The NeceMoon Album

Technologies and Strategies to Keep Moving Forward

Chapter 7
C# .NET Programming

Featuring Holty Sow

By Necemon Yai

First Edition

TheAlbum.NeceMoon.com

Evasium ®

The NeceMoon Album

Technologies and Strategies to Keep Moving Forward

By Necemon Yai

First edition

Published by Evasium ®

April 2018

London, UK

TheAlbum.NeceMoon.com

© Necemon Yai

necemon@gmail.com

www.necemonyai.com

Version 1.0.7.235

To the ones I lost, to the ones I got back

For each one that gets lost, ten get laid back

Table of Contents

Introduction

About The NeceMoon Album: what is this all about?

The NeceMoon™ is a Blog about Technology and Strategy. The Album is The Best-Of, a compilation of the most popular articles on The NeceMoon™.

The main objective of The NeceMoon™ is to share tips and insights on a sensible range of topics, in order to let others learn from my mistakes and successes, and hopefully to make things easier for the next person. The main objective of The Album is to promote an optimal access to that content. The Blog format does not always do justice to techno-strategic content. Originally, Blogs were designed in a journalistic spirit and are more suitable to chronological events and (more or less trivial) discussions around daily news. Even if the usefulness and the importance of an analysis persist in time, it becomes almost impossible to find and difficult to consult, as more articles keep stacking.
That is why the best articles have been cherry-picked, reviewed and arranged in a logical order that better matches the layout of a book. The Album is free of charge.

The NeceMoon™ can be accessed from NeceMoon.com (or necemonyai.com/blog)
The NeceMoon™ Album can be downloaded in various file formats, in full from TheAlbum.NeceMoon.com (or necemonyai.com/Blog/page/The-Album.aspx). The available formats are PDF, EPUB, MOBI/AZW3/KF8 (Amazon Kindle) and MP3. Furthermore, the various chapters and volumes can be downloaded independently/separately, according to your interests.

About the Author: who is Necemon Yai?

I am a Software Development Engineer extensively involved in Microsoft .NET technologies. Full time developer. Part time digital artist, strategist, essayist and entreprenerd. I majored in computer science from NIIT, Christ University and Swansea University (Master of Engineering, Computing).
At the time of publishing this, I have worked for a Europe leading E-Commerce company, a major UK financial group, the General Electric global corporation and a few tech start-ups that you probably never heard of.
Over the past decade or so, I have been running The NeceMoon blog, where I describe my experiences, my observations and my reflexions. I mostly talk about Technology and Strategy. Here I share my most popular articles.

About the Contributors: who is in your War Council?

I invited the best writers in my network to include some contributions in this book, especially some of their most relevant insights in terms of Technology and Strategy. These top authors are, Ahou The African Chick, Antoine Mian, Cyriac Gbogou, Darren Mart, Edith Brou, Holty Sow, Israel Yoroba, Jean Luc Houedanou, Jean-Patrick Ehouman, Karen Kakou, Monty Oum, Nanda Seye, Nnenna Nwakanma, Olivier Madiba, Vanessa Lecosson and Yehni Djidji.
Along with their respective writings, you can find links to their own web pages. In addition, most of these contributors introduce themselves and provide you with some tactics in our exclusive interviews that you will also find in this book.

About You, Dear Reader: who is this book for? What's in for you?

In The Album, there are 9 chapters organised in 2 volumes. Each chapter deals with a specific topic. You don't have to read everything. If you are interested (to one extend or another) in one or more of these topics, you would possibly appreciate the related chapter(s):

Volume 1: Moon Light (softcore)
- Chapter 1: Strategy and Tactics
- Chapter 2: Digital Marketing and Web Visualisation
- Chapter 3: Corporate Worlds and Emerging Markets
- Chapter 4: Quick Wins, Tricks and Tips
- Chapter 5: Transition - Extra Thoughts and Sharp Fantasy

Volume 2: Full Moon (hardcore)
- Chapter 6: Software Development and Engineering
- Chapter 7: C# .NET Programming
- Chapter 8: Epic Prototypes, Classic Projects, Historic Genre
- Chapter 9: Research and Case Studies

If you want to, you can download and read only the chapter(s) and volume(s) that you are interested in. Several file formats are available on TheAlbum.NeceMoon.com
(or necemonyai.com/blog/page/The-Album.aspx)
All the Web links in this document are working, feel very welcome to click on them.

For All Practical Purposes

This document contains the Chapter 7 of The Album: "C# .NET Programming". If you care, 8 other chapters are also available. Depending on your interests, you may download, (re-)read or share any of the various chapters and volumes independently/separately. The PDF, EPUB, MOBI/AZW3/KF8 (Amazon Kindle) and MP3 formats are available.

To get them, just click on any of the links you like below or go to TheAlbum.NeceMoon.com
(or necemonyai.com/Blog/page/The-Album.aspx)

The Album is available in French as well at Album.NeceMoon.com
(or necemonyai.com/Blog/page/L-Album.aspx)

Chapter 7

C# .NET Programming

Featuring Holty Sow

Why I like C# so much

By Necemon

If you are not into programming, I need to start by telling you that a programming language is basically an artificial language (system of communication) designed to communicate instructions to a machine, typically a computer.

Now there are a lot of programming languages out there. Some are more popular than others, some are more recent, some are more powerful to some extent.

In an ideal world, each programming language serves a specific purpose. So an engineer should be able to adapt to the on going project and choose the optimal technologies. But the truth is, we very often tend to feel comfortable with some languages and find some others kind of painful. It depends on the features of the language and how long we have been using it. It's a bit like natural languages (human languages). You may learn many languages (French, Italian, Spanish, German, Japanese, etc.) and wherever you go, there will be one language that would be more relevant and that you would have to use, but as a native English speaker, you would mostly feel more comfortable speaking English than anything else. If you get in a non English speaking country, you may sure have to adapt to the local language but you would feel some relief when you meet people you can speak English with, as this is what comes naturally to you.

The difference is that you don't choose your main human language. It's generally the language they speak at the place you were born and grew up, the language your parents speak, the language your friends and teachers speak at your school, etc.

Programming languages are a different matter. Programmers do choose to learn a language, even though their motivations may be different. What I want to discuss here is, why and how people choose their programming languages ? What's the best way to go about it ?

From what I observed, there are 2 main reasons people go for a specific language:

- social proof: I guess this concern mostly the beginners, when you want to learn programming for the first time, you don't know much about the languages but you got to start somewhere. So you just go for the language that's popular among your friends, your lecturers/mentors or at your school/college. In short, you go for the languages you are most exposed to or you take advice from people around you. The upside is that you would definitely be surrounded with people who are into the same technologies so they will be able to guide and support you, work along with you on same projects. It's a bit like choosing to buy a specific video game console because all your friends have the same. Suppose you get stuck at some level, there is probably one of your friends that can tell you what to do. Plus you can exchange games with them, discuss game news & cheat codes, enjoy playing together. In short, you become part of the community and it goes pretty much the same when it comes to choosing a language.

- project requirement: throughout their career, developers come across many projects they are compelled to do. The requirements may lead them to learn other languages and technologies, for example, if you work for some company and they assign you to work on that new Python project, you would need to learn Python. Even if you are an independent developer, you may learn another language as it becomes popular with your customers or as it better meets the needs of your users (in terms of speed or user experience for example).

Personally, I think both reasons are valid points. Regarding the first reason, I would just add that it's not about following the trend just for the sake of it. You should make some research to find out which language will help you better do what you are trying to do. Regarding the second reason, I realise it is required to do things that you don't really like but, as often as possible, I would advise you to use the technologies and languages that you like better. If programming is your job, you better enjoy it. Do what you love, really.

Coming to my own experience, I guess I went for C#.NET firstly because it is very popular in the institution where I started getting serious about programming (NIIT). But this is only the reason why I got introduced to C#. There are many other reasons that keep me going.

The main reason being that I find it comfortable but not just because I had been using the most that time. Anyway it wasn't my first programming language. I did program in VB and Pascal before, so it's not that I got stuck to the first language I liked to use. By comfortable, I mean that I enjoy writing C# code. Personally, I would rather write code if I have a lot of fun doing it than write code in some obscur language that's just painful. It's true that we need to worry about things like performance and user experience but I believe that one should enjoy what they are doing.

Now what's so cool about C# ? I won't be discussing how C# compare to other programming languages and whether it's technically better or not. That's not the point of this article and as I said before there is no perfect language, there is just a right language for the right circumstances. I will just say what make C# so awesome to me (and probably to you as soon as you give it a try):

I can't help it, I need to mention that Visual Studio, the work environment for C#, is probably the world best IDE. Automated features like IntelliSense and controls drag and drop save a lot of time and effort. It's not being lazy, it's really about productivity. Whether you work solo or as a team, you always have some important tasks that are not necessarily programming oriented. Automation helps you avoid spending time and effort on the obvious, frequent, basic tasks and focus on the things that are most important.

The language itself is easy to grab, it follows the same kind of syntax that C, C++, Java, etc. So anyone with a similar background can quickly get going. Also, it's known to be simple and elegant.

C# is powerful. It helps me make ANYTHING I want, whether it's a website, a web application, a web service, a smart client application, a game, a windows application, a windows service or a in-browser (Silverlight) application, etc. While most of the languages are used just for a specific purpose, either for the

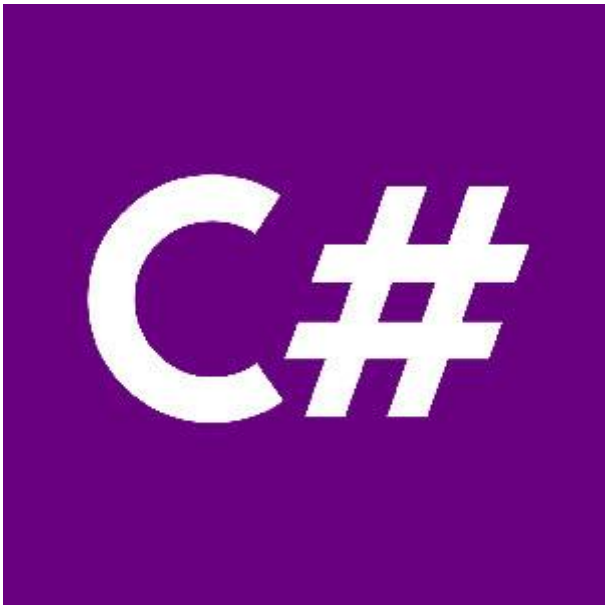
web server, for the client or for the browser, C# does it all. The obvious advantage is that you don't have to learn a new language when you want to start a new kind of application.

Interoperability and language integration: C# applications and services can talk to each other. In fact, they can communicate with any other .NET applications. For example, you can easily make a WPF C# application get data from a WCF VB.NET service and pass it to a Silverlight C# application.

Well, now you know why I like C# so much...

C# Free Learning Resources For Beginners And Professionals

By Necemon



C# (C Sharp) is Microsoft flagship programming language, hence its popularity in the tech industry. Used by a large and growing number of professionals, it helps when building all kinds of applications.

A frequently asked question is, where do I start with C#, what are the best free courses?

Here are a few suggestions.

Microsoft Virtual Academy
C# fundamentals for absolute beginners. Step through 24 practical and easy-to-understand C# training episodes. Tune in to learn the basics of the C# language, and learn to apply them in your programming endeavors, like video games, mobile environments, and client apps.

learncs.org
Free interactive C# tutorial. Whether you are an experienced programmer or not, this website is intended for everyone who wishes to learn the C# programming language. There is no need to download anything, just click on the chapter you wish to begin from, and follow the instructions.

Solo Learn
The best way to learn to code is to code. Gain an understanding of C# concepts by going through short interactive texts and follow-up fun quizzes. Their beautifully designed code editor lets you make changes to existing code or write and run your own custom code and see the output on your mobile device. You can code while going through the core lessons or as a stand-alone learning activity. The more you play, the better you get!

Codeasy
A free interactive online course for learning to program C# language. It is designed for absolute beginners and does not require any prior knowledge to start. It is really fun to learn with Codeasy just by reading an adventure story about fighting machines in the future. While reading, the user meets challenges, which require real coding to solve. User can write code directly on the website. The course consists of chapters, like a real book. Each chapter has several tasks to solve by coding. The final goal is to become a programmer and to save the world.

Visual Studio Dev Essentials
Free tools and free training. Free access to technical training from industry leaders such as Pluralsight, Wintellect, and Xamarin University.

Channel 9
A Microsoft community site for Microsoft customers. It hosts video channels, discussions, podcasts, screencasts and interviews. This includes courses like C# Fundamentals for Absolute Beginners.

Bonus : Top Youtube playlists and tutorials on C#
Youtube 1
Youtube 2
Youtube 3
Youtube 4
Youtube 5

How to Track Mysterious Bugs with Visual Studio

By Necemon

Debugging is the process of finding and resolving defects or problems within the program that prevent correct operation of computer software or a system. Debugging tactics can involve interactive debugging, control flow analysis, unit testing, integration testing, log file analysis, monitoring at the application or system level, memory dumps, and profiling.

Some bugs may be easy to track. I don't want to talk about those. Let's discuss the really mysterious ones.

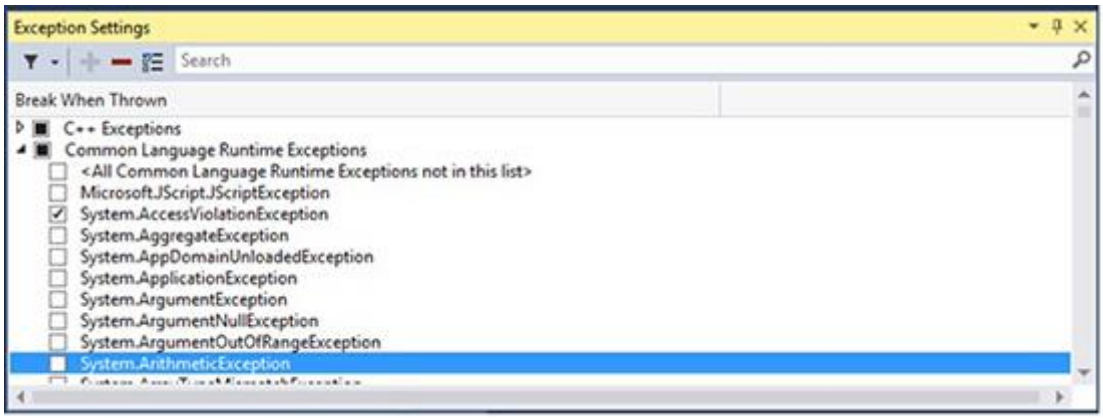
Like the saying goes, Theory is when one knows everything but nothing works; Practice is when everything works but nobody knows why. Sometimes Theory and Practice go hand in hand: nothing works and nobody knows why. In such cases, it's worth remembering about some effective techniques that often allow us to reach the root of the problem and to display its finer details. Over time, here are the 3 techniques that have been most useful when debugging with Visual Studio:

1. Managing relevant exceptions with the Exception Settings Window

An exception is an indication of an error state that occurs while a program is being executed. You can and should provide handlers that respond to the most important exceptions, but it's important to know how to set up the debugger to break for the exceptions you want to see. You can use the Exception Settings window to specify which exceptions (or sets of exceptions) will cause the debugger to break, and at which point you want it to break. You can add or delete exceptions, or specify exceptions to break on. Open this window when a solution is open by clicking **Debug / Windows / Exception Settings**.

You can find specific exceptions by using the Search window in the Exception Settings toolbar, or use search to filter for specific namespaces (for example System.IO). The debugger can break execution at the point where an exception is thrown, giving you a chance to examine the exception before a handler is invoked.

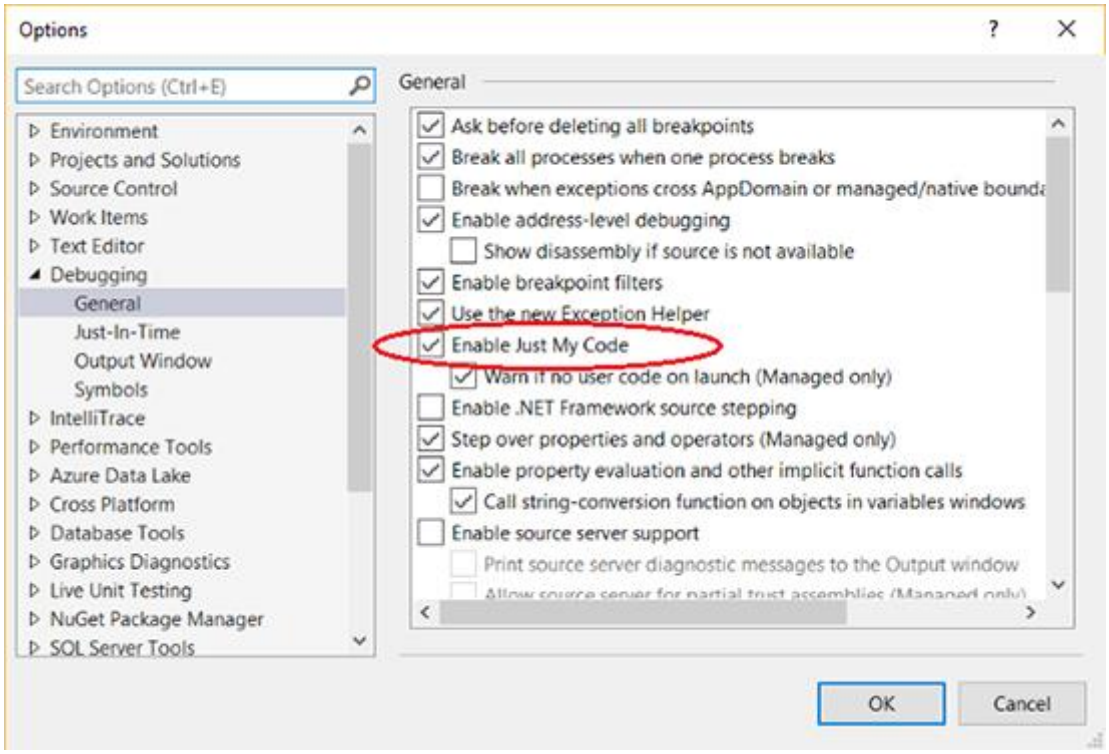
In the Exception Settings window, expand the node for a category of exceptions (for example, Common Language Runtime Exceptions, meaning .NET exceptions), and select the check box for a specific exception within that category (for example System.AccessViolationException). You can also select an entire category of exceptions.



2. Not "Just My Code"

By default, the Visual Studio debugger only breaks on exceptions generated from your own (user) code, hence skipping other system, framework, and other non-user calls. The feature that enables or disables this behavior is called "Just My Code". Depending on what you are debugging, you may want to disable it, because the source or description of the issue might well be outside of "your" code.

To disable (or enable) Just My Code, choose the **Tools > Options** menu in Visual Studio. In the **Debugging > General** node, clear (or choose) **Enable Just My Code**.



3. Recommended tools for Tracing and Error Logging

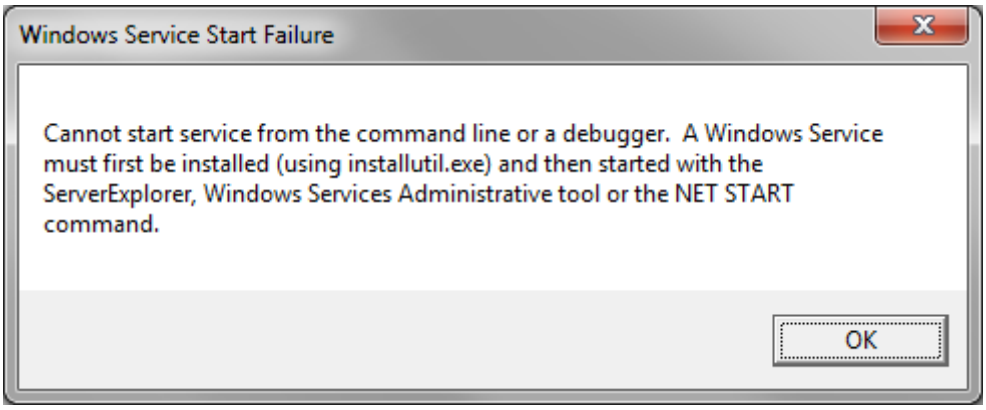
Sometimes you need to record and analyse the full details of the errors, the events and the inner exceptions: that **tracing** involves a specialized use of logging to record information about a program's execution, typically for debugging purposes. Here are my favorite logging and tracing tools:
- Systems.Diagnostics
- Microsoft Entreprise Library
- NLog
- Elmah
- Log4net

Voilà.

How to Debug a Windows Service in Visual Studio

By Holty Sow

When you create a Windows Service project under Visual Studio, you may have noticed that the following dialog box appears when you try to run the service:



In summary, it is simply impossible to run a **Windows Service** in Visual Studio, you must necessarily go through the **NET START** command to start the service after having previously installed it with the command **INSTALLUTIL**. Except that this method prevents us from easily debugging the Windows Service. As a matter of fact, to debug our code, we must install the service, start it, then link the debugger to the relevant service process from Visual Studio. And by the way, let's keep in mind that we will also have to stop, recompile and restart the windows service in order to load up any change we make to the code. In short, it's kinda annoying :D

There is a simpler solution though. We can formulate different compilation directives to detect which mode we are running: **RELEASE** or **DEBUG**. If we are in:
- **DEBUG** mode: we will treat our windows service as a simple client application by displaying a dialog box indicating that the service has started
- **RELEASE** mode: the operation will remain the same as when we tried to run our Windows Service under Visual Studio. In other words, this mode should be used in production, after the debugging session is complete

Our use case will be very simple, we will just be running a Windows service and debugging the WCF service that it hosts. Here are the steps you need to follow:

1. Tweaking the code of the windows service: we will add two methods, **StartWCFService** and **StopWCFService**, which should start and stop listening to WCF incoming requests respectively. The first method will be called in the **OnStart** method definition, and the second one will come from the **OnStop** method. Below is the code:

```
private ServiceHost host;

public MyWindowsService()
{
    InitializeComponent();
}

protected override void OnStart(string[] args)
{
    StartWCFService();
}

protected override void OnStop()
{
    StopWCFService();
}

public void StartWCFService()
{
    host = new ServiceHost(typeof(IWCFService));
    host.Open();
}

public void StopWCFService()
{
    if (host != null && host.State == CommunicationState.Opened)
    host.Close();
}
```

2. Tweaking the **Program.cs** file: it's in the **Main** method that we would detect our selected compilation mode. If we are in **RELEASE** mode, then the windows service is running as usual and if we try to run it in Visual Studio within this mode, we will obviously get the so-called dialog box. If, on the other hand, we are in **DEBUG** mode, then it gets easier, we directly call the **StartWCFService** method of the instance of our windows service (so the **OnStart** method will not be called) and then we display a dialog box to notify the user. When they close the dialog box, the **StopWCFService** method is called to stop the WCF service (therefore, the actual **OnStop**method of the Windows service will not be called).

Here is the code of the **Main** method:

```
static void Main()
{
    MyWindowsService service = new MyWindowsService();
    #if DEBUG
        service.StartWCFService();
        MessageBox.Show("Le service a démarré...");
        service.StopWCFService();
    #else
        ServiceBase[] ServicesToRun;
        ServicesToRun = new ServiceBase[]
        {
            service
        };
        ServiceBase.Run(ServicesToRun);
    #endif
}
```

I hope this post has been helpful ;-)

My Essentials : Top 12 Tech Courses on Pluralsight for .NET Developers

By Necemon

Pluralsight is the largest online tech and creative library on the planet: an online education platform that offers a variety of video training courses for software developers, IT administrators, and creative professionals through its website.

I have been learning quite a bit from Pluralsight, and here are my favorites so far.

Learning Technology in the Information Age

So much to learn, so little time... This course will show both beginners and experts how to learn more in less time.

Would you write code without a design? Build hardware without a schematic? Configure a server without a plan? Of course not. Yet, how many of us learn technology without any planning, choosing resources at random in the hope that one of them will be worthwhile? This is horribly inefficient, and worse, can leave critical holes in your knowledge and skills. In this course you'll learn how to design a plan for learning any technology effectively and efficiently based on your own needs and goals.

Reprogramming the Developer Mind

Explore the habits and career tactics that create remarkable developers. Take control of your career, and set yourself apart from the pack. Become an Outlier.

This course is about making a paradigm shift in how you manage your career. We'll discuss concrete activities and skills that transform average developers into outliers. You'll learn why developers can't afford cable, ways to improve your "luck surface area", and techniques to compress your career through accelerated development. You'll learn the foundational skills for becoming an outlier: command your time, hack your image, and own your trajectory. Prepare to think about your development career in a whole new way.

Encapsulation and SOLID

This course teaches how to write maintainable and flexible object-oriented code. Learn how to write maintainable software that can easily respond to changing requirements using object-oriented design principles. First, you'll learn about the fundamental object-oriented design principle of Encapsulation, and then you'll learn about the five SOLID principles, also known as 'the principles of object-oriented design.' While this course is aimed at beginner to intermediate developers, it's based on decades of experience, so even advanced programmers can learn a thing or two. There are plenty of code examples along the way; while they're written in C#, they should be easily understandable to readers of C, C++ or other curly-brace-based languages.

Date and Time Fundamentals

This course will help you to understand dates and times, and how they should be used in software development.

Managing dates and times properly is one of the most difficult things to get right in software. This is mostly due to how humans have introduced nuance into our calendars and clocks. In this course, you can straighten it all out. You will learn about UTC, daylight saving time, time zones, and calendar systems. You will also learn how date and time values are represented and manipulated in various programming languages. We will look closely at the different kinds of time zone data, and discuss various fallacies and gotchas that are commonly encountered. We will deep dive into how date and time are handled in the .NET Framework, and in JavaScript. We will also look at various libraries that make things slightly more bearable. Throughout the course, you will learn about real-world situations that require deeper thought about how date and time are handled in your applications, and I will give you practical advice on how to solve them.

Web Security and the OWASP Top 10: The Big Picture

Security on the web is becoming an increasingly important topic for organisations to grasp. This course takes you through a very well-structured, evidence-based prioritisation of risks and most importantly, how organisations building software for the web can protect against them.

Visual Studio : Essentials to the Power-User

This course introduces Visual Studio and includes productivity boosters for everyone to make writing and reading code easier and more fun. Visual Studio is an integrated development environment you can use to create applications and libraries with many different frameworks and languages. It has a rich feature set, including an intelligent editor, built in compiler (and related tools), and context sensitive help. This course starts with basic concepts like projects and solutions, shows you how to make Visual Studio look and work the way you want it to, and demonstrates how to use the most popular tool windows and dialogs. It goes further into tips and shortcuts that will save you time every day. Using Visual Studio is about more than writing code or reading code written by others. To be truly productive, you need to debug well and understand the designers that help you build your user interface. This course also shows you how to add helpful extensions that make Visual Studio even better. When you've completed it, you'll know how to use the tool itself and can focus on a specific language or framework as your next step.

Bootstrap 3

Twitter's Bootstrap 3 can help you achieve a great looking and performing web site. Building great looking websites that work well with different sized devices can be a challenge. By utilizing Bootstrap 3 framework, you can meet that challenge head-on. Bootstrap 3 is a mobile-first responsive design framework for structuring your website's HTML. It includes a great grid system, responsive design, CSS typography and components to solve many of the most common design challenges that face web developers today.

Building Your First Xamarin.Android App from Start to Store

Xamarin is a cross-platform development tool. It solves dilemmas many developers face when developing cross-platform apps: separate coding languages and UI paradigms. With Xamarin, you can use C# for iOS, Android, and Universal Windows apps.

In this course, you'll learn how to build a complete, fully-working Xamarin.Android app using C#. Xamarin.Android is covered in a very practical, hands-on way that you can follow along.

More Effective LINQ

Learn how to fully harness the power of LINQ by exploring best practices and avoiding common pitfalls by solving some fun and challenging problems. In this course, you will learn how to take full advantage of the power and capabilities of LINQ. You will see how the LINQ extension methods can be combined together to solve complex problems in a simple and succinct "pipeline." Throughout the course, you will learn to solve some "LINQ Challenges" and pick up lots of pro tips that will take your LINQ skills to the next level, including how to extend, debug, optimize, and test LINQ.

Async C#

In this advanced series, Jon Skeet shows us the new Asynchronous goodness available in C# 5.0. Managing threads, awaiting asynchronous calls - these are all made simple with a few new keywords and types. Asynchronous coding in a static language is not exactly "simple," and rather than try to simplify complex topics, Jon have, instead, decided to go deep and see how things work at a deeper level.

Again, this is an advanced production! If you are new to C# (or new to programming in general), you may want to become familiar with the basics of C# before you tackle this one.

TypeScript Fundamentals

TypeScript is an open source language that provides support for building enterprise scale JavaScript applications. Although several patterns exist that can be used to structure JavaScript, TypeScript provides container functionality that object-oriented developers are familiar with, such as classes and modules. It also supports strongly-typed code to ensure inappropriate values aren't assigned to variables in an application.

In this course, John Papa (link to his site) will walk you through the key concepts and features that you need to know to get started with TypeScript, and use it to build enterprise scale JavaScript applications. You'll learn the role that TypeScript plays as well as key features that will help jump-start the learning process.

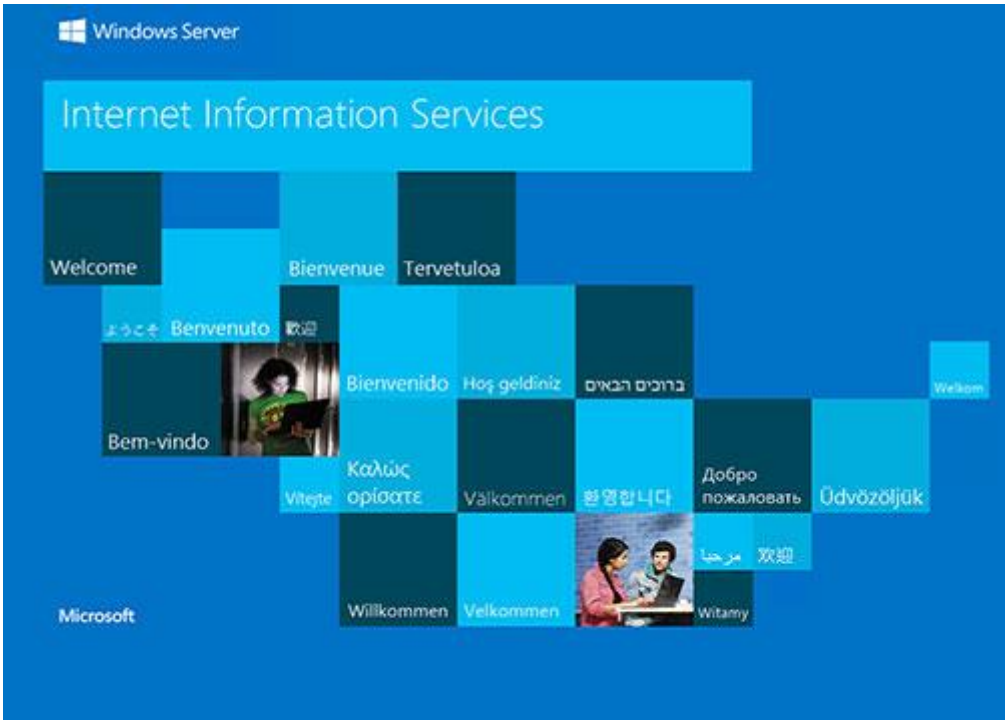
Understanding ASP.NET Core

This course will help you quickly grasp how to get going with ASP.NET Core with a compact, practical presentation, covering all the best new features of ASP.NET since its release.

This latest release of .NET has a lot to be excited about, it not only supports development for a lightweight version of .NET called .NET Core, but for the first time you can also target non-Windows platforms. In this course, Understanding ASP.NET Core, you'll get to see what concepts are most important to get you quickly up to speed. You'll start with seeing what's changed within the project structure and mechanics in Visual Studio. Then, you'll get to see what's new in MVC for ASP.NET Core along with sharing code across existing .NET frameworks and newer .NET Core frameworks. Finally, you'll get to see it's versatility first-hand when an application is deployed to multiple operating systems. By the end of this course, you'll be ready to take advantage of the best new features of this latest release of .NET.

Deploying a Web App to IIS / Windows Server In 7 Clever Steps

By Necemon



Here is a simple deployment routine, for a reliable and smooth release :-)

1. Create/configure the website and its application pool

2. Copy the files over : setup an FTP(S) connection, as well as the client access (FileZilla, Visual Studio, etc.)

3. Assign writing permissions to the application identity on the relevant folders (for logging, files uploads, etc.)

4. Set time outs to appropriate values
Idle Timeout: you can change it from the default of 20 to however many minutes you want. You can also adjust the setting to 0 (zero) which effectively disables the timeout so that the application pool will never shut down due to being idle. This can be configured in the Advanced Settings of the application pool.
ConnectionTimeOut: specifies the time (in seconds) that IIS waits before it stops a connection that is considered inactive. This can be configured in of the Advanced Settings of the Administrative Tools (system.applicationHost/weblimits).

5. Configure Auto-Start
A common problem is the need to perform initialization tasks and "warm up" tasks for a web application. Larger and more complex web applications may need to perform lengthy startup processing, prime in-memory caches, generate content, etc... prior to serving the first HTTP request. One way to fix this is to twist a couple of properties in the application initialization module:
- Set the application pool StartMode property to AlwaysRunning
- Set PreloadEnabled to true, and specify the application pool and path

6. Configure SQL Server / Set up automatic data Backups
Create a maintenance backup plan with SQL Agent

7. HTTPS / SSL Certificate Installation
HTTPS provides security, identity, SEO, access to HTML5 powerful features and more.

Launching Your Application At Windows Startup Without Hacking The Registry

By Holty Sow

In this post, I will show you two methods to configure your .NET applications so that they would launch as soon as Windows starts. These two methods do not require any change to the registry, hence you don't need to worry about cleaning up that database if the user uninstalls your application.

First method:

From the Windows Installer Deployment Project, follow these steps:

In the **File System** of the project, add a special folder called **User's Startup** Folder.

Then a shortcut of the project output is created in the Application Folder and renamed to give it a more meaningful name.

We cut (CRTL + X) the shortcut we just created and paste it (CRTL + V) in the User's Startup Folder.

Second method:

The other solution is to trigger an automatic start from within the code of the application. To do this, I created an activation function, and another one for deactivation.

N.B.: For this code to work, you will need to add a reference to the **Windows Script Host Object Model** assembly in your project.

The activation function can be written as follow:

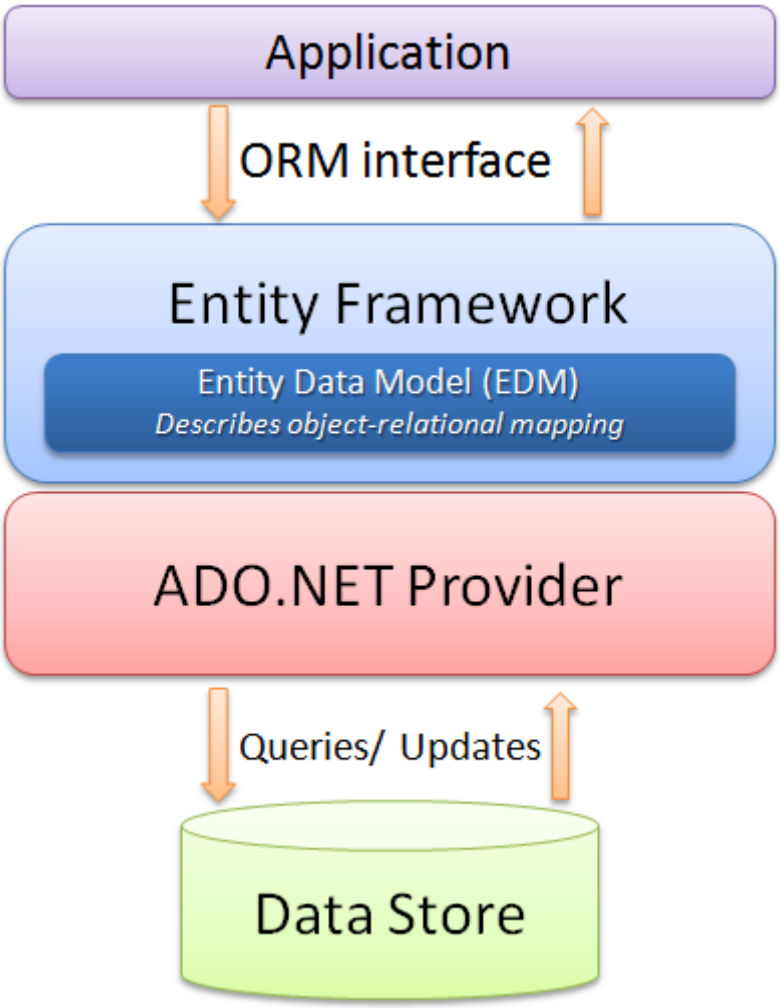
```
public void EnableApplicationStartup()
{
    string shortcutPath =
System.IO.Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Startup),
 "MyShortcut.lnk");
    if (System.IO.File.Exists(shortcutPath)) return;
    WshShell wshShell = new WshShellClass();
    IWshShortcut shortcut = (IWshShortcut)wshShell.CreateShortcut(shortcutPath);
    shortcut.TargetPath = Assembly.GetEntryAssembly().Location;
    shortcut.Description = "My first shortcut";
    shortcut.Save();
}
```

Here is the deactivation function:

```
public void DisableApplicationStartup()
{
    string shortcutPath =
System.IO.Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Startup),
"MyShortcut.lnk");
    if (!System.IO.File.Exists(shortcutPath)) return;
    System.IO.File.Delete(shortcutPath);
}
```

My 12 Favorite Entity Framework Tricks

By Necemon



The Entity Framework is an ORM system, a set of technologies that support the development of data-oriented software applications. It's part of the .NET Framework. I have been playing with it for a couple of years and here are my top tips :

1. Extending the TimeOut value

When loading large amounts of data, the operations happen to fail every time they hit the default time out, so it may be a good idea to extend it to a few hours.
Doing it when initializing the Context object ensures that it's all set before any database call.

2. Second Level Caching

That's a caching of query results. The results of SQL commands are stored in the cache, so that the same SQL commands retrieve their data from the Cache rather than executing the query again against the underlying provider. This can have a performance boost for your application and results in less activity against your database. To enable this feature, you can download and implement this open source project.

3. Dynamic Code First and database migration

EF allows you to program against a model without having to deal with a database directly. With the Code-First approach, you can focus on the domain design and start creating classes. Code-First APIs will create/update the database on the fly based on your entity classes and configuration.

By the way, while we are on the topic on configuration, you can do it all within your C# code. Here are a couple of handy properties:

AutomaticMigrationEnabled : Set it to true to enable the Code-First magic

AutomaticMigrationDataLossAllowed : a value indicating if data loss is acceptable during automatic migration. If set to false an exception will be thrown when data loss may occur as part of an automatic migration.

4. Previewing Code-First change details

There is an easy way to generate the SQL scripts that EF plans to execute, before/without actually commiting those changes. It goes like this:

```
var configuration = new MigrationConfiguration();
var migrator = new DbMigrator(configuration);
var scriptor = new MigratorScriptingDecorator(migrator);
string script = scriptor.ScriptUpdate(null, null);
```

It could be useful if you want to make some changes to that script, or if you are debugging an issue or if you are just curious about what's going on :-)

5. MARS

Multiple Active Result Sets (MARS) is a feature that allows the execution of multiple batches on a single connection. To say it in a simple way, you can make a connection to server and then submit multiple requests to server at the same time, and then read results from these requests in whatever way you want. Just include "MultipleActiveResultSets=True" in your web.config file.

6. Setting all the properties constraints in a single place

For example, instead of adding an attribute to each string property, you could set a default string maximum length as follow:

```
protected override void OnModelCreating(DbModelBuilder modelBuilder)
{
   modelBuilder.Properties<String>().Configure(p => p.HasMaxLength(360));
}
```

7. Inheritance strategy

EF has many ways of dealing with inheritance

-Table per hierarchy : a single table with all based and derived class properties (A discriminator property being used to differentiate between them)

-Table per type : base properties on the base table, and for each derived class, derived properties on a separate table

-Table per concrete type : each derived class gets its own table with all (based or derived) properties.

Pretty extensive, the only case I am not sure about : is there a way to make EF not deal with inheritance at all ? I mean, assuming A derives from B, is there a way to make EF treats A and B as completely different classes and ignore the fact that that one inherits from the other ?

Not a very common scenario, but just for the sake of avoiding duplication, I could want to use inheritance in the C# code so that I don't copy the properties twice in different classes but I wouldn't want to involve any reaction from EF. TPC comes close to that, but the classes still share the same primary key.

8. EntityFunctions Methods

When using LINQ to Entity Framework, your predicates inside the Where clause get translated to SQL, which doesn't have a translation for some complex constructions like DateTime.AddDays(). That's where EntityFunctions methods come in the picture.

9. LINQKit

LINQKit is a free set of extensions for LINQ to SQL and Entity Framework power users. I mostly use it to dynamically build predicates and insert expression variables in subqueries, but it also allow to :

- Plug expressions into EntitySets and EntityCollections
- Combine expressions (have one expression call another)
- Leverage AsExpandable to add your own extensions.

10. Lazy Loading

One of the most interesting Entity Framework features is the Lazy Load function, which is the process whereby an entity or collection of entities is automatically loaded from the database the first time that a property referring to the entity/entities is accessed.

To put it simply, lazy loading means delaying the loading of related data, until you specifically request for it. This could be enabled by setting the Configuration.LazyLoadingEnabled property to true.

11. AsNoTracking

Entity Framework exposes a number of performance tuning options to help you optimise the performance of your applications. One of these tuning options is .AsNoTracking(). This optimisation allows you to tell Entity Framework not to track the results of a query. This means that Entity Framework performs no additional processing or storage of the entities which are returned by the query.

There are significant performance gains to be had by using AsNoTracking().

12. Staying away from performance traps and other EF gotchas

The object context manager can lead to situations where EF behaves in surprising ways. It's good to be aware of the patterns that you can follow to avoid these pitfalls.

ASP.NET MVC: Avoid Polluting The Model Of The View With Error Messages

By Holty Sow

I recently answered a question that came up on the StackOverflow website. Here was the situation.

Let's consider the following model:

```
public class ForgotPasswordMV
{
    [Display(Name = "Enter your email"), Required]
    public string Email { get; set; }
}
```

This model is used in a controller action as follows:

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Search(ForgotPasswordMV viewModel)
{
    if(Temp.Check(viewModel.Email))
        return RedirectToAction("VerifyToken", new { query = viewModel.Email });
    else
    {
        ViewBag.ErrorMessage = "Email not found or matched";
        return View();
    }
}
```

The question was whether using the ViewBag dynamic property of the controller to expose the error message was a good practice, as the OP investigations suggested that it was necessary to expose properties from the model.

Obviously it is highly recommended NOT to use the ViewBag property since it does not provide any Strong Typing. If you want to communicate with the view you should always involve a typed model. The suggested solution is therefore legitimate in that respect, however, it is not a good practice when it comes to model error handling in ASP.NET MVC.

The solution for exposing error messages (as in the previous example, which does not use data annotation attributes) derives from the use of the AddModelError method, from the ModelStateDictionary class. We do not need to instantiate this class because a ModelState property containing an instance of this class already exists in the Controller.

Therefore, the right solution is the following:

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Search(ForgotPasswordMV viewModel)
{
    if
    {
        // ...
    }
    else
    {
        this.ModelState.AddModelError("Email", "Email not found or matched");
        return View(viewModel);
    }
}
```

It should be noted that this method receives as first parameter the name of the property of the model to which the error message is associated. So, in order to display this error message in the view near the Email field, you can simply add the following line next to it:

```
@Html.ValidationMessageFor(m => m.Email)
```

However it is possible to have a general error message for the whole model, that is, an error message which is not attached to any property of the model. For this, it would be necessary to use an empty String as first parameter:

```
ModelState.AddModelError(String.Empty, "Email not found or matched");
```

In the Razor View, use the following line:

```
@Html.ValidationSummary(true, "The following error has occured:")
```

The first Boolean parameter indicates that you do not want to display error messages that are already attached to model properties.

In this post, we saw that we do not need to pollute our model, nor do we need to attach our error messages to specific properties. It's all about using ASP.NET MVC tools to make things easy.

I hope this post has been helpful to you.

Limiting The Execution Of An Action To AJAX Requests Only

By Holty Sow

In ASP.NET MVC, we manipulate Views, among other things. Some of these Views represent full pages and some are just page parts. These parts or areas belonging to a view are called partial views, and they are also returned by controller actions. Since these partial views should only be used within a view, the ASP.NET MVC framework allows us to protect any call to these partial actions by decorating them with the **ChildActionOnly** attribute. This attribute makes sure that the action:
- cannot be used as an entire view and the application developers will always run it using the **HtmlHelper.Action** or **HtmlHelper.RenderAction** methods.
- has a URL that will not be accessible via the address bar, if a user somehow becomes aware of the existence of this URL.

However, as with any dynamic site, we will have AJAX requests that can also make requests for HTML content without having to load the page completely. This content also represents a part, and when we receive the response from the web server we have to embed this piece of HTML somewhere in the page. The AJAX request sent to the server will certainly invoke a controller action. This action, like those marked with the **ChildActionOnly** attribute, must have these constraints:
- should only go through AJAX requests.
- inaccessible via the browser address bar.

But the ASP.NET MVC framework does not offer any attributes that allow us to apply these restrictions to an action, but it gives us the tools to create them. For this, we need to code a filter that will be executed just before the execution of the action in question. If the incoming request complies with the requirements of a request made in AJAX then we let the action continue its way. In case the conditions are not met, we return a 404 page (as if the URL didn't exist).

The code of the new filter that I will call **AjaxOnlyAttribute** (derived from the **ActionFilterAttribute** class) is as follows:

```
[AttributeUsage(AttributeTargets.Class | AttributeTargets.Method, AllowMultiple = false)]
public class AjaxOnlyAttribute : ActionFilterAttribute
{
    public override void OnActionExecuting(ActionExecutingContext filterContext)
    {
        if (filterContext.HttpContext.Request.IsAjaxRequest())
        {
            base.OnActionExecuting(filterContext);
        }
        else
        {
           filterContext.HttpContext.Response.StatusCode = 404;
           filterContext.Result = new HttpNotFoundResult();
        }
    }
}
```

The new class only overrides the method that we are interested in, that is, the **OnActionExecuting** method that is called just before the start of the action invoked by the incoming request. The attribute can be set on the controller to handle the set of actions where an AJAX request is mandatory.

When getting new developers onboard, they might not immediately understand why those requests keep returning **404** messages. To avoid wasting time in recurrent explanations, I think it would be worth having the **DEBUG** mode enabled by default for developers. The code of our class would then look like this:

```
[AttributeUsage(AttributeTargets.Class | AttributeTargets.Method, AllowMultiple = false)]
public class AjaxOnlyAttribute : ActionFilterAttribute
{
    public override void OnActionExecuting(ActionExecutingContext filterContext)
    {
        if (filterContext.HttpContext.Request.IsAjaxRequest())
        {
            base.OnActionExecuting(filterContext);
        }
        else
        {
            #if DEBUG
            filterContext.Result = new ViewResult { ViewName = "AjaxOnly" };
            #else
            filterContext.HttpContext.Response.StatusCode = 404;
            filterContext.Result = new HttpNotFoundResult();
            #endif
        }
    }
}
```

Since developers have to compile the application in **DEBUG** mode most of the time, that **AjaxOnly** View can provide them with some explanations and instructions. Note that this version of the class only works if an **AjaxOnly.cshtml** View has been added to the **Shared Views** directory of the application.

Alternatively, we do not have to return a partial view. We can simply replace the following code:

```
filterContext.Result = new ViewResult { ViewName = "AjaxOnly" };
```

With some simple text content:

```
filterContext.Result = new ContentResult
{
    Content = $"This action '{filterContext.HttpContext.Request.RawUrl}' was designed for
AJAX requests only"
};
```

However, the version that uses an actual View has an advantage, it's the ability to systematically render the relevant page "**_Layout**", typically defined at the application level.

[Interview] In The Bubble Of Holty Sow: "for more efficiency, I prefer to work in an agile team"

1. Quick (but effective) introduction of the personage and her achievements?

My name is Holty Samba SOW. After having started my higher studies in Maths-Physics, then completed a professional degree in Computer Science in Senegal, I went on to further my studies in France and obtained a Master's degree in Electronic Documents and Information Flow in 2009. This allowed me to get into the professional world of computer programming. Currently I am a consultant and developer, mainly on the .NET platform, at SoftFluent, which is a software publisher but also a service company.

2. What are your main goals in life?

At the moment, my main goal is to keep my passion for computer programming for as long as possible by accumulating a maximum of experiences in the Web and Mobile world. I hope to enable my native country, Senegal, to benefit from this wealth of experience.

3. What tools and techniques do you use to accomplish things efficiently?

My favorite IDEs: VS Code for anything related to CSS, JavaScript and TypeScript. For everything else, I use the Visual Studio IDE with the ReSharper extension for more efficiency.
My favorite technologies are generally anything that involves back-end technologies on the .NET platform, like ASP.NET MVC, ASP.NET Web API and recently ASP.NET Core.
The methodology I prefer to work in a team is Scrum/Agile.

4. Any recommendations for your juniors?

A book: "C# in a Nutshell" to thoroughly understand the C# programming language.

Twitter and Blogs: following influencers (tech speakers, CTO, etc.) via Twitter or through their blogs, can help in staying aware of some good stuff. [links]

Websites:
docs.microsoft.com (I think the new Microsoft documentation website is quite well developed);
stackoverflow.com (I think that trying to answer questions from other developers can sometimes help in uncovering tips and tricks that we would not have thought of otherwise);
pluralsight.com (a lot of videos about programming, really interesting... The only catch : there's a charge for it);
channel9.msdn.com (same as Pluralsight, but it's free and the speakers are Microsoft employees).

5. What is the best way to contact you?

Email: holty.sow@gmail.com
Blog : hssow.wordpress.com
LinkedIn : linkedin.com/in/holty-samba-sow-43042715
Twitter: twitter.com/CodeNotFound
Facebook : facebook.com/CodeNotFound

The Xamarin Revolution

By Necemon

What is Xamarin?
Xamarin is a free, cross-platform development tool. It solves dilemmas many developers face when developing cross-platform apps: separate coding languages and UI paradigms. With Xamarin, you can use C# for iOS, Android, and Universal Windows apps. And with Xamarin Forms, interface design for all three platforms can be accomplished within its XAML-based framework.

The Xamarin platform consists of a number of elements that allow you to develop applications for iOS and Android:
C# language – Allows you to use a familiar syntax and sophisticated features like Generics, Linq and the Parallel Task Library.
Mono .NET framework – Provides a cross-platform implementation of the extensive features in Microsoft's .NET framework.
Compiler – Depending on the platform, produces a native app (e.g. iOS) or an integrated .NET application and runtime (e.g. Android). The compiler also performs many optimizations for mobile deployment such as linking away un-used code.
IDE tools – The Xamarin Studio IDE and the Xamarin in Visual Studio allow you to create, build and deploy Xamarin projects.

In addition, because the underlying language is C# with the .NET framework, projects can be structured to share code that can also be deployed to Windows Phone.

Here is some documentation I would recommend if you consider getting started with Xamarin :

Developer Home Page

Cross Platform Guide

10 minutes introduction for Android

Android Guide

10 minutes introduction for iOS

iOS Guide

The Xamarin Show

Xamarin University

Pluralsight courses

Lynda Courses

Top 8 Tricks with Xamarin Programming

By Necemon

If you don't know what Xamarin is, may I suggest that you start by reading my introductory notes before proceeding with this?

For those in the now, let's get into it without further ado:

Xamarin Forms

Xamarin.Forms is a cross-platform UI toolkit that allows developers to easily create native user interface layouts that can be shared across Android, iOS, and Windows Phone. You can build native UIs for iOS, Android and Windows from a single, shared C# codebase.This means that applications can share a large portion of their user interface code and still retain the native look and feel of the target platform. Xamarin.Forms allows for rapid prototyping of applications that can evolve over time to complex applications. Because Xamarin.Forms applications are native applications, they don't have the limitations of other toolkits such as browser sandboxing, limited APIs, or poor performance. Applications written using Xamarin.Forms are able to utilize any of the API's or features of the underlying platform, such as (but not limited to) CoreMotion, PassKit, and StoreKit on iOS; NFC and Google Play Services on Android; and Tiles on Windows. In addition, it's possible to create applications that will have parts of their user interface created with Xamarin.Forms while other parts are created using the native UI toolkit.

Custom Renderers

Xamarin.Forms user interfaces are rendered using the native controls of the target platform, allowing Xamarin.Forms applications to retain the appropriate look and feel for each platform. Custom Renderers let developers override this process to customize the appearance and behavior of Xamarin.Forms controls on each platform.
Custom renderers provide a powerful approach for customizing the appearance and behavior of Xamarin.Forms controls. They can be used for small styling changes or sophisticated platform-specific layout and behavior customization. This article provides an introduction to custom renderers, and outlines the process for creating a custom renderer. Xamarin.Forms Pages, Layouts and Controls present a common API to describe cross-platform mobile user interfaces. Each page, layout, and control is rendered differently on each platform, using a Renderer class that in turn creates a native control (corresponding to the Xamarin.Forms representation), arranges it on the screen, and adds the behavior specified in the shared code.

SQLite

Most applications have some requirement to save data on the device locally. Unless the amount of data is trivially small, this usually requires a database and a data layer in the application to manage database access.
SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects.
iOS and Android both have the SQLite database engine "built in" and access to store and retrieve data is simplified by Xamarin's platform.

ACR User Dialogs

A cross platform library that allows you to call for standard user dialogs from a shared/portable library : Actionsheets, alerts, confirmations, loading, login, progress, prompt, etc.

Xlabs

Xamarin Forms Labs is a open source project that aims to provide a powerful and cross platform set of controls and helpers tailored to work with Xamarin Forms : AutoCompleteView, Calendar Control, ImageButton, HyperLinkLabel, etc.

UI Sleuth

A Xamarin.Forms debugging tool and UI inspector. If you've ever made a web site, it's similar to Microsoft's F12 tools or Chrome Developer Tools. You can use it to efficiently track down layout issues, prototype a new design, and remotely control a device.

GenyMotion Android Emulator

Fast and easy Android emulator. The most powerful Android emulator for app developers and testers. It's available for free or premium on Windows, Mac and Linux.

Remoted iOS Simulator

To test and debug iOS apps within Visual Studio on Windows. Most modern Windows computers have touch screens, and the remote iOS simulator lets you touch the simulator window to test user interactions in your iOS app. This includes pinching, swiping, and multiple-finger touch gestures - things that previously could only be easily tested on physical devices.

Final Reminder

Before we conclude, I just thought I would remind you that this document contains the Chapter 7 of the Album, asnd that 8 other chapters are also available. You may download, (re-)read or share any of the various chapters and volumes independently/separately, depending on your interests. The PDF, EPUB, MOBI/AZW3/KF8 (Amazon Kindle) and MP3 formats are available.

To get them, just click on any of the links you like below or go to TheAlbum.NeceMoon.com
(or necemonyai.com/Blog/page/The-Album.aspx)

The NeceMoon Album (complete)

Volume 1: Moon Light (softcore)

Volume 2: Full Moon (hardcore)

The Album is available in French as well at Album.NeceMoon.com
(or necemonyai.com/Blog/page/L-Album.aspx)

Conclusion

This is Good Bye. However, we can stay in touch. Feel very welcome to add me on LinkedIn, Twitter or Facebook. If you also have any tools and tactics that help you achieve things efficiently, I would like you to tell me about that. My email address is necemon@gmail.com. You are more than welcome to write to me and report any possible mistake in this document, or to suggest any improvement, or to tell me about your favourite parts. However, if you don't like The Album, don't bother writing to me.

Anyways, I wish you all the best for your ongoing and next projects. Thank you for reading. And many thanks to Wikipedia, MSDN, IconFinder and FreeDigitalPhotos for the clarifications and the graphic resources. Special thanks to the Evasium Team. Thanks to all those who contributed, thank you Ahou The African Chick, Antoine Mian, Cyriac Gbogou, Darren Mart, Edith Brou, Holty Sow, Israel Yoroba, Jean Luc Houedanou, Jean-Patrick Ehouman, Karen Kakou, Nanda Seye, Nnenna Nwakanma, Olivier Madiba, Vanessa Lecosson and Yehni Djidji.
Thank you Monty Oum, rest in peace.

Share The Album with Your Mates

Just click on any relevant icon below. It only takes a couple of seconds and it's free for everyone. Alternatively, you can share this link on any social media website or email it to your contacts that would benefit from reading it: http://TheAlbum.NeceMoon.com