The NeceMoon : L'Album

Le Rêve à Temps Plus Que Partiel, le Travail à Temps Carrément Complet

Chapitre 7
Programmation informatique avec C# .NET

Avec la participation de Holty Sow

Par **Necemon Yai**

Première édition

Album.NeceMoon.com

Evasium ®

The NeceMoon : L'Album

Le Rêve à Temps Plus Que Partiel, le Travail à Temps Carrément Complet

Par Necemon Yai

Première édition

Publié par Evasium ®

Avril 2018

Londres, UK

Album.NeceMoon.com

© Necemon Yai

necemon@gmail.com

www.necemonyai.com

Version 1.0.7.186

A celles que j'ai perdues, à celles que j'ai retrouvées

Une de perdue, dix de retournées

Table des Matières

Introduction

A propos de l'Album "The NeceMoon" : ça c'est quoi ça encore ?

The NeceMoon™ est un Blog à propos de Technologie et de Stratégie. L'Album est le Best-Of, une compilation des articles les plus populaires sur The NeceMoon™.

L'objectif principal de The NeceMoon™, c'est de partager des analyses judicieuses sur divers sujets, pour permettre aux autres d'apprendre de mes erreurs et succès, et j'ose espérer, faciliter la tâche à celui ou celle qui entreprend des choses similaires.
L'objectif principal de l'Album est de favoriser un accès optimal à ce contenu. Le format Blog ne rend pas toujours justice au contenu techno-stratégique. A la base, les Blogs ont été conçus dans un esprit journalistique et sont mieux adaptés aux évènements chronologiques et aux discussions (plus ou moins banales) sur l'actualité. Même si l'utilité et l'importance d'une analyse perdurent dans le temps, l'empilement progressif des articles la rend quasiment introuvable et difficilement consultable.
C'est pourquoi les meilleurs articles ont été repris, revus et agencés dans un ordre logique correspondant mieux à celui d'un livre. L'Album est gratuit.

The NeceMoon™ est disponible sur NeceMoon.com (ou necemonyai.com/blog)
L'Album peut être téléchargé en entier dans différents formats à l'adresse Album.NeceMoon.com (ou necemonyai.com/blog/page/L-Album.aspx). Les formats PDF, EPUB, MOBI/AZW3/KF8 (Amazon Kindle) et MP3 y sont disponibles. Aussi, les différents chapitres et volumes peuvent être téléchargés indépendamment/séparément, selon tes intérêts.

A propos de l'Auteur : qui est Necemon Yai ?

Je suis un ingénieur logiciel à fond dans les technologies Microsoft .NET. Développeur à temps complet. Artiste digital, stratégiste, essayiste et entreprenerd à temps partiel. Je me suis spécialisé en informatique depuis NIIT, Christ University et Swansea University (Master en Génie Informatique). A l'heure où j'écris ces mots, j'ai travaillé pour l'une des principales entreprises de E-Commerce en Europe, pour l'un des groupes financiers les plus importants d'Angleterre, pour la multinationale General Electric et pour quelques startups technologiques dont tu n'as probablement jamais entendu parler.
Depuis une dizaine d'années , je tiens le blog "The NeceMoon", où je décris mes expériences, mes observations et mes réflexions. Je parle surtout de Technologie et de Stratégie. Je partage ici mes articles les plus populaires.

A propos des Contributeurs : qui est dans ton Conseil de Guerre ?

J'ai invité les meilleurs auteurs de mon réseau à inclure des contributions dans ce livre, notamment certaines de leurs pensées les plus pertinentes en termes de technologie et de stratégie. Il s'agit en l'occurrence de, Ahou l'Africaine, Antoine Mian, Cyriac Gbogou, Darren Mart, Edith Brou, Holty Sow, Israël Yoroba, Jean Luc Houédanou, Jean-Patrick Ehouman, Karen Kakou, Monty Oum, Nanda Seye, Nnenna Nwakanma, Olivier Madiba, Vanessa Lecosson et Yehni Djidji.
Au niveau de leurs textes respectifs, tu peux retrouver des liens vers leurs pages Web. De plus, la plupart de ces contributeurs et contributrices se révèlent et te livrent quelques tactiques dans des interviews exclusives que tu trouveras aussi dans ce livre.

A propos de Toi, cher lecteur, chère lectrice : à qui s'adresse ce livre ? Qu'est-ce que tu y trouves ?

Dans L'Album, il y a 9 chapitres organisés en 2 volumes. Chaque chapitre aborde un thème différent. Tu n'es pas obligé(e) de tout lire. Si tu es une personne qui s'intéresse de près ou de loin à un ou plusieurs de ces thèmes, tu apprécierais éventuellement le(s) chapitre(s) concerné(s) :

Volume 1 : Clair de Lune (softcore)
- Chapitre 1 : Stratégies et Tactiques
- Chapitre 2 : Marketing Digital et Visualisations Web
- Chapitre 3 : Carrières et Emergence
- Chapitre 4 : Bons Plans et Petites Victoires Faciles
- Chapitre 5 : Dégammage - Délires et Réflexions Rapides

Volume 2 : Pleine Lune (hardcore)
- Chapitre 6 : Génie Logiciel - Quelques Notions Remarquables
- Chapitre 7 : Programmation informatique avec C# .NET
- Chapitre 8 : Prototypes Epiques, Projets Classiques, Genre Historique
- Chapitre 9 : Recherches et Etudes de Cas

Si tu veux, tu peux télécharger et lire uniquement le(s) chapitres ou volume(s) qui t'intéressent.
Plusieurs formats sont disponibles sur Album.NeceMoon.com (ou necemonyai.com/blog/page/L-Album.aspx)
Tous les liens Web de ce document sont fonctionnels, n'hésite pas à cliquer dessus.

A Toutes Fins Utiles

Ce document contient le Chapitre 2 de l'Album : « Programmation informatique avec C# .NET ». Si ça t'intéresse, 8 autres chapitres sont également disponibles. En fonction de tes intérêts, tu peux télécharger, (re)lire et partager chaque chapitre ou volume indépendamment/séparément. Les formats PDF, EPUB, MOBI/AZW3/KF8 (Amazon Kindle) et MP3 sont disponibles.

Il suffit de cliquer sur le(s) document(s) qui t'intéresse(nt) ci-dessous ou d'aller sur Album.NeceMoon.com (ou necemonyai.com/blog/page/L-Album.aspx).

L'Album est également disponible en Anglais à l'adresse TheAlbum.NeceMoon.com (ou necemonyai.com/blog/page/The-Album.aspx).

Chapitre 7

Programmation informatique avec C# .NET

Avec la participation de Holty Sow

Pourquoi j'aime tant C#

Par Necemon

Si tu ne t'y connais pas en programmation, je devrais commencer par te dire qu'un langage, c'est juste un système de communication et un langage de programmation, au fond, c'est un langage artificiel conçu pour communiquer des instructions à une machine, typiquement un ordinateur.

Il existe de nombreux langages de programmations. Certains sont plus populaires que d'autres. Certains sont plus récents, certains sont plus puissants dans une certaine mesure.

Dans un monde idéal, chaque langage de programmation a un but précis. Donc un ingénieur devrait être capable de s'adapter au projet en cours et d'utiliser les technologies optimales pour ce projet. Mais la vérité est que, très souvent, nous avons tendance à nous familiariser avec tel ou tel langage et nous trouvons que certains langages sont pénibles ; ça dépend des caractéristiques du langage et du temps qu'on a passé à les utiliser. C'est un peu comme les langages naturels (les langages humains). Tu peux apprendre plusieurs langues (anglais, italien, espagnol, allemand, japonais, etc.) et où que tu ailles, il y aura un langage qui sera adéquat et que tu devrais utiliser, mais si ta langue de base c'est le français, pour t'exprimer, tu serais toujours plus à l'aise en français. Si tu te retrouves dans un pays où on ne parle pas français, tu devrais bien sûr t'adapter et parler la langue locale, mais en vérité tu serais soulagé d'y rencontrer des francophones et de parler français avec eux, vu que les mots te viennent plus naturellement.

La différence est que tu ne choisis pas ton langage humain de base. C'est généralement la langue que les gens parlent là où tu nais et grandit, le langage que tes parents parlent, le langage que tes amis et tes enseignants parlent dans ton école, etc.

Pour les langages de programmation, c'est différent. Les programmeurs peuvent faire le choix d'apprendre un langage en particulier même si leurs motivations peuvent être différentes. La problématique que je veux relever ici, c'est pourquoi et comment les programmeurs choisissent leurs langages de programmation ? Quelle est la meilleure façon de s'y prendre ?

D'après ce que j'ai observé, il y a 2 raisons principales qui poussent les gens à choisir un langage spécifique:

- Suivre la tendance: je suppose que ce sont surtout les débutants qui le font. Quand tu veux apprendre à programmer au début, tu ne sais pas grand-chose sur les différents langages mais il te faut bien commencer quelque part. Donc souvent, tu commences par le langage qui est populaire dans ton école, parmi tes profs/mentors, ou juste parmi tes amis. Bref, tu choisis les langages auquel tu es le plus exposé ou tu suis juste les conseils de tes proches. L'avantage c'est que tu aurais forcément autour de toi des gens qui se spécialisent dans les mêmes technologies, donc ils seront là pour te guider, te soutenir et vous pourriez travailler sur les mêmes projets. C'est un peu comme choisir d'acheter un jeu vidéo parce que tous tes amis ont le même jeu. Si tu es bloqué à un niveau dans le jeu, il y a probablement un de tes amis qui sait ce qu'il faut faire. En plus, vous pouvez échanger des disques, discuter de l'actualité de jeux, prendre plaisir à jouer ensemble, etc. En somme, tu deviens membre de la communauté et c'est à peu près la même chose quand tu choisis un langage de programmation.

- Exigence du projet: tout au long de leur carrière, les développeurs rencontrent plusieurs projets qu'ils se doivent de terminer. Les exigences de ces projets peuvent les amener à apprendre de nouveaux langages, de nouvelles technologies. Par exemple, si tu travailles pour une compagnie et que tu es affecté sur un nouveau projet en Python, tu aurais besoin d'apprendre Python. D'autant plus que dans certaines SSII, on se retrouve consultant expert dans n'importe quel domaine en quelques heures... Même si tu développes à ton propre compte, tu peux avoir à apprendre un nouveau langage imposé par tes clients ou qui satisfait mieux les besoins de tes utilisateurs (en terme de vitesse ou d'expérience utilisateur par exemple)

Personnellement, je pense que les deux arguments sont valides. Par rapport au premier argument, j'ajouterais juste qu'il ne s'agit pas de suivre la tendance juste pour suivre la tendance. Il faut faire des recherches sur les langages pour savoir quel est celui qui t'arrange le plus. Pour ce qui est de la deuxième raison, je comprends qu'il faille parfois faire des choses qu'on n'aime pas mais je conseillerais d'utiliser les langages et les technologies où l'on se sent le plus à l'aise, tant que c'est possible. Si la programmation doit être ton travail quotidien, c'est mieux de prendre plaisir à le faire. Fais ce que tu aimes, sérieux.

Pour ce qui est de mon expérience personnelle, je suppose que j'ai opté pour C#.NET tout d'abord parce que c'était très populaire dans l'institut ou je me suis sérieusement mis à la programmation (NIIT). Mais ça, c'est juste la raison pour laquelle j'ai commencé C#. Il y a plusieurs autres raisons qui font que je continue de l'utiliser. La raison principale est que je trouve que la programmation sous C# est "confortable", mais au début ce n'était pas parce que c'était le langage que j'utilisais le plus souvent. D'ailleurs C# n'était pas mon premier langage de programmation. J'avais programmé en VB et en Pascal avant ça, donc ce n'est pas comme si je suis resté bloqué sur mon premier langage. Quand je dis "confortable", je veux dire que je prends plaisir à écrire des applications en C#. Je préfèrerais apprécier mes heures de coding plutôt que d'écrire du code dans un langage bizarre que je trouve pénible. C'est vrai qu'il faut se soucier de certains aspects de l'application tels que la performance mais je crois qu'il faut surtout aimer ce qu'on fait.

Mais qu'est ce qui est si cool à propos de C# ? Je ne vais pas comparer C# aux autres langages et affirmer que c'est techniquement mieux que les autres. Ce n'est pas le but de cet article et comme je l'ai dit précédemment, il n'y a pas de langage parfait, il y a juste des langages adéquats pour les circonstances qui se présentent. Ce que je vais faire ici, c'est d'expliquer en quoi C#.NET améliore ma productivité (et probablement la tienne aussi, quand tu voudras bien l'essayer):

Je ne peux pas m'empêcher de mentionner que Visual Studio, l'environnement de travail de C#/.NET est probablement le meilleur IDE au monde. Ces fonctionnalités d'automation telles que l'IntelliSense et le glisser-déposer des contrôles permettent d'économiser beaucoup de temps et d'effort. Ce n'est pas de la paresse, c'est vraiment à propos de productivité. Que tu travailles en solo ou en équipe, tu fais toujours face à d'importantes tâches qui ne sont pas nécessairement liées à la programmation. Les automations t'évitent de perdre du temps sur les tâches évidentes et fréquentes, ce qui te permet de te concentrer sur les choses qui sont vraiment importantes.

Le langage en lui même est facile à apprendre, il suit la même syntaxe que C, C++, Java, etc. Donc quiconque connaît un de ces langages similaires peut s'y retrouver rapidement. Aussi, on dit souvent que C# est simple et élégant.

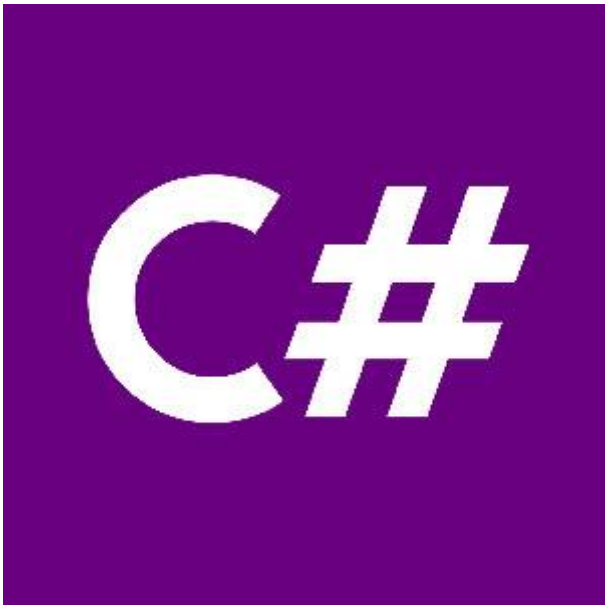
C# est puissant. Il me permet de faire tout ce que je veux, que ce soit un site Web, une application Web, un service Web, une application cliente, un jeu, une application Windows, un service Windows, un service web, une application dans le navigateur (Silverlight), etc. Tandis que la plupart des langages ne peuvent être utilisées que pour un domaine précis, soit le serveur web, soit le navigateur, soit le client, C# peut tout faire. L'avantage évident est que tu n'as pas besoin d'apprendre un nouveau langage quand tu veux commencer un nouveau genre d'application.

Interoperabilité et intégration des langages: les applications et les services C# peuvent aisément parler entre eux. En fait ils peuvent communiquer avec les autres applications .NET. Par exemple, tu peux facilement faire une application C# WPF qui reçoit des données d'un service VB.NET WCF et passer ces données à une application C# Silverlight.

Ok, maintenant tu sais pourquoi j'aime tant C#...

Condensé De Ressources Gratuites En Français Pour Apprendre Et Maitriser C#

Par Necemon



Le C# (Prononcez "cé charpe" à la française ou "ci charpe" à l'anglaise) est le langage de programmation phare de Microsoft, donc un langage très professionnalisant. Utilisé par un nombre important et grandissant de professionnels, il permet de réaliser toutes sortes d'applications.

Une question qui revient souvent est, comment débuter en C#, quels sont les meilleurs cours gratuits pour les francophones ?

Voici quelques suggestions.

Developpez.com
Ce cours est le fruit de plusieurs années d'enseignement en école d'ingénieurs à l'université d'Angers, plus particulièrement dans la formation "Génie des systèmes industriels : automatique et génie informatique". Aussi disponible en format pdf.

Open Classrooms - Apprendre à développer en C#
Ce cours a pour but de t'apprendre les rudiments du langage C# et est ouvert à tous les débutants. D'abord, apprendre comment on crée des applications informatiques et plus particulièrement celles utilisant le Framework .NET; et puis tu pourras te familiariser avec la syntaxe de base du C# pour commencer à créer des applications avec Visual Studio. À la fin de ce cours, tu maîtriseras les bases de la programmation en C# et saura créer une application capable d'interagir avec un utilisateur, de lire ses saisies au clavier et d'afficher des choses à l'écran via une console.

Open Classrooms - Programmer en orienté objet avec C#
Ce cours est la suite du cours "Apprendre à développer en C#", qui présente la syntaxe de base de C# et comment créer des applications simples utilisant le Framework .NET avec Visual Studio.
C'est là que tu peux apprendre les bases de la programmation orientée objet ainsi que la syntaxe C# à utiliser pour créer des classes et manipuler tes objets. Tu découvriras également comment utiliser les différents types du Framework .NET, comment créer des bibliothèques de code réutilisables ou encore comment créer des tests unitaires.
À la fin de ce cours tu sauras quasiment tout ce qu'il faut pour commencer à réaliser des applications d'envergure.

Open Classrooms - Créer sa première application connectée en C#
Découvre comment C# te permet de développer rapidement des applications connectées, entre autres, des applications clientes Windows ainsi que des applications serveurs robustes.

Cours langage C# sur misfu.com
Un panel de documents susceptibles de t'aider à maîtriser la programmation en C Sharp. Certaines formations s'adressent plus particulièrement aux débutants, d'autres sont plus avancées. Ces cours sont accessibles gratuitement et sont téléchargeables au format pdf.

Bonus 1 : quelques exemples de codes et d'applications
Comment ça marche (codes sources)

Bonus 2 : quelques séries de tutoriaux vidéo sur Youtube
Youtube 1
Youtube 2
Youtube 3

Comment Traquer les Bugs Mystérieux avec Visual Studio

Par Necemon

En informatique, un bug (ou bogue) est un défaut de conception (ou d'exécution) d'un programme informatique (ou d'une application, d'un logiciel, etc.), qui est à l'origine d'un dysfonctionnement.

Un débogueur (ou débugueur, de l'anglais debugger) est un logiciel qui aide un développeur à analyser les bogues d'un programme. Pour cela, il permet d'exécuter le programme pas-à-pas, d'afficher la valeur des variables à tout moment, de mettre en place des points d'arrêt sur des conditions ou sur des lignes du programme. Il s'agit de l'application à la programmation informatique du processus de troubleshooting, c'est à dire, un processus de dépannage, de recherche logique et systématique de résolution de problèmes.

Certains bugs sont faciles à traquer. Je ne vais pas m'attarder sur ceux-là. Parlons plutôt de ceux qui sont vraiment mystérieux.

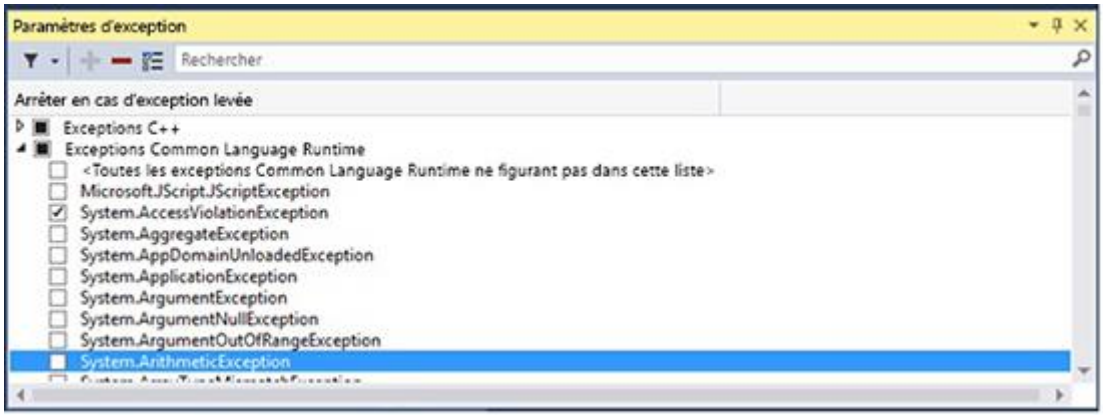
Un vieil adage disait que la Théorie, c'est quand on comprend tout, mais rien ne fonctionne ; la Pratique c'est quand on ne comprend rien, mais tout fonctionne. Mais parfois la Théorie et la Pratique se rencontrent : rien ne fonctionne et personne ne sait pourquoi. Dans ces cas là, il est bon de se rappeler de certaines techniques efficaces qui permettent bien souvent de toucher le fond du problème, et d'en afficher les détails. Au fil du temps, voici les 3 tactiques qui se sont montrées particulièrement utiles quand il s'agit de déboguer avec Visual Studio:

1. Gestion des exceptions pertinentes avec la fenêtre "Paramètres d'exception"

Une exception est une indication d'un état d'erreur qui se produit pendant qu'un programme est en cours d'exécution. Tu peux (et tu dois) fournir des gestionnaires qui répondent aux exceptions les plus importantes, mais il est important de savoir comment configurer le débogueur pour qu'il s'arrête sur les exceptions que tu veux afficher. Tu peux utiliser la fenêtre Paramètres d'exception pour spécifier les exceptions (ou ensembles d'exceptions) qui provoqueront l'arrêt du débogueur, et à quel point tu veux qu'il s'arrête. Tu peux ajouter ou supprimer des exceptions, ou spécifier les exceptions sur lesquelles effectuer un arrêt. Ouvre cette fenêtre lorsqu'une solution est ouverte en cliquant sur **Déboguer/Fenêtres/Paramètres d'exception**.

Tu peux trouver des exceptions spécifiques à l'aide de la fenêtre Rechercher de la barre d'outils Paramètres d'exception ou en utilisant la fonction de recherche pour filtrer des espaces de noms spécifiques (par exemple, System.IO). Le débogueur peut interrompre l'exécution à l'endroit où une exception est levée, ce qui vous permet d'examiner l'exception avant qu'un gestionnaire soit appelé.

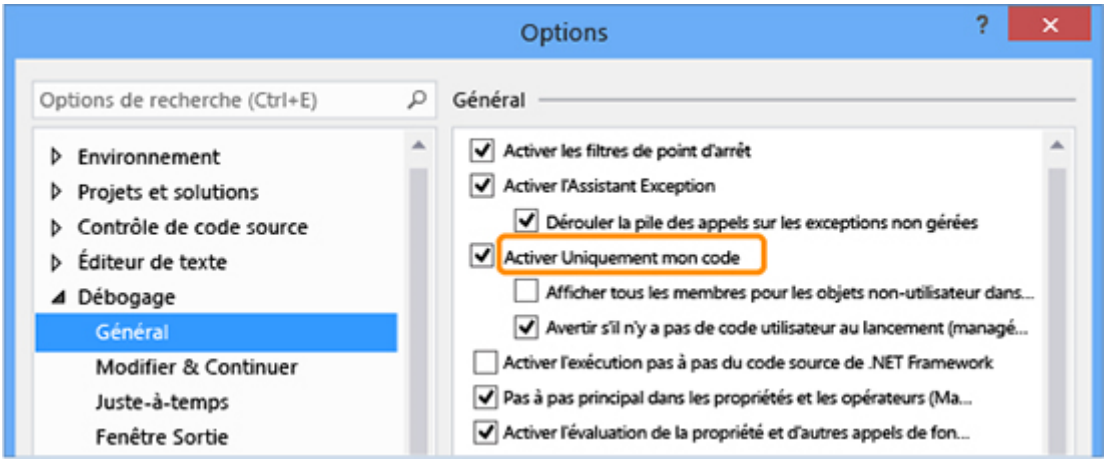
Dans la fenêtre Paramètres d'exception, développe le noeud d'une catégorie d'exceptions (par exemple, Exceptions Common Language Runtime, c'est-à-dire les exceptions .NET), puis coche la case correspondant à une exception spécifique de cette catégorie (par exemple, System.AccessViolationException). Tu peux également sélectionner une catégorie entière d'exceptions.



2. Pas "Uniquement Mon Code"

Par défaut, le débogueur de Visual Studio ne s'arrête que sur les exceptions générées par ton propre code d'utilisateur, survolant ainsi les appels externes et non-utilisateurs, par exemple les appels de systèmes et d'infrastructure. La fonctionnalité qui active ou désactive ce comportement est appelée "Uniquement Mon Code". Selon ce que tu débogues, tu pourrais penser à la désactiver, puisque la source ou la description du problème pourrait bien être située hors de "ton" code.

Pour désactiver (ou activer) Uniquement Mon Code, choisis le menu **Outils > Options** dans Visual Studio. Avec la commande **débogage > général**, annule (ou sélectionne) **Activer Uniquement Mon Code**.



3. Outils recommandés pour le traçage et la journalisation des erreurs

Parfois, il te faut enregistrer et analyser les détails complets des erreurs, des évènements et les séries d'exceptions internes : ce traçage implique un usage spécialisé de journalisation, typiquement pour des raisons de débogage. C'est l'historique, l'enregistrement séquentiel dans un fichier ou une base de données de tous les événements affectant un processus particulier.

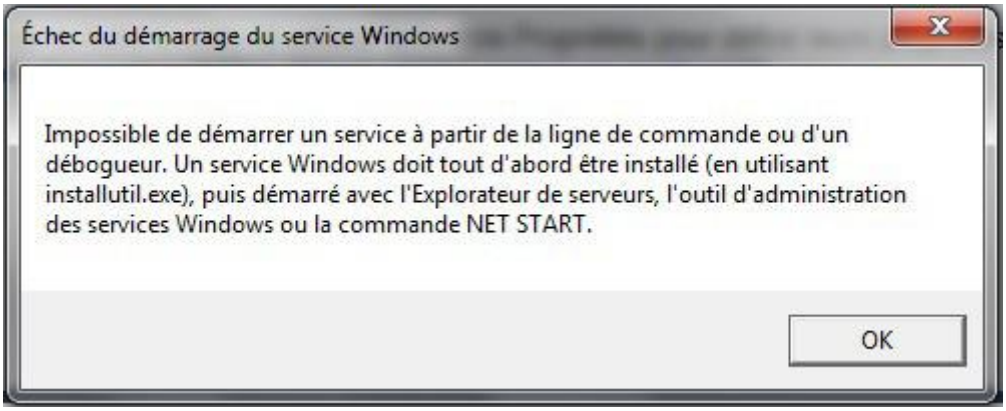
Voici mes outils de journalisation et de traçage préférés:
- Systems.Diagnostics
- Microsoft Entreprise Library
- NLog
- Elmah
- Log4net

Voilà.

Comment Déboguer un Service Windows sous Visual Studio

By Holty Sow

Lorsque vous créez un projet de type **Service Windows** sous Visual Studio, vous avez remarqué que la boîte de dialogue ci-dessous s'affiche quand on essaie d'exécuter le service :



En résumé il est tout simplement impossible d'exécuter un service Windows sous Visual Studio, on doit obligatoirement passer par la commande **NET START** pour démarrer le service après avoir préalablement installé celui-ci grâce à la commande **INSTALLUTIL**. Sauf que cette méthode nous empêche de faire facilement un débogage du service Windows. En effet il faut installer le service, le démarrer, puis dans Visual Studio attacher un processus (qui sera bien sûr le processus du service Windows) pour pouvoir déboguer notre code. Sans oublier qu'il va falloir aussi arrêter le service Windows pour pouvoir compiler suite à quelques modifications qu'on aura effectuées dans le code du service. Bref c'est un peu lourd :D.

Il y a plus simple. On utilisera les symboles de compilation pour détecter dans quel mode nous sommes : **RELEASE** ou **DEBUG**. Si nous sommes en :
- DEBUG : nous allons traiter notre service Windows comme une simple application Windows Forms en affichant une boîte de dialogue indiquant que le service est démarré
- RELEASE : le fonctionnement sera le même que quand nous avons essayé d'exécuter notre service Windows sous Visual Studio. En d'autres termes, ce mode doit être utilisé en production une fois le débogage terminé

Pour notre cas d'utilisation ce sera très simple. Il s'agira ici de rendre possible l'exécution du service Windows et de déboguer un service WCF qu'il héberge. Voici les étapes à suivre :

1. Modification du code du service Windows : nous allons ajouter deux méthodes **StartWCFService** et **StopWCFService** qui ont pour tâches respectives de démarrer et d'arrêter l'écoute des requêtes entrantes WCF. La première méthode sera appelée dans la redéfinition de la méthode **OnStart** et la seconde dans celle de la méthode **OnStop**. Ci-dessous le 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();
}
```

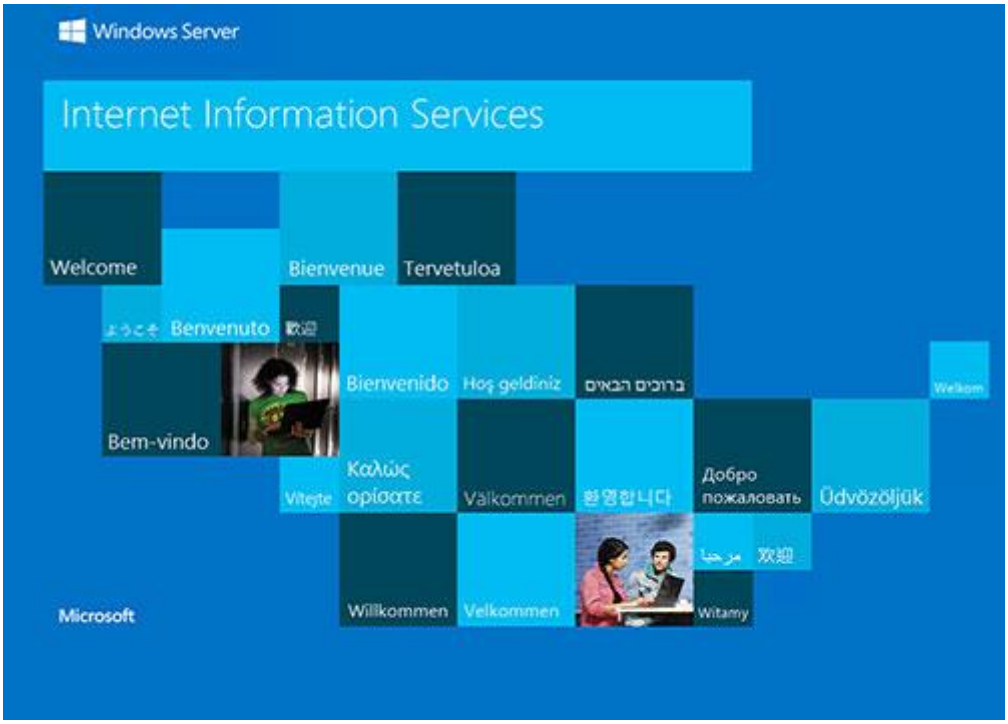
2. Modification du fichier **Program.cs** : c'est dans la méthode **Main** que nous allons faire la détection du mode de compilation dans lequel nous sommes. Si nous sommes en mode **RELEASE**, alors le service Windows s'exécute comme d'habitude et forcément on aura la fameuse boîte de dialogue si on essaie de l'exécuter sous Visual Studio avec ce mode. Si par contre, nous sommes en mode **DEBUG**, alors c'est très simple on fait appel directement à la méthode **StartWCFService** de l'instance de notre service Windows (donc la méthode **OnStart** ne sera pas appelée) puis on affiche une boîte de dialogue pour en informer l'utilisateur. si ce dernier ferme la boîte de dialogue alors la méthode **StopWCFService** est appelée pour arrêter le service WCF (donc la méthode **OnStop** du service Windows ne sera pas appelée).
Ci-dessous le code de la méthode **Main** :

```
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
}
```

J'espère que ce billet vous a été utile :-)

Déployer une application web sur IIS / Windows Server en 7 étapes ingénieuses

Par Necemon



Une routine simple, pour un déploiement robuste et efficace :-)

1. Créer/configurer le site web et son pool d'application

2. Copier les fichiers vers le serveur : installer une connexion FTP(S), ainsi que les accès clients correspondant (FileZilla, Visual Studio, etc.)

3. Attribuer les permissions d'écriture à l'application sur les dossiers appropriés (pour le journal d'erreurs, la sauvegarde des fichiers d'utilisateurs, etc.)

4. Fixer les délais de temps mort
Delai d'inactivité: Tu peux changer la valeur par défaut de 20 au nombre de minutes que tu veux. Tu peux également régler le paramètre à 0 (zéro), ce qui désactive effectivement le délai tel que le pool d'applications ne va jamais s'arrêter à cause de son inactivité. Ceci peut être configuré dans les Paramètres Avancés du pool d'applications.
Delai d'expiration de session: spécifie le temps (en secondes) que IIS attend avant d'interrompre une connexion qui est considérée comme inactive. Ceci peut être configuré dans les Paramètres avancés des Outils d'administration (system.applicationHost / weblimits).

5. Configurer le paramètre Auto-Start
Un problème commun est la nécessité d'effectuer des tâches d'initialisation et l'"échauffement" des tâches pour une application Web. Les applications Web les plus larges et les plus complexes peuvent avoir besoin d'effectuer de longues procédures de démarrage, de mise en mémoire cache, de création de contenu, etc. avant de servir la première requête HTTP. Une façon de résoudre ce problème est d'ajuster quelques propriétés dans le module d'initialisation d'applications :
- Mets la propriété StartMode du pool d'applications à AlwaysRunning
- Mets la propriété PreloadEnabled à True et précise le pool d'application

6. Configurer SQL Server / les sauvegardes automatiques
Créer une procédure de planification de sauvegarde avec l'Assistant Plan de Maintenance

7. Installation de Certifcat HTTPS / SSL
HTTPS améliore la sécurité, l'identification, le SEO, l'accès à certaines fonctionalités avancées de HTML5 et bien d'autres choses.

Lancer son application au Démarrage de Windows sans Bidouiller avec la Base de Registre

Par Holty Sow

Dans ce billet je vais vous présenter deux méthodes permettant de configurer son application .Net pour qu'elle soit lancée au démarrage de Windows. Ces deux méthodes nous évitent de toucher à la base de registre et ainsi nous n'allons pas oublier de nettoyer cette base s'il arrivait que l'utilisateur désinstalle notre application.

La première méthode :

Dans le projet d'installation (projet de déploiement avec Windows Installer) suivre les étapes suivantes :

Dans le **File System (Système de fichiers)** du projet on ajoute un dossier spécial appelé **User's Startup** Folder (Dossier de démarrage de l'utilisateur)

On crée un raccourci de la sortie de projet qui se trouve dans le dossier Application Folder (Dossier de l'application). On renomme ce raccourci pour lui donner un nom plus explicite.

On coupe (CRTL+X) le raccourci qu'on vient de créer et on le colle (CRTL+V) dans le dossier User's Startup Folder.

Deuxième méthode :

Cette solution consiste à rendre configurable le démarrage automatique à partir du code de l'application. Pour cela j'ai créé deux fonctions : une pour l'activation et l'autre pour la désactivation.

N.B. : Pour que le code fonctionne, il faudra ajouter la référence à l'assembly COM Windows Script Host Object Model dans votre projet.

La fonction d'activation est la suivante :

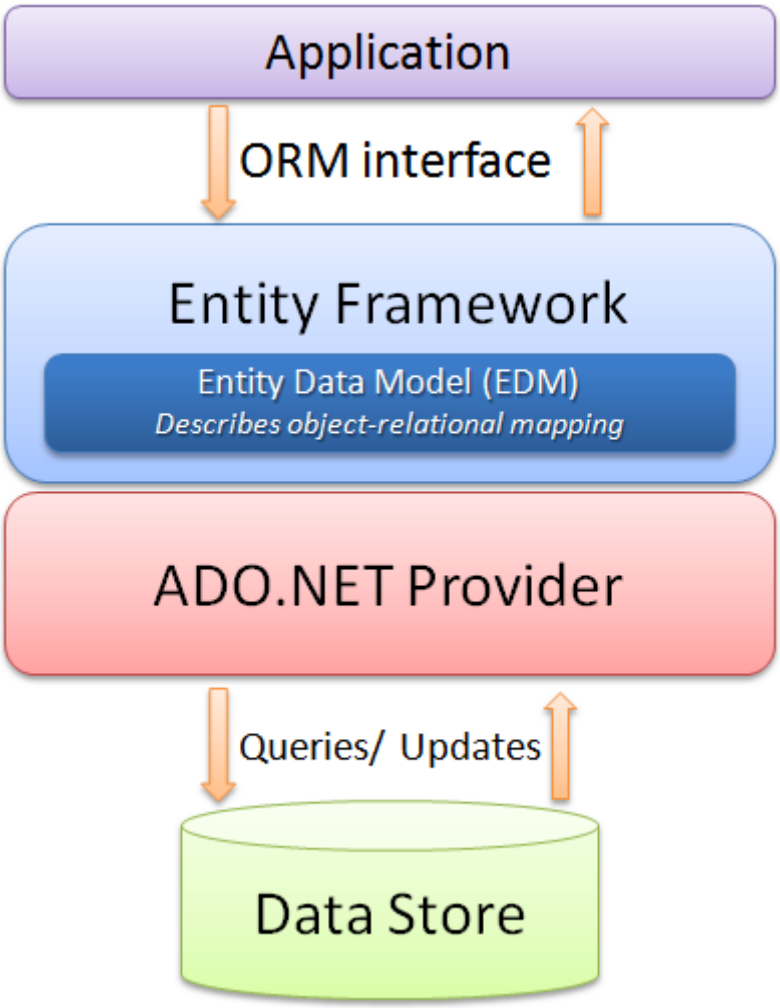
```
public void EnableApplicationStartup()
{
    string shortcutPath =
System.IO.Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Startup),
 "MyShortcut.lnk");
    if (System.IO.File.Exists(shortcutPath)) return;
    WshShell wshShell = new WshShellClass();
    IWshShortcut shortcut =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

La fonction de désactivation :

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

Mes 12 Astuces Préférées Avec Entity Framework

Par Necemon



Entity Framework est un mappeur objet/relationnel qui permet aux développeurs .NET d'utiliser des données relationnelles à l'aide d'objets spécifiques au domaine. Il rend inutile la plupart du code d'accès aux données que les développeurs doivent généralement écrire. Je joue avec ça depuis deux ou trois ans, et voici mes techniques préférées:

1. Prolongement de la valeur TimeOut

Lors du chargement de grandes quantités de données, il peut arriver que certaines opérations échouent chaque fois qu'elles atteignent le temps mort par défaut, du coup ça pourrait être une bonne idée d'étendre cette durée à quelques heures. C'est bien de le faire pendant l'initialisation de l'objet Contexte, donc avant tout appel de base de données.

2. Cache de deuxième niveau
C'est la mise en cache des résultats de la requête. Les résultats de commandes SQL sont stockés dans le cache, de sorte que les commandes répétées récupèrent leurs données à partir du cache au lieu d'exécuter la requête à nouveau contre la base de données. Ceci peut présenter un gain de performances pour l'application et moins d'activités au niveau de la base de données. Pour activer cette fonctionnalité, tu peux télécharger et utiliser ce projet open source.

3. Code First dynamique et migration de bases de données
EF permet de programmer contre un modèle sans avoir à toucher la base de données. Avec la technique Code-First, tu peux te concentrer sur la conception des modèles et commencer à créer des classes. Les APIs de Code-First vont créer et/ou mettre à jour la base de données à la volée en fonction de tes classes d'entités et de ta configuration.
D'ailleurs, pendant qu'on parle de configuration, tu peux faire tout ça depuis ton code C#. Voici quelques propriétés pratiques:
AutomaticMigrationEnabled : pour activer la magie de Code-First
AutomaticMigrationDataLossAllowed : Une valeur indiquant si la perte de données est acceptable lors de la migration automatique. Si la valeur est "false", une exception sera levée quand la migration automatique est susceptible de créer une perte de données.

4. Aperçu et anticipation des changements de Code-First
Ceci est un moyen facile de générer les scripts SQL que EF compte exécuter, sans confirmer ou avant de confirmer ses changements. Voici comment ça marche:
var configuration = new MigrationConfiguration();
var migrator = new DbMigrator(configuration);
var scriptor = new MigratorScriptingDecorator(migrator);
string script = scriptor.ScriptUpdate(null, null);
Ça peut être utile si tu veux faire des changements au script qui va être exécuter, ou si tu es en train de déboguer un problème ou simplement si tu es curieux de savoir ce qui se passe :-)

5. MARS
Multiple Active Result Sets (MARS) est une fonctionnalité qui permet l'exécution de plusieurs paquets sur une seule connexion. Pour le dire d'une manière simple, tu peux établir une connexion au serveur, puis soumettre plusieurs commandes au serveur en même temps, puis lire les résultats de ces demandes de la manière que tu veux. Il suffit d'inclure "MultipleActiveResultSets = true" dans ta ConnectionString.

6. Réglage de toutes les propriétés en un seul endroit
Par exemple, au lieu d'ajouter un attribut à chaque propriété String, tu peux définir une longueur maximale par défaut comme suit:

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

7. Stratégie d'héritage

EF a plusieurs façons de gérer l'héritage

-Table par Hiérarchie : une seule table avec toutes les propriétés des classes de base et des classes dérivées (Une propriété discriminatrice étant utilisée pour les différencier)

-Table par Type : les propriétés de base dans la table de base, et pour chaque classe dérivée, les propriétés dérivées dans une table séparée

-Table Par type concret : chaque classe dérivée obtient sa propre table avec toutes les propriétés (base ou dérivées).

C'est assez complet, le seul cas où je ne suis pas sûr: y a-t-il un moyen de faire en sorte que EF évite complètement l'héritage ? Je veux dire, en supposant que A dérive de B, est-il possible de faire en sorte que EF traite A et B comme des classes complètement différentes et ignorer le fait que l'une hérite de l'autre ? Pas un scénario très commun, mais juste pour éviter la duplication, ce serait alors possible d'utiliser l'héritage dans le code C# sans copier les propriétés deux fois dans des classes différentes, sans réaction de la part EF. TPC se rapproche de cela, mais les classes partagent toujours la même clé primaire.

8. Les Méthodes EntityFunctions

Lorsqu'on utilise LINQ to Entity Framework, tes prédicats dans la proposition Where sont traduits en SQL, mais SQL n'a pas d'équivalent pour certaines constructions complexes telles que DateTime.AddDays(). C'est là que les méthodes EntityFunctions entrent en jeu.

9. LINQKit

LINQKit est un ensemble gratuit d'extensions pour LINQ to SQL et Entity Framework. Je l'utilise surtout pour construire dynamiquement des prédicats et insérer des variables d'expression dans les sous-requêtes, mais il permet aussi de:

- Combiner des expressions (permettre à une expression d'en appeler une autre)
- Insérer des expressions dans les EntitySets et EntityCollections
- Créer tes propres extensions

10. Lazy Loading

L'une des fonctionnalités les plus intéressantes de Entity Framework, c'est la fonction Lazy Loading. C'est le processus par lequel une entité ou une collection d'entités sont automatiquement chargés à partir de la base de données la première fois qu'une propriété faisant référence à l'entité ou aux entités est accessible. Pour le dire plus simplement, Lazy Loading signifie: retarder le chargement des données, jusqu'à ce que tu les appelles explicitement.

Ceci pourrait être activé en activant la propriété Configuration.LazyLoadingEnabled.

11. AsNoTracking

Entity Framework expose un certain nombre d'options d'optimisation des performances pour t'aider à peaufiner tes applications. L'une de ces options de réglage est .AsNoTracking(). Cette optimisation te permet de dire à Entity Framework de ne pas traquer les résultats d'une requête. Cela signifie que Entity Framework n'effectue aucun traitement supplémentaire ou stockage des entités qui sont retournés par la requête.

Il y a des gains de performance considérables découlant de l'utilisation de AsNoTracking().

12. Rester loin de pièges de performance et autres complications de EF

Le gestionnaire de contexte d'objets peut parfois mener à des situations où EF se comporte de manière étrange. C'est bien d'être informé des méthodes que tu peux suivre pour éviter ces pièges.

ASP.NET MVC: Eviter de Polluer le Modèle de la Vue avec des Messages d'Erreurs

By Holty Sow

J'ai récemment eu à répondre à une question posée sur le site StackOverflow. Je pose le contexte. Nous avons un modèle suivant:

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

Ce modèle est utilisé dans l'action d'un contrôleur comme suit :

```
[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();
   }
}
```

La question était de savoir si le fait d'utiliser la propriété dynamique **ViewBag** du contrôleur pour exposer le message d'erreur était une bonne pratique et que les recherches effectuées par le questionneur lui ont fait savoir qu'il fallait exposer une propriété au niveau du modèle.
Evidemment il est fortement recommandé de ne pas utiliser la propriété **ViewBag** étant donné qu'on ne bénéficie pas du typage fort. Si on veut communiquer avec la vue il faut toujours passer par un modèle typé. La solution proposée est donc légitime mais n'est pas une bonne pratique dans le cas de la gestion des erreurs du modèle dans le framework ASP.Net MVC.
La solution pour exposer les messages d'erreurs (comme dans l'exemple précédent qui n'utilise pas les attributs d'annotations de données) vient par l'utilisation de la méthode **AddModelError**de la classe **ModelStateDictionary**. Nous n'avons pas besoin d'instancier cette classe étant donné qu'une propriété **ModelState** contenant une instance de cette classe existe déjà au niveau du contrôleur.
Du coup la bonne solution est de faire comme suit:

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

Il faut noter que cette méthode reçoit en premier paramètre le nom de la propriété du modèle à laquelle le message d'erreur est associé. Ainsi pour que ce message d'erreur soit affiché dans la vue à côté du champ **Email** il faut ajouter juste à côté de ce dernier la ligne suivante :

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

Cependant il est possible d'avoir un message d'erreur général à l'ensemble du modèle c'est à dire que ce message d'erreur n'est rattaché à aucune propriété du modèle. Pour cela il faudra utiliser une chaîne vide comme premier paramètre :

```
@Html.ValidationSummary(string.Empty, ""Email not found or matched")
```

Dans la vue Razor, il faudra utiliser la ligne suivante :

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

Le premier paramètre booléen indique qu'on ne veut pas afficher les messages d'erreur déjà rattachés à des propriétés du modèles.

Dans ce billet nous avons vu qu'on n'a pas besoin de polluer notre modèle et d'exposer des propriétés spécifiques aux messages d'erreur. Il suffit juste d'utiliser ce que nous offre le Framework ASP.Net MVC pour nous faciliter la tâche.

J'espère que ce billet vous a été utile.

Limiter l'exécution d'une action uniquement aux requêtes AJAX

Par Holty Sow

En ASP.Net MVC nous manipulons entre autres des vues. Parmi ces vues certaines représentent des pages complètes et d'autres ne sont que des parties de page. Ces parties ou zones appartenant à une vue sont appelées des vues partielles et elles sont renvoyées elles aussi par des actions du contrôleur. Ces vues partielles ne devant être utilisées qu'à l'intérieur d'une vue alors le Framework ASP.Net MVC nous permet de protéger l'appel à ces actions partielles en les décorant avec l'attribut **ChildActionOnly**. Cet attribut permet d'être sûr que l'action :
- ne pourra pas être utilisée comme une vue entière et que les développeurs de l'application l'exécuteront toujours en passant par les méthodes **HtmHelper.Action** ou **HtmlHelper.RenderAction.**
- a une URL qui ne sera pas accessible via la barre d'adresse si un utilisateur, je ne sais par quel moyen, serait au courant de l'existence de cette URL.

Cependant comme tout site dynamique nous aurons des requêtes AJAX qui pourront faire elles aussi des demandes de contenu HTML sans qu'on ait besoin de charger la page de façon complète. Ce contenu aussi représente une partie puisqu'à la réception de la réponse du serveur Web nous devons incorporer ce bout de HTML quelque part dans la page. La requête AJAX envoyée au serveur invoquera certainement une action du contrôleur. Cette action comme celles marquées avec l'attribut **ChildActionOnly** doit posséder des restrictions, soit :
- obligation de passer par une requête faite en AJAX.
- inaccessible via la barre d'adresse du navigateur.

Cependant le Framework ASP.Net MVC n'offre aucun attribut nous permettant d'appliquer ces restrictions sur une action mais il nous fournit les outils pour en créer. Pour cela nous devons coder un filtre qui sera exécuté juste avant l'exécution de l'action concernée. Si la requête entrante respecte les conditions d'une requête faite en AJAX alors on laisse l'action continuer son chemin. Dans le cas où les conditions ne seraient pas respectées nous envoyons une page 404 (comme quoi l'URL est inexistante).

Le code du nouveau filtre que je nommerai **AjaxOnlyAttribute,** classe dérivée de la classe **ActionFilterAttribute,** est le suivant :

```
[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();
      }
   }
}
```

La nouvelle classe ne fait que surcharger la méthode qui nous intéresse. C'est à dire la méthode **OnActionExecuting** qui est appelée juste avant le début de l'exécution de l'action demandée par la requête entrante. L'attribut peut être posé sur le contrôleur pour atteindre l'ensemble des actions ou unitaire sur chaque action où la requête AJAX est obligatoire.

Pour éviter des soucis de pertes de quelques petites minutes qui peuvent arriver dans le cas où l'un des développeurs ayant rejoint l'équipe en cours de route et qui ne comprendrait pas pourquoi sa requête renvoie du **404** alors je pense que ce serait un avantage d'avoir un mode **DEBUG** activé par défaut pour les codeurs. Le code de notre classe ressemblerait finalement à celui-ci :

```
[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
      }
   }
}
```

Etant donné que les développeurs sont amenés à compiler l'application la plupart du temps en mode **DEBUG** du coup la vue **AjaxOnly** contenant le texte qui va bien leur expliquera ce qu'il faut faire. Il faut noter que cette version de la classe fonctionne uniquement si une vue **AjaxOnly.cshtml** a été ajoutée dans le répertoire **Views\Shared** de l'application. Aussi on n'est pas obligé de renvoyer une vue partielle. On peut tout simplement remplacer le code suivant :

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

par juste du contenu textuel :

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

La version utilisant la vue **AjaxOnly** a l'avantage de prendre en compte automatiquement le rendu **_Layout** par défaut défini au niveau de l'application.

[Interview] Dans La Bulle de Holty Sow : "pour plus d'efficacité, je préfère travailler avec une équipe agile"

1. Présentation rapide (mais efficace) du personnage et de ses réalisations ?

Je m'appelle Holty Samba SOW. Après avoir débuté mes études supérieures en Maths-Physique puis effectué une licence professionnelle en Informatique au Sénégal, je les ai poursuivies en France pour obtenir un Master en Documents Electroniques et Flux d'informations en 2009. Ce qui m'a permis par la suite de me lancer dans le monde professionnel de la programmation informatique. Actuellement je suis consultant et développeur principalement sur la plateforme .NET chez SoftFluent, un éditeur de logiciels mais aussi une société de services.

2. Quels sont tes objectifs principaux dans la vie ?

Mon principal objectif est pour le moment de garder aussi longtemps que possible ma passion dans la programmation informatique en accumulant un maximum d'expériences dans le monde du Web et du Mobile. J'espère grâce à la richesse de cette expérience en faire bénéficier mon pays natal le Sénégal.

3. Quels outils et techniques utilises tu pour accomplir des choses efficacement ?

Pour les IDE préférés : VS Code pour tout ce qui touche au CSS, JavaScript et TypeScript. Pour tout le reste j'utilise l'IDE Visual Studio accompagné de l'extension ReSharper pour plus d'efficacité.
Mes technologies préférées sont principalement tout ce qui touche au Back-end sur la plateforme .NET donc ASP.NET MVC, ASP.NET Web API et récemment ASP.NET Core.
La méthodologie avec laquelle je préfère travailler en équipe est Scrum/Agile.

4. Des recommandations pour tes juniors ?

Un livre : "C# In a Nutshell" pour comprendre le langage de programmation C# de façon approfondie.

Twitter et Blogs : suivre, via Twitter ou à travers leurs blogs, les influenceurs (speakers, CTO, etc.) évoluant dans le monde informatique permet d'être au courant de pas mal de choses.

Sites Web :
docs.microsoft.com (je trouve que le nouveau site web de documentation de Microsoft est assez bien développé)
stackoverflow.com (je trouve qu'essayer de répondre aux questions posées par d'autres développeurs permet parfois de découvrir des trucs et astuces auxquels on n'aurait pas pensé)
pluralsight.com (pas mal de vidéos sur la programmation vraiment très intéressant. Seul hic : c'est payant)
channel9.msdn.com (idem que Pluralsight mais c'est gratuit et les speakers sont des employés de Microsoft).

5. Quel est le meilleur moyen de te contacter ?

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

La Révolution Xamarin

Par Necemon

Xamarin, c'est quoi ?
Xamarin est une technologie qui te permet de concevoir des applications natives pour différentes plateformes mobiles telles que Android, iOS ou encore Windows Phone et ce, en n'utilisant qu'un seul langage de programmation, le C#.
Il ne vous sera donc pas nécessaire d'avoir les bases en Java, utilisé habituellement pour développer sous Android ou encore en Objective C pour iOS, en revanche il vous sera très utile de connaître globalement le fonctionnement de chacune des plateformes visées (cycle de vie de l'application, directives générales, etc.)

Ici, Xamarin prend le contre-pied des autres technologies multiplateforme. Le développeur commence par créer une base de code commune. Elle contient notamment la logique métier, le stockage en base de données, les appels réseaux, les éléments d'interface communs. Ce projet peut être facilement encadré par des tests unitaires car son code est indépendant de tout système spécifique. Ensuite, un projet est crée par plateforme cible. Il contient l'interface graphique, la navigation et les composants propres à chaque SDK. Ainsi, on peut tirer parti des spécificités propres à Android ou iOS sans réduire l'expérience utilisateur au plus petit commun dénominateur.

Avec l'avènement des technologies mobiles et des smartphones, Xamarin devient de plus en plus populaire, surtout depuis sa mise à disposition gratuite dans Visual Studio. Voici quelques docs que je peux recommander pour ceux qui pensent à s'y mettre :

Ce e-book gratuit et complet en Français

Xamarin sur MSDN

Channel9

Plus de ressources en anglais

8 instruments efficaces pour programmeurs Xamarin

Par Necemon

Si tu ne sais pas ce qu'est Xamarin, puis-je te suggérer de commencer par lire mon article d'introduction avant de continuer avec celui-ci ?

Pour ceux qui savent, entrons dans le vif du sujet :

Xamarin Forms

Xamarin.Forms est une librairie de code qui permet de construire des interfaces graphiques natives qui peuvent être partagées sur Android, iOS et Windows Phone, complètement en C#, à partir d'une base de code C# unique et partagée. Les interfaces graphiques sont affichées à l'aide des contrôles natifs de la plate-forme cible, ce qui permet aux applications Xamarin.Forms de conserver l'aspect et la sensation appropriés pour chaque plate-forme. Cela signifie que les applications peuvent partager une grande partie de leur code d'interface utilisateur et conservent toujours l'apparence native de la plate-forme cible. Xamarin.Forms permet un prototypage rapide des applications qui peuvent évoluer au fil du temps vers des applications complexes. Étant donné que les applications Xamarin.Forms sont des applications natives, il est possible de créer des applications qui auront certaines parties de leur interface utilisateur créées avec Xamarin.Forms tandis que d'autres parties seraient créées à l'aide d'outils UI natifs.

Les Custom Renderers

Comme on le voyait précédemment, les interfaces utilisateur Xamarin.Forms sont rendues à l'aide des contrôles natifs de la plate-forme cible, ce qui permet aux applications Xamarin.Forms de conserver l'apparence appropriée pour chaque plate-forme. Les Custom Renderers permettent aux développeurs d'annuler ce processus pour personnaliser l'apparence et le comportement des contrôles Xamarin.Forms sur chaque plate-forme. Ils peuvent être utilisés pour de petits changements de style ou pour une personnalisation sophistiquée de la mise en page et du comportement spécifique à la plate-forme.

SQLite

La plupart des applications ont besoin d'enregistrer des données sur le périphérique, localement. À moins que la quantité de données ne soit trivialement petite, il faut généralement une base de données et une couche de données dans l'application pour gérer l'accès à la base de données. SQLite est un moteur de base de données relationnelle accessible par le langage SQL.
Contrairement aux serveurs de bases de données traditionnels, comme MySQL ou SQL Server, sa particularité est de ne pas reproduire le schéma habituel client-serveur mais d'être directement intégrée aux programmes. Facile d'usage, l'intégralité de la base de données (déclarations, tables, index et données) est stockée dans un fichier indépendant sur la plateforme.
SQLite est le moteur de base de données le plus utilisé au monde, grâce à son utilisation dans de nombreux logiciels grand public comme Firefox, Skype, dans certains produits d'Apple et d'Adobe. De par son extrême légèreté (moins de 300 Ko), il est également très populaire sur la plupart des smartphones modernes.
IOS et Android ont tous deux le moteur de base de données SQLite "intégré" et l'accès aux données est simplifié par la plateforme Xamarin.

Boites de Dialogue ACR

Une bibliothèque multiplate-forme qui te permet d'invoquer les boites de dialogue habituelles à partir d'une bibliothèque partagée et portable: feuilles d'actions, alertes, confirmations, demandes de confirmation, chargement, connexion, progression, etc.

Xlabs

Un projet open source qui vise à fournir un ensemble puissant et multiplate-forme de contrôleurs et d'assistants ajustés pour travailler avec Xamarin Forms: suggestions semi-automatiques, contrôles de Calendrier, boutons-images, étiquettes d'hyperliens, etc.

UI Sleuth

Un outil de débogage et un inspecteur d'interface utilisateur. Si tu as déjà fait un site Web, c'est très similaire aux outils F12 de Microsoft Edge ou aux outils de développement de Google Chrome. Tu peux l'utiliser pour analyser efficacement les problèmes de mise en page, pour prototyper un nouveau design, ou encore contrôler un appareil à distance.

Émulateur Android GenyMotion

Si tu veux tester Android, mais sans un smartphone Android, il y a plusieurs solutions disponibles comme l'émulateur officiel d'Android qui vient avec le SDK (kit de développeur), le populaire Bluestacks App Player qui est plus axé sur l'exécution des applications Android sur PC. Cependant, l'émulateur du SDK est connu pour être très lent tandis que Bluestacks est limité en fonctionnalités si tu veux faire du développement. Si tu cherches un émulateur Android plus rapide et complet pour ordinateur, Genymotion est ce qu'il te faut. Genymotion est un émulateur pour Android, rapide et facile, pour tester le fonctionnement et les performances d'applications. C'est l'émulateur Android le plus puissant pour les testeurs et les développeurs d'applications. Conçu par la start-up française Genymobile, Genymotion permet l'émulation d'un grand nombre de marques et de modèles de tablettes et de smartphones. S'appuyant sur des technologies de virtualisation grâce à VirtualBox, il crée une machine virtuelle qui émule en temps réel Android sur ton PC Windows, Mac ou Linux. il est ainsi possible d'émuler différents terminaux comme la gamme des Samsung Galaxy Note, Google Nexus, HTC One ou encore les tablettes Sony Xperia. GenyMotion offre la possibilité de paramétrer facilement la résolution d'écran, le type de connexion Internet, le GPS, le statut de la batterie (simuler un déchargement), etc.

Simulateur iOS à distance

Pour tester et déboguer les applications iOS dans Visual Studio sous Windows. La plupart des ordinateurs Windows modernes ont des écrans tactiles et le simulateur iOS distant vous permet de toucher la fenêtre du simulateur pour tester les interactions de l'utilisateur dans votre application iOS. Cela inclut les pincements, les glissements et les gestes tactiles avec plusieurs doigts, des choses qui, auparavant, ne pouvaient être testées que sur iPhone, directement.

Rappel Final

Avant de conclure, je voulais juste te rappeler que ce document contient le Chapitre 7 de L'Album et que 8 autres chapitres sont également disponibles. Tu peux télécharger, (re)lire et partager chaque chapitre ou volume indépendamment/séparément, en fonction de tes intérêts. Les formats PDF, EPUB, MOBI/AZW3/KF8 (Amazon Kindle) et MP3 sont disponibles.

Il suffit de cliquer sur le(s) document(s) qui t'intéresse(nt) ci-dessous ou d'aller sur Album.NeceMoon.com (ou necemonyai.com/blog/page/L-Album.aspx).

The NeceMoon, L'Album Complet

Volume 1 : Clair de Lune (softcore)

Volume 2 : Pleine Lune (hardcore)

L'Album est également disponible en Anglais à l'adresse TheAlbum.NeceMoon.com (ou necemonyai.com/blog/page/The-Album.aspx).

Conclusion

C'est le moment de se quitter, mais on peut rester en contact. N'hésite pas à m'ajouter sur LinkedIn, Twitter ou Facebook. Si toi aussi, tu as des outils ou des techniques pour accomplir les choses efficacement, je voudrais bien que tu m'en parles. Mon adresse email, c'est necemon@gmail.com. N'hésite pas à m'écrire pour signaler d'éventuelles erreurs dans ce document, pour suggérer des améliorations ou pour me faire part de tes passages préférés. Par contre, si tu n'aimes pas L'Album, ce n'est pas la peine de m'écrire.

Dans tous les cas, bon courage pour la suite, et merci d'avoir lu. Et merci à Wikipédia, MSDN, IconFinder et FreeDigitalPhotos pour les clarifications et les ressources infographiques. Remerciements spéciaux à la Team Evasium. Merci à tous ceux et celles qui ont contribué, merci Ahou l'Africaine, Antoine Mian, Cyriac Gbogou, Darren Mart, Edith Brou, Holty Sow, Israël Yoroba, Jean Luc Houédanou, Jean-Patrick Ehouman, Karen Kakou, Nanda Seye, Nnenna Nwakanma, Olivier Madiba, Vanessa Lecosson et Yehni Djidji.
Merci Monty Oum, repose en paix.

Partage L'Album avec Tes Potes

Clique simplement sur l'icône qui te convient ci-dessous. Ça ne prend que quelques secondes et c'est gratuit pour tout le monde. Alternativement, tu peux partager ce lien sur toute Plateforme Web/Sociale ou l'envoyer par e-mail à tes contacts qui pourraient en bénéficier : http://Album.NeceMoon.com