ISSUE 11 | MAR -APR 2014

DNCMagazine

www.dotnetcurry.com

WPF 4.5 DataBinding Features

Create rich LOB apps using the WPF 4.5 DataBinding features

4 New Features in IIS 8.5

4 New and Exciting Features of the latest version of IIS

Issue Management Using JIRA

Issue Management Service using JIRA - an Atlassian product

SOFTWARE GARDENING: Watering your Garden

Become an aspiring Software Craftsman and begin to Water your Software

Using jsPerf to test jQuery Selectors

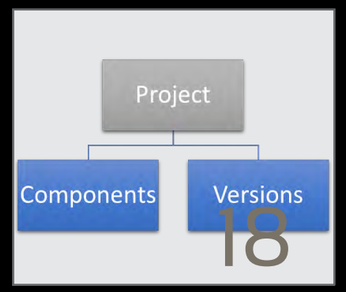
Compare jQuery Selector performance using the online jsPerf tool

Windows Phone App Studio

Create Windows Phone Apps using configurable themes and templates

Extend Frameworks using C# Extension Methods

Extend Framework classes and use Extension Methods with ASP.NET WebAPI, MVC



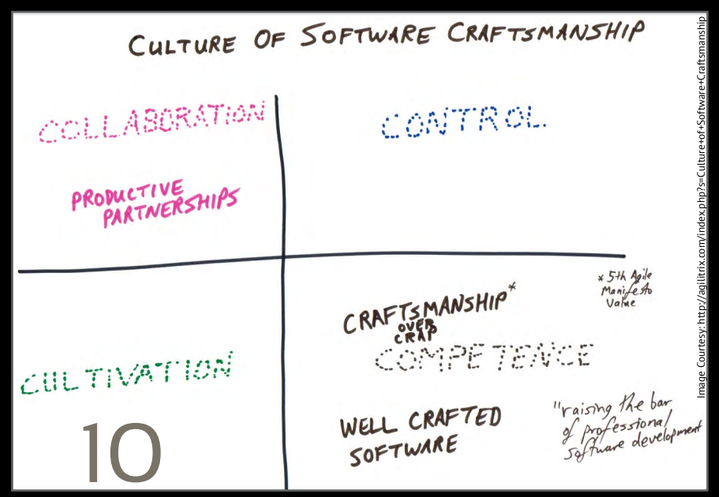
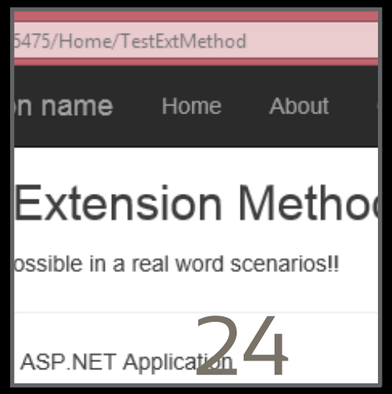
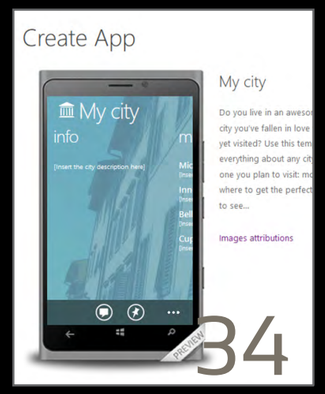


Image Courtesy: http://agilitrix.com/index.php?s=Culture+of+Software+Craftsmanship





Articles

stay

connected

www.Facebook.com/DotNetCurry

@dotnetcurry

www.DotNetCurry.com/magazine

LETTER FROM THE EDITOR

Hello Readers, Welcome to the Eleventh edition of the DNC .NET Magazine. The Microsoft world is alive with ebullience and volatility, with a new CEO talking the helm, some rapid changes in the top leadership including ScottGu as the Cloud Chief, Windows XP support ending on April 8th, the SkyDrive to OneDrive leap, the Build Developer conference announced in April; it's all Live and Happening here.

Back here at the DNC Magazine, as always, we have new tech-topics, new authors and some exclusive content. This month we have Windows Phone MVP Vikram Pendse joining our author club, with his quickfire introduction to creating Windows Phone Apps using the Windows Phone App Studio tool. Welcome Vikram!

Pravin Dabade walks us through some new, exciting and powerful features of IIS 8.5. Pravin also demystifies Extension Methods in a separate article and explains how this underused but useful feature can be used to extend frameworks like LINQ, MVC and Web API.

'Software Gardening' columnist Craig Berntson explains the culture of Software Craftmanship and what it takes to become a Software Craftsman. Subodh Sohoni talks about JIRA, an Issue Management Service tool and does a feature comparison with Microsoft's ALM Tools.

A much awaited WPF article is authored by our very own Mahesh Sabnis who does a thorough walkthrough of the new DataBinding features in v4.5. To wrap up this edition, I do a beginner level article for jQuery fans introducing them to jsPerf and demonstrating how to use the tool to test jQuery Selectors.

The C# Extensions Methods, WPF and a Beginner level jQuery article were introduced in this edition as a result of some of you writing back to us and demanding them. We love to hear feedback, and not just hear, we act upon them! E-mail us at suprotimagarwal@dotnetcurry.com or tweet @dotnetcurry and keep the feedback coming!

Suprotim Agarwal

Editor in Chief

Editor In Chief • Suprotim Agarwal
suprotimagarwal@dotnetcurry.com

Art Director • Minal Agarwal
minalagarwal@a2zknowledgevisuals.com

Contributing Writers • Craig Berntson, Mahesh Sabnis, Pravin Dabade, Subodh Sohoni, Suprotim Agarwal, Vikram Pendse

Writing Opportunities • Carol Nadarwalla
writeforus@dotnetcurry.com

Advertising Director • Suprotim Agarwal
suprotimagarwal@dotnetcurry.com

Next Edition • May 1st 2014

4 NEW FEATURES OF IIS 8.5

Pravinkumar Dabade discusses some new, exciting and powerful features of IIS 8.5. He also compares these new features with IIS 8.0 which was released with Windows Server 2012, almost a year before R2 was released.

WHAT'S NEW IN IIS 8.5?

IIS has been around for more than 17 years, ever since its first release in Windows NT 3.51. It started out as a basic HTTP server and has evolved to a fully configurable, highly secure and high-performing webserver. You can pretty much host anything on IIS; right from media streaming to hosting scalable web applications.

With the release of Windows Server 2012 R2 and Windows 8.1, a new version of IIS – version 8.5 has been launched which contains several new and exciting features. Some of the new features are listed below –

- Dynamic Website Activation
- Idle worker process Page-Out
- Enhanced Logging
- Logging to Event Tracing for Windows

In this article, we will discuss some of these new features and also see them in action. We will compare these new features with IIS 8.0 which was released with Windows Server 2012, almost a year before R2 was released. Although I have explored these new features in Windows 8.1, you can easily replicate them in Windows Server 2012 R2.

DYNAMIC WEBSITE ACTIVATION IN IIS 8.5

While using IIS 8.0 in Windows Server 2012, if you are hosting a number of websites, IIS will activate all the websites which are configured in it, during a system boot-up. The good thing about *automatic activation* of all the configured sites is that they are ready to respond whenever the first request arrives. But there is also a downside to automatic activation. The first being, it takes a bit of time to load the IIS configuration file and activate all the sites. This is because the Windows Process Activation Service (WAS) has to load the configuration of all websites hosted on IIS. Second being the sites which are activated, consume a lot of memory. Many of these sites may be requested instantly but what about the sites which are not at all called or are called after a certain time period. The activation time and memory consumption is unnecessary in that situation.

Let's observe IIS 8.0 activation. I have preconfigured 12 sites on IIS 8.0. Using PowerShell, I wrote the following command to check the service state running under IIS 8.0 –

```
PS C:\> netsh http show servicestate
```

After running this command, you will see all the 12 sites are queued up as shown below –

```
Administrator: Window
        Version: 2.0
        State: Active
        Request queue 503 verbosity level: Limited
        Max requests: 1000
        Number of active processes attached: 0
        Controller process ID: 1176
        Process IDs:

    Request queue name: MvcDemo2
        Version: 2.0
        State: Active
        Request queue 503 verbosity level: Limited
        Max requests: 1000
        Number of active processes attached: 0
        Controller process ID: 1176
        Process IDs:

    Request queue name: MvcDemo3
        Version: 2.0
        State: Active
        Request queue 503 verbosity level: Limited
        Max requests: 1000
        Number of active processes attached: 0
        Controller process ID: 1176
        Process IDs:

    Request queue name: MvcDemo4
        Version: 2.0
        State: Active
        Request queue 503 verbosity level: Limited
        Max requests: 1000
        Number of active processes attached: 0
        Controller process ID: 1176
        Process IDs:
```

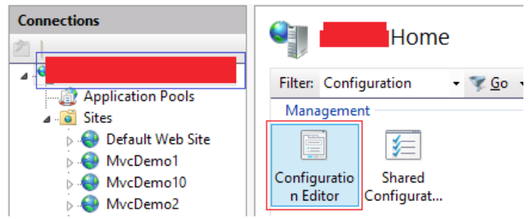
When you have a couple of hundred of websites hosted on your server, it makes sense to activate the website when the first request is made for it. Addressing this problem, IIS 8.5 has now introduced a new feature called as the "Dynamic Website Activation".

With *Dynamic Website Activation*, IIS 8.5 does not activate all sites during boot-up time. Rather IIS will activate the site when the first request is received by it. This activation may take some time initially, but after that all other subsequent requests will respond as expected. This will reduce the consumption of memory and unused resources which would unnecessarily occur during boot-up time, especially for those websites that are accessed rarely or not accessed at all.

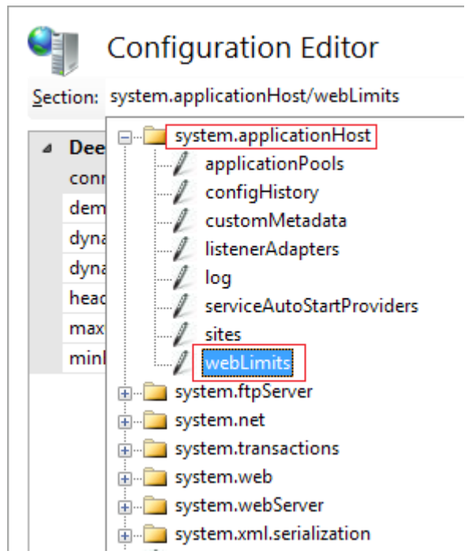
Setting Threshold

IIS 8.5 now lets you set the threshold while configuring dynamic website activation. Let's see how this feature can be utilized.

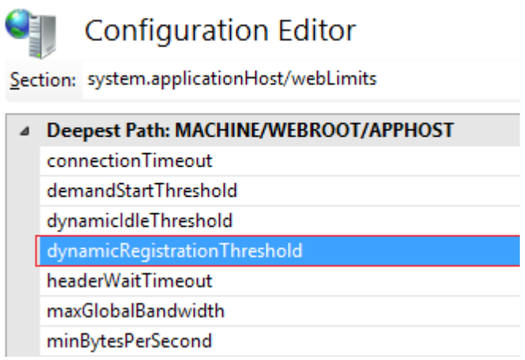
Open IIS Manager. Choose Server in a Connection Pane located on the left hand side and from the Management Section in the right side, choose Configuration Editor as shown here:



Once you open the Configuration Editor, from the section dropdown list, choose *system.applicationHost* and then the *webLimits* section as shown here:



This brings up the dynamic registration threshold which is by default set to 100 as shown below –



You can now change the value as per your requirement. When the threshold is set to 100 [this is default value] and if the number of sites are less than 100, IIS will by default activate all the sites. However in cases where you have 100 or more than 100 sites, IIS will activate each site when the first request is received for the site. When you change the threshold value, make sure you restart IIS.

Now let's test the service state again now for IIS 8.5. I have preconfigured IIS 8.5 with the same set of 12 sites which I had configured in IIS 8.0, a short while ago. I have set the dynamic registration threshold to 5 just for testing purpose. On running the NetSh command in PowerShell –

```
PS C:\> netsh http show servicestate
```

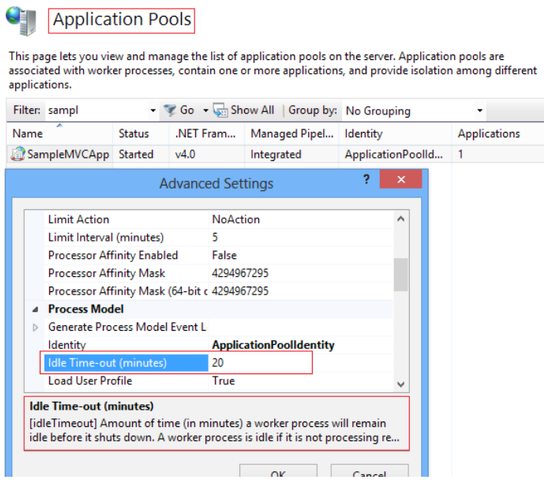
The output is as shown below –

```
Request queue name: Request queue is unnamed.
    Version: 2.0
    State: Active
    Request queue 503 verbosity level: Full
    Max requests: 1000
    Number of active processes attached: 1
    Process IDs:
        784

Request queue name: Request queue is unnamed.
    Version: 1.0
    State: Active
    Request queue 503 verbosity level: Basic
    Max requests: 1000
    Number of active processes attached: 1
    Process IDs:
        2640

Request queue name: W3SvcInspection
    Version: 2.0
    State: Active
    Request queue 503 verbosity level: Basic
```

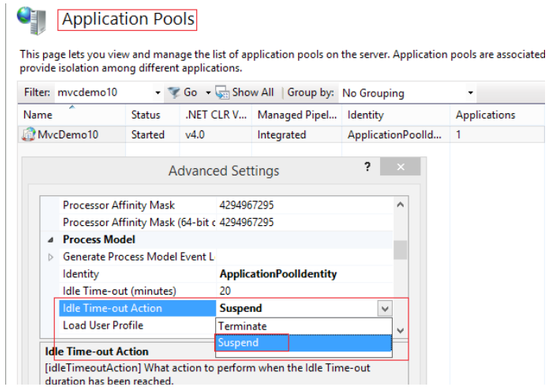
I have a total of 5 sites activated currently. If I start one more, my count will go to 6. This way IIS is only activating the site when the first request comes in, thereby reducing the consumption of resources.

IDLE WORKER PROCESS PAGE-OUT IN IIS 8.5

In IIS 8.0, the administrator can set the timeout for the worker process when it is in an idle state. In one way, it is a good option as the process gets terminated due to inactivity, thereby freeing the resources associated with it. On the other hand, it is a disadvantage as well. When the site is accessed the next time, the worker process takes time to complete its startup process, which keeps the user waiting. Let's check the timeout option of Application Pool in IIS 8.0 as shown below –

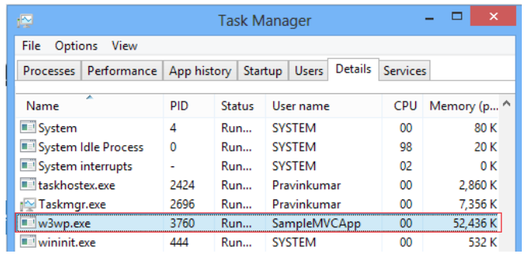


Now let's compare the same with IIS 8.5 Idle worker process timeout setting as shown here –

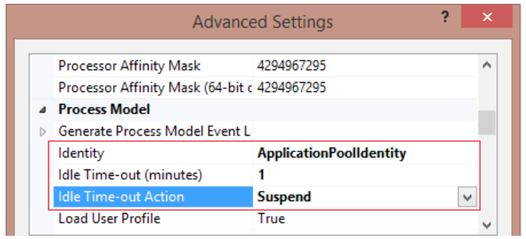


As you might have observed, we now have an *Idle Time-out Action* setting which we can set to either Suspend or Terminate. The default value is *Terminate*. You can change that to *Suspend*.

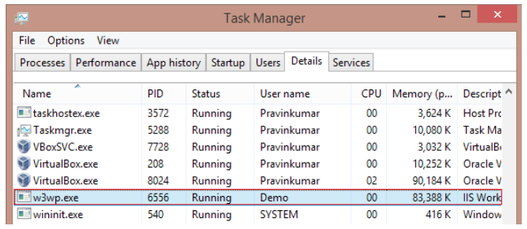
Just for comparison, I am changing the Idle Time-out (minutes) to 1 minute and then observe the worker process in IIS 8.0 by accessing the site. The observations are as follows –



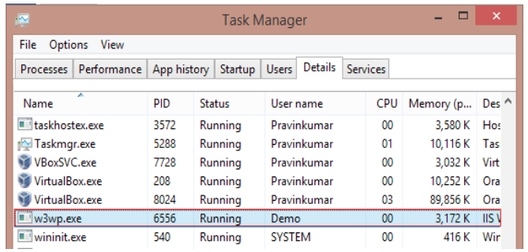
After one minute, you will observe the worker process has been terminated. When you try accessing the site once again, it will take some time to start up. Now let's observe the same demonstration in IIS 8.5. I have set the Idle time-out action to "Suspend". Let's run the site and see the difference –



After starting the website, following is the observation –



When the time-out occurs, you will see the same worker process now consuming less memory:



The observation is that the worker process has not been terminated. It is still alive but has been paged out to the disk. As it gets paged out to the disk, it ends up in less utilization of resources. Check the worker process memory consumption before going to the idle state and after going to idle state.

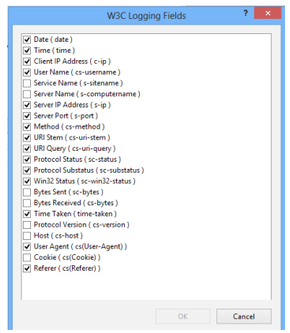
You can configure the same in the applicationHost.config file as shown here –

```
<system.applicationHost>

    <applicationPools>
        <add name="DefaultAppPool" autoStart="true" managedRuntimeVersion="v4.0"
        <add name="Classic .NET AppPool" managedRuntimeVersion="v2.0" managedPip
        <add name=".NET v2.0 Classic" managedRuntimeVersion="v2.0" managedPipeli
        <add name=".NET v2.0" managedRuntimeVersion="v2.0" />
        <add name=".NET v4.5 Classic" managedRuntimeVersion="v4.0" managedPipeli
        <add name=".NET v4.5" managedRuntimeVersion="v4.0" />
        <add name="Demo" autoStart="true" managedRuntimeVersion="v4.0">
            <processModel idleTimeout="00:01:00" idleTimeoutAction="Suspend" />
        </add>
```

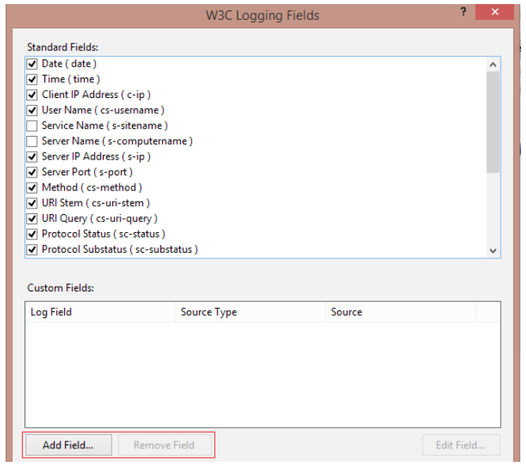
So a combination of idleTime-out action and the app initialization feature illustrates a quick startup time for the application in IIS 8.5.

ENHANCED LOGGING IN IIS 8.5

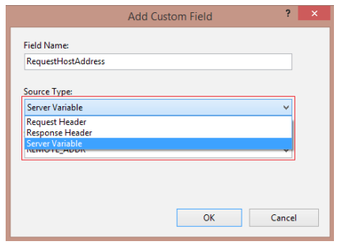
IIS 8.0 has some logging features which are configurable. But this logging system has very limited capabilities. For example, it provides you the standard logging capabilities which we cannot extend or customize during logging. The administrator of IIS has very limited options. Take a look at the following IIS 8.0 logging screen –



Now let's compare the same Logging Fields screen in IIS 8.5.



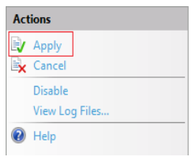
As you can see, we can add a custom field while configuring logging:



Add the field name and choose the Source Type. It gives us three different options –

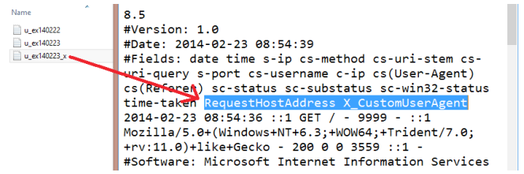
- Request Header.
- Response Header.
- Server Variables.

I chose the source type as "Server Variables" and the source as "REMOTE_ADDR. Make sure to click the Apply button in the Action pane on the right hand side as shown here:



Once you apply the changes, make a request to the site and check the log files.

The log files are generated under "C:\ inetpub\logs\LogFiles" [your system drive] as shown here:

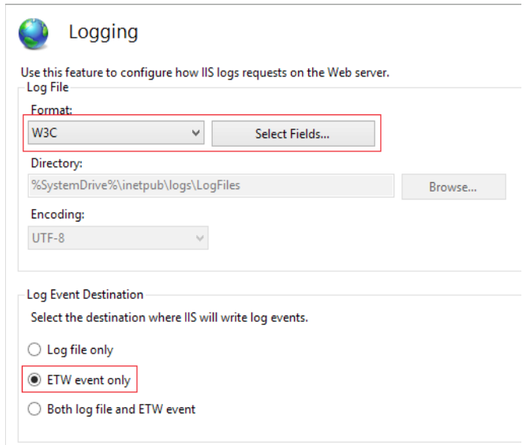


Now IIS administrators have the choice of adding custom fields within the Request Header, Response Header and/or Server variables to IIS logging to extend the logging capabilities.

EVENT TRACING FOR WINDOWS (ETW) IN IIS 8.5 IN THE LOGGING PROCESS

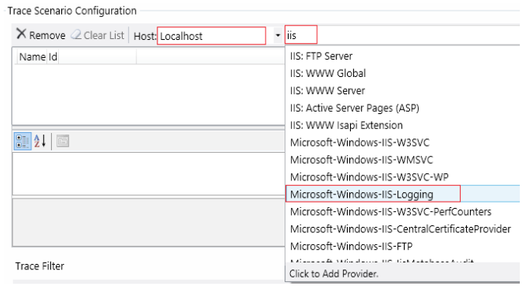
Earlier versions of IIS had fewer logging capabilities and IIS took some time to flush the log data into log files. Hence administrators could not monitor IIS logging information in real time. We also just saw that the log data is plain text which increases the time to process and analyse them. In IIS 8.5, administrators can now send logging data to ETW. You as a developer can create your own custom tools to read the data or use some ETW tools to analyse the logging data.

I am using Microsoft Message Analyzer tool to see real-time logging information. Let's first configure IIS log to send the information to ETW as shown here –

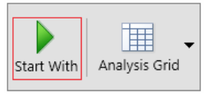


Please note that this configuration is now not going to log the information in log files.

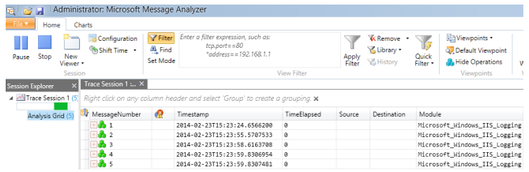
Do not forget to click the *Apply* button in the Action pane on the right hand side. Now start Microsoft Message Analyzer and configure it for Microsoft – Windows – IIS – Logging as shown here –



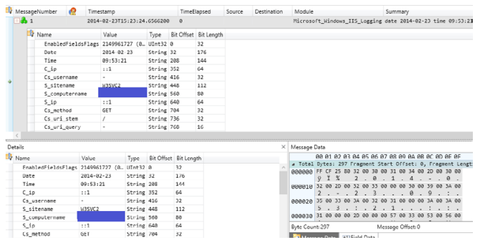
Click the button *Start With* located on the right hand bottom side –



Make some requests to the website and check the Microsoft Message Analyzer. You should now see the log information as shown here –



I made 5 requests. Now double click any one of the request and you will see information about the logging as shown here –



This way with ETW support turned on in IIS 8.5; events are shown immediately as event tracing is now built-in. It's a very important feature for Website owners and IIS admins to better monitor IIS in real time.

Conclusion

Most of the changes in IIS 8.5 reflect the Microsoft Cloud OS initiative that's in full swing. In the coming versions, it will be interesting to see how IIS fits into cloud-based scenarios! ■

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

Watering Your Garden

In the Jan 2014 issue of DNC Magazine, I wrote about the *Soil of Software Gardening*. As a quick review, soil is good Agile practices. Here are the four tenets from the Agile Manifesto:

- Individuals and Actions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Agile is an excellent guideline and is a key part of Software Gardening. But there is one problem here. Do you see it? Go ahead, read the four points of the Agile Manifesto again. In case you did miss it, the Agile Manifesto in reality has nothing to do with actually writing code. The Manifesto is all about *Project Management*.

For Software Gardeners, we need something that helps us write better code. It turns out; this problem has already been put forth by people who also realized the Agile Manifesto tells us nothing about how to write code. They came up with a manifesto to do just that. It's called the *Software Craftsmanship Manifesto*. **This is the water for Software Gardeners**.

Before we get into the manifesto, let's look at what defines a craftsman. In medieval times, a person had to go through a series of steps to earn a title befitting a craftsman. Often times that person would start out as an apprentice, then perhaps move to a journeyman level, then on to something else, and eventually become an expert or craftsman.

Often times this path was part of a guild. According to Bing Dictionary, a guild was a *"medieval trade association: an association of merchants or craftspersons in medieval Europe, formed to give help and advice to its members and to make regulations and set standards for a particular trade."*

Even today, in some industries, to become a craftsman, you have to go through a similar path. For example, in the USA, to become a carpenter, electrician or plumber, you will often join a union (the modern version of the old-time guild), and progress through a series of levels before becoming a Master Carpenter, Master Electrician, or Master Plumber.

A Master is someone who is highly skilled in their profession. The term craftsman means that and more. A craftsman is someone who is seen as taking great pride in their work. Look at someone who makes furniture as a hobby. This person keeps refining their skills to the point that not only is the furniture sturdy and well built, but the designs carved into the wood are intricate, elegant, and beautiful.

The idea of a Software Craftsman comes directly from the concept of guilds and apprentices. Wikipedia tells us, *"The movement traces its roots to the ideas expressed in written works. The Pragmatic Programmer by Andy Hunt and Dave Thomas and Software Craftsmanship by Pete McBreen explicitly position software development as heir to the guild traditions of medieval Europe."* (If you were with me in my first column here in DNC Magazine (Nov 2013), the concept of Software Gardening traces its roots back to The Pragmatic Programmer.)

Here's a picture of a Craftman's shop in the Witte Museum in San Antonio, TX.

Now, without further ado, here are the four tenets of the Software Craftsmanship Manifesto. As you read them, pay attention to how they intertwine with the Agile Manifesto:

- Not only working software, but also **well-crafted software**
- Not only responding to change, but also **steadily adding value**
- Not only individuals, but also a **community of professionals**
- Not only customer collaboration, but also **productive partnerships**

Let's look at each one in more detail to learn what Software Craftsmanship is all about.

WELL-CRAFTED SOFTWARE

How many projects have you had that is new development? How many that are enhancing an existing application? If you do more new development than maintenance, you're one of the lucky few. Most of us work far more on existing applications. And we always hate the code and come to say bad things about the person that originally wrote the code, often times that person was yourself.

But there is a better way. If we write the code to be more maintainable in the beginning, the maintenance would be easier. This means interfaces, small methods, single purpose classes, and more. I'll discuss these topics and more in future columns.

STEADILY ADDING VALUE

When you have to add code to that poorly written module, do you cram the new code in or do you rework the old code so that it's better? The Boy Scouts have a rule to leave the campground better than they found it. A good software craftsman should leave modified code better than he found it.

COMMUNITY OF PROFESSIONALS

Do you learn from others? Do you help others to learn? Do you encourage good practices? Do you attend user groups, conferences, code camps, etc? Do you have any non-work friends that are also developers? If so, do you talk about how to improve your skills? Do you read blogs and magazines? Do you keep abreast of new technologies or techniques? You should, if you wish to become a Software Craftsman.

PRODUCTIVE PARTNERSHIPS

Are you a detriment to your coworkers? What about your customers? How about your employer? Do you want to help them out? Do you want to help them solve their business problems? Do you do this by delivering the code as quickly as you can or do you work to have a low bug count and well-crafted code so that it will be easy to maintain.

Agilitrix lays out these four areas in quadrants that make it easy to see and understand.

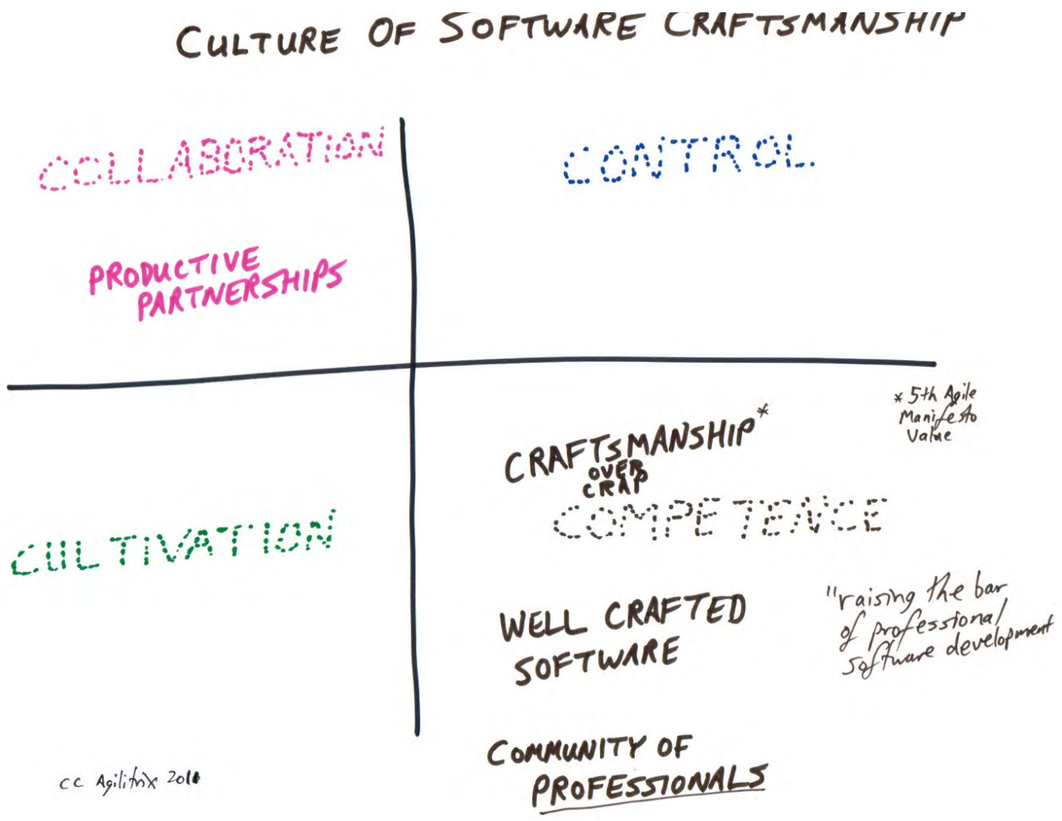


Image Courtesy: http://agilitrix.com/index.php?s=Culture+of+Software+Craftsmanship

There's another aspect to Software Craftsmanship that we can't overlook. Returning to Wikipedia's entry of Software Craftsmanship, we read in the first paragraph, "*Software craftsmanship is an approach to software development that emphasizes the coding skills of the software developers themselves. It is a response by software developers to the perceived ills of the mainstream software industry, including the prioritization of financial concerns over developer accountability.*" What this tells us is the industry typically puts money as more important than making the developer responsible for what he/she does. I don't totally agree with the quote from Wikipedia. Keeping the financial needs of the business is extremely important. But as developers, we need to be accountable for what we do. That includes the financial side of the business, the costs of development, as well as doing high quality work. My advice, make sure you understand business basics in general and more specifically your customer's or employer's business model and policies.

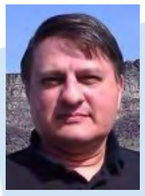
At this point you may be wondering *how you can become a Software Craftsman*. The short answer is, I'm not sure most developers ever do. There seems to always be a new language or technique to master. The long answer is you have to take the route of those people from medieval times. You need to learn, practice, and re-learn. You need to teach others as well as learn from them. And don't forget to learn about not only business in general, but about your customer's business or the business where you're employed. At each step of the way, you need to apply what you know. When you make mistakes, you need to understand what was wrong and then not do it again.

Now that I've covered what Software Craftsmanship is, I need to give you a warning. A Software Craftsman also knows when to do the simplest thing. It becomes easy to over-engineer a solution and this is just a bad as writing bad code. K. Scott Allen addressed this in a recent blog post.

In his post, Scott goes through an example that applies Software Craftsmanship guidance and shows how easy it is to over-engineer, then give the simple solution that turns out is easier to write, read, test, and maintain.

In the first paragraph of his post, Scott also says, "*Sometimes people will approach me and ask "what is it like to be a software craftsman?". I'll usually answer with: "Pffft, don't ask me, I just cranked out 500 lines of script that are harder to read than Finnegans Wake". At times though, I like to pretend what it might be like to be a software craftsperson.*"

Like Scott, I see myself as an aspiring Software Craftsman. I work every day to improve my skills and knowledge. If you head down this path, you too can become an aspiring Software Craftsman and begin to water your software. After all, it's a good thing to have software that is *green, lush and vibrant* ■



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

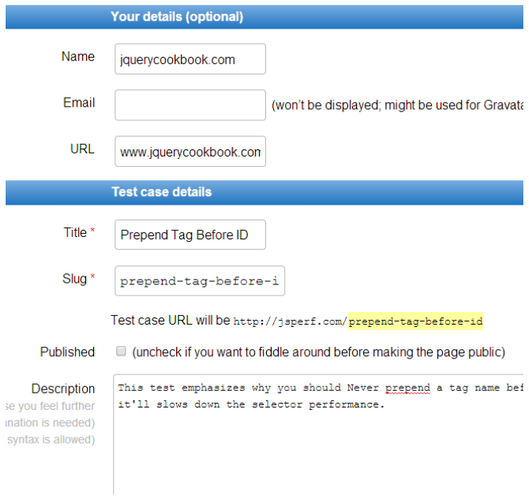
PERFORMANCE TEST YOUR **JQUERY SELECTORS** USING **JSPERF**

In recent years, Developers have started focusing on client-side performance, which essentially means testing your JavaScript code. One way to improve client-side code is to test blocks of code and optimize it. jsPerf.com provides a simple way to test the performance of JavaScript code. All you need to do is provide two or more code snippets that you want to compare for performance and it will test them to see which performs better and across different browsers.

Now we have all used CSS selectors to access DOM elements. For example using *.classname* allows you to access all elements that have the class .classname applied to them. jQuery uses *Sizzle* (which is a pure-JavaScript CSS Selector engine) to select elements from the DOM. Since there are multiple ways to use selectors, there are some techniques laid down to speed up the performance of your selectors.

We will demonstrate one such technique that says you should *never prepend a tag name before an ID as it slows down the selector.* So essentially that means if you have a DIV element with an id *description*, do not use *div#description* to select the element. Instead use only *#description* which is better performing than the latter. But how do we test this theory? Let's use jsPerf.

Go to jsPerf.com and set up some basic info for your test case. I have set it up over here http://jsperf.com/prepend-tag-before-id. Start by mentioning a Title and Description of our test case.



Scroll down to write some preparation code which is essentially some HTML code on which our selectors will work.

Preparation code

```
<h2>Why You Should Not Prepend Tag </h2>
<div id="description">
This example demonstrates the difference
Prepending the tag before an ID VS Not p
a Tag to improve Selector Performance.
</div>

<script src="//ajax.googleapis.com/ajax/
</script>
```

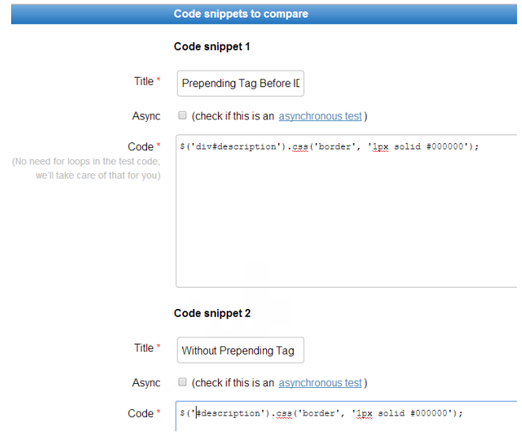
The next step is to write some code snippets to compare. We are essentially comparing two pieces of code:

```
$('div#description').css('border', '1px solid #000000');
```

Vs

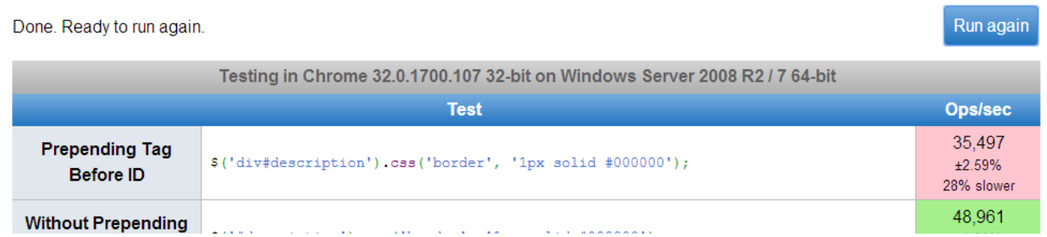
```
$('#description').css('border', '1px solid #000000');
```

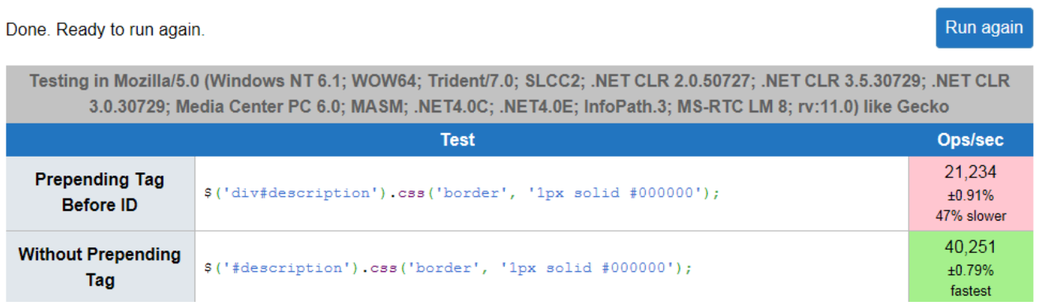
..where in the first snippet we are prepending the Div tag *before* ID (div#elementid) VS selecting the element *directly* by its ID (#elementid)



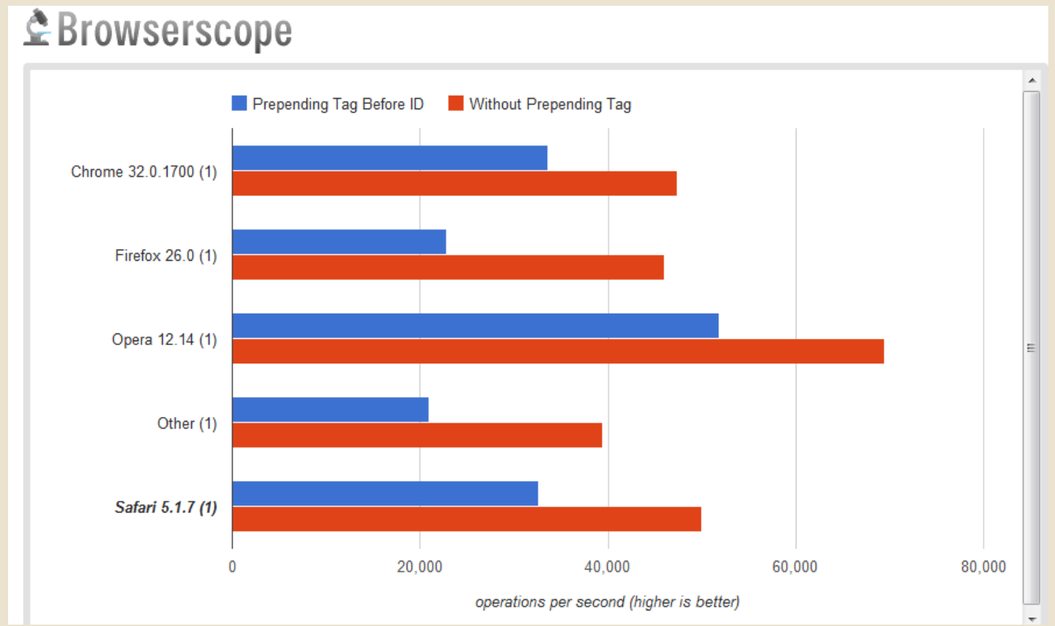
Scroll down to the bottom and click on 'Save Test Case'. In our case, the jsPerf test got saved as http://jsperf.com/prepend-tag-before-id

Run the tests and see which code performs better. Here's a test run on Chrome v32.0 and IE 11.0 respectively





jsPerf keeps a track of each browser result, so you can compare benchmark tests across environments. Here's a snapshot of the same test run in different browsers and the performance comparison across browsers.



As you can see, after running some tests on jsPerf.com, we can see that the snippet which *does not* prepend tag before ID (shown in red) performs better than the snippet which prepends tag before ID (shown in blue). The higher the number, the better it is as it conveys a higher number of operations per second. Hence using this simple jsPerf test, we can reach the conclusion that if you *prepend the tag name before an ID, it slows down the selector.*

I introduced you to jsPerf.com so that you are *aware* of a tool that can help you test your code snippets when you have empirical evidence that your code is not performing well. Use it meticulously!

This article was taken from my upcoming *The Absolutely Awesome jQuery Cookbook* at www.jquerycookbook.com ■

AS DEVELOPERS, SHOULD OUR FOCUS BE CODE READABILITY OR PERFORMANCE?

This question lingers the mind of most developers out there. One point to always remember and as quoted by Donald Knuth is *Premature optimization is the root of all evil (or at least most of it) in programming.* Your focus should be on writing the correct algorithm. Write it in a way that's correct and readable. Don't spend too much time on optimization unless you exactly know what to optimize.

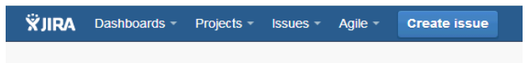
Suprotim Agarwal, ASP.NET Architecture MVP, is an author and the founder of popular .NET websites like dotnetcurry.com, devcurry.com and the DNC .NET Magazine that you are reading. You can follow him on twitter @suprotimagarwal or learn more about his new book www.jquerycookbook.com

Issue Management Using JIRA

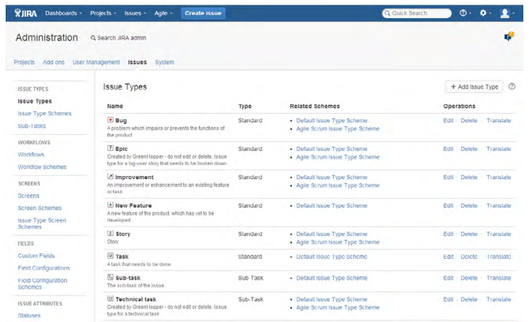
In the field of Application Lifecycle Management, products provided by Atlassian are regarded as highly feature rich which help Software teams to plan, track and collaborate. Gartner in their recent study has bracketed these products along with similar products of Microsoft and IBM. I set out to check the capabilities of Altassian products and note my first experiences with those. At the hub of the Atlassian ALM product suite is JIRA, one of the well known Atlassian product - https://www.atlassian.com/software/jira

JIRA provides issue management service. *Issue* is a data structure that stores the metadata of entities that we come across in software development. By monitoring the changes in the metadata, we can track the related entity. Concept of issue is somewhat similar to work item concept in Microsoft Team Foundation Server. JIRA supports many types of issues:

1. New Feature
2. Improvement
3. Bug
4. Task



Each issue is defined by the data it stores. For storing the appropriate data, there are fields defined for each issue type. Some of the fields are common to all issue types like the Project that issue belongs to, Key as a unique identifier of the issue, Summary to provide title of the issue, priority, reporter as person who originally authored that issue, current assignee, estimated efforts and status etc.



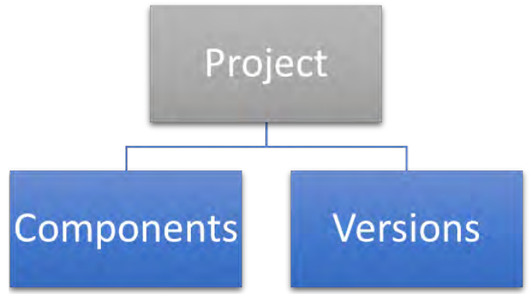
Issues are either standard or are Sub-Task which is a child of another issue like a *New Feature*. This way of classification is different than TFS work items, which are all of same level but can be structured in a hierarchy with various relations and links.

Issues follow certain workflow. They reflect the states of entity that they represent, which transition from one state to another. For example a requirement may be 'New' when it is added and then it may be listed in 'To Do' and then transitions to 'In Progress' when team starts working on it. When it is fulfilled, it is set to 'Done' state. The same states are reflected by New Feature or Improvement issue that represent the requirement. Workflow is defined by JIRA for each issue.

Fields and status need to be viewed by assignees and other stakeholders. Screens are defined for issues to show snapshot of that data. When we install JIRA, default fields, workflow and screen are assigned to all issue types. We may customize those as needed.

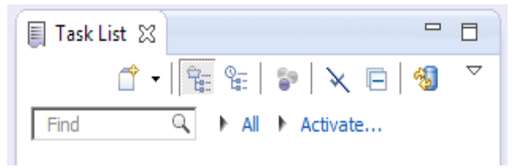
This definition of issues is quite similar to the way work items are defined in TFS. In the issue design fields, workflow and screens can be predefined in schemes and applied to appropriate issue types.

Issues under JIRA can be grouped based upon some logical criteria. Top level of grouping mechanism is project. A project is organizational reason to create collection of issues. It can be software development project or a software maintenance project or infrastructure deployment project. Each project may contain a number of components. Each component is a logical criteria defined by the organization for grouping of issues. Modules in the software can be a good example of components. Concept of component matches that of Area in TFS work items. Each project may also have number of versions. Usually versions map to releases of the software. Each version may have state transitions from Unreleased to Released to Archived.

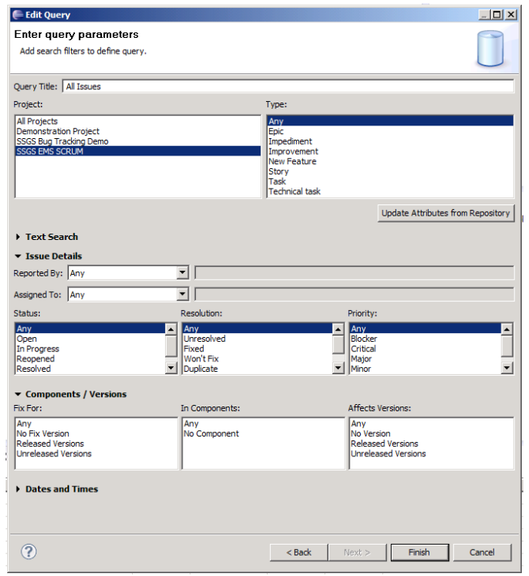


Once the issues are created, they can be viewed in a tabular way or in details. For the purpose of managing few issues at a time, we can create and save filters. Filters are complex criteria that show only issues that match that criteria. JIRA provides a filter editor to create filters based upon the fields and their values. I felt the filter mechanism to be little inadequate since it does not provide a hierarchical view with multi-layer filtering mechanism that TFS does provide.

User Interface of JIRA by default is a browser based application. Compared to TFS, it seems a little clunky but functionally looking at the features, it is adequate and it works. It is also possible to get an IDE plugin for integrating JIRA with IDEs like Eclipse and Visual Studio https://www.atlassian.com/software/ide-connectors/overview . In Eclipse, the plugin extends the services provided by Mylyn. In the Mylyn perspective of Eclipse, it provides a window to view existing tasks, create new tasks and update data of tasks. Tasks of Mylyn are the issues that are created in JIRA.

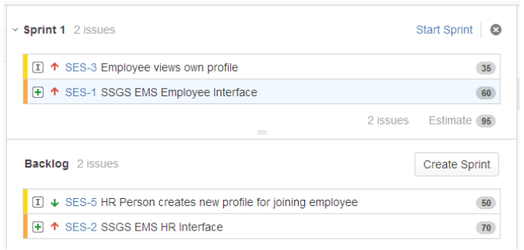


It is also possible to run queries on issues and create new queries with the help of query editor in Mylyn perspective.



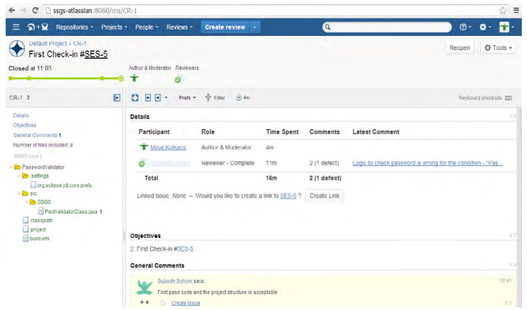
Features of JIRA are extended by installing the JIRA Agile plugin. It adds functionality that makes possible to add and manage Backlog, Sprints, Epics, Stories and Sprint Backlog based upon their story points. These features are part of TFS out of box and does not require any additional installation. JIRA Agile requires separate installation and license to be purchased. As a foundation, it adds the issue types for Epics and Stories. Epic is essentially a story that is not possible to be completed in one sprint. Epic usually will spread over many sprints and any other issue type like story that fit into one sprint, can be made as part of the defined epic. Story represents a Use Case in the real life. One of the biggest advantages of issue type story is that it provides a field named "Story Points". This field is used first for estimating efforts involved in implementing that story.

Story points can be provided in any unit of team's choice. It is a relative term. Once the stories are assigned some story points and priority that represents the value a business has for that story , the team can plan the sprint. In JIRA Agile, team can define sprints and then drag and drop stories from product backlog onto the sprint to create sprint backlog. Every time a new story is dropped on the sprint, JIRA Agile recalculates the total story points for the sprint and shows it. If you know the velocity of the team (story points that the team can get done in a sprint) then you can plan those may stories in the sprint. This feature and the user interface is quite neat.



The next important ALM service is Source & Version Control (SCM). Unlike Microsoft TFS, Atlassian does not offer any mechanism for source and version control. Instead of that, it provides integration with common SCM like Git, SVN, TFS etc. This integration is provided by another piece of software named *Fisheye* from Atlassian. https://www.atlassian.com/software/jira. For me the most important functionality of Fisheye is ability to link source code check-ins with issues. It is an important factor in traceability of source from requirements. Additionally Fisheye also provides ability to search, compare, view history and visualize branching structure.

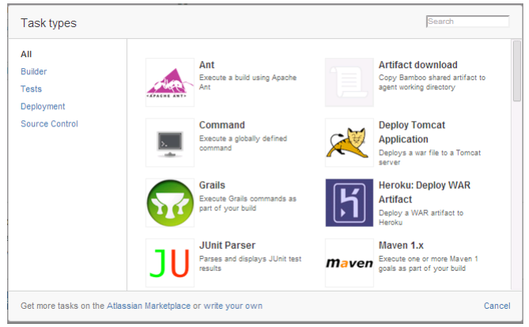
Along with Fisheye, another software that is bundled is *Crucible*. It provides code review features. When an issue is linked to source, the form of that issue shows the tab of Source. Under that tab of source, we can find the details of check-ins done which had that issue associated. For each of them, there is a button to create a review. We can then provide the reviewers name. It is also possible to create review from home page of Crucible. While creating that, it is possible to attach Changeset or files from code repository or any other files to be reviewed. Review comments can be given for the entire scope of the files under review or each line in each code file that is in changeset or attached. These comments can be replied by team members. Review is completed by reviewer and closed by the person who had requested review. Functionality of Crucible is fairly similar to the code review functionality offered by TFS 2012 onwards out of box.



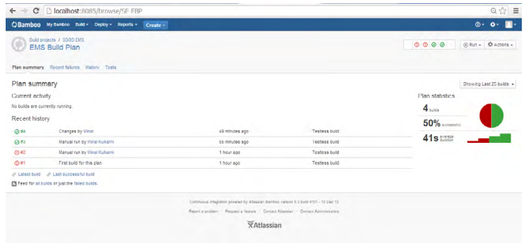
Atlassian provides another plugin called Stash (https://www.atlassian.com/software/stash/overview) to connect to any Git repository that you may have. Stash does not require Fisheye to connect but does not have many capabilities that Fisheye has.

Another very important part of Application Lifecycle Management is an Agile practice – Continuous Integration (CI). Set of tools that support CI should trigger the Build – Test – Deploy process against the event of check-in. In TFS this triggering mechanism as well as execution of build workflow, are part of the services. Bamboo is an Atlassian product that supports such trigger but leaves the process execution responsibility with other tools like Ant, Nant, Maven and even MSBuild. While configuring Bamboo, we have to provide the URL of the source control repository with credentials to access source controlled files from that repository. After configuration, initial activity in Bamboo is to define a plan which actually is the named workflow of the CI process. In this plan, then we can add individual tasks that define the activities of the workflow. Vast number of task types are supported in Bamboo. It supports tasks that are for:

- Source Control – Check-out, Branching, Tagging
- Build – Ant, MSBuild, NAnt, Maven, Grails, Visual Studio, Command Prompt based scripts
- Testing – MSTest, NUnit, JUnit, MBUnit, PHPUnit etc.
- Deployment – Tomcat, Heroku, SCP and SSH scripts



This process usually will start from source check-out and then build, test, deploy tasks will follow. It is either triggered automatically when any commit in the linked repository happens or it can be also manually triggered. Results of the builds are shown on the Plan Summary page. We may also view Recent failures, History of the builds and test results.



The last integration I would like to mention is between JIRA and Confluence. Both of these products if installed together integrate without any efforts. It is possible to add:

1. Link to a JIRA issue on any Confluence Wiki page. A built in macro helps us to search and select an issue on the page.
2. Add an issue in JIRA from Confluence Wiki page.
3. Show a Confluence Wiki page on the dashboard of JIRA.
4. Link a Confluence Wiki page to an existing JIRA issue.

My primary observation is that the integration of TFS with SharePoint is much richer compared to that provided in integrated pair of JIRA and Confluence.

SUMMARY

In my opinion, Atlassian has created excellent products which work as stand-alone servers for issues management, do agile project management, link source control to issues, manage builds and support team collaborate with integrated Wiki with issues management. It may not have attempted to make its own source control and build mechanism but rather has provided plugins that can integrate its core product of issues management with third party mechanisms of source control and build. I found a couple of points where improvement may be necessary, but overall I was pleased with what I saw.

Integrating all products is not trivial task. It took some efforts from my side to install and integrate all products of Atlassian, IDE like Eclipse, source control mechanism – SVN and build mechanism – Ant. For all of them to work together continuously, a dedicated administrator may be required. Atlassian can think of creating an integrated product for all of these. TFS does that quite seamlessly.

Another challenge that I would like to point out is about integrated security. If I want to integrate my organizations active directory domain or a LDAP server with all these products, I need to add another product called Crowd that can talk to these external authentication agencies. If the same can be integrated in JIRA, it will be more convenient ■

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

C# EXTENSION METHODS DEMYSTIFIED

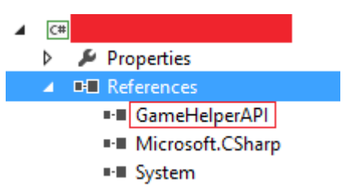
Praveen Dabade demystifies the controversial yet powerful Extension Methods and demonstrates some cool examples of extending classes, interfaces, collection classes, enums and frameworks using Extension Methods.

Extension Methods allow an existing type to be extended with new methods without changing the definition of the original type. Before diving into Extension Methods, I want to discuss some challenges we face while adding new functionality to existing API's. Let's assume I am creating a Console Application with a class called as Sales. We will define this class as a *sealed* class to prevent other classes from inheriting from it. Our class is defined as follows:

```
public sealed class Sales
{
 public double COGS { get; set; }
public double Expenses { get; set; }
public double ActualSales { get;
set; }

public double SalesNetProfit(double
COGS, double Expense, double
actualSales)
{
 return actualSales - (COGS +
Expense);
}
}
```

Now at a later date, one of your team members decides that he/she would like to extend the functionality of Sales Class by adding a new method called as *CalculateServiceTax*. As a usual practice, we would inherit the class and build our functionality. But this is not possible as our class is a sealed class [Not Inheritable class]. Another way could be that your team member may request the owner of the Sales class to add an additional method for calculating service tax in it. Although this would do the job, there is a higher chance that this may break the compatibility of other modules and projects with this class. So in most cases, this request will be denied by the owner of this class.

Take another instance where you are using a third party DLL in your project. In my console Application, I will use a hypothetical popular third party DLL called as *GameHelperAPI*. Now I am expecting that this DLL has a class with the *TextExplode* method in it, which I want to use. So I will first add a reference to this DLL in my project –



When I create an instance of the class *PrintTextHelper* present in the *GameHelperAPI* DLL, I find that the TextExplode method is not available in the third party API! Alas, that's not what I had expected as I thought this method is available in the class.

The question is can I extend the class PrintTextHelper from the GameHelperAPI to add the TextExplode method? It's obvious that I do not have access to the source code of the third party API [not applicable in special cases]. So I cannot change the original type to add a new method. I will try and inherit the class only to find out that I cannot do it because the class has been declared as *sealed* by the third party provider.

These kinds of issues are frequently encountered in our development scenarios where we are restricted to extend existing types. Similarly the .NET Framework classes also have similar restrictions. So is this is a dead end?

In .NET 3.5, a new concept called *Extension Methods* was introduced. Extension methods allow you to add new methods or properties to an existing type like a class or structure [includes Custom APIs, Framework APIs and third party APIs] *without* inheriting or modifying these existing types directly.

In the Microsoft .NET framework, the most common extension methods can be found in LINQ which extends the System. Collections.IEnumerable / System.Collections.IEnumerable<T> for query operators.

In this article, we will use extension methods to extend –

- LINQ Framework classes
- Custom classes
- Interfaces
- Nested classes
- ASP.NET MVC classes
- Web API classes

We will also see how to declare extension methods in –

- Standard Class Library.
- Portable class library

and use them in various scenarios like –

- Using extension methods in different languages. For example extension methods declared in C# will be used in VB.NET.
- Using Extension methods in Windows Store App
- Using Extension methods in Silverlight Applications
- Using Extension methods in standard .NET applications

USING EXTENSION METHODS IN LINQ

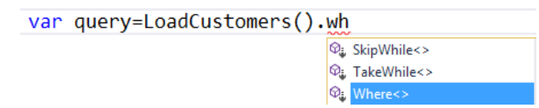
Let's kick start our example by directly using the existing Extension methods available in the LINQ framework. I have created a simple console application which is querying the Object collection. The customer class in the console application is declared as follows:

```
public class Customers
{
    6 references
    public string CustomerID { get; set; }
    6 references
    public string ContactName { get; set; }
    7 references
    public string City { get; set; }
}
```

The main method and customers collection is as declared here:

```
private static List<Customers> LoadCustomers()
{
    return new List<Customers>()
    {
        new Customers()
        {CustomerID="ALFKI",ContactName="Maria Andurs",City="Berlin"},
        new Customers()
        {CustomerID="ANJAL",ContactName="Anjala Honey",City="London"},
        new Customers()
        {CustomerID="SMITH",ContactName="Maria Smith",City="London"},
        new Customers()
        {CustomerID="ALISH",ContactName="Alisha Johns",City="New York"},
        new Customers()
        {CustomerID="MARKK",ContactName="Mark K.",City="Berlin"},
    };
}

static void Main(string[] args)
{
    var query = LoadCustomers().Where(c => c.City == "London");
    foreach (Customers customer in query)
    {
        Console.WriteLine("{0} : : {1} : : {2}",
            customer.CustomerID,customer.ContactName,customer.City);
    }
    Console.ReadKey();
}
```

Now filter the collection data using the where<> method, which is an extension method:



The .NET framework has made use of Extension methods for extending the framework classes as we just saw in our example.

CUSTOM EXTENSION METHODS

We will now try creating our own custom Extension method. I have created a console application which we will use to test our custom Extension methods.

In the demonstration here, we will extend the existing DateTime type of .NET framework. To do so, we will add a class called as HealthcarePolicyExtensions and then add some Extension methods to it as shown here –

```
public static class HealthcarePolicyExtensions
{
   public static int Age(this DateTime date, DateTime
                          birthDate)
  {
     int birthYear = birthDate.Year;
     int currentYear = DateTime.Now.Year;
     if (birthYear>=currentYear)
     {
        throw new Exception("Please enter correct birth
        date!!");
     }
    else
    {
        return currentYear - birthYear - 1;
    }
}

public static decimal ClaimPolicy(this DateTime date,
DateTime claimDate,decimal claimAmount=120000)
{
  DateTime currentDate = DateTime.Now;
  if (claimDate >= currentDate)
  {
     throw new Exception("Please enter correct Claim
     date!!");
  }
  else
  {
     return claimAmount;
  }
}
}
```

In the above class we are extending DateTime type with two methods –

- Age()
- ClaimPolicy()

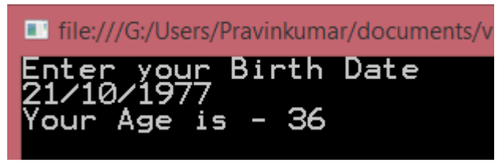
When we create an extension method, we have to follow some basic rules as listed below –

- Extension method are declared in a static class.
- All the extension methods must be static.
- The first parameter of the Extension method is a type which we want to extend, so it must be declared with this keyword.

We will now try to call our Custom extension methods in our main function. Observe how the DateTime class now has additional methods, as shown here –

```
static void Main(string[] args)
{
try
{
        Console.WriteLine("Enter your Birth Date");
        DateTime d = Convert.ToDateTime(
                        Console.ReadLine());
        DateTime calculateAge = new DateTime();
        int age = calculateAge.Age(d);
        Console.WriteLine("Your Age is - {0}", age);
}
catch (Exception ex)
{
        Console.WriteLine(ex.Message);
}
Console.ReadKey();
}
```

The output of the program is as shown below –



Using extension methods, we can extend classes, structures, interfaces, collection classes etc.

EXTENDING AN INTERFACE USING EXTENSION METHODS

Let's see how to extend an interface using Extension methods. We will declare a class called as "PurchasedProducts" as shown here –

```
public class PurchasedProduct
{
public int ProductCode { get; set; }
public string ProductName { get; set; }
}
```

Now declare an interface called as IProduct –

```
public interface IProduct
{
    IEnumerable<PurchasedProduct> GetProducts();
}
```

The interface declares a method called as GetProducts which returns an IEnumerable of PurchasedProduct. Create two classes which will implement the IProduct class as shown here –

```
public class FoodProducts : IProduct
{
public IEnumerable<PurchasedProduct> GetProducts()
{
    return new List<PurchasedProduct> {
        new PurchasedProduct{
        ProductCode=2900, ProductName="Chai"},
        new PurchasedProduct{
        ProductCode=2901,ProductName="Coffee"},
        new PurchasedProduct{
        ProductCode=2902,ProductName="Milk"},
    };
}
}

public class ElectronicProducts : IProduct
{
public IEnumerable<PurchasedProduct> GetProducts()
{
    return new List<PurchasedProduct>
    {
        new PurchasedProduct{
        ProductCode=3000,ProductName="Oven"},
        new PurchasedProduct{
        ProductCode=3001,ProductName="Washing
        Machine"},
    };
}
}
```

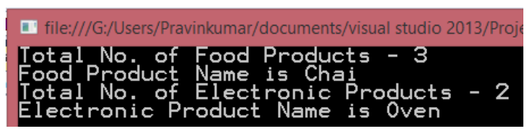
We will add an Extension method to our interface IProduct as shown here –

```
public static class IProductExtensions
{
 public static IEnumerable<PurchasedProduct>
 GetProductByCode(this IProduct product,int proCode)
 {
     return product.GetProducts().Where(p =>
      p.ProductCode == proCode);
  }
}
```

The IProductExtension class extends an interface IProduct which returns the product by passing the product code. Try using the interface extension method in our Main method –

```
static void Main(string[] args)
{
  try
  {
     IProduct product = null;
     product = new FoodProducts();
     Console.WriteLine("Total No. of Food
     Products - {0}", product.GetProducts().Count());
     Console.WriteLine("Food Product Name is {0}",
        (product.GetProductByCode(2900).
        SingleOrDefault()).ProductName);
        product = new ElectronicProducts();
      Console.WriteLine("Total No. of Electronic
        Products - {0}", product.GetProducts().
                        Count());
      Console.WriteLine("Electronic Product Name is
        {0}", (product.GetProductByCode(3000).
                        SingleOrDefault()).ProductName);

  }
  catch (Exception ex)
  {
    Console.WriteLine(ex.Message);
   }
Console.ReadKey();
}
```

The output of the above program is as shown here -



Working with Private/Protected members of a Class

In some scenarios, while creating Extension methods you may need to access methods or data members which are not directly available outside the class. For example, Private/Protected members of the class which is being Extended. Let's take a look at the following example –

```
public class DisplayMessage
{
        private DateTime _captureCurrentDate;
        public void SetTime()
        {
                _captureCurrentDate = DateTime.Now;
        }
        public string GetIndiaMessage()
        {
                if (_captureCurrentDate.Hour<12)
                {
                        return "Good Morning!!";
                }
                else
                {
                        return "Good Evening";
                }
        }
}
```

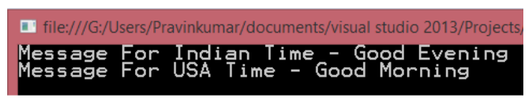
We will extend the DisplayMessage class by calling the private data member _captureCurrentDate into our Extension methods. In this scenario, we will have to read the value of the variable using Reflection. Here's our Extension Method

```
public static class DisplayMessageExtensions
{
  public static string GetCountrySpecificMessage(this
  DisplayMessage message,string country)
  {
      var privateField = typeof(DisplayMessage).
      GetField("_captureCurrentDate", BindingFlags.
                Instance | BindingFlags.NonPublic);
      var currentDateTime = (DateTime)privateField.
                            GetValue(message);
     if (country == "USA" && currentDateTime.Hour < 12)
     {
        return "Good Evening!!";
     }
     else
     {
            return "Good Morning";
     }
  }
}
```

Let's make use of the Extension method we just created -

```
static void Main(string[] args)
{
  try
  {
    DisplayMessage displayMsg = new DisplayMessage();
    displayMsg.SetTime();
    Console.WriteLine("Message For Indian Time - {0}",
    displayMsg.GetIndiaMessage());
    Console.WriteLine("Message For USA Time - {0}",
    displayMsg.GetCountrySpecificMessage("USA"));
  }
  catch (Exception ex)
  {
        Console.WriteLine(ex.Message);
  }
  Console.ReadKey();
}
```

The output of the above program is as follows :



EXTENSION METHODS WITH NESTED CLASSES

A Nested class is a class within a class. Nested classes by default are private, but can be made public, protected internal, protected or internal. Here's a sample nested class –

```
public class ParentClass
{
   public string MessageFromParent()
   {
        return "This Message is from Parent!!";
   }
   private class ChildClass
   {
        public string MessageFromChild()
        {
                return "This Message is from Child!!";
        }
   }
}
```

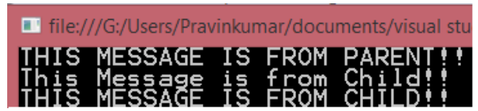
In the code, we have a ParentClass which has a nested class ChildClass declared as private. The question is how to add an Extension method to ChildClass? The answer is to use reflection to extend the nested child class and add an Extension method. The extended class is shown here –

```
public static class NestedClassExtensions
{
  public static string ToUpperCaseParentMessage(this
                                    ParentClass pObj)
  {
    return pObj.MessageFromParent().ToUpper();
  }
  public static string ToUpperCaseChildMessage(this
                                        object o)
  {
      var childUpper = string.Empty;
      var privateClass = typeof(ParentClass).
      GetNestedType("ChildClass", BindingFlags.
                                NonPublic);
      if (o.GetType()==privateClass)
      {
      var callMethod = privateClass.
                    GetMethod("MessageFromChild");
      childUpper = (callMethod.Invoke(o, null)
                                        as string).
                    ToUpper();
      }
      return childUpper;
  }
}
```

In the NestedClassExtensions class, we are using reflection to add an Extension method to our private nested class. We will now use the nested class and access its Extension method in our Main method shown below –

```
static void Main(string[] args)
{
   try
  {
    ParentClass p = new ParentClass();
    Console.WriteLine(p.ToUpperCaseParentMessage());
    var privateClass = typeof(ParentClass).
        GetNestedType("ChildClass", BindingFlags.
                                NonPublic);
    var c = Activator.CreateInstance(privateClass);
    var callMethod = privateClass.
        GetMethod("MessageFromChild");
    Console.WriteLine(callMethod.Invoke(c, null));
    Console.WriteLine(c.ToUpperCaseChildMessage());
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
Console.ReadKey();
}
```

The output of the program is shown here –



The extension method is available under the scope of the namespace in which it is declared. If you want to use these Extension methods in another namespace, you will have to import the namespace.

CREATING AN EXTENSION METHODS LIBRARY

You can also create a separate library of Extension methods which you can reuse into a number of applications as per your requirement. Let's create a standard class library which we will use in various applications. I have added a class library to the project with the name "ExtensionStandardLibrary". Write the following code in our class library –

```
public static class StringExtensions
{
   public static bool FindWord(this string s, string
                  message, string wordToFind)
   {
        return Regex.IsMatch(message,
                                  wordToFind);
   }
}
```

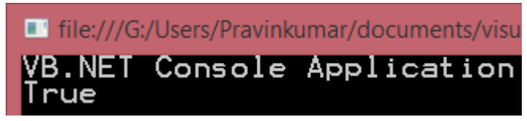
So here we are extending the framework's string type which will find a given word in a string. Now this is a very simple method which I would like to use in various applications. So let's compile the library and try using it in a VB.NET console application.

Using a C# Extension Methods Library in a VB.NET application

Let's add a VB.NET console application to test our C# Extension method library as shown here [Add a reference to the extension method library in our VB.NET application] –

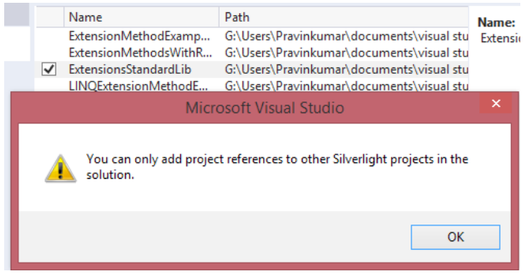
```
Module Module1

Sub Main()
        Console.WriteLine("VB.NET Console Application")
        Dim msg As String = "WelCome To India. India
        is growing as a popular IT Destination for
        big development companies"
        Dim res As String = msg.FindWord(msg, "IT")
        Console.WriteLine(res)
        Console.ReadKey()
End Sub

End Module
```

The output of this VB.NET program is shown here –

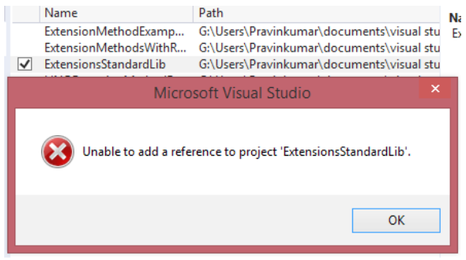


Using the Extension Methods library in a Silverlight and Windows Store application

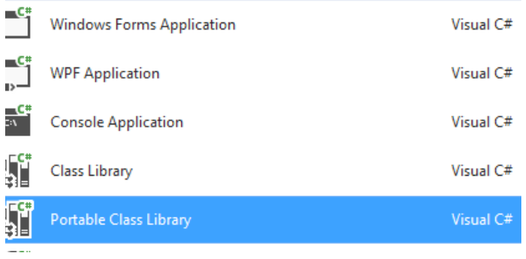
Now let's make use of this Extension library in other applications. For example I would like to use this library in a Silverlight as well as Windows Store Application too. To do so, add the Extension library reference into our projects respectively. When we try adding a reference of the standard extension library, we get the following error –



Similarly try adding the library into a Windows Store App and by doing so, we will receive the following error –



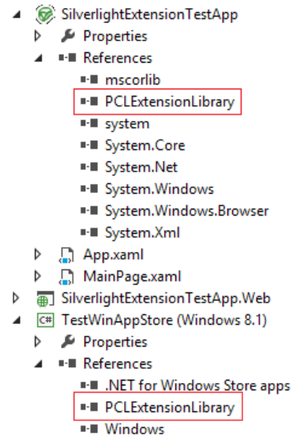
An important point to remember here is that to make use of Extension methods as an library, you cannot make use of a standard class library. Instead, we will have to make use of the Portable Class Library as shown below –



Write the same code in our portable class library as shown here –

```
public static class StringExtensions
{
public static bool FindWord(this string s, string
message, string wordToFind)
  {
    return Regex.IsMatch(message, wordToFind);
  }
}
```

Compile the library and try adding the same into a. NET application as well as into a Silverlight and Windows Store App. Now the library reference will be added to the project as shown here



Let's make use of this library in our Windows Store app. The design of the Windows Store App is as following –

```
<StackPanel Background="Black" Margin="100">
<TextBox Text="Wel Come To India!!" FontSize="20"
Name="txtString"/>
<Button Content="Search Word" Name="btnSearch"
Margin="50" Click="btnSearch_Click"/>
<TextBlock Name="txtResult" FontSize="20"/>
</StackPanel>
```

In the code file, we will first import the library reference and then write some code on the button click event –

```
private void btnSearch_Click(object sender,
RoutedEventArgs e)
{
  string searchText = txtString.Text;
  bool result = searchText.FindWord(searchText,
                                    "India");
  if (result==true)
  {
        txtResult.Text = "Word Match Found";
  }
  else
  {
        txtResult.Text = "Didn't find the word match";
  }
}
```

Run the program and you will see the following output -

EXTEND ASP.NET MVC CLASSES USING EXTENSION METHODS

We will extend an ASP.NET MVC framework class by using Extension methods. We will extend the HTML Helper by adding an Extension method with the name "ReplaceWord". The primary job of this method will be to find out the words which we would like to delete from the content.

We will create an ASP.NET MVC application and then add a class in our ASP.NET MVC application with the name HTMLHelperExtensions and write the method "ReplaceWord" –

```
public static class HTMLHelperExtensions
{
  public static MvcHtmlString ReplaceWord(this
  HtmlHelper htm, string msg, string findWord)
  {
   bool result=Regex.IsMatch(msg, findWord);
   return new MvcHtmlString(msg.Replace(findWord, "<del
   style='background-color:red'>" + findWord + "</
   del>"));
  }
}
```

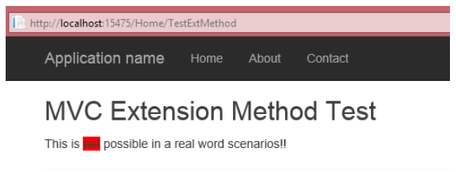
Now we will use this Extension method in our Home Controller. Add an Action method in our Home Controller as shown here –

```
public ActionResult TestExtMethod()
{
  ViewBag.Message = “This is not possible in a real
   word scenarios!!”;
   return View();
}
```

Once you add the action method, we will now create a view for the TestExtMethod. Once the view is ready, we will use our Extension method in the view. To use the Extension method, we will first import the namespace in our view and then code it in the following manner –

```
@using MVCExtensionMethod
@{
    ViewBag.Title = "Test Extension Method";
}

<h2>MVC Extension Method Test</h2>
<p>@Html.ReplaceWord((string)ViewBag.Message,"not")</p>
```

Run your application and see the output. It should look the following –



EXTEND ASP.NET WEB API USING EXTENSION METHODS

Similarly you can use Extension methods to extend the ASP.NET Web API too. Start by creating a new Web API project in Visual Studio 2013. Add a class with the name WebAPIExtensions and write the following code –

```
public static class WebAPIExtensions {
public static void ClientIPHeader(this
HttpResponseMessage response, HttpRequestMessage
request) {
  var httpContextWrapper = request.Properties[“MS_
  HttpContext”] as HttpContextWrapper;
  if (httpContextWrapper != null)  {
    var ipAddress = httpContextWrapper.Request.
                            UserHostAddress;
    response.Headers.Add(“clientIP”, ipAddress);
}
}
}
```

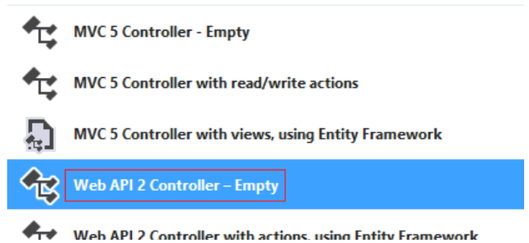
This Extension method adds a client IP address with a response message. Let’s use this Extension method in our Web API. Add a class with the name Customer into our Models folder and write the following code in it

```
public class Customer {
        [Key]
        public int CustomerID { get; set; }
        public string Name { get; set; }
}
```

Install Entity Framework using the NuGet Package in the project. To create a database with the table, we will have to create a data context class as shown here –

```
public class TestEntities:DbContext {
  public TestEntities():base(“Data
  Source=localhost;Initial Catalog=Test;Integrated
  Security=true;”)   {  }
  public DbSet<Customer> Customers { get; set; }
}
```

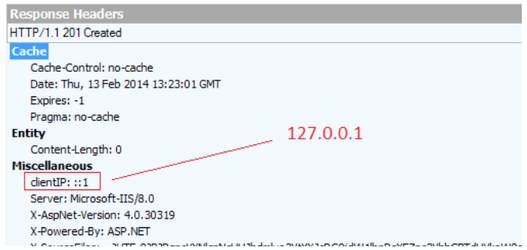
Add a new Web API with the name CustomersController as shown here –



Write the following code in our Customers Controller –

```
public class CustomersController : ApiController {
  TestEntities dataContext = new TestEntities();
  public IEnumerable<Customer> Get()   {
    var query = from cust in dataContext.Customers
    select cust;
    return query.ToList();
  }

public HttpResponseMessage Post([FromBody]Customer c) {
  dataContext.Customers.Add(c);
  dataContext.SaveChanges();
  var res = new HttpResponseMessage(HttpStatusCode.
                                        Created);
   res.ClientIPHeader(Request);
   return res;
}
}
```

I am using Fiddler to inspect the response header to check the client IP address. The output is as shown here –



Likewise you can extend Framework classes, Custom classes and even Third party classes without inheriting them. Extension methods are a powerful means by which you can extend classes, interfaces, collection classes, enums etc.

Extension Method Advantages –

• Extension methods extend the existing type system whether it is a framework class, your own custom class or third party API. You can also use Extension methods to extend sealed classes [Not Inheritable].

• The advantage of using Extension method is that it appears in Visual Studio Intellisense, when you access an object of type which you are extending.

• Extension methods can be used to extend Classes, Interfaces, Collections, Structures, Enum etc.

Extension Method Limitations –

• You cannot write Extension Methods with the same signature. For example you are extending a framework class Object and your team member is also extending the Object class using Extension method. Accidentally if the signature of both the method is the same, then it throws you an ambiguous exception when you will call the method.

• Avoid using Reflection while working with Extension methods. Sometimes you may want to call the non-public members of the class which you are extending using Extension methods, however that’s not possible because Extension methods can only access public members of the class. In such a situation, you may use Reflection. This will work but in case the class members get changed, then there will be an issue while calling the Extension methods. For example you are calling a method Foo() from the class in an extension method using reflection and someone changed the method from Foo to Foo1, it will throw you an exception.

• What if you declare an Extension method which is already declared by the class? In this situation, the compiler will give priority to the method which is declared in the class and not to an Extension method.

• Extension methods are scoped at namespace level. For example, if you are extending an object class using Extension method in ABC namespace, you cannot use this extension method in XYZ namespace. For this, either add an extension method in System namespace or import the namespace. In our case it is ABC.

Extension methods are a special kind of static method, but are called as if they were instance methods on the extended type. Use them sparingly and only where required! ■

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

BUILDING WINDOWS PHONE APPS USING WINDOWS PHONE APP STUDIO

WINDOWS PHONE MVP *VIKRAM PENDSE* INTRODUCES THE WINDOWS PHONE APP STUDIO FOR WINDOWS 8 PHONE & DESKTOP AND WALKS US THROUGH THE FEATURES OF THIS POWERFUL LITTLE TOOL.

Mobile Phones have become nearly ubiquitous in modern life. As developers, we are faced with an exciting opportunity of creating mobile applications for millions of customers using Windows Phone, Google Android or the Apple iOS platform, amongst a few other ones. With Microsoft's pairing with Nokia, developing apps for Windows Phone has become all the more attractive.

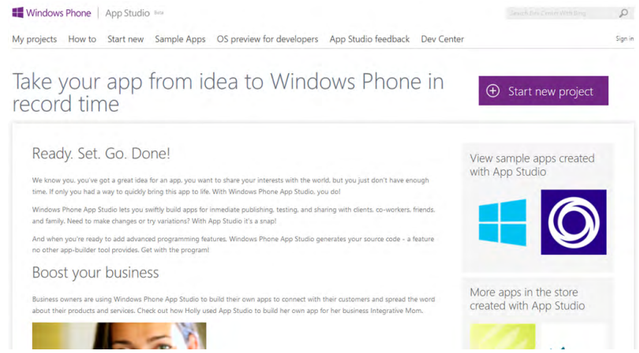
So far, developers have been using Visual Studio and Expression Blend to create Windows Phone apps. Combined these two tools help developers to create mobile apps with great ease. Additionally Microsoft has been taking some huge steps to improve the User and Developer experience on the Windows Phone Platform. Windows Phone App Studio is one such initiative. Currently in Beta, it's an Online Phone Application Building Tool. Another initiative from Microsoft is its recently announced Project Siena which is a Metro App to design Metro Apps. Both these tools have got the windows phone developers excited because of their powerful but easy-to-use features. In this article we are going to discuss more about some new features in the Windows Phone App Studio.

WHAT IS WINDOWS PHONE APP STUDIO?

Windows Phone App Studio is a Web-based tool for potential Developers, Hobbyists and Enthusiast who wish to develop Windows Phone Apps with little efforts. It provides some unique built-in design templates as Start kits based on specific Themes. Users can then further customize these themes according to their needs.

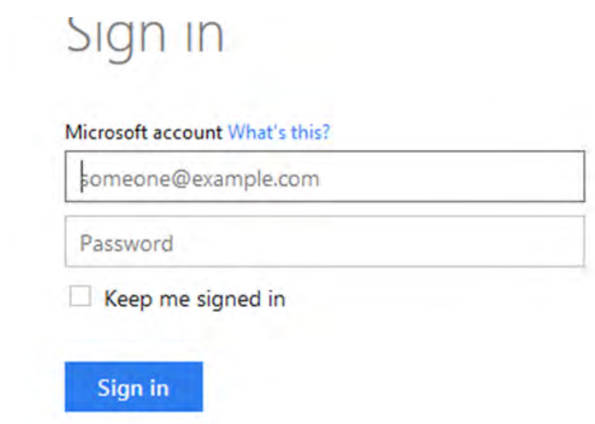
It is a FREE Tool for all. The biggest advantage of the tool is for hobbyists and enthusiasts to create next generation apps without much prior development knowledge. All you need is a valid Windows Live ID, Enroll using it and once it is activated, Login and start building Apps. You can get started over here: http://appstudio.windowsphone.com/

The Portal as shown here gives you couple of options to know more about the Windows Phone Platform and guidelines for Application Development. Another advantage of this tool is that it work across browsers like Internet Explorer, Mozilla Firefox etc. Since this is "Beta" version, there may be couple of Bugs/Issues in the portal, which will eventually get fixed.

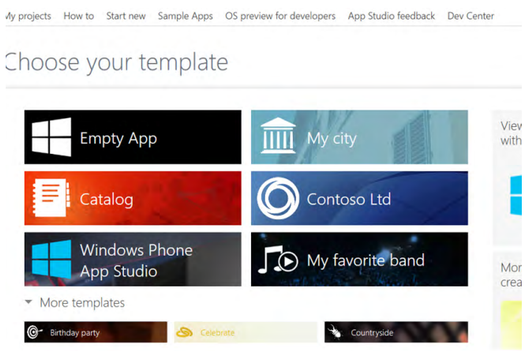


BUILDING AN APPLICATION FOR WINDOWS PHONE USING APP STUDIO

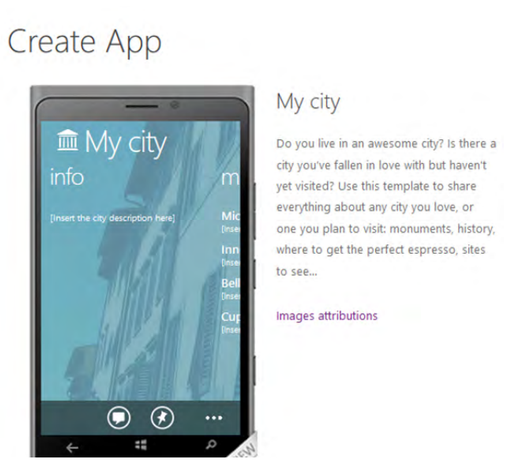
To start building Apps, you need to sign-in with your Live ID and login at http://appstudio.windowsphone.com/



Once you input your Live ID and Password, you will be taken to the Dashboard of App Studio:

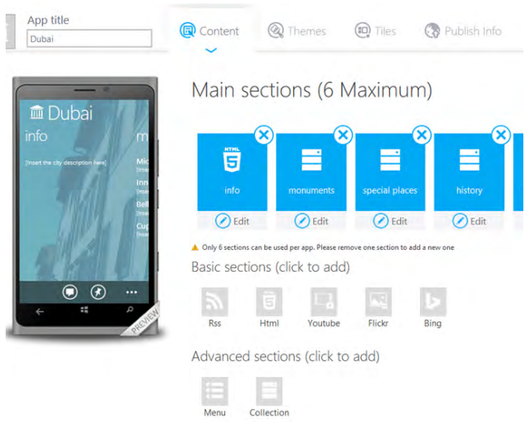


As you can see, your Dashboard enables you to choose a Template for your app. It can be an Empty Template if you wish to customize it yourself and have a total control over it. Alternatively you can choose from "More templates" and pick any predefined template theme and start customizing it as per your needs. For our demo, we will pick the "My City" template and build an App using this template. Once you click on Create, you will see a step by step Wizard. Let us go through each step.



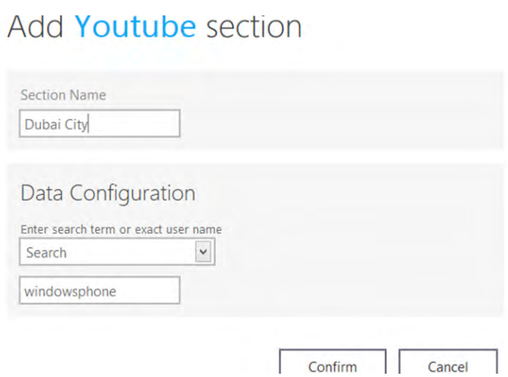
Step 1 : Content

Once you click on Create, the wizard moves to Step 1 "Content". Here you can set the overall content of the Application as shown in the screenshot here.

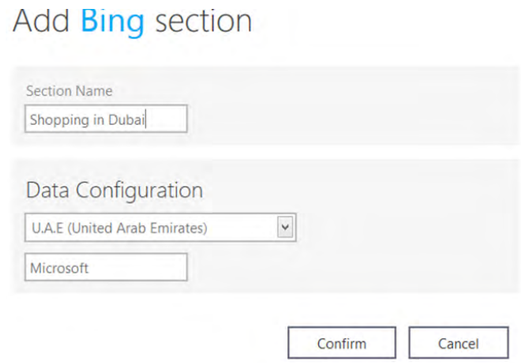


The screen consists of Info Block which can be customized using HTML5 (the first block). There are some other blocks like "monuments", "special Places", "history" etc. You can also add some additional Blocks like "RSS", "Html", "YouTube", "Flicker" and "Bing" to leverage individual services offerings for your App. Here I am considering "Dubai" as City for my "My City" app. You can set the Title in "App title" box on the top left corner and change Images or replace the default images either from local image source or even from the Cloud. We will see this in the next steps.

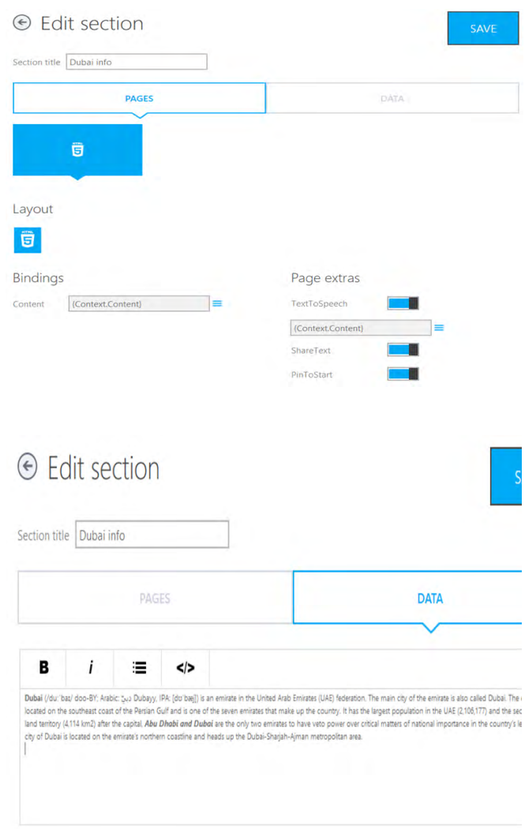
Let's first quickly build some Info and basic structure of the App using the Main Sections. It allows maximum 6 blocks, but for our demo we will reduce to 4. We will also add the "YouTube" and "Bing" Section in our App. The YouTube Videos will be added with the term "Dubai City", so we will configure that as shown here. You can filter YouTube either with a User or Search Term as per your need. We are going for "Search" for this example.



Similarly we will configure the "Bing" section with term "Shopping in Dubai" and we will filter Data Configuration by UAE as default country :



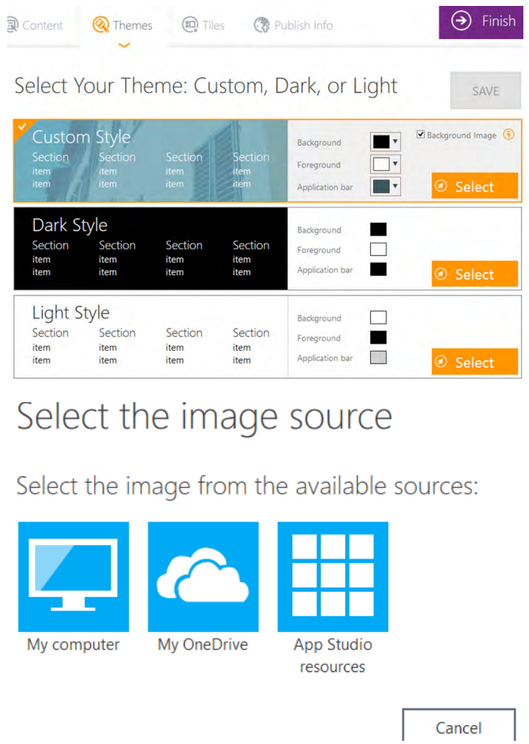
Note: the Edit option helps you to customize the content for an individual Section. We will click it to edit the "info" section followed by "special places" as shown here:



So here we get 2 Sections in Info as "Pages" and "Data". Pages allows you to do Bindings for Content and also helps you to configure "Page extras" like Text To Speech, PinToStart and ShareText. This will appear on the AppBar. Section gives you a kind of RichTextBox which allows you to use your text with some basic Bold, Italic and Bullet Font effects. One you are done, just press "Save" which will take you to next Step.

Step 2: Theme

Themes allows you to change Images and apply different Styles to your content. Here I will be changing the background and Images.

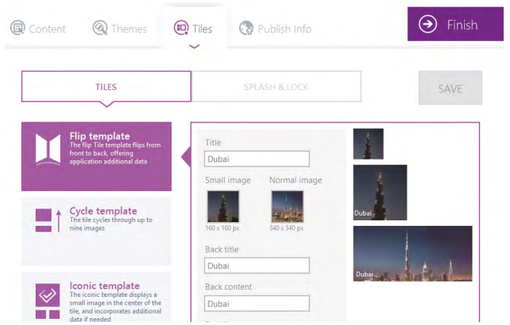


We will select Local Images from my Machine. You can also get images from OneDrive (Earlier it was SkyDrive) and App Studio Resources.

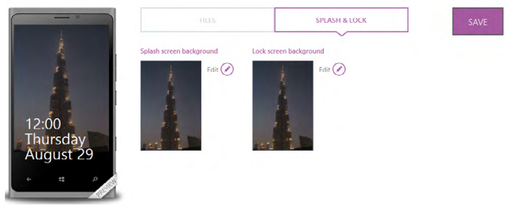
Step 3: Tiles

In Step 3 "Tiles" helps you to complete the Iconography and Lock Screen setup for your app. For those new to Windows Phone, a Tile is an image that represents your app on the Start

screen. Here you have a choice to select The Template for Tiles such as "Flip", "Cycle" and "Iconic". I am going for Flip.



Once you select and configure it, click on Save to move ahead for Splash & Lock.



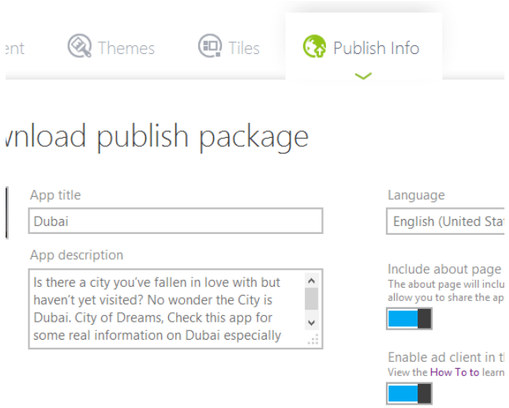
Splash & Lock allows you to configure Splash screen and Lock Screen. Note that the Iconography is as per the Windows Phone Marketplace guidance, so don't worry about size of the icons much. It will even convert your JPEG images to PNG internally as per policy and UI design guidance.

Note: If you need some tips on building Marketplace ready apps, check this article Building Store Ready Apps for Windows Phone.

Step 4: Publish Info

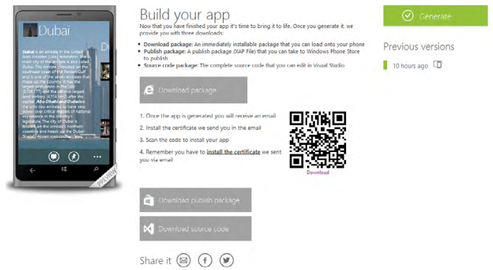
After saving these changes, now we are all set to publish our Application to your phone and eventually to the app store, so the last step is to "Publish Info".

In this final step, we will edit Publish Package information like App Description, Enable Ads, Setting the default App Language etc. as shown here:

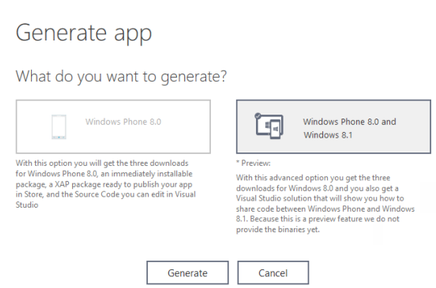


GENERATE SOURCE CODE AND PUBLISH APP

Once you click on "Finish", you will be redirected to the final page of the wizard where you can download the Source Code (Generated by App Studio in the background).

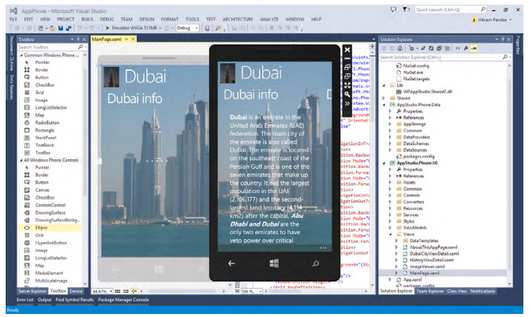


You need to click on "Generate" button to complete the Code Generation Process. By clicking on "Generate", App Studio will ask you whether you want to publish App only for Phone or for both Phone and Windows 8.1 (Desktop), here we will choose for Phone and Windows 8.1 (Desktop)

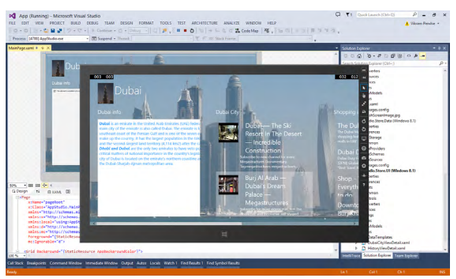


Once you click on "Generate", the wizard will generate a QR Code for you along with "Download Publish Package" and "Download Source Code" to download on your local Machine, if you wish to customize it further . If you scan the QR Code and install the Certificate which you get on your LiveID via email, you will be able to install the Application on your Phone directly. You can also share the Application on social platforms like Facebook and Twitter, Email.

After downloading the Source Code, you can then open it in Visual Studio 2012 / 2013 (It is assumed that your local machine already has the Windows Phone SDK installed and configured). Once you open the Solution (Either Phone or Desktop), you can compile the Source Code and even test in the Emulator. The source code generated by App Studio will be in the Model View ViewModel (MVVM) Pattern. You can further customize the XAML and C# code as per your requirement and publish the Application to Windows Phone Marketplace as per the standard guidance of publishing Windows Phone Apps to the Marketplace.



For Windows 8.1 Desktop you can set the Windows 8.1 Project as Startup Project and Customize and Test it just like the Windows Phone Project



And there you go! This way you can build next generation Apps for Windows Phone using Windows Phone App Studio Beta.

SUMMARY

The Windows Phone App Studio is a unique tool which helps enthusiasts without a programming background, realize their app ideas and get a chance to enter the mobile app ecosystem . Even Developers who are ready to add advanced programming features but wish to have some cool designs complimenting their code, can use this tool and its templates.

Windows Phone App Studio comes with an Interactive built-in emulator which shows real time Content and supports dynamic text updates. Please note that this tool is useful only for Building Apps and Generating Source Code. For publishing Apps to Windows Phone App Store, it is recommended to revisit the Marketplace App Submission guidance provided by Microsoft. It is also highly recommended to use Marketplace Test Kit available in Visual Studio for passing the basic test cases before actual submission of Application to Windows Phone Store.

I hope this article provides you the jumpstart needed to get you to publish apps on the App Store. So what's holding you back? ■

Vikram Pendse is currently working as a Subject Matter Expert in Avanade (Accenture, India) for .NET, Azure & Windows Phone Projects. He is responsible for Providing Estimates, Architecture, Supporting RFPs and Deals. He is Microsoft MVP since year 2008 and currently a Windows Phone MVP. He is a very active member in various Microsoft Communities and participates as a "Speaker" in many events. You can reach him at: vikrampendse@hotmail.com or Follow on Twitter: @VikramPendse

DataBinding Features in WPF 4.5

Mahesh Sabnis explores the DataBinding Features in WPF 4.5 for building maintanable, loosely coupled , Rich Line-Of-Business data-driven applications

Windows Presentation Foundation (WPF) is Microsoft's premier technology for creating a broad range of Windows Desktop Apps. From its very first release 12 years ago (with the code name "Avalon"); WPF has provided several new features to develop robust and rich business applications.

WPF does a very good job in managing the display and editing of complex data using powerful data-binding techniques. Along with Data Binding, the support for Model-View-View-Model (MVVM) in WPF has supported code-less development.

In almost all the types of Business applications, a common and an important requirement is where the UI makes a call to an external service for reading data (in most cases a large amount of data). The call in such scenarios must be handled asynchronously to keep the UI responsive during the process, and once the data is available, it must be automatically synchronized (updated) to the UI.

Another requirement is performing data validations asynchronously, typically when there are more than one validation scenarios to be checked on the data-bound property or where validation has to be performed against complex business rules.

In this article, we will discuss how these requirements can be achieved using two different approaches in WPF 4.5.

AUTOMATICALLY POPULATING THE UI WITH COLLECTIONS ON NON-UI THREADS

Populating the UI using Collections is a very common requirement in Business applications. In .NET 4.0 the *Task* class provided in the Task Parallel Library (TPL) encapsulates all async operations. This class contains the *ContinueWith()* method, which creates a continuation that executes asynchronously when the Task is complete.

Using Task class (the old way)

To demonstrate the old way, a WCF service makes a call to the SQL Server Database using ADO.NET EF. The database table schema is as shown below:

Column Name	Data Type	Allow Nulls
EmpNo	int	☐
EmpName	varchar(50)	☐
Salary	decimal(18, 0)	☐
DeptName	varchar(50)	☐
Designation	varchar(50)	☐
	nchar(10)	☐
		☐

Step 1: Open Visual Studio 2012/2013 and create a blank solution. In this solution, add a WCF Service Application and name it as 'WCF_DataService'. In this service, add an ADO.NET EF data model using the wizard and use the *EmployeeInfo* table shown above.

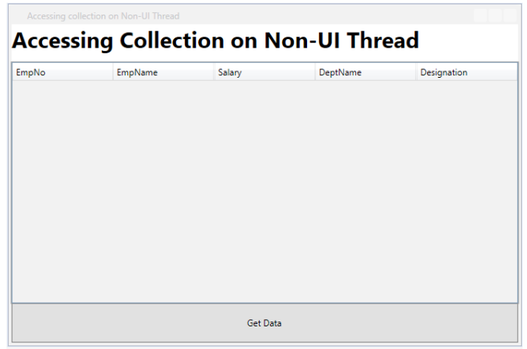
Step 2: Implement the *ServiceContract* interface and the *Service* class as shown here:

```
[ServiceContract]
public interface IService
{
    [OperationContract]
    int CreateEmployee(EmployeeInfo emp);
    [OperationContract]
    EmployeeInfo[] GetEmployees();
}

public class Service: IService
{
    CompanyEntities objContext;
    public Service()
    {
        objContext = new CompanyEntities();
    }
    public int CreateEmployee(EmployeeInfo emp)
    {
        objContext.AddToEmployeeInfoes(emp);
        objContext.SaveChanges();
        return emp.EmpNo;
    }
        public EmployeeInfo[] GetEmployees()
        {
            return objContext.EmployeeInfoes.ToArray();
        }
}
```

Step 3: Build the project and make sure that it is error-free.

Step 4: In the same solution, add a new WPF 4.5 project and call it 'WPF45_DataBinding'. Add the WCF service reference in this project. Delete MainWindows1.xaml and add a new WPF Window and call it 'MainWindows_Update_Collection_UI'.

Step 5: Design the MainWindows_Update_Collection_UI with DataGrid, Button and TextBlock. Set the AutoGeneratedColumns property of the DataGrid to false and define columns for each property from the EmployeeInfo class. The design should be similar to the following:



Step 6: In the code-behind, define an instance of the WCF Proxy as shown here:

```
MyRef.ServiceClient Proxy;
public MainWindows_Update_Collection_UI()
{
        InitializeComponent();
        Proxy = new MyRef.ServiceClient();
}
```

Step 7: In the code-behind, add a method for making a call to the WCF service using the Task class:

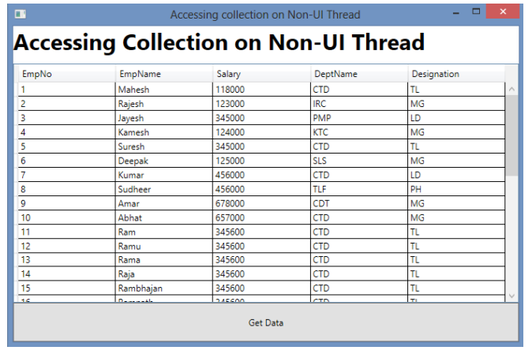
```
/// <summary>
/// The method uses the Task class
/// to update the DataGrid on UI after
/// making call to the WCF Service
/// </summary>
void UpdatingOlderWay()
{
    Task taskGetEmployees =
    Task.Factory.StartNew
    <ObservableCollection<EmployeeInfo>>((obj) =>
    {
        ObservableCollection<EmployeeInfo>
        Employees = new
        ObservableCollection<EmployeeInfo>();
        var Emps = Proxy.GetEmployees();

        foreach (var item in Emps)
        {
            Employees.Add(item);
        }
            return Employees;
    }, CancellationToken.None).ContinueWith(emp =>
        {
            dgemp.ItemsSource = emp.Result;
        },TaskScheduler.
            FromCurrentSynchronizationContext());
}
```

If you observe the code, the Task class makes a call to the *GetEmployees()* method of the WCF service. The return data is then added to the Employees *ObservableCollection* declared locally in the Task call. The *ContinueWith* method then accepts the Employees collection and assigns it to the DataGrid *dgemp*. The *TaskSchedular.FromCurrentSynchronizationContext()* creates a *TaskScheduler* associated with the current synchronization context.

Step 8: Call the above method in the click event of the Get Data button:

```
private void btngetdata_Click(object sender,
RoutedEventArgs e)
{
try
    {
        UpdatingOlderWay();
        //UpdatingUsingWPF45();
        //dgemp.ItemsSource = NewEmployees;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

Step 9: Run the project and click on the 'Get Data' button:



In WPF 4.5, the data is synchronized by accessing the collection using *EnableCollectionSynchronization* method of the *BindingOperations* class. The EnableCollectionSynchronization method enables a collection to be accessed across multiple threads. A specific lock object must be used for synchronizing the access to the collection.

IMPLEMENTING SYNCHRONIZATION WITH NEW TECHNIQUES IN WPF 4.5

Step 1: In the code-behind of the window we just created, declare the collection object and the synchronization lock object as shown here:

```
//Defining the lock object used for Synchronization
private static object lockObject = new object();

//Defining the Collection object to receive data from
external service
ObservableCollection<EmployeeInfo> NewEmployees = new
ObservableCollection<EmployeeInfo>();
```

Step 2: In the constructor of the code-behind, add this code:

```
BindingOperations.
EnableCollectionSynchronization(NewEmployees,
lockObject);
```

Step 3: Then add a new method for making a call to the WCF service:

```
/// <summary>
/// This method will demonstrate the mechanism
/// of accessing the collection on non-UI thread
/// on the UI to Update the UI
/// </summary>
void UpdatingUsingWPF45()
{
    Task taskGetEmployees = Task.Factory.
    StartNew<ObservableCollection<EmployeeInfo>>
    ((obj) =>
    {
        var Emps = Proxy.GetEmployees();
        foreach (var item in Emps)
        {
            NewEmployees.Add(item);
        }
            return NewEmployees;
    },null);
}
```

It is a simple snippet of code to make a call to the WCF service. Once the call is completed, the received data is added into the NewEmployees collection.

Step 4: Add the following code on the click event of the Get Data button,

```
private void btngetdata_Click(object sender,
RoutedEventArgs e)
{
    try
    {
        //UpdatingOlderWay();
        UpdatingUsingWPF45();
        dgemp.ItemsSource = NewEmployees;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

Step 5: Run the application and the result will be exactly the same as we saw in our previous example. The distinctive feature in WPF 4.5 is that the collection source can be shared across threads where the code does not require an explicitly declared thread and it also reduces the coding complexity.

PERFORMING ASYNCHRONOUS DATAVALIDATION

Any user interface that accepts user input has to validate it, to ensure that valid data gets into the back-end. We will discuss how WPF uses the *IDataErrorInfo* interface and the *INotifyErrorDataError* interface that was introduced in the .NET Framework 4.5, to perform validation. The IDataErrorInfo interface provides functionality to implement custom error information on the model object bound to the user interface. The INotifyDataErrorInfo interface is used for performing data validation synchronously and asynchronously. This interface was originally introduced in Silverlight considering its asynchronous behavior for dealing with external objects.

The INotifyDataErrorInfo has the following properties:

- **HasErrors:** This is a read-only property to notify about the validation errors on the object, if it has any.

- **GetErrors:** Returns validation errors for a given property.

- **ErrorChanged:** This event must be raised when a validation error is detected on each property of the source object.

Here's an implementation of the following:

Step 1: In the recently created WPF project, add a new window and name it as 'MainWindows_Asynchronous_DataValidation'.

Step 2: In the WPF project add a new model class with the following code:

```
using System.Threading.Tasks;

namespace WPF45_DataBinding
{

/// <summary>
/// The model class implementing the
INotifyDataErrorInfo interface and
INotifyPropertyChanged
/// </summary>
public class Employee : INotifyDataErrorInfo,
INotifyPropertyChanged
{
    private Dictionary<string,
    List<string>> modelerrors =
    new Dictionary<string, List<string>>();

    public event PropertyChangedEventHandler
PropertyChanged;

    /// <summary>
    /// The property changed
    /// </summary>
    /// <param name="pName"></param>
    void onPropertyChanged(string pName)
    {
        if (PropertyChanged != null)
        {
            PropertyChanged(this,
            new PropertyChangedEventArgs(pName));
            PerformValidation();
        }
    }

    int _EmpNo;

    public int EmpNo
    {
        get { return _EmpNo; }
        set
        {
            _EmpNo = value;
            onPropertyChanged("EmpNo");
        }
    }
    string _EmpName;

    public string EmpName
    {
        get { return _EmpName; }
        set
        {
            _EmpName = value;
            onPropertyChanged("EmpName");
        }
    }
    decimal _Salary;

    public decimal Salary
    {
        get { return _Salary; }
        set
        {
            _Salary = value;
            onPropertyChanged("Salary");
        }
    }
    string _DeptName;

    public string DeptName
    {
        get { return _DeptName; }
        set { _DeptName = value; }
    }
    string _Designation;

    public string Designation
    {
        get { return _Designation; }
        set { _Designation = value; }
    }

    /// <summary>
    /// Method to get errors on a specific property.
    /// </summary>
    /// <param name="propertyName"></param>
    /// <returns></returns>
    public System.Collections.IEnumerable
    GetErrors(string propertyName)
    {
        List<string> errorsForProperties =
        new List<string>();
        if (propertyName != null)
        {
            modelerrors.TryGetValue(propertyName,
            out errorsForProperties);
            return errorsForProperties;
        }
        else
        {
            return null;
        }

    }

    /// <summary>
    /// The Error Info on the object if any
    /// </summary>
    public bool HasErrors
    {
        get {
                try
                {
                    var errInfo = modelerrors.Values.
                    FirstOrDefault(rec => rec.Count >
                    0);
                    if (errInfo != null)
                        return true;
                    else
                        return false;
                    }
                    catch
                    {
                    }
                    return true;            }
    }

    /// <summary>
    /// Perform Validation Asynchronously
    /// </summary>
    private void PerformValidation()
    {
        Task taskasyncvalidate = new Task(() =>
        DataValidation());
        taskasyncvalidate.Start();
    }

    /// <summary>
    /// Logic for Validation
    /// </summary>
    private void DataValidation()
    {
        //Validation for EmpName
        List<string> empNameErrors;
        if (modelerrors.TryGetValue("EmpName",
        out empNameErrors) == false)
        {
            empNameErrors = new List<string>();
        }
        else
        {
            empNameErrors.Clear();
        }

        if (String.IsNullOrEmpty(EmpName))
        {
            empNameErrors.Add("The EmpName can't be
            null or empty.");
        }
        modelerrors["EmpName"] = empNameErrors;

        if (empNameErrors.Count > 0)
        {
            PropertyErrorsChanged("EmpName");
        }
        //Ends Here

        //Validations for Salary

        List<string> salErrors;
        if (modelerrors.TryGetValue("Salary",
        out salErrors) == false)
        {
            salErrors = new List<string>();
        }
        else
        {
            salErrors.Clear();
        }
        if (Salary <= 0 || Salary>70000)
        {
            salErrors.Add("The Salary must be greater
            than zero. and less than 70000");
        }
        else
        {
            salErrors.Clear();
```

```
    }
    modelerrors[“Salary”] = salErrors;
    if (salErrors.Count > 0)
    {
        PropertyErrorsChanged(“Salary”);
    }
    //Ends Here
}

public event
EventHandler<DataErrorsChangedEventArgs>
ErrorsChanged;

/// <summary>
/// The ErrorChaged handler which will be on each
    property.
/// </summary>
/// <param name=”propertyName”></param>

public void PropertyErrorsChanged
(string propertyName)
{
    if (ErrorsChanged != null)
        {
            var evtargs = new
            DataErrorsChangedEventArgs
            (propertyName);
            ErrorsChanged.Invoke(this, evtargs);
        }
    }
}
}
```

- The Employee class contains properties EmpNo, EmpName, Salary, DeptName and Designation.

- The *modelerrors*, the Dictionary<K,V> object is used to store validation errors information for the properties declared in the Employee class.

- The class implements *INotifyDataErrorInfo* and *INotifyPropertyChanged* interfaces.

- The *DataValidation()* method defines logic for the data validation on EmpName and Salary properties. The local List<string> empNameErrors is used to store error messages for the validations on the EmpName property. If there are validation errors on this property, then the error message will be added in the empNameErrors and this list is then passed to the *modelerrors* dictionary for the key as ‘EmpName’. Similarly the validations on the Salary property is declared for Salary less than or equal to zero and greater than 70000.

- The *PerformValidation ()* method uses the Task object to run the *DataValidation ()* method asynchronously.

- The *GetErrors()*, method checks for the validation errors on property in the *modelerrors* dictionary, using *TryGetValue()* method of the dictionary object. This method accepts the property name as key and the value returned will be the list of errors on the property.

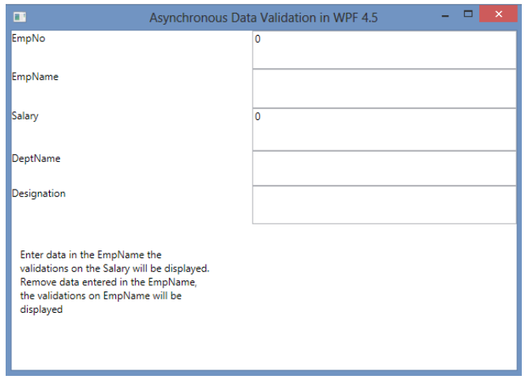
Currently validation logic is written for EmpName and Salary but it can be implemented for rest of the properties as per the requirements.

Step 3: In the MainWindows_Asynchronous_DataValidation add the following xaml:

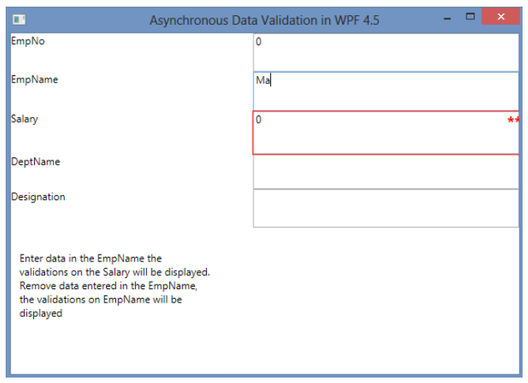
```
<Window x:Class=”WPF45_DataBinding.
    MainWindows_Asynchronous_DataValidation”
    xmlns=”http://schemas.microsoft.com/winfx/2006/
    xaml/presentation”
    xmlns:x=”http://schemas.microsoft.com/
    winfx/2006/xaml”
    xmlns:src=”clr-namespace:WPF45_DataBinding”
    Title=”Asynchronous Data Validation in WPF 4.5”
    Height=”436.917” Width=”610.15”>
    <Window.Resources>
     <src:Employee x:Key=”eds”></src:Employee>
      <Style TargetType=”{x:Type TextBox}”>
      <Setter Property=”Validation.
      ErrorTemplate”>
       <Setter.Value>
        <ControlTemplate>
         <DockPanel LastChildFill=”True”>
           <TextBlock DockPanel.Dock=”Right”
           Foreground=”Red” FontSize=”14pt”
           Margin=”-15,0,0,0” FontWeight=”Bold”>***
           </TextBlock>
           <Border BorderBrush=”Red”
            BorderThickness=”1”>
             <AdornedElementPlaceholder
             Name=”controlWithError”/>
      </Border>
     </DockPanel>
    </ControlTemplate>
   </Setter.Value>
  </Setter>
  <Style.Triggers>
   <Trigger Property=”Validation.HasError”
   Value=”true”>
    <Setter Property=”ToolTip”
    Value=”{Binding RelativeSource={x:Static
    RelativeSource.Self},
      Path=(Validation.Errors)[0].ErrorContent}”/>
   </Trigger>
  </Style.Triggers>
 </Style>
</Window.Resources>

<Grid DataContext=”{Binding Source=
{StaticResource eds}}”>
    <Grid.RowDefinitions>
        <RowDefinition Height=”46*”/>
        <RowDefinition Height=”47*”/>
        <RowDefinition Height=”51*”/>
        <RowDefinition Height=”42*”/>
        <RowDefinition Height=”46*”/>
        <RowDefinition Height=”174*”/>
</Grid.RowDefinitions>
<Grid.ColumnDefinitions>
    <ColumnDefinition Width=”41*”/>
    <ColumnDefinition Width=”45*”/>
</Grid.ColumnDefinitions>

<TextBlock TextWrapping=”Wrap” Text=”EmpNo”/>

<TextBlock Grid.Row=”1” TextWrapping=”Wrap”
Text=”EmpName”/>

<TextBlock Grid.Row=”2” TextWrapping=”Wrap”
Text=”Salary”/>

<TextBlock Grid.Row=”3” TextWrapping=”Wrap”
Text=”DeptName”/>
<TextBlock Grid.Row=”4” TextWrapping=”Wrap”
Text=”Designation”/>

<TextBox Grid.Column=”1” TextWrapping=”Wrap”
Name=”txteno” Text=”{Binding EmpNo}”/>

<TextBox Grid.Column=”1” Grid.Row=”1”
TextWrapping=”Wrap” Name=”txtename”
Text=”{Binding EmpName,Mode=TwoWay,
ValidatesOnNotifyDataErrors=True,
UpdateSourceTrigger=PropertyChanged}”/>

<TextBox Grid.Column=”1” Grid.Row=”2”
TextWrapping=”Wrap” Name=”txtsal”
Text=”{Binding Salary,Mode=TwoWay,
ValidatesOnNotifyDataErrors=True,
UpdateSourceTrigger=PropertyChanged}”/>

<TextBox Grid.Column=”1” Grid.Row=”3”
TextWrapping=”Wrap” Name=”txtdname”
Text=”{Binding DeptName}”/>

<TextBox Grid.Column=”1” Grid.Row=”4”
TextWrapping=”Wrap” Name=”txtdesig”
Text=”{Binding Designation}”/>

<TextBlock HorizontalAlignment=”Left”
Margin=”10,27,0,0” Grid.Row=”5”
Text=”Enter data in the EmpName the validations
on the Salary will be displayed. Remove data
entered in the EmpName, the validations on
EmpName will be displayed”

   VerticalAlignment=”Top” Width=”225”
   Height=”137”
   TextWrapping=”WrapWithOverflow”/>

    </Grid>
</Window>
```

The xaml is bound with the Employee model class. Textboxes are bound with the properties from the Employee class. The validations are checked using ValidatesOnNotifyDataErrors property of the Binding class.

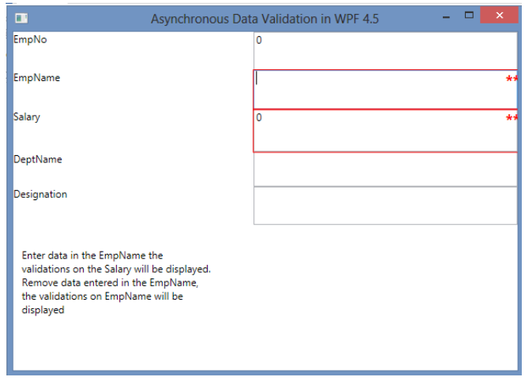
Step 4: Run the application to bring up the following window:



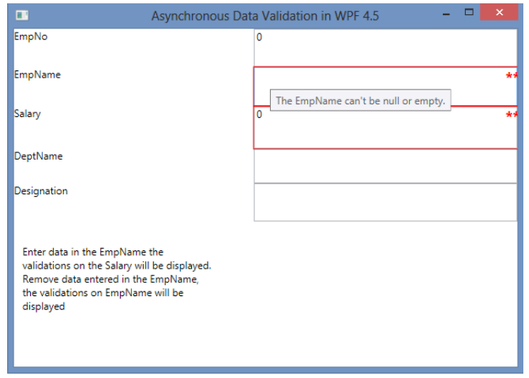
Enter some text in the EmpName TextBox and the validation gets executed on the Salary box as seen here:



Now remove text entered for the EmpName and the validation on the EmpName triggers:



The tootip for the error gets displayed as shown here:



Typically Asynchronous validations can be performed when the input data is to be validated based upon some time consuming logic, like checking for a strong password based on some complex business rules or checking if a user already exist by making a call to the service and so on.

Conclusion

The new DataBinding features provided in WPF 4.5 are very useful for creating Rich Line-Of-Business applications where time and again, the UI needs to be either updated with a collection of data received from external repositories or to perform complex validation checks based upon some complex time consuming logic ■

Download the entire source code from our GitHub Repository at bit.ly/dncm11-wpfdatabinding

Mahesh Sabnis 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