoke with any of my colleagues. I've read the C# spec more closely than most people, I have a gift for deliberately abusing language features, and my communication skills are pretty good (through much practice), but beyond that I'm not that special. I have strong opinions on API and language design, but ask me to write a full application and I'd be hopeless.

If folks really want to tread a path in the same general direction as me, they should get plenty of practice at writing – write a blog, participate on Stack Overflow, write a book! Communication is really important. Take all advice (including this) with a large pinch of salt, write lots of code and reflect on what's worked well and what's worked badly. Combine that with reading specifications carefully and the love of an amazing (and infinitely more talented) partner, and you're set. Simple, really! ■



BUILDING A RESPONSIVE UI USING TWITTER BOOTSTRAP & ASP.NET MVC

DOWNLOAD FILES >

bit.ly/dncmag-twbs

Responsive UI in Web Development is a term that loosely correlates to a design that can adapt itself to various screen sizes. With the proliferation of 'Web consumption' devices, most internet sites are no longer viewed on large screen desktops only. Today people view sites from Mobile devices like smartphones with 3" screens to tablets with 10" screens and a variety of sizes in between. The web application you build is expected to be able to handle all these screen sizes gracefully and 'adapt' to them. Such design is referred to as Responsive UI.

WAYS TO BUILD RESPONSIVE UI

Sometime back, the ASP.NET site was revamped to be responsive to various screen sizes. We can see how it changes if we resize our browser.

A scaled version of the two images is shown in the next page.

The behavior of the ASP.NET site is achieved via CSS3 Media Queries and view-port width detection. Thus irrespective of which browser you are viewing it in, the view changes to accommodate itself to the best possible fit. In this article, we will see in-depth how we can leverage CSS3 Media Queries and view-port detection to build responsive UI.

However, another way to support smaller screens is by having independent views for different devices based on Browser/Device detection techniques. Libraries like jQuery.Mobile leverage this. ASP.NET MVC 4 has support for this built-in via the ViewSwitcher Controller and using targeted mobile CSS + JavaScript libraries jQuery.Mobile. It is okay to choose one way over the other depending on the intended audience and level of interaction required with the site. If the site is all about consumption of information, a CSS3 Media Query approach works fine. However if user interaction involves heavy two way interaction, then a more targeted approach like jQuery.Mobile can be adopted. Finally for high precision activities like Gaming, native applications are a very good idea.

"... a media query is essentially a condition on a particular 'media feature'..."

THE META VIEWPORT AND CSS MEDIA QUERIES

The Meta ViewPort Tag

Lets get a little deeper into the viewport Meta tag and see how it works. The viewport tag is defined as follows:

```
<meta name="viewport" content="width=device-width" />
```

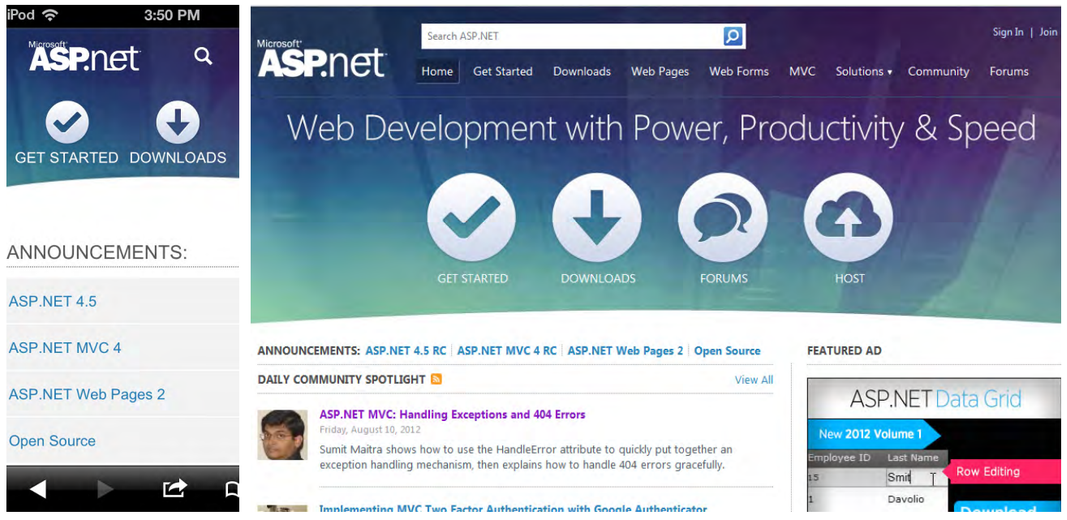
So what does content="width=device-width" mean? It means that the width of the 'area' of the browser is set to whatever value that current device returns as optimal. Why is this important? If we consider a device with a small screen like iPhone, it return it's width as 320px, so now if in our web page, we set width to 80%, the width will automatically be set to 80% of 320px. Similarly for a desktop browser, the browser usually reports its current width as the device-width. As we can see in the screenshots from our sample, different browser widths elicit different width responses.

Okay great, now we know the width of our page how does this help in making them appear 'nice' on each of these devices? Answer is *CSS Media Queries*.

CSS Media Queries

Back in HTML4/CSS2 days, there were two different media types supported, 'screen' and 'print'. These allowed explicit labeling of Style Sheets are per the target purpose.

In HTML5, the concept of Media queries was introduced to enable checking of conditions for one or more media type 'features'. At the time of this article, W3C has thirteen features described, amongst which the most prominent ones



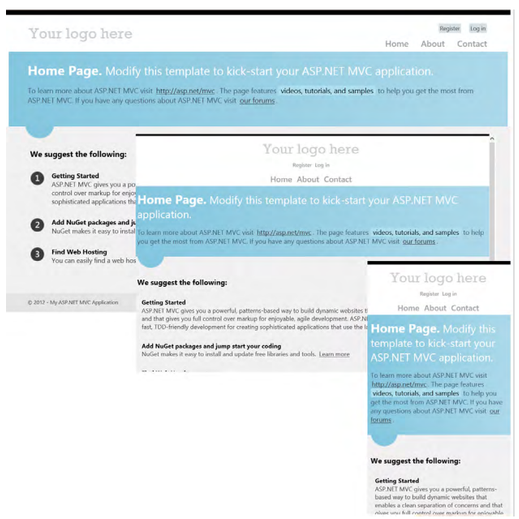
are 'width', 'height' and 'color'.

If we look a little deep, the picture begins to clear up for us. We can get device-width via JavaScript, and we could then use the width media type feature to check the width and apply different CSS as required.

For example:

```
@media (max-width: 767px) {
 .section-width{
   width: 747px;
   margin-left: 10px;
   margin-right: 10px;
 }
}

@media (max-width: 480px) {
 .section-width{
   width: 470px;
   margin-left: 5px;
   margin-right: 5px;
 }
}
```

As we see above, we have used the @media (...) condition to specify a certain condition based on which all classes defined in it are picked up and used.



In the above condition, we have a css class section-width. This sets the width and margin of whatever it's applied to (e.g. a). We have setup the media queries such that if the device reports max-width as 480 pixels, the class sets the width to 470 and give a margin of 5px. However if the max-width is reported to be 767 pixels, the class sets the width to 747 and provides a left and right margin of 10px.

The query itself is very simple in this case, wherein we are checking for max-width media feature. As per W3C specs, there are thirteen such media features. You can refer to the spec here - http://www.w3.org/TR/css3-mediaqueries/

So a media query is essentially a condition on a particular 'media feature' based on which a completely different set of CSS styles can be picked for each condition that is checked.

However building multiple sets of CSS for each possible device is actually a non-trivial exercise. This is where boilerplate like the one for default ASP.NET MVC4 or Twitter Bootstrap come in and make our lives easier. These templates provide us with a prebuilt set of styles and scripts that have the most common sizes considered and are easy to extend. We are all familiar with the ASP.NET MVC4 template. See the image to your left for a default ASP.NET sample in action.

The default ASP.NET CSS bootstrap is a good starter template, but is kind of muted in terms of design aesthetics. Towards that end, Twitter Bootstrap is an HTML5 + CSS boilerplate, which encapsulates the design aesthetic of Twitter and adds-on Responsive UI elements to help build a modern looking web site.

HELLO, TWITTER BOOTSTRAP

The Twitter Bootstrap package consists of predefined CSS styles, Components and jQuery plugins. It is licensed under Apache 2.0 and is free for commercial use. We take a quick look at the Components and plugins available to us before jumping into actually using these components in an ASP.NET MVC application.

Components

Following are the Component Categories at the time of writing –

- Buttons

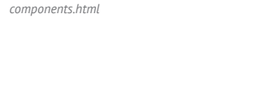
o Button Groups

o Button Dropdown

o Split Button Dropdown

- Nav, tabs, and pills

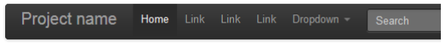
o Basic tabs



o Basic pills

o Other variations like stacked pills, Tabs with dropdowns, tabs to the left or right.

- Navigation Bar



- Labels

o Inline Labels

- Badges

Name	Example	Markup
Default	1	`<span class="badge">1</span>`
Success	2	`<span class="badge badge-success">2</span>`
Warning	4	`<span class="badge badge-warning">4</span>`
Important	6	`<span class="badge badge-important">6</span>`
Info	8	`<span class="badge badge-info">8</span>`
Inverse	10	`<span class="badge badge-inverse">10</span>`

Source: Twitter Bootstrap Documentation at http://twitter.github.com/bootstrap/components.html

- Other Components

There are many more components like Typography (that marks a section of special highlighting), Thumbnails (to show actual image and thumbnails in various layouts), Alerts (classes for highlighting various alert conditions), Progress bars (showing progress of actions).

Plugins

Apart from the CSS styling classes, the template also comes along with a number of jQuery plugins that are in the bootstrap.js file. These are

- Modals: for modal dialogs (of course)
- Dropdowns for creating dropdown menus on buttons etc.
- Scroll Spy: A nifty plugin that updates 'nav' targets as you scroll a long page.
- Tab: Toggling for tabbed interfaces.
- Tooltips: Nifty tooltip plugin.
- Popover: Extension of tooltip for more extensive popup notes.
- Alert Messages: A tiny plugin that adds a 'close' button to alert message components.
- Buttons : Allows grouping and toggling of grouped buttons.
- Collapse: A plugin that gives the traditional 'accordion' functionality to a set of layout panels.
- Carousel: A generic plugin for cycling through a list of contents like images.
- Typeahead: A simple autocomplete plugin that uses an array as its data source.

Combined with the Components, Plugins and responsive CSS3 based media queries, Twitter Bootstrap offers a potent platform for building nice looking HTML5+Responsive UI web applications quickly.

We will now go through what it takes to build an ASP.NET MVC Application using the Twitter Bootstrap framework and make some tweaks so that it looks 'cool' on small devices.

ASP.NET MVC AND TWITTER BOOTSTRAP

Now that we have got an idea of the capabilities of Twitter Bootstrap, we will see a hands-on of how to integrate Twitter Bootstrap in our ASP.NET application. We will dig deep into the Styles and scripts we use and once done, we will wrap it into a nice Visual Studio project template for future use.

Updating to Twitter Bootstrap

- We start off with an ASP.NET MVC 4 Web Application's Internet Template. First couple of things to do is to install the Bootstrap package from Nuget and update the jQuery package if its version is less than 1.7.2.

```
PM> install-package Twitter.Bootstrap
PM> update-package jQuery
```

- Next we update the BundleConfig.cs class to include the twitter BootStrap scripts and css.

```
bundles.Add(new ScriptBundle("~/bundles/bootstrap").Include("~/Scripts/bootstrap.min.*"));

bundles.Add(new StyleBundle("~/Content/bootstrapcss").Include("~/Content/bootstrap-responsive.min.css", "~/Content/bootstrap.min.css"));
```

- We include the newly added bundles to our _Layout.cshtml page and remove the default css bundle (jQuery UI overlaps certain twitter bootstrap styling)

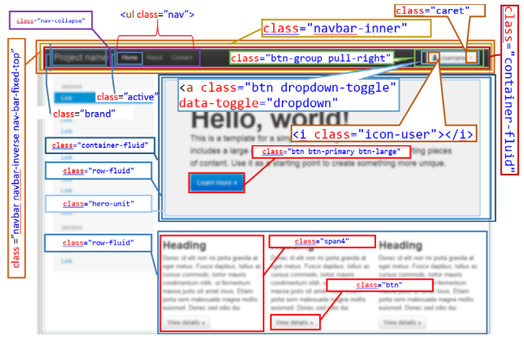
```
@Styles.Render("~/Content/bootstrapcss")
```

- At the bottom of the page, we add the BootStrap script bundle. It's important that bootstrap scripts be loaded after the jQuery bundle because it uses jQuery.

```
@Scripts.Render("~/bundles/jquery")
@Scripts.Render("~/bundles/bootstrap")
```

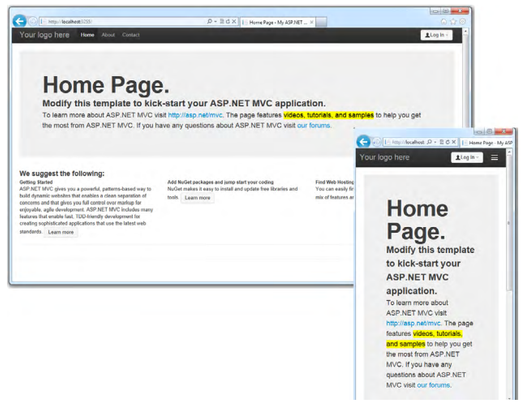
- We are all set with the initial setup. Now we'll go hands on with the layout changes. We are aiming for Twitter Bootstrap's Responsive UI sample look. Like the ASP.NET default template, it also has a 'Home', 'About' and 'Contact' links and a Login Widget on the top right. The default ASP.NET sample does not have the left hand side list of links so we'll ignore it for now.

- The following graphic highlights the various css classes used. The labels with class definitions only imply elements. Some of the other elements have been explicitly specified.



BootStrap Release 2.1: While this article was in production, BootStrap 2.1 was released. Default Navigation bar is now white in color. However the navbar-inverse class restores the original black Navigation bar. The nuget package for the final release of the article has been updated to BootStrap 2.1.

- When we update the _Layout.cshtml, Index.cshtml and _LoginPartial.cshtml to use the above styles instead of the default styles, our UI comes up as follows and we can see the changes that happen when a browser is resized.

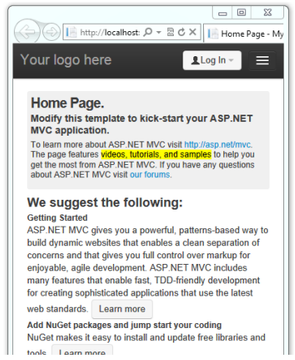


- Now as we can see the 'feature' on the home page is taking up the entire mobile screen space. It seems the 'hero-unit' does not respond to media size changes. We can reduce the size of the 'featured' section. To do this we need to update the bootstrap-responsive.css

- We add the required classes in the media section that's for small smartphone screens and setup sizes appropriately

```
@media (max-width: 480px)
{
.hero-unit {
  padding: 5px;
  margin-bottom: 5px;
  background-color: #eeeeee;
  -webkit-border-radius: 3px;
    -moz-border-radius: 3px;
        border-radius: 3px;
}

.hero-unit h1 {
  margin-bottom: 2px;
  font-size: 20px;
  line-height: 1.2;
  letter-spacing: -1px;
  color: inherit;
}
.hero-unit h2 {
  margin-bottom: 5px;
  font-size: 14px;
  line-height: 1.2;
  font-weight:bold;
  color: inherit;
}
.hero-unit p {
  font-size: 12px;
  line-height: 1.2;
  font-weight: 200;
  color: inherit;
}
```

- When we run our application in a small browser window it looks as follows

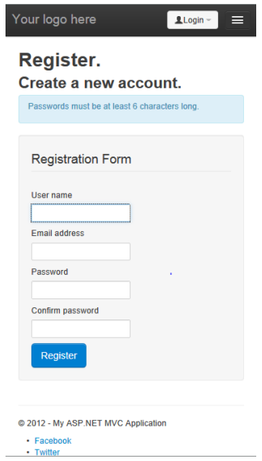


As we see here, now the entire content of the page has fit in the small browser space. This takes care of the home page. Let us update the Registration and Login pages with the Form classes in the bootstrap

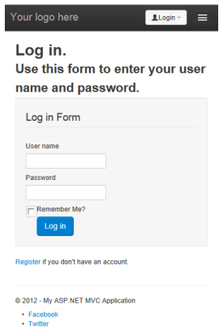
- We update the markup of the Register.cshtml as follows. We have removed the <ol> for all the fields and added the class 'span3' for the textboxes. We applied styling for the button and wrapped the registration fields in a div with the class set to "well" as built in component.

```
<div class="alert alert-info">
   <p class="message-info">
      Passwords must be at least
      @Membership.MinRequiredPasswordLength characters long.
   </p>
</div>
@using (Html.BeginForm())
{
   @Html.ValidationSummary()
   <div class="well">
      <fieldset class="control-grouperror">
         <legend>Registration Form</legend>
         @Html.LabelFor(m => m.UserName)
         @Html.TextBoxFor(m => m.UserName,
               new { @class = "span3" })
         @Html.LabelFor(m => m.Email)
         @Html.TextBoxFor(m => m.Email,
               new { @class = "span3" })
         @Html.LabelFor(m => m.Password)
         @Html.PasswordFor(m => m.Password,
               new { @class = "span3" })
         @Html.LabelFor(m => m.ConfirmPassword)
         @Html.PasswordFor(m => m.ConfirmPassword,
               new { @class = "span3" })
         <input type="submit" value="Register"
            class="btn btn-primary btn-large" />
      </fieldset>
   </div>
}
```

- Result of the above change is shown on the next page:



- We update the Login page similarly

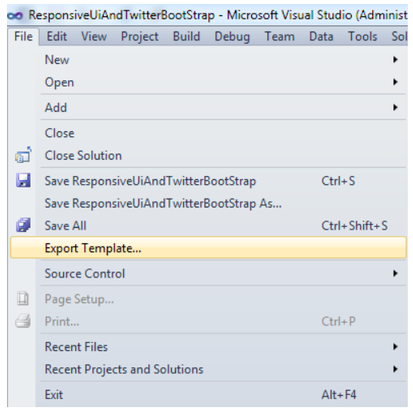


CREATING A VISUAL STUDIO PROJECT TEMPLATE

Now that all the default pages have been updated and we have got a handle of how to use the Bootstrap boilerplate, we would like our MVC projects to start off looking like this without having to go through the exercise of doing all these changes again. Visual Studio provides an easy way to create a Visual Studio project template. Just export it as a template and you are good, you have a neat ASP.NET MVC template that starts off with the Twitter Bootstrap.

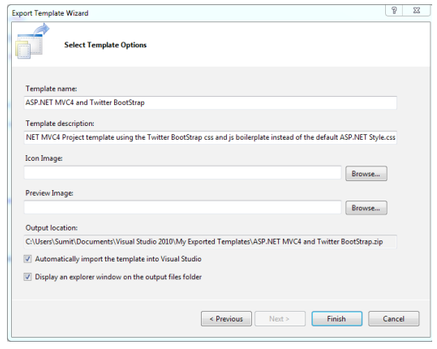
- In Solution Explorer select the Project

- From File Menu of VS select "Export Template..."

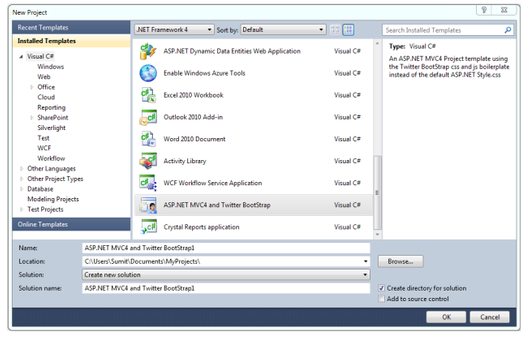


- Select Project Template and Click Next

- Next update the details like name and description as required and hit Finish.



- Start new instance of Visual Studio and select Installed Templates > Visual C#. You will see the newly created Template



- There you have it, your very own project template that creates MVC4 + Twitter Bootstrap based web applications.

CONCLUSION

We saw a host of things in this article today, starting with the theory behind Responsive UI and how CSS3 and HTML5 support building of Responsive web UI easily. Next we saw how Media queries worked. We took a look at a well established boiler plate for Responsive UI design – the Bootstrap, by Twitter. We also saw how to replace default ASP.NET styling with Twitter Bootstrap and finally we created a Project template that can server as starter MVC projects using Bootstrap.

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

INTRODUCING WINDOWS AZURE CACHING

Shiju Varghese introduces the new Windows Azure Caching (Preview) introduced in the Windows Azure June Release - SDK 1 and demonstrates how to use Windows Azure Caching for your Windows Azure apps for building high performance cloud applications.

DOWNLOAD FILES >

bit.ly/dncmag-azc

Today, scalability and performance are two of the most challenging areas in web application development. The performance optimizations are very critical for web apps when they are being accessed by several thousands of users concurrently. Caching solutions can have a key role for building high performance web apps which can avoid frequent round trips to backend sources, thus giving you better performance and better scalability.

This article will give an introduction to the new Windows Azure Caching (Preview) introduced in the Windows Azure June Release - SDK 1.7 and will demonstrate how to use Windows Azure Caching in your Windows Azure apps for building high performance cloud applications. This article will cover the following:

- Introduction to Windows Azure Caching
- Introduce the different type of deployment options available for Cache and how to configure Cache Cluster
- How to build Cache Clients for accessing Windows Azure Caching
- Working with the Windows Azure Caching
- How to use Windows Azure Caching as the storage mechanism for ASP.NET session state and ASP.NET Page Output Caching

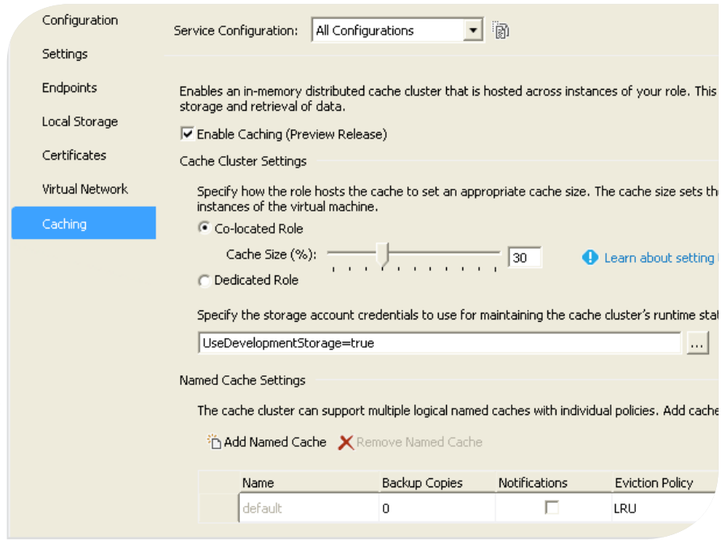
WINDOWS AZURE CACHING

Windows Azure is providing a caching service which can be used for your Cloud apps hosted on Windows Azure. The Caching service is a highly scalable, distributed, in-memory Cache which allows the developers to cache all data types, thus reducing the load on databases and improving the performance of the Windows Azure applications.

Using Windows Azure Caching, you can do the following:

- Cache data on Windows Azure Caching
- Use as storage mechanism for ASP.NET session state and ASP.NET page output caching

The new Windows Azure Caching (Preview) allows you to deploy Cache on the virtual machines in which you have hosted your Windows Azure Cloud Services. Now you can cache on Web Roles and Worker Roles that can be easily scaled, managed and monitored just like your Cloud Services. You can also use the existing Windows Azure Shared Caching along with the new Caching option on the Cloud Services.



CACHE CLUSTER

Cache Cluster is the Cloud Service role instance that you use for hosting your Cache.

The following are the two types of deployment options available for configuring the cache.

- **Co-located Role caching** - The Cache shares the Virtual Machine resources with the Cloud Service application. In this option, you can share the resources of Web Role and Worker Role for Cache.

- **Dedicated Role caching** - The role instances dedicated only for your caching. In this way, you would be deploy the Cache in a dedicated VM by using a Cache Worker Role.

Configuring Co-located Role Cache Cluster

If you want to configure your cache along with your Role instances, you can choose the co-located Role cache cluster as the deployment option.

The following steps will configure the Windows Azure Role for using co-located Role Caching

1. In Visual-Studio, right-click on the Windows Azure Role, and click properties.

2. In the properties window, choose the *Caching* tab.

3. In the Caching tab, check the *Enable Caching (Preview Release)* checkbox.

4. Choose *Co-located Role* for the Cache Cluster settings. By default, this will be Co-located Role when we enable Caching.

5. Choose the percentage of Cache size. By default Cache size will be 30 percentage. This means that 30 percentage of VM memory is allocated for Cache. You can change the Cache size to whatever you want for your caching solution.

6. Configure the settings of Caches. We can specify the *Time to Live (min)* for the time interval of Cache for expiry and *Expiration Type* for specifying the expiration type of Cache object.

By default, value of "The Time to Live" would be 10 minutes and the Expiration Type would be Absolute. The Absolute expiration type is specifying that the timer for the expiration would start after the data is added into Cache object. If the expiration time is 10 minutes, the cache will be expire after 10 minutes after the item is added into Cache. The other

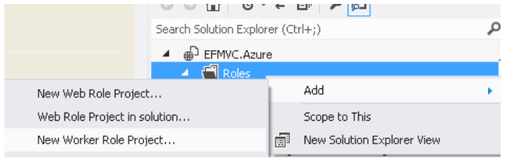
expiration types are *Sliding Window* and *None*. If the Expiration Type is Sliding Window, the timer for the expiration time would be reset after each item is retrieved from the Cache. If expiration type is *None*, items in the cache will not be expired. In this scenario, you have to specify "Time to Live (min)" as 0

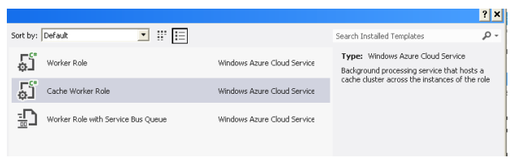
Configuring Dedicated Role Cache Cluster

If you want to host your cache in a dedicated VM, you can choose the Dedicated Role cache cluster as the deployment option.

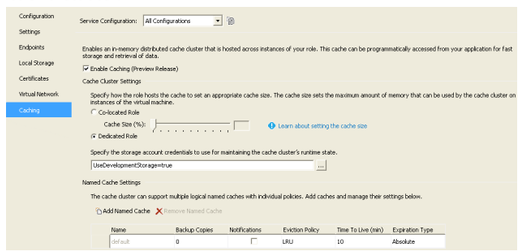
The following steps will configure the Dedicated Role cache cluster

1. Add Cache Worker Role project - To configure Dedicated Role cache cluster, you need to add a Cache Worker Role to your solution. Add a New Cache Worker Role project onto your Windows Azure project.

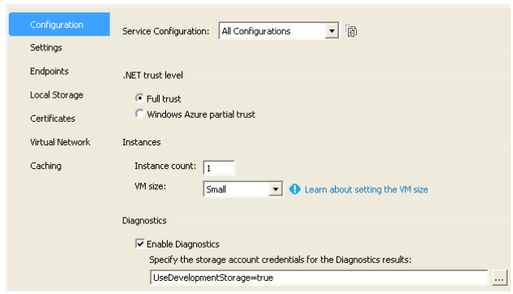




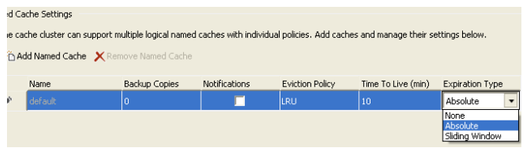
When we are adding a new Cache Worker Role project, it will automatically enable Caching with Dedicated Role as the Cache cluster.



2. Configure the settings for Dedicated Cache Role - Right-click on the Cache Worker Role from the Windows Azure project, and click properties and choose Configuration.



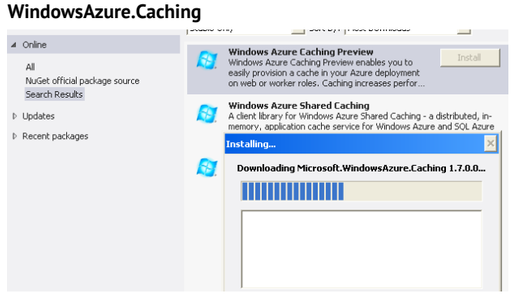
By default, the VM instance count would be 1 and Virtual Machine size would be small. You can change these setting based on your hosting plan for Dedicated Role Cache Cluster.



3. Configure the Cache settings for expiration time and expiration type.

WORKING WITH CACHE CLIENTS

In order to work with Windows Azure caching from Cloud Services applications, we need to install Windows Azure Caching client library onto your Web Role and Worker Role applications. The following steps will add a Windows Azure client library to a Web Role application. We can install the Windows Azure Caching client library by using NuGet. From the following NuGet packages, Install **Microsoft.WindowsAzure.Caching**



The Windows Azure Caching Preview NuGet package will install the necessary assembly reference and will also add the configuration setting on the configuration file. You can also manually add the necessary assmeblies from the folder C:\Program Files\Microsoft SDKs\Windows Azure\.NET SDK\2012-06\ref\CachingPreview if you have installed Windows Azure SDK 1.7 on your system.

The Windows Azure Caching NuGet package will add a configuration element dataCacheClients into the web.config file.

The following configuration setting in the web.config file will configure a cache client for a web role application.

```
<configSections>
 <sectionname="dataCacheClients"type="Microsoft.ApplicationServer.
Caching.DataCacheClientsSection, Microsoft.ApplicationServer.
Caching.Core" allowLocation="true" allowDefinition="Everywhere" />
 </configSections>
 <dataCacheClients>
  <tracing sinkType="DiagnosticSink" traceLevel="Error" />
  <dataCacheClient name="default">
   <autoDiscover isEnabled="true" identifier="EFMVC.
        CacheWorkerRole" />
   <!--<localCache isEnabled="true" sync="TimeoutBased"
        objectCount="100000" ttlValue="300" />-->
  </dataCacheClient>
 </dataCacheClients>
```

The *identifier* in the config settings specifies the name of the role that hosts the cache cluster

PROGRAMMING WITH WINDOWS AZURE CACHING

In order to work with Windows Azure Caching from client apps, you need to add the following namespace in your client app:

```
using Microsoft.ApplicationServer.Caching;
```

To programmatically access the Windows Azure Caching, you first need to create an instance of DataCache class.

The following code block will create a cache client by creating an instance of DataCache which will take the settings from configuration file.

```
//Creating a cache object
DataCacheFactory cacheFactory = new DataCacheFactory();
DataCache dataCache = cacheFactory.GetDefaultCache();
```

The above code block will create the cache client object by using DataCacheFactory object. The GetDefaultCache method of DataCacheFactory object will return the DataCache object.

Adding and Retrieving Data from the Windows Azure Cache

To add an item to Cache object, we can use either Add or Put method of DataCache object. The Add method adds an object to the Cache object with a specified key and will generate an exception if the item already exists with the same key. The put method adds the object to the Cache object with a specified key if the key does not exist, and replaces if the key is already exists.

The following code block adds the collection of data objects to Cache:

```
//Query from DB and adding data to Cache
var categories = categoryRepository.GetAll();
dataCache.Add("categories", categories);
```

In the above code block, we are querying data from Database and adding this data to Cache object with a key name "categories". We can take the data from Cache for the subsequent requests for the same data which can avoid database round trips and improve the performance of the application.

Using the overloads of the Add and Put methods of DataCache object, we can override the timeout interval, which is specified in the Cache Roles settings.

The following code block adds the data to the Cache with time span that allows to specify when the object should expired from Cache.

```
dataCache.Add("categories", categories, TimeSpan.FromMinutes(45));
```

The above code block adds the data to the Cache object with a key category and specifies the timeout as 45 minutes.

The following code block retrieves the data from Cache:

```
//Getting the Cache item with key "categories"
IEnumerable<Category> category = dataCache.Get("categories") as
IEnumerable<Category>;
```

The Get method of the DataCache object retrieves the data from Cache where you have to give the key as the parameter value for the Get method. The above code block retrieves the data and casts it to the appropriate data type.

Working with a Cache Provider Class

In the previous steps, we have discussed how to perform data caching with Windows Azure Caching. Let's create a Cache provider class that will help us better manage Data in the cache. Let's create a contract type for the Cache provider.

```
public interface ICacheProvider
{
    object Get(string key);
    void Add(string key, object data);
    void Put(string key, object data);
    void Add(string key, object data, TimeSpan? timeout);
    void Put(string key, object data, TimeSpan? timeout);
    bool IsSet(string key);
    void Remove(string key);
}
```

The Get method returns the cached item based on the given key and Add and Put methods adds and puts an item to Cache. The IsSet method will return true if an item is added on the Cache. The Remove method will remove an item from the Cache.

Let's create a concrete implementation for the ICacheProvider. The below AzureCacheProvider Class is an implementation of ICacheProvider for Windows Azure Caching.

```
public class  AzureCacheProvider : ICacheProvider
{
    private DataCache dataCache=null;
    public AzureCacheProvider()
    {
        //Creating a cache object by using DataCacheFactory
        DataCacheFactory cacheFactory = new DataCacheFactory();
        dataCache = cacheFactory.GetDefaultCache();
    }
    //Get cached item
    public object Get(string key)
    {
      return dataCache.Get(key);
    }
    public void Add(string key, object data)
    {
        Add(key, data, null);
    }
    public void Add(string key, object data, TimeSpan? timeout)
    {
      //Add item to Cache
      if (timeout.HasValue)
      {
          dataCache.Add(key, data, timeout.Value);
      }
      else
      {
          dataCache.Add(key, data);
      }
    }
    public void Put(string key, object data)
    {
        Put(key, data, null);
    }
    public void Put(string key, object data, TimeSpan? timeout)
    {
      //Put item to Cache
      if (timeout.HasValue)
      {
          dataCache.Put(key, data, timeout.Value);
      }
      else
      {
          dataCache.Put(key, data);
      }
    }
}
```

In the code block shown below, we are using the ICacheProvider from an ASP.NET MVC application

```
private readonly ICacheProvider cache;
public CategoryController(ICategoryRepository
categoryRepository,ICacheProvider cache)
{
    this.categoryRepository = categoryRepository;
    this.cache = cache;
}
 public ActionResult Index()
    {
        IEnumerable<Category> categories;
        var cachedCategories=cache.Get("categories");
        if (cachedCategories != null)
        {
            categories = cachedCategories as IEnumerable<Category>;
        }
        else
        {
            categories = categoryRepository.GetAll();
            cache.Put("categories", categories);
        }
        return View(categories);
    }
```

In the Index action method, we are taking the data from Cache, if the cached item exists. If the item does not exist, we retrieve the data from Database and put it in Cache so that we can take the data from the Cache, for the subsequent requests. In this sample code, we are injecting an instance of AzureCacheProvider class through constructor for the contract type ICacheProvider.

Using Windows Azure Caching for ASP.NET Session State and ASP.NET Page Output Caching

In addition to the data caching, we can also use Windows Azure Caching as the storage mechanism for ASP.NET Session State and ASP.NET Page Output Caching. The Session State Provider for Windows Azure Caching and the Output Cache Provider for Windows Azure Caching are the out-of-process storage for ASP.NET applications that works great with Windows Azure load balancers. To use Windows Azure Caching as the storage mechanism for ASP.NET Session State and ASP.NET Page Output Caching, first you need to configure Cache Cluster (using one of the two approaches explained above) and then configure your ASP.NET application to use the Windows Azure Caching. After you have configured the Cache Cluster, you can use Windows Azure Caching for ASP.NET Session State and ASP.NET Page Output Caching by adding necessary configuration setting in the web.config file.

The following configuration setting in the web.config allows to use the Session State Provider for Windows Azure Caching for web applications.

```
<sessionState  mode="Custom"  customProvider="AppFabricCacheSessionStoreProvider">
  <providers>
    <add name="AppFabricCacheSessionStoreProvider"
      type="Microsoft.Web.DistributedCache.DistributedCacheSessionStateStoreProvider,  Microsoft.Web.DistributedCache" cacheName="default" useBlobMode="true" dataCacheClientName="default" />
  </providers>
</sessionState>
```

The cacheName specifies the name of the Cache used for persisting Session State data. The following configuration setting in the web.config configures the Session State Provider for Windows Azure Caching for web applications.

```
<caching>
  <outputCache defaultProvider="DistributedCache">
    <providers>
      <add name="DistributedCache"
          type="Microsoft.Web.DistributedCache.DistributedCacheOutputCacheProvider,  Microsoft.Web.DistributedCache"  cacheName="default"
          dataCacheClientName="default" />
    </providers>
  </outputCache>
</caching>
```

SUMMARY

The Windows Azure Caching (Preview) introduced in the Windows Azure June Release - SDK 1.7 is a high performance distributed, in-memory caching solution. Using Windows Azure Caching (Preview), you can host your Cache in the Virtual Machine of your Web Role and Worker Role. You can deploy the Cache in a Co-located Role which allows you use some portions of the VMs memory as Cache. You can also deploy to Dedicated Role where you can use a VM specifically dedicated for Caching. Windows Azure Caching comes with ASP.NET providers for Session state and Page Output Caching and can thus be used as storage mechanism for these. Thus Windows Azure Caching enables you to build high performance cloud apps on the Windows Azure platform. ■

Shiju Varghese is a Microsoft MVP and a Technical Architect specializing in Windows Azure and Web technologies. His current technology focus is on Windows Azure, ASP.NET MVC, Node.js, REST, CQRS and Event Sourcing. Shiju blogs at http://weblogs.asp.net/shijuvarghese and can follow him on twitter @shijucv

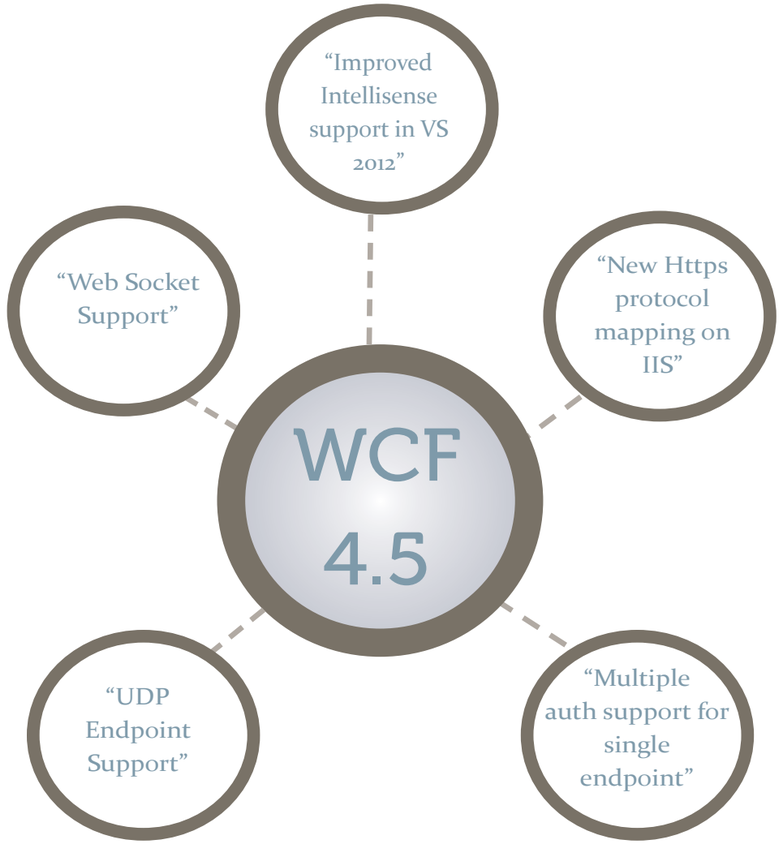
WHAT'S NEW IN WCF 4.5

DOWNLOAD FILES >

bit.ly/dncmag-wcf45

Mahesh Sabnis highlights the new features of WCF 4.5 with significant improvements with respect to setup, configuration and protocol support.

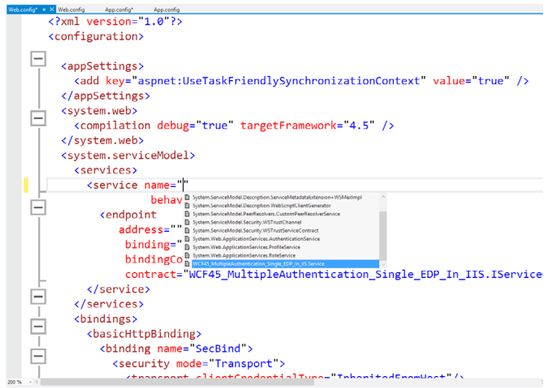
WCF is a popular technology for developing distributed applications. Over the past two releases, new features have been added that increased the flexibility of the framework. For example in WCF 3.5, REST was introduced and WCF 4.0 introduced features like Routing, Discovery, Help page with REST etc. With the latest release of WCF in .NET 4.5, several new features have been added. We will explore these new features in this article.



IMPROVED INTELLISENSE SUPPORT IN VS 2012

Intellisense in Configuration

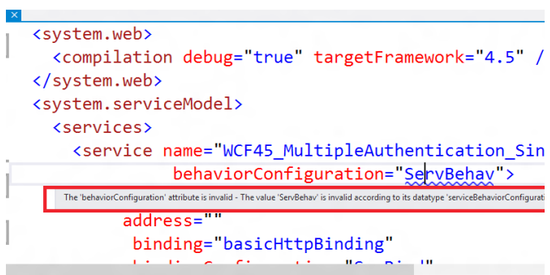
As we know, Intellisense support for code is a major productivity boost when developing in Visual Studio. However, while creating configuration files, Intellisense support has been 'weak' to say the least. Many a times, we make mistakes in writing Service name, Binding name, Contract name, Behavior name etc. resulting in runtime errors. With WCF 4.5, Visual Studio 2012 provides significantly improved Intellisense support in configuration files too. For example, as we can see here, we have auto completion of Service Names.



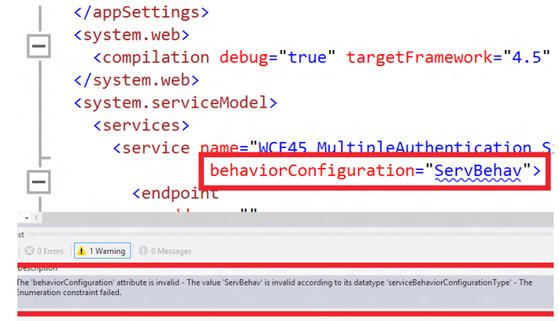
This type of Intellisense allows you to easily create WCF configuration files.

Improved Validations with Configuration

Validation of the configuration file is another important feature in VS 2012. This reduces possible validation mistakes. Any mistakes in configuration, will be displayed as shown below:



Also the configuration errors come up in the Tasks window as warnings:



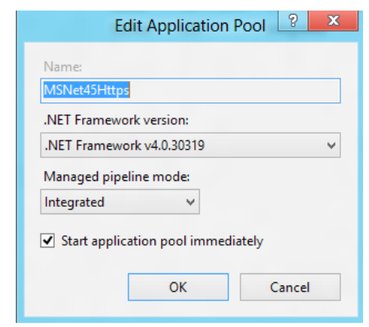
The validation message of the configuration file will be shown as above.

These subtle improvements in VS 2012 improve developer productivity and take away some of the pains associated with WCF configuration.

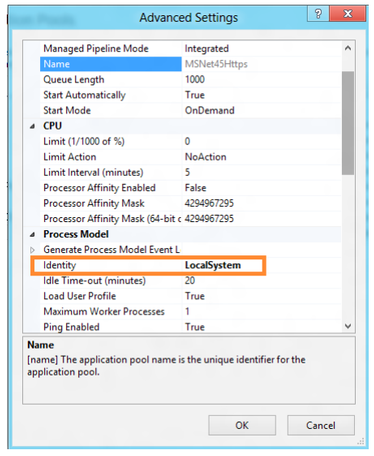
NEW HTTPS PROTOCOL MAPPING ON IIS

Transport security is an important factor for a WCF service. In WCF 4.0, we were provided with default endpoint feature which defined protocol mapping for 'basicHttpBinding'. In WCF 4.5, if IIS is enabled for SSL and if the service does not have any explicit endpoint defined for a specific binding, then WCF can be hosted on IIS with default HTTPS enabled i.e. 'basicHttpsBinding'.

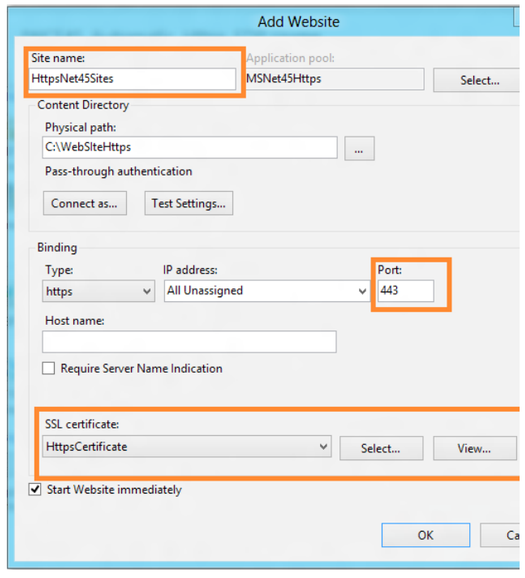
To experience the automatic HTTPS protocol mapping on IIS, you need to create an Application Pool with Identity as 'LocalSystem'. This must be targeted to .NET 4.0 framework as shown below:



The Identity property is set to the LocalSystem as shown below:



Now we need to create a Web Site under the application pool which will contain the WCF 4.5 service published in it as shown here:



The web site must be provided with the SSL certificate which you need to create. The port is set to 443.

Now Open Visual Studio 2012 and create a new WCF Service application, and set a Service Name. When you browse the service.svc, you will get the 'basicHttpBinding' endpoint in WSDL as shown here:

```
- <wsdl:service name="Service1">
  - <wsdl:port name="BasicHttpBinding_IService1" binding="tns:BasicHttpBinding_IService1">
      <soap:address location="http://localhost:4249/Service1.svc"/>
    </wsdl:port>
</wsdl:service>
```

Now, publish the WCF service on IIS in the site created above and browse the service.svc. You will find the endpoints:

```
- <wsdl:service name="Service1">
  - <wsdl:port name="BasicHttpBinding_IService1" binding="tns:BasicHttpBinding_IService1">
      <soap:address location="http://mahesh-pc:8070/WCF45_Automatic_Https_EDP/Service1.svc"/>
    </wsdl:port>
  - <wsdl:port name="BasicHttpsBinding_IService1" binding="tns:BasicHttpsBinding_IService1">
      <soap:address location="https://localhost/WCF45_Automatic_Https_EDP/Service1.svc"/>
    </wsdl:port>
</wsdl:service>
```

This new feature reduces the amount of configuration required. Although we had SSL support in previous versions of WCF, but it was necessary to declare an explicit endpoint. Now in WCF 4.5, using the default endpoint, we can now have smaller configuration codes.

MULTIPLE AUTHENTICATION SUPPORT FOR SINGLE ENDPOINT HOSTED ON IIS

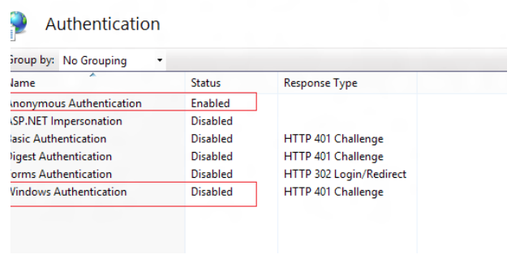
Technically, when a WCF service is used by multiple client applications over a large network, the recommended host is IIS 7.x with Windows Process Activation Service (WAS). This enables multi-protocol support for WCF services and various client applications can make calls to a WCF service over these protocols.

To setup authentication on the WCF service, the authentication values need to be applied in the Web.config file of the Service application and also on IIS. If there are any authentication value mismatch between the Endpoint authentication and the IIS authentication, then authentication mismatch error will occur.

Consider, a scenario where, the WCF service is configured to use 'basicHttpBinding' endpoint with Security mode as 'Transport' and ClientCredentialType as 'Windows'

```
    <endpoint
      address=""
      binding="basicHttpBinding"
      bindingConfiguration="SecBind"
      contract="WCF45_MultipleAuthentication_Single_EDP_In_IIS.IServic
  </service>
</services>
<bindings>
  <basicHttpBinding>
    <binding name="SecBind">
      <security mode="Transport">
        <transport clientCredentialType="Windows"/>
      </security>
    </binding>
  </basicHttpBinding>
</bindings>
```

The WCF service is published on IIS with authentication set similar to the one shown below:



When we try to access the service, we will get the following error:


Server Error in '/WCF45_Multiple_Auth' Application.

The authentication schemes configured on the host ('Anonymous') do not allow those configured on the binding 'BasicHttpBinding' ('Negotiate'). Please ensure that the SecurityMode is set to Transport or TransportCredentialOnly. Additionally, this may be resolved by changing the authentication schemes for this application through the IIS management tool, through the ServiceHost.Authentication.AuthenticationSchemes property, in the application configuration file at the <serviceAuthenticationManager> element, by updating the ClientCredentialType property on the binding, or by adjusting the AuthenticationScheme property on the HttpTransportBindingElement.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

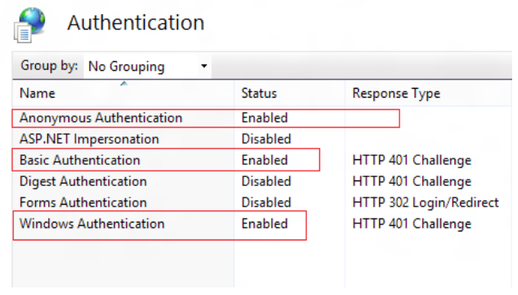
Exception Details: System.NotSupportedException: The authentication schemes configured on the host ('Anonymous') do not allow those configured on the binding 'BasicHttpBinding' ('Negotiate'). Please ensure that the SecurityMode is set to Transport or TransportCredentialOnly. Additionally, this may be resolved by changing the authentication schemes for this application through the IIS management tool, through the ServiceHost.Authentication.AuthenticationSchemes property, in the application configuration file at the <serviceAuthenticationManager> element, by updating the ClientCredentialType property on the binding, or by adjusting the AuthenticationScheme property on the HttpTransportBindingElement.

This happens because, the WCF service endpoint is expecting Windows Authentication and the IIS is enabled for anonymous authentication only. To support multiple clients for making call to WCF service with different authentication, developer has to define multiple endpoints for the WCF service.

The solution here is that WCF needs to be configured to inherit the authentication types provided by IIS. This can be achieved using the ClientCredentialType value on the transport security set to 'InheritedFromHost'. Now the configuration of the endpoint will be as below:

```
bindings>
 <basicHttpBinding>
   <binding name="SecBind">
     <security mode="Transport">
       <transport clientCredentialType="InheritedFromHost"
     </security>
   </binding>
 </basicHttpBinding>
/bindings>
```

The 'InheritedFromHost' value for the ClientCredentialType is newly introduced in WCF 4.5. This configures the single endpoint WCF service hosted on IIS to use the authentication types defined in IIS. Now the IIS authentication can be set as follows:



If you now browse to the service.svc, you will be able to get the WSDL. (Note: The endpoint uses Transport security, so for publishing on IIS 7.x, you need to have Web site enabled for SSL). The client application can have various ClientCredentialType values e.g.

```
:em.serviceModel>
:bindings>
  <basicHttpBinding>
    <binding name="BasicHttpBinding_IService">
      <security mode="Transport">
        <transport clientCredentialType="None" />
      </security>
    </binding>
  </basicHttpBinding>
:/bindings>
:client>
    <endpoint address="https://localhost/WCF45_Multiple_Auth/Se
      binding="basicHttpBinding" bindingConfiguration="BasicH
      contract="MyRef.IService" name="BasicHttpBinding_IServi
:/client>
:tem.serviceModel>
```

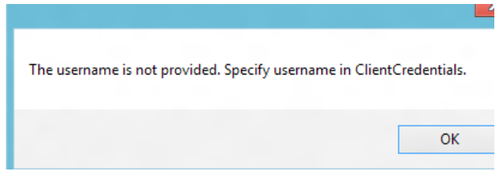
Here the client applications need not pass any credential information to WCF service, because IIS has authentication type Anonymous, as enabled. The response from the WCF service, will be successfully delivered to client If the value of the ClientCredentialType is set to Windows as below:

```
ngs>
asicHttpBinding>
  <binding name="BasicHttpBinding_IService">
    <security mode="Transport">
      <transport clientCredentialType="Windows" />
    </security>
  </binding>
basicHttpBinding>
ings>
```

In this case also, the response will be successfully delivered to client, because the Service on IIS has set the Windows Authentication as Enabled. If the client application set the ClientCredentialType to 'Basic' as below:

```
<bindings>
    <basicHttpBinding>
        <binding name="BasicHttpBinding_IService">
            <security mode="Transport">
                <transport clientCredentialType="Basic" />
            </security>
        </binding>
    </basicHttpBinding>
</bindings>
```

This mandates that the client application should send the credentials to WCF service for successful communication otherwise the following exception will be displayed:



So to ensure successful communication between client and the WCF service, the credentials are passed as shown below:

```
Proxy = new MyRef.ServiceClient();
Proxy.ClientCredentials.UserName.UserName = "Admin";
Proxy.ClientCredentials.UserName.Password = "Password123";
```

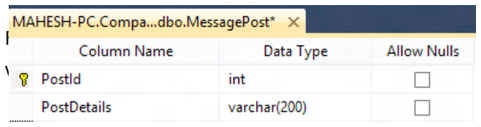
The advantage of this feature is that, for single endpoint WCF service hosted on IIS with multiple authentication type set, multiple client applications with different credentials can communicate with WCF service successfully and hence developer is free from providing several endpoints making WCF service less complex and more manageable.

SUPPORT FOR UDP ENDPOINT

One of the most attractive and beneficial features of WCF 4.5 is the out-of-box support for UDP protocol. We have been provided with 'udpBinding'. In some cases the UDP is proven far better than TCP and HTTPS which are the most popular protocols. Typically TCP is used in the non-time critical (may be non-real-time) applications, but UDP is very useful for real-time applications e.g. Games or the applications which requires fast data communication. Since TCP ensures data-packets are sent and received in order, it is slower that UDP. In UDP, packet delivery is not guaranteed and loss or out-of-arrival of packets is acceptable. Thus it is faster that TCP.

Using TCP vs. UDP

Let us setup a simple comparison between the TCP and UDP binding for database communication.


MAHESH-PC.Compa...dbo.MessagePost*

Column Name	Data Type	Allow Nulls
PostId	int	☐
PostDetails	varchar(200)	☐

Step 1: Open VS2012 and create a blank solution, name it as 'WCF45_UDP_Comparision'. IN this solution add a new WCF Service application; name it as 'WCF45_UPD_Binding_Service'. Rename, IService1.cs to IService.cs and Service1.svc to Service.svc. Add the following methods in the IService interface:

```
using System.Runtime.Serialization;
using System.ServiceModel;

namespace WCF45_UPD_Binding_Service
{
    [ServiceContract]
    public interface IService
    {
        [OperationContract(IsOneWay=true)]
        void PostMessages(MessagePost message);
    }

    [DataContract]
    public class MessagePost
    {
        [DataMember]
        public string MessageDetails { get; set; }
    }
}
```

Step 2: Implement the IService interface in the service class as below:

```
using System.Data.SqlClient;

namespace WCF45_UPD_Binding_Service
{
    public class Service : IService
    {
        SqlConnection Conn;
        SqlCommand Cmd;
        public Service()
        {
            Conn = new SqlConnection("Data Source=.;Initial
            Catalog=Company;Integrated Security=SSPI");
        }

        public void PostMessages(MessagePost message)
        {
            Conn.Open();
            Cmd = new SqlCommand();
            Cmd.Connection = Conn;
            Cmd.CommandText = "Insert into MessagePost Values(@
            PostDetails)";
            Cmd.Parameters.AddWithValue("@PostDetails",message.
            MessageDetails);
            Cmd.ExecuteNonQuery();
            Conn.Close();
        }
    }
}
```

The fact about the UDP binding is that it is not supported by IIS/WAS because there is no UDP shared listener available on this host, so we need to host it via a managed Exe (.NET application). Also one important thing to note is the service contract defines one-way message i.e. Fire and forget, which is the most suitable mechanism for UDP based Communication. Other important facts about the UDP Binding are:

- The binding is supported only for .NET client applications. SOAP-based service messaging is not supported.
- No support for transport or message security is available.
- Only text encoding is supported.

Step 3: In the solution created, add a new Console Application and add the App.Config file with the following configuration:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

  <appSettings>
    <add key="aspnet:UseTaskFriendlySynchronizationContext"
     value="true" />
  </appSettings>
  <system.web>
    <compilation debug="true" />
  </system.web>
  <system.serviceModel>
    <services>
      <service name="WCF45_UPD_Binding_Service.Service">
        <endpoint address="http://localhost:3401/MyServHttp"
                  binding="basicHttpBinding"
                  contract="WCF45_UPD_Binding_Service.IService">
        </endpoint>

        <endpoint address="net.tcp://127.0.0.1:3402/MyServTcp"
                  binding="netTcpBinding"
                  contract="WCF45_UPD_Binding_Service.IService">
        </endpoint>

        <endpoint address="soap.udp://localhost:3403/MyServUdp"
                  binding="udpBinding"
                  contract="WCF45_UPD_Binding_Service.IService">
        </endpoint>

      </service>
    </services>
    <behaviors>
      <serviceBehaviors>
        <behavior>
          <serviceMetadata httpGetEnabled="true" httpGetUrl=
                "http://localhost:3400/MyServ"/>
          <serviceDebug includeExceptionDetailInFaults="False" />
        </behavior>
      </serviceBehaviors>
    </behaviors>
  </system.serviceModel>

</configuration>
```

Note that in the above configuration, 3 endpoints HTTP, Tcp and UDP are defined for comparison. Note that the address schema defined for the UDP endpoint is "soap.udp".

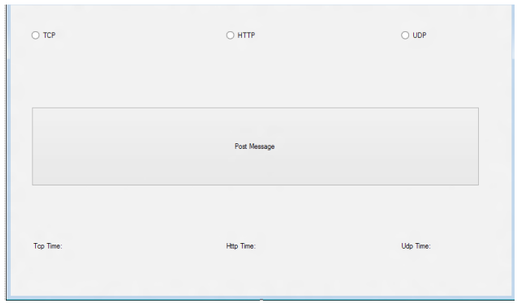
Following is the code for hosting it in the Program class:

```
class Program
{
    static void Main(string[] args)
```

```
    {
        ServiceHost Host = new ServiceHost(typeof(WCF45_
                                UPD_Binding_Service.Service));
        Host.Open();
        Console.WriteLine("Service Started...");
        Console.ReadLine();
        Host.Close();
        Console.ReadLine();

    }
}
```

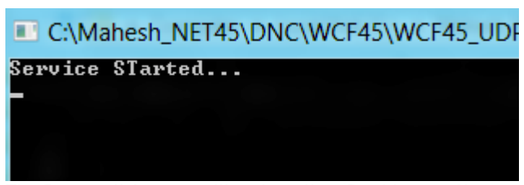
Step 4: Add the service reference of the above WCF service in Windows application by adding Windows application in the solution created above, name it as 'WCF45_UDP_Comparision_Client'. The reference will be added using the following URI 'http://localhost:3400/MyServ'. In the App.Config file of the client application, you will get 3 endpoints, one each for Tcp, Udp and HTTP.

Step 5: Design the WinForm as shown below:



Step 6: Write the following code in the Form Code behind:

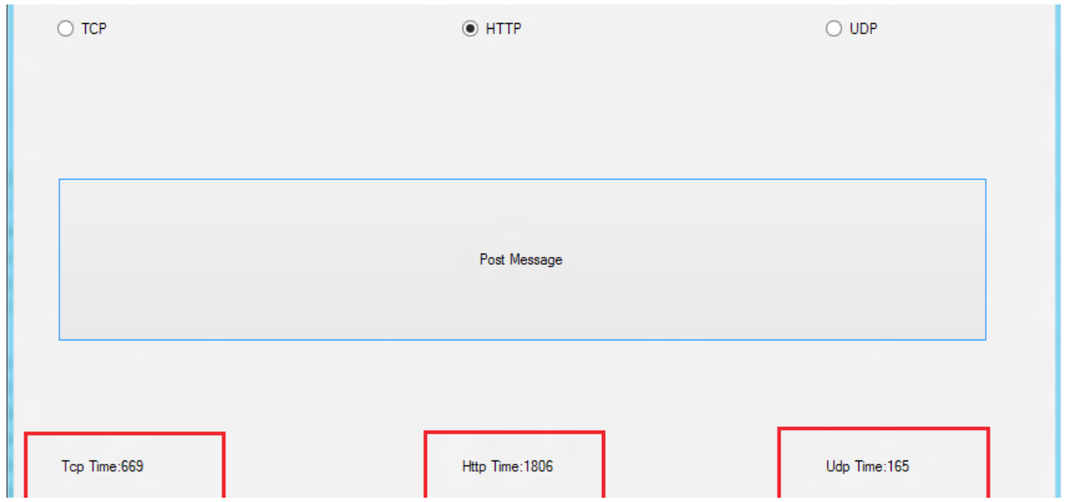
```
if (Protocol == string.Empty)
{
MessageBox.Show("Please select the Protocol");
}
else
{
switch (Protocol)
{
case "Tcp":
        Proxy = new
        MyRef.ServiceClient("NetTcpBinding_IService");
        break;
case "Http":
Proxy = new
MyRef.ServiceClient("BasicHttpBinding_IService");
break;
case "Udp":
Proxy = new
MyRef.ServiceClient("UdpBinding_IService");
break;
}
}


Stopwatch stopWatch = new Stopwatch();
stopWatch.Start();
for (int i = 0; i < 2000; i++)
{
        MyRef.MessagePost msg = new MyRef.MessagePost()
        {
                MessageDetails = "My Message Post No" +
                i.ToString() + "with Protocol " + Protocol
        };
        Proxy.PostMessages(msg);
}
stopWatch.Stop();
}
stopWatch.Stop();
if (Protocol == "Tcp")
{
        lblTcpTime.Text += stopWatch.ElapsedMilliseconds;
}
if (Protocol == "Http")
{
        lblHttpTime.Text += stopWatch.ElapsedMilliseconds;
}
if (Protocol == "Udp")
{
        lblUdpTime.Text += stopWatch.ElapsedMilliseconds;
}
```



The Button click event will make call to 'PostMessage()' method of the WCF service based upon the Protocol selected from the Radio Button on the UI.

Step 7: To test the comparison, run the Host Application and run the Client application in multiple instances:
Run the client application, and select various Protocols e.g. UDP, TCP and HTTP and click on the 'Post Message' button, the result is shown below:

It is clearly shown that for 2000 Database Insert transactions, UDP takes only 165 Milliseconds whereas TCP takes 669 and HTTP takes 1806 milliseconds. Hence it is proven that UDP is better performing as compared to other protocols.



One more feature worth mentioning is support for WebSockets using NetHttpBinding.

NETHTTPBINDING AND NETHTTPSBINDING

This is provided for WebSocket programming. WebSocket is the new protocol for bidirectional communication over port 80 and 443. This protocol has provided enhanced performance like TCP protocol. Typically we can make use of this for Duplex communication with WCF 4.5.

CONCLUSION

WCF has been considered more verbose with respect to configuration and setup when compared to other Remote communication mechanism like the new WebAPI.

However the latest WCF release makes significant improvements with respect to setup, configuration and protocol support making it easier to develop and manage WCF applications. ■

Mahesh is a Microsoft Certified Trainer (MCT) since 2005 and has conducted various Corporate Training programs for .NET Technologies (all versions). Follow him on twitter @maheshdotnet. Mahesh blogs regularly on Azure, SharePoint, Metro UI, MVC and other .NET Technologies at http://bit.ly/HsS2on

HOW TESTING IMPROVES QUALITY OF SOFTWARE DEVELOPMENT

• ***Gouri Sohoni*** *reviews the overall process of software testing and how Microsoft Tools can help achieve it.*

We often hear people say – "Testing increases time for a software development project". It is true that testing requires time, but it also improves quality of software. Testing helps in detecting software defects before it gets shipped to customers (or in cases like the NASA's Mars mission, before they blast off into space). Apart from detecting bugs, performance testing helps us determine system boundaries with respect to load handling and responsiveness of the system.

With testing, we can fix bugs before they can create a cascading effect in production (Like crashing a probe into a far off planet because different teams were using different units of measure). When we identify and fix bug in such a manner, it leads to less time to complete the development process; complete being the keyword here.

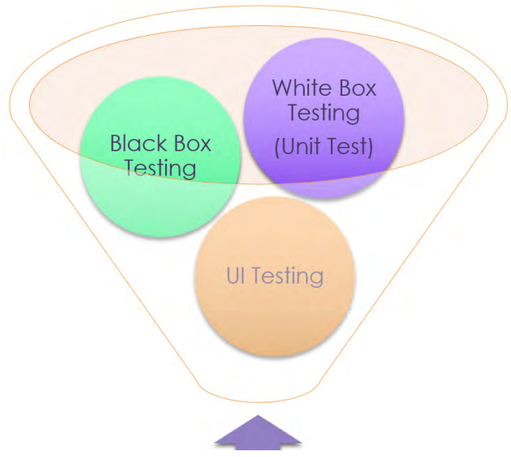
If we omit testing from the software development life cycle, we may be able to deliver the software faster, but the quality of the software will always be at stake.

Test implementations usually depend on the software development process. Often testing is implemented after the code is written (with typical waterfall model). In Agile, it is implemented as an on-going activity. The final goal however remains same - Ensure Software meets the requirements of the customer.

APPROACHES TO SOFTWARE TESTING

There are various approaches to testing, like White Box Testing (Unit Test), Black box testing, UI Testing and many more.

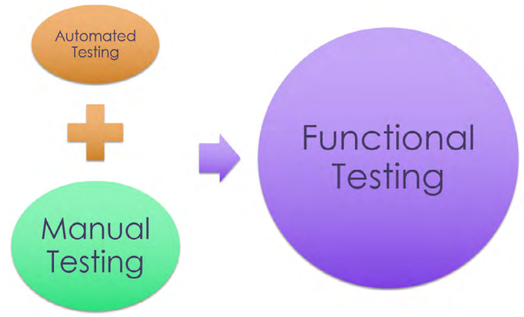
With white box testing, we test the code, internals of the program & performance of the application can also be tested. With black box testing, we do not check any code or internals. We provide input and check if the expected output is received. We do not measure any performance here.



FUNCTIONAL TESTING

Functional testing can be bifurcated into two major types:

- Manual Testing and
- Automated Testing



MANUAL TESTING

Let us take a first look at manual testing. World over, majority of the tests are conducted manually. Testing usually begins by the action of creation of test case. People usually use MS Excel to create and store the test cases. Although MS Excel is an excellent software, it falls short of being a good tool to document the testing process. While running a test stored in an Excel worksheet, the tester has to shuffle between test case, input data and the application under test. That reduces the tester's efficiency.

Microsoft Test Runner which is a part of Microsoft Test Manager 2012 lets testers create, store and view the steps of the test case while the test is being run. It means that the tester can view the test steps, the data to be used in testing and execute the application through the same UI. This increases the tester's efficiency of running the test.

It may so happen that the tester needs to execute the same test again. Time required to re-execute the manual test is usually the same as first time execution. Microsoft Test Manager provides a fast-forward playback functionality. Tester can record the actions of execution for the first time. Thus, when a test case fails, the test engineer already has a recording of the previous steps. With fast forward play back, tester can run those steps till the point of previous failure. Thus the time required for next round of test execution is now considerably reduced. Tester can then continue manually and record the next steps till the next breaking scenario is encountered. Thus overall testing activity is completed in less time.

TEST RESULT LOGGING AND BUG FILING

Filing a bug is one of the major time consuming activities during testing. A bug or a defect should be accompanied with the supporting data that can be used by the developer to fix it. Entering this data requires a lot of time. If a tool can collect and associate that data to the bug, then the time taken by the tester to enter this data will be saved. With Microsoft Test Manager creating a bug is a very easy process. It decreases the time required by a tester to file a 'Rich Bug' drastically.

After a bug is filed, the major task is fixing it. Many a times ping-pong of a bug occurs when tester says there is a bug and developer cannot reproduce it. Microsoft Test Manager helps to create what is referred to as a Rich Bug. MTM helps add additional meta-information to a bug, as shown in the above diagram. The automatic capture of this additional information helps tester file a detailed bug. For the developer, a Rich bug often implies it is easier to reproduce and thus fix the bug.

AUTOMATED TESTING

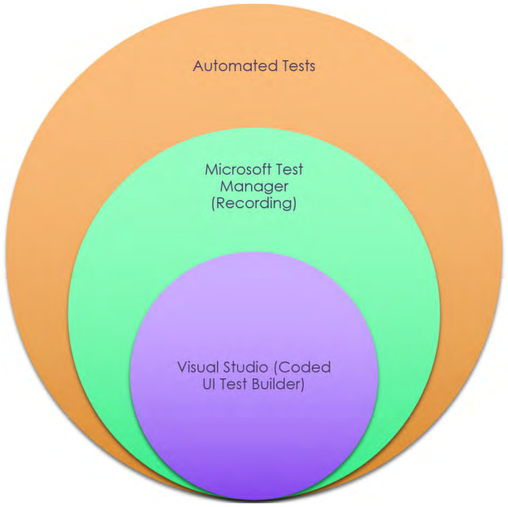
Most large software projects now-a-days adopt an incremental and iterative development process. This requires that a Test suite be run before every release iteration to ensure regression issues don't occur. In such cases, instead of running the tests for existing features manually, it is preferred that the features which were untouched in the current release be tested automatically. Automating a test requires significant time and technical skills, hence it is not a fit for testing functionality that's under active development sprints. Tools that allows you to create automated tests usually use the record and playback paradigm. Record the software execution with one set of data and then play it back with different sets of data that needs to be tested. You will also want to specify certain validations and conditions when the test passes or fails. Microsoft provides a feature called Coded UI Test (CUIT) to achieve all these. CUIT can convert manual test to its code so that time required to record the test is zero. It also allows Coded UI Test Builder actions to be recorded and converted. Such a test can participate as a build verification test also.

TEST LAB MANAGEMENT

We often need to test an application on different platforms and developers come back with the response to bugs saying 'it works on my box'. This is true mostly for hard-to-reproduce bugs. This 'works on my box' situation arises due to difference in the environment in which tester works and the environment on which the developer works.

As a tester, it is imperative that a bug be reproducible. As a developer it is important to review the exact problem encountered by the tester. We cannot keep multiple machines locked till a developer has a look at the problem. Microsoft provides a wonderful way of creating and running tests in different environments. This feature is called Lab Management (introduced with Visual Studio 2010). With certain hardware and software specific pre-requisites, we can create different environments for testing. The tester tests with one environment which can comprise of different virtual machines playing their respective roles. If the testing goes as expected, the machines can go back to the library. Otherwise the tester can take a snapshot of the environment which will keep the exact state of all the machines when a problem was encountered. This snapshot can be made available to developer who can then easily reproduce the bug in the very same environment. This technique takes the 'works on my box' scenario out of the picture thus reducing the time required for fixing a hard-to-reproduce bug.

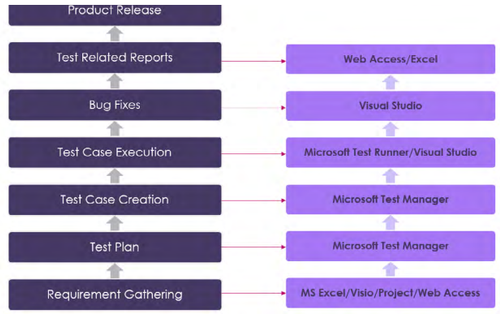
Lab Management can also be used for executing automated tests with a particular build. The build can be deployed on a particular machine in the environment by creating a build definition.



TEST PLANNING AND MANAGEMENT

Test resource planning is also a time and cost consuming activity during testing. First thing which needs to be done here is to create a Test Plan. Test plan is a collection of all the test cases that need to be executed, grouped by certain common properties. Those properties can be the schedule dates, platform on which the application is to be tested or the build number that is going to be tested. A project may have multiple test plans, each with a different set of identified properties and each plan may contain different set of test cases. A plan can match the schedule of iterations or may have its own schedule. The test plan can have different test suites or sets as named collections of test cases. These collections are created to identify the test cases adhering to other common rules. One test suite can be for all test cases that check the fulfillment of a certain requirement. Similarly some other test suite may be for all test cases that test a specific feature of an application. Team Foundation Server 2012 facilitates planning, recording and monitoring of the test plans.

Thus Microsoft's tools help us in each step to plan, manage, execute and iterate tests better. The following diagram roughly translates the tool set used in the entire testing lifecycle



CONCLUSION

To conclude, we reviewed the overall process of software testing. The intention was to highlight that software testing does not reduce, rather increases the productivity of a team developing non-trivial software. Everyone agrees to the fact that testing helps in improving the quality of the application. We also saw how Microsoft tools for testing, help maintain quality, reduce the time required for testing, bug fixing and improve the overall quality software and productivity of development. ■

Gouri Sohoni, is a Visual Studio ALM MVP and a Microsoft Certified Trainer since 2005. Check out her articles on TFS and VS ALM at http://www.dotnetcurry.com/Author.aspx?AuthorName=Gouri Sohoni

HADOOP ON AZURE

HIVE JOURNEY FOR THE SQL JOURNEYMAN

Govind Kanshi gives a bird's eye view of Hadoop and then dives into 'Hive' - A Data Warehousing platform.

In recent times, with the proliferation of social media, a term that we often hear with respect to large scale data storage and analysis is 'Big Data'.

Along with 'Big Data' almost synonymously comes the most predominant Big Data platform - Hadoop. In this article, we will take a bird's eye view of Hadoop and then dive into 'Hive' - A Data Warehousing platform (built by engineers at Facebook) on top of the Hadoop software ecosystem. We look at Hive from the point of view of a person who is exposed to SQL as a dialect for DDL, DML operations. We use the HadoopOnAzure platform as we explore Hive's features and limitations.

This article does not cover installation of Hadoop distribution or provisioning of cluster or for that matter MapReduce centric introduction to Hadoop. For introduction and detailed training refer to wiki.hadoop.org or at http://bit.ly/hd-tcnet.

TARGET AUDIENCE

This article targets regular .net developers and casual DBAs who want to understand what Hive is and where and how it differs from a regular database.

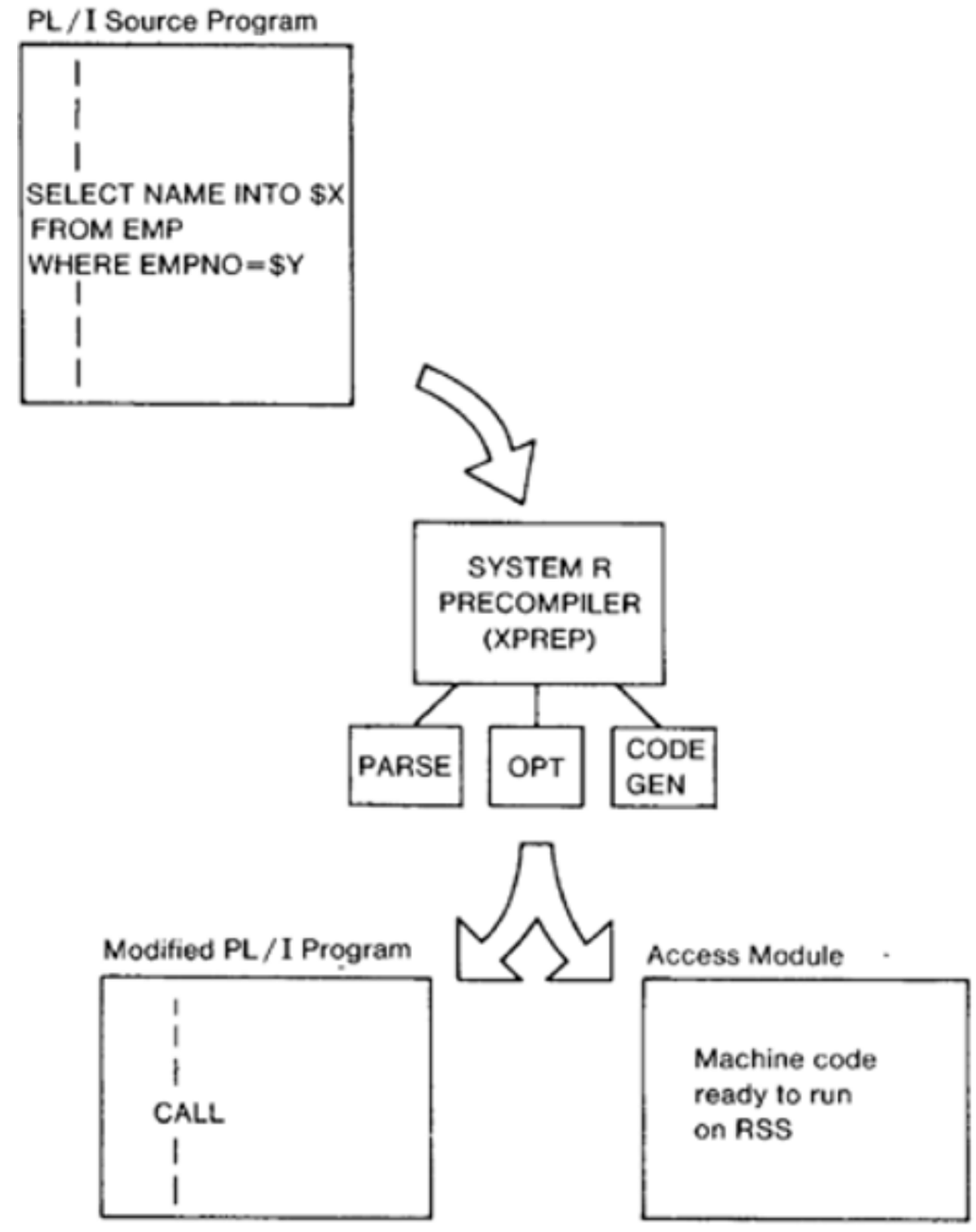
WHAT IS HIVE?

To paraphrase from the excellent whitepaper by Ashish, Joydeep et al. -

Hive (is), an open-source data warehousing solution built on top of Hadoop.

For a person who is familiar with RDBMS and High Performance Data Warehousing solutions – Hive scales compute and storage via Hadoop (specifically Hadoop Distribute File System HDFS) and provides availability via HDFS replication. It provides a server which can listen for HiveQL queries and translate them into execution plan using Metastore information about schema. The query planner/optimizer also refers Metastore to optimize and create the map-reduce jobs. These map-reduce jobs are scheduled by Hadoop on the underlying file system to get results. Thus Hive brings to us a System for effectively querying Petabytes of data on Hadoop via HiveQL, a high level SQL like querying language. HiveQL brings Big Data closer to analysts by removing the low-level map-reduce programming requirement for analyzing data on Hadoop. Having said that, let's take a peek under the hood of traditional DBMS and then compare it to the Hadoop Ecosystem.

5 MINUTE RECAP OF DATABASE EVOLUTION



Right from system R where we all have origins - we have storage subsystem and query execution subsystem differentiated in layered approach. Idea of DBMS was to provide increased level of abstraction for business professionals so that they could query the storage for tuples of their interest (either in aggregate or other mode). SQL, as that dialect for abstraction for query, makes life very easy for us.

Storage system abstracted the physical layout to help scale (partition) or access (index). It also imposed a persistence mechanism more suited for access of b-tree variants. It also created a viable model to take advantage of main memory in term of buffer pool to store data pages, query plans, locks etc. Since its focus was on consistency and isolation it also included a model for transaction (gray etc.) which implied usage of locks. Over period of time evaluation of right query plan pushed usage of various ways to increase the efficiencies by using statistics (wide, columnar etc.) of columns and advent of cost based optimizers over rule/heuristics based. (Evolution over gamma/exodus, starburst, volcano, cascades).

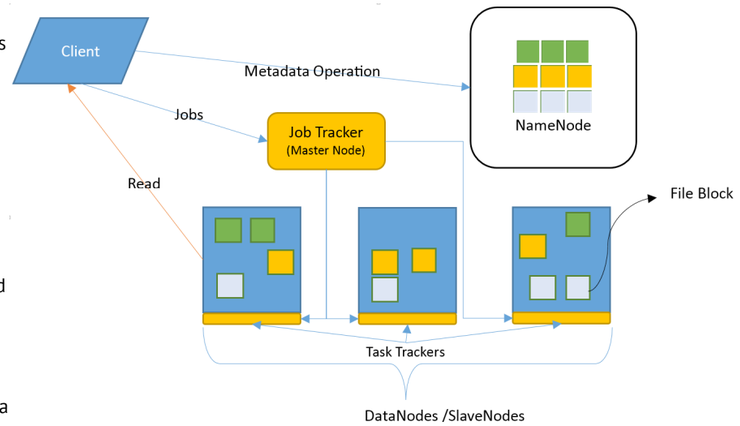
In simple words - when we execute a query we do not care where the tables are stored, what is the persistence of the data and how the data is actually retrieved. This is of immense help for end user. SQL is the most successful DSL of our times. The differentiation of OLTP vs. OLAP has become more apparent now that we have massive tomes of older data to "report" on and take care of. The latter world also implies lot of pre-generated aggregates, massive data, and slice/dice from any attribute.

In RDBMS world - scale is first done vertically in terms of resources (CPU/Memory/IO) if required, efficiency can also be sought in terms of data partitioning and scale. In shared nothing databases - best way to scale is balance out reads or functional sharding. In a shared database, some kind of a synchronization mechanism needs to be in place to keep all nodes in sync to allow horizontal scale. In general this approach has challenges in case of very large data for storage/query.

30K FEET OVERVIEW OF HADOOP

Hadoop provides a software framework that supports distributed data storage via HDFS and distributed computing via the Map-Reduce abstraction. It comes built in with a job scheduler to initiate and monitor the map-reduce jobs. Hadoop also has data replication built in allowing it to recover data and jobs in case of hardware failure.

Hadoop architecture comprises of two important Hardware and Service combinations. First, a Namenode that provides the abstraction of a File System. Second, the Datanode where data blocks are physically stored. The system scales horizontally by adding more Datanodes. The Job Tracker service allows scheduling of the jobs that are executed via the Task Trackers. Task Trackers are the job executioners and reside on the same hardware as the Datanodes. As a result jobs are executed as close to the data as possible



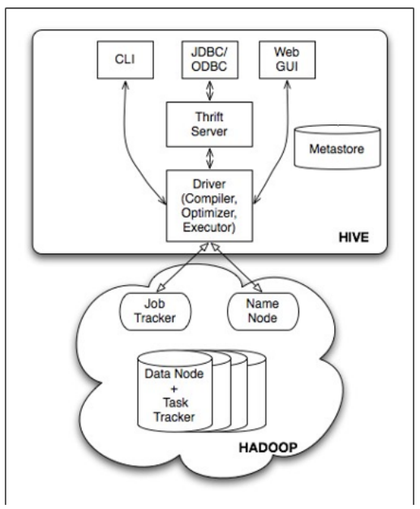
To mine information stored in HDFS system an analyst has to use the Map-Reduce framework. Using Map-Reduce typically involves writing a series of jobs written in Java or other programming languages. The input to a job is a programmed specification that will yield key-value pairs. Each job consists of two stages: first, a user-defined map function is applied to each input record to produce a list of intermediate key-value pairs. Second, a user-defined reduce function is called once for each distinct key in the map output and passed the list of intermediate values associated with that key. The Map Reduce framework automatically parallelizes the execution of these functions and ensures fault tolerance.

Hence data mining on Hadoop by default imposes an entry barrier or learning curve of understanding and utilizing the Map Reduce framework efficiently. This is where meta-layers like Hive come in and as we will see in this article, help lower the barrier for data analysis on large clusters of distributed data.

Thus, when compared to a traditional DBMS, the Hadoop ecoystem builds on the System R architecture and extends the notion of abstracted storage and query to a very distribted level. At the same time it sacrifices some of the properties of traditional DBMS (like query response times) along the way.

HIVE WALKTHROUGH

Why Hive? Because it comes closest to the abstraction level a database person is familiar with in terms of tables/tuples/filters etc. It also allows interaction with large amount of data without the necessity of being familiar with Map/Reduce paradigm. It provides command line/web interface to write Hive queries using HQL - Hive Query language – very similar to SQL. The real difference is HQL queries are transformed into map-reduce jobs underneath for execution. This in a way allows Hive to operate on very large datasets.



We will do a walkthrough of interaction with hive using a dataset from - http://bit.ly/hd-ilpd link

WHAT IS THE HIVE METASTORE?

Metastore is the place where hive stores information about tables, serialization formats, indexes, security (roles) etc. This is referred to anytime for query resolution and plan generation. It is updated whenever there is change/addition in schema or items in Hive.

Hive's Metastore is configured to store metadata locally in an embedded Apache Derby database. Unfortunately, this configuration only allows a single user to access the Metastore at a time. Here the user can be such as the Hive Web UI or CLI client. In HadoopOnAzure one can eliminate this limitation by using SqlAzure. In an 'on-premise' cluster it can be any relational store.

INTERACTION WITH HIVE ON HADOOPONAZURE

At present HadoopOnAzure provides a

1. Web based Hive frontend
2. Web based JavaScript frontend
3. Terminal services login into the node

For viewing the logs of the submitting query HadoopOnAzure streams back the result of the job.

In this example we will use mixture of JavaScript/hive frontend in the browser itself.

LOADING OF DATASET

Once the patient dataset is available locally, it can be uploaded in multiple ways.

1. FTP to the cluster – this requires opening of the ftp port on the cluster
2. Load into the azure storage – which in turn can then be accessed
3. Front end based load via JavaScript – for small sample data (< 6MB)

Loading via JavaScript console involved #put command which will prompt to load the local file to the cluster. #ls command on JavaScript console can verify that file is present.

```
# ls
-rw-r--r-- 3 hiveql supergroup 23755 2012-08-13 09:55 /user/hiveql/
ip.csv
```

I usually tend to copy sample data to avoid multiple uploads and #cp command can help in the JavaScript console.

SCHEMA OF DATA STORAGE

Hive allows creation of schema to constrain the data. This can be done via the hive front end on web for the cluster. The following command defined the meta-structure called livpatient. This information is stored in Hive's Metastore which is typically a RDBMS.

```
create table livpatient(
age int,
gender string,
totBil float,
dirBill float,
alkphos int,
sgpt  int,
sgot int,
totProt float,
aLB float,
aG float,
sel int
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
```

Data Storage - Interesting points

Hive does not "own" the persistence of data into HDFS in terms of file-formats. Users can plug in any format and provide readers/writers for the same. Hive parses the data file stored in HDFS using Serializer/Deserializer (SerDe) mechanism. A table in Hive is basically a directory with the data file(s) in it. When defining a table in Hive we need to provide following two (which are stored in the Metadata Store):

1. The folder location that contains the data files
2. How to "parse" the data for reading and writing to the file

This adds interesting "twists" - Depending on Hive configuration, simply adding more files to the table's folder in the file system adds that data to the table. In the special case that the table is partitioned, then each partition in the table is a sub-folder within the table's folder.

Hive Local Data

Data can be loaded into Hive using the following command. This moved the data into the Hive folder on HDFS.

```
LOAD DATA INPATH 'ip.csv' INTO TABLE livpatient
```

Certain points to keep in mind while loading data into Hive

- By default loading data overwrites existing data.
- In 0.8 version an INSERT INTO was introduced which kept the data intact or allowed append to existing data.
- 0.9 Version supports INSERT OVERWRITE with (IF NOT EXISTS) allowed overriding of OVERWRITE.

Since Hive started off largely as a system to read and analyze data its DDL/DML functions were barebones compared to standard RDBMS. However as things mature and Hive gets more popular these "RDBMS like" features are getting resolved and prioritized slowly over period of time. It also shows the legacy of "design decisions based on specific requirement at given time".

External tables

One can also define External tables in Hive. External Tables are useful when you have data being updated by an external source like log files being moved into HDFS from a remote location. In case of External tables Hive does not move the data into the default Hive location. Another important distinction is that Hive does not delete any data file when an external table is dropped. Syntax for creating an external table is as follows

```
CREATE EXTERNAL TABLE datad (
id BIGINT,
active TINYINT,
dt String
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LOCATION 'ASV://datad/ ';
```

LOCATION is where we store the data files. If one creates an External table with the same name as the folder name, it allows addition of files according to given constraint in that location for instant query.

Partitioning Tables

After sometime it becomes imperative that we want to have partitions for ease of loading a batch of data or querying smaller section of data. Hive provides that too. Let us say in this case we want to store date wise information.

```
CREATE EXTERNAL TABLE datad (
  id BIGINT,
  active TINYINT,
  dt String
)
PARTITIONED BY (dt String)
```

Loading of the data then is

```
LOAD INPATH 'ip.csv' OVERWRITE INTO TABLE user PARTITION
(dt=2012-08-17)
```

The data can be found in "./user/2012-08-17". Each Partition can be further divided into buckets. These buckets are based on hash value of column in the table.

Interaction with hive data

This can be done through hive frontend on the web.
describe livpatient; shows the metadata of the table

```
Output (stdout)
age int
gender string
totbil float
dirbill float
alkphos int
sgpt int
sgot int
totprot float
alb float
ag float
sel int
```

Interesting output apart from the regular description of the metadata is the log file and time it takes to execute it.

```
Hive history file=C:\Apps\dist\logs\history/hive_job_log_
hiveql_201208160942_388214921.txt
OK
Time taken: 3.938 seconds
```

Almost every interaction with cluster gets logged into the history and allows nice way to go back to look at what all we executed.

```
select * from livpatient where gender ='Male'
```

We will skip the output part as it is obvious and instead focus on log output.

```
Hive history file=C:\Apps\dist\logs\history/hive_job_log_
hiveql_201208131348_12183633988.txt
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_201208130644_0001, Tracking
URL = http://10.114.254.96:50030/jobdetails.
jsp?jobid=job_201208130644_0001
Kill Command = C:\Apps\dist\bin\Hadoop.cmd job -Dmapred.job.
tracker=10.114.254.96:9010 -kill job_201208130644_0001
2012-08-13 13:48:28,485 Stage-1 map = 0%, reduce = 0%
2012-08-13 13:48:40,610 Stage-1 map = 50%, reduce = 0%
2012-08-13 13:48:43,657 Stage-1 map = 100%, reduce = 0%
2012-08-13 13:48:55,688 Stage-1 map = 100%, reduce = 100%
Ended Job = job_201208130644_0001
OK
Time taken: 48.968 seconds
```

Since this is introductory article for SQL/DB folks – we will get into little detail here. What is happening here is data is stored on HDFS, Hive gets the HiveQL a SQL like query language and coverts into map-reduce jobs after creating a plan and doing little bit of optimization.

Till now MapReduce was completely hidden from us and this is good thing from an Analyst point of view. Tools like Hive/Pig/Cascading/Cascalog hide the underneath complexity and allow us to work at much higher level of abstraction.

DIFFERENCES BETWEEN HIVE AND TRADITIONAL DBMS

By providing a SQL meta-layer on HDFS Hive easily gets pulled into comparisons with traditional DBMS systems. However there are some key differences that should be acknowledged to start off with.

- Hive is based on Hadoop which is a batch processing system. This system does not and cannot deliver low latencies on queries. The paradigm here is strictly of submitting jobs and being notified when the jobs are completed as opposed to real time queries. As a result it should not be compared with systems like traditional DBMS.
- Hive allows creation of arrays/maps types and allowing custom definition of types. There is no in built notion of constraints but can be enforced by custom data type.
- Hive allows inserts into multiple tables.
- Hive has NO explicit Update statement.
- There is no "ACID" as we are familiar with databases. This implies concept of locks is not available. Shared/ Exclusive locks are suggested at this juncture.
- Hive stores meta-information about partitions, tables etc. in a relational DBMS. This meta-data this is stored in proprietary format but exposed via public views or meta-tables.
- Concept of index is making its way through. Bitmap index has made it in 0.8 version.
- This simply means physical access is never going to be fast enough when compared to traditional DBMS
- The filters have not been pushed down to the bottom most level (again proposed to be fixed). This means execution plans really can't be very efficient and you can see lot of data getting pulled in to satisfy a query. It also has dependencies on how is storage handled - natively or externally. In latter case that Storagehandler needs to implement - configureTableJobProperties and other friends.
- Related to above issue is lack of column level statistics
- There is no concept of foreign keys to constrain a given collection of tuples.
- It allows actual storage to be pluggable via StorageHandlers so one can store data in HBase or other desired locations (HyperTable, Cassandra, MongoDB etc.)
- It allows storage of data in specialized compressed format called sequencefile, textfile, rcfile or custom input/ output format.
- It allows atomic creation of table as result of execution of a select query
- SELECT statement allows "regex" based column specification

SELECT {{(d)?+.+}} FROM livpatient

TOOLS TO INTERACT

Currently a lot of Third-Party vendors like Karmasphere, DataMeer provide frontends do a better job of creating scalable Metastore and visual layout of queries. DataMeer for example provides a Rich Excel like interface to pull in sample data and define formulae and functions to work on the sample data. Once complete, these are then converted into appropriate low level queries to work on the complete data set on HDFS.

Interaction with Hive on HadoopOnAzure through ODBC

Microsoft has released a Hive ODBC driver for Excel and other usage (.Net developers can use it too). One significant eye-sore will be time taken for basic queries will overwhelm the first time user of hive.

Configuration

Today HadoopOnAzure provides JavaScript and hive interface to hive thrift server. It does not expose all the configurable properties from hive-default.xml or for that matter HDFS related information. Command Line Interface via RDP into master node is better way to change the settings.

Handling failure of data nodes

HDFS takes care of failure of node using replication of blocks across multiple nodes. By default replication factor has value of 3 which means a data block is replicated on other two locations. This also implies one is using 3X the space on the cluster. Datanodes send heartbeat to NameNode and if they don't respond within 10 minutes, NameNode will mark them dead. Nodes can die due to hardware failures or need to be taken down for software updates or replica corruption. The NameNode constantly tracks which blocks need to be replicated and initiates replication whenever necessary.

NameNode is single point of failure. Secondary NameNode is unfortunately in this version just checkpoint collector. Which means it can't replace Primary NameNode in event of failure. It only provides access to last known check pointed image. Primary Namenode's regular checkpoints are the key here which can help in restart of Namenode on same or different machine. Machines running NameNode and Tasktracker need resources in terms of memory especially as the cluster namespace is maintained in memory.

Scalability of data nodes

Again HDFS allows very easy scaling out of the nodes. In on premise world adding the new node's DNS name to the config/slaves file on the master node followed by execution of commands such as –

```
start datanode
```

and Tasktracker makes new node to be part of the cluster. In some cases –

```
hadoop HDFSadmin –refreshNodes
```

and

```
hadoop mradmin –refreshNodes
```

need to be run to make NameNode/Tasktracker aware of the new Datanode.

HadoopOnAzure does not at present provide this ability in beta release on the web GUI.

Concept of balancing the Datanodes is required so that data storage load is fairly shared across the cluster. In event of new addition of node HDFS by itself does not do automatic rebalancing of the cluster by moving the blocks. Rebalancer can help here and has to be run separately. It should be done with care on production clusters.

Monitoring/Profiler

The Hive explain command provides an AST for the query and dependencies between various stages.

```
explain select * from livpatient where gender = "Male"
```

```
Output (stdout)
ABSTRACT SYNTAX TREE:

(TOK_QUERY (TOK_FROM (TOK_TABREF (TOK_TABNAME
livpatient))) (TOK_INSERT (TOK_DESTINATION (TOK_DIR TOK_
TMP_FILE)) (TOK_SELECT (TOK_SELEXPR TOK_ALLCOLREF))
(TOK_WHERE (= (TOK_TABLE_OR_COL gender) "Male"))))

STAGE DEPENDENCIES:
Stage-1 is a root stage
Stage-0 is a root stage

...(contd...)
```

HadoopOnAzure shows the history of jobs on the cluster. To monitor the cluster in little bit more detail than what is displayed on HadoopOnAzure dashboard Jobtracker and NameNode provide more information. Jobtracker is present at - http://localhost:50030/ and NameNode information is present at http://localhost:50070 (replace 'localhost' with name of server hosting the services). You can do this once you do RDP into the cluster Masternode. Logs for Datanode, Tasktracker are available which can also open up wealth of information.

WHAT COULD BE POSSIBLE CHALLENGES OF RUNNING HIVE?

Fair allocation of resources – thus the QoS. How does one prevent big job hogging all the resources? Fair Scheduler and Capacity Scheduler are available in the contributions. They can be utilized by enabling their usage in configuration-in config/hadoop-site.xml & pools.xml.

SECURITY

Security of data at rest vs. data in transfer, access/authorization is all evolving for hive. HDFS has Unix-like user/group authorization.

PLANNING FOR HIVE -

OTHER ITEMS

- Adding incremental data
- Adding new nodes/Bringing down few
- Adding a Fair scheduler
- Monitoring the job progress/cancellation
- Identifying bottlenecks in JVM/HDFS settings
- Integration with SSAS if on-premise deployment to take advantage of low latency response
- Which Hadoop distribution to use and support
- Backup of Hadoop Metadata, data & file system layout (especially for hive)

At times comparison with MPP Database like PDW should be done to do fair justice to problem at hand and compare cost/returns/skillset availability/maintenance.

WHAT KIND OF APPLICATIONS ARE USING HIVE

Facebook uses it for

Reporting (daily, weekly aggregation of impressions, click logs)
Ad-hoc analysis
Machine learning (Ad-optimization)
Spam detection
Facebook stores summaries into relational database.

CNET uses it for

Data-mining, log analysis

CONCLUSION

Hive is best suited for very large data which needs interaction via SQL like query mechanism in scalable way. It is not right choice for low latency storage or as well as queries as of today. It is best suited for OLAP workloads where data can be aggregated and stored in faster/better system like SSAS for throughput.

End of the day for overall Hadoop adoption one will have to do lot of work around how to leverage pieces of the ecosystem. At present ease afforded by HadoopOnAzure in terms of provisioning, monitoring, storage (Azure storage) is good factor to consider it as part of solution.

Credits

1. System R – History and Evaluation of System R – ACM -1981 , Chamberlin, Gray & others) - image
2. Hive – A warehousing solution over map-reduce framework – VLDB -2009, Ashish, Namit, Zhao, Joydeep and others) - Image
3. Exodus - The EXODUS Optimizer Generator
4. Volcano - www.sis.pitt.edu/~vladimir/classes/infs-ci2711/literature/volcano.pdf
5. Cascade - http://www.informatik.uni-trier.de/~ley/db/journals/debu/Graefe95a.html
6. BigData team at Microsoft for review and feedback (Brad Sarsfield, Cindy Gross, John Gordon, Denny lee's Klout work integrating Hadoop and SSAS-presentation) ■

Govind Kanshi works as a senior architect in the Microsoft Technology Center. You can follow him on twitter at @govindk