Visual Studio 2013 開発する上で知っておいたほうがいいこと

佐野　尚之

変更履歴

- 2014/7/23
  - 新規作成

目次

- キャメルケース
- varキーワード(暗黙的に型付けされたローカル変数)
- Path.Combine メソッド(フォルダ名とファイル名を結合して絶対パスを作成)
- 自動実装するプロパティ
- ラムダ式
- Entity Framework
- テストクラス
- Microsoft Fakes Framework(フェイクス フレームワーク)
- MicrosoftのDIコンテナ「Unity」(ユニティ)
- Unityを使ったAOP使用例
- Microsoft Enterprise Library
- Microsoft Azureについて
- おまけ

キャメルケース

- 一般的なことなので、とりあえず紹介。CamelCase。
  - 最初の単語はすべて小文字
  - それ以降の単語は頭だけ大文字、頭以外は小文字
  - 例：fooBar
- 参考URL

  大文字の使用規則

  http://msdn.microsoft.com/ja-jp/library/ms229043.aspx

  名前に関するガイドライン

  http://msdn.microsoft.com/ja-jp/library/ms229002.aspx

varキーワード
(暗黙的に型付けされたローカル変数)

- メソッドのスコープで宣言される変数が、var という暗黙の型を持つことができます。 暗黙的に型指定されたローカル変数は、型を宣言した場合と同様に厳密に型指定されますが、コンパイラが型を決定します。
- 使用例

```
var testUser = “テストユーザー”;

var l = 1;
```

Path.Combine メソッド(フォルダ名とファイル名を結合して絶対パスを作成)

- C:¥ParentDirectoryとChildDirectory¥FileName.txtを連結させる場合、うしろに「¥」を付けるとか付けないのとか考えなくていい。
- 使用例

```
using System.IO;

var testFolderPath =
Path.Combine(@"C:¥ ParentDirectory","ChildDirectory","FileName.txt");
```

- 実行後

```
C:¥ParentDirectory¥ChildDirectory¥FileName.txt
```

自動実装するプロパティ

- プロパティ アクセサーで追加のロジックが必要ない場合に、プロパティ宣言が簡単！
- 使用例

```
public string Name { get; set; }

public int Age { get; set; }
```

ラムダ式

- ラムダ式はデリゲート型または式ツリー型を作成するために使用できる匿名関数です。
- 使用例

```
Func<string, string> fullPath = (folderName) =>
{
        return Path.Combine(@"C:¥ ParentDirectory", "ChildDirectory", folderName);
};
var excelFolderPath = fullPath("Excel.txt");
var accessFolderPath = fullPath("Access.txt");
```

- 参考URL

  ラムダ式 (C# プログラミング ガイド)

  http://msdn.microsoft.com/ja-jp/library/bb397687.aspx

  【ラムダ式】 ラムダ式記事一覧【目次】

  http://csharp30matome.seesaa.net/category/7142936-1.html

Entity Framework

- Entity Framework (EF) は、.NET 開発者がドメイン固有のオブジェクトを使用してリレーショナル データを処理できるようにするオブジェクト リレーショナル マッパーです。
- 遅延読み込みは覚えておいたほうがいいです。SELECTの結果を.toList()で取得するとデータ量によってはメモリ使用量が大変なことになります。コードレビューでとっても怒られます。遅延読み込みの場合は、foreachでまわしながらデータを取得しにいくイメージなので、メモリ消費量が少ないらしいです。SQLの結果を取得した時には結果を取得していない状態で、ループでまわるたびに１行ずつDBから読み込むイメージなんだと思います。たぶん。
- 参考URL

  Entity Framework の俺的まとめ

  http://shiba-yan.hatenablog.jp/entry/20101010/1286636999

  Entity Framework のお役立ち記事まとめ

  http://fnya.cocolog-nifty.com/blog/2014/01/entity-frame-13.html

  4. データの挿入、読み出し、更新、削除

  http://densan-labs.net/tech/codefirst/adddelete.html

テストクラス

- Visual Studio のテスト実行環境は、テストを実行する度に、TestResultディレクトリ配下に、新たなフォルダが作成され、そこでテストが行われる。
- テストクラスは並列で動く。
- Visual Studio上 で プライベートなメンバやメソッドなど非公開メンバの単体テストに対して実施する場合、 リフレクションを使用する代わりにPrivateObject, PrivateTypeを使用する。
- 参考URL

  単体テストを使用したコードの検証

  http://msdn.microsoft.com/ja-jp/library/dd264975.aspx

  VisualStudio2013単体テスト作成方法

  http://rururu.sakura.ne.jp/doc/VisualStudio2013単体テスト作成方法.pdf

  【Visual Studio】ファイルを使う機能の単体試験について

  http://blogs.yahoo.co.jp/dk521123/31532337.html

  [VS2010] Visual Studio の単体テストでプライベートなメソッドやフィールドにアクセスする

  http://handcraft.blogsite.org/Memo/Article/Archives/294

Microsoft Fakes Framework (フェイクス フレームワーク)

- 単体テストのフレームワーク。「スタブ」と「shim」という機能があります。shim は、コンパイル済みアセンブリの挙動を単体テスト用に置き換える機能。
- 残念ながらVisual StudioのPremiumかUltimateしか使うことができない。
- 通常のモックライブラリがinterfaceやvirtualなメソッド類しか上書きできないのに対して、FakesはStaticメソッドや普通の非virtualメソッドすらも上書きできる。
- 参考URL

  Moq ＆ Fakes Frameworkを使った実践的ユニットテスト ［スライド＆動画］

  http://www.buildinsider.net/hub/bioff/o3

  Microsoft Fakes Frameworkの使い方

  http://neue.cc/2012/11/03_387.html

MicrosoftのDIコンテナ「Unity」(ユニティ)(1/3)

- Dependency Injection（以下、DI）。DIとはクラス間の直接的な依存関係を排除する開発手法であり、DIを実現するツールのことをDIコンテナと言います。
- Javaだと、Spring FrameworkやJBoss Seam、Sarsar2。.NETの場合は、NinjectやUnity、Seasar.NETなどがあります。
- 参考URL

  猿でも分かる! Dependency Injection: 依存性の注入

  http://qiita.com/hshimo/items/1136087e1c6e5c5b0d9f

  Unityを利用したDI実装のサンプル(Project Silk)

  http://silk.codeplex.com/

  Project Silk が本気すぎてやばい

  http://shiba-yan.hatenablog.jp/entry/20111006/1317914206

  Project SilkにならってUnityを使ってみる

  http://miso-soup3.hateblo.jp/entry/20120618/1340034949

MicrosoftのDIコンテナ「Unity」(ユニティ) (2/3)

- http://codezine.jp/article/detail/6378より引用

図1　DI実装前

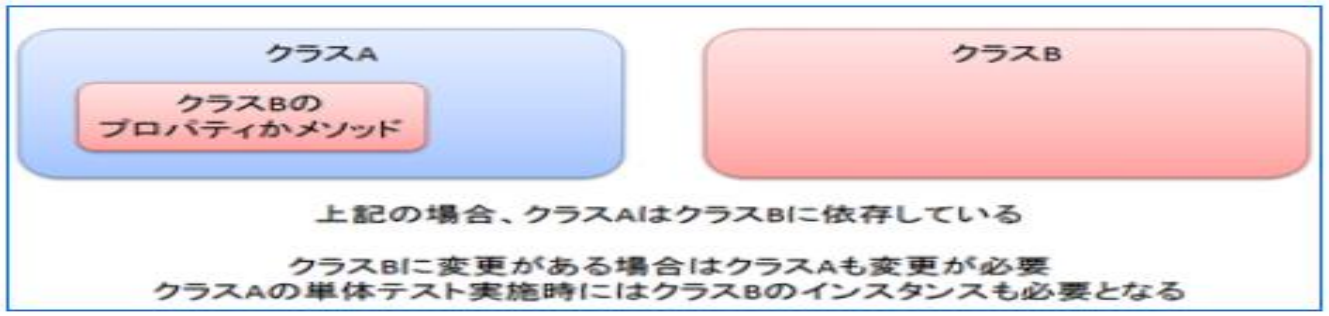
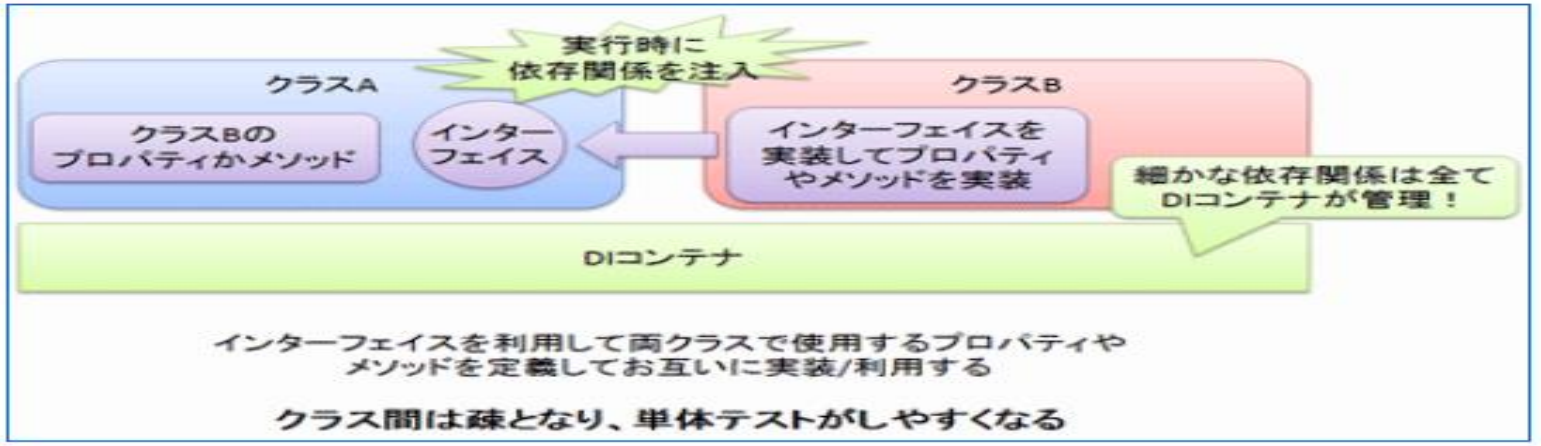


図2　DI実装後



MicrosoftのDIコンテナ「Unity」(ユニティ) (3/3)

- 参考URL

  Unityで属性を使ってクラス間の依存関係を定義する方法

  http://code.msdn.microsoft.com/windowsdesktop/Unity-f36c3550

  UnityのLifetimeManagerをカスタマイズする

  http://code.msdn.microsoft.com/windowsdesktop/UnityLifetimeManager-d703afff

  Unityで配列のインジェクションを行う

  http://code.msdn.microsoft.com/windowsdesktop/Unity-64387871

  Unityでコンテナ登録時とコンテナからのオブジェクト取り出し時に値を設定する

  http://code.msdn.microsoft.com/windowsdesktop/Unity-811c2109

  Unityで簡単に型を登録する方法（一括登録）

  http://code.msdn.microsoft.com/windowsdesktop/Unity-8f13dfbd

  Unityコンテナに型を登録して取得する

  http://code.msdn.microsoft.com/windowsdesktop/Unity-dca369f1

Unityを使ったAOP使用例

- Unityの拡張機能としてInterceptionというものを使ってメソッドやプロパティの処理の前後に任意の処理を行うことができるようになるらしいです。
- 詳細は、以下のURLを参照。

  UnityでAOPをしよう

  http://code.msdn.microsoft.com/windowsdesktop/UnityAOP-0ff53983

Microsoft Enterprise Library

- Enterprise Libraryとは、Microsoftのpattern & practiceチームが開発しているエンタープライズアプリケーションの開発のベストプラクティスを集めたライブラリです。このライブラリの特徴は、UnityというDIコンテナを軸にしてEnterprise Libraryが提供している機能(Application Blockと呼ばれる)を組み合わせて使用できるという点です。ログ出力機能はゼロから実装すると結構めんどくさいので、こういうライブラリを使ったほうがいいのかなぁと思います。
  - Caching Application Block　アプリケーション内でキャッシュ機能を提供します。
  - Cryptography Application Block　データの暗号化・複合とハッシュを生成する機能を提供します。
  - Data Access Application Block　データベースへのアクセス機能を提供します。
  - Exception Handling Application Block　例外処理の機能を提供します。
  - Logging Application Block　ログ出力機能を提供します。
  - Policy Injection Application Block　機能横断的なPolicyをアプリケーションに適用する機能を提供します。
  - Security Application Block　承認の規則（操作の許可や拒否など）を構成・管理する機能を提供します。
  - Validation Application Block　値の妥当性検証の機能を提供します。
- 参考ＵＲＬ

  @IT > Insider.NET > Enterprise Library

  http://www.atmarkit.co.jp/fdotnet/entlib/index/index.html

  Microsoft エンタープライズ ライブラリ開発者向けガイダンス

  http://msdn.microsoft.com/ja-JP/library/ff953181%28v=pandp.50%29.aspx

  EntLib

  http://okazuki.hatenablog.com/category/EntLib

Microsoft Azureについて(1/2)

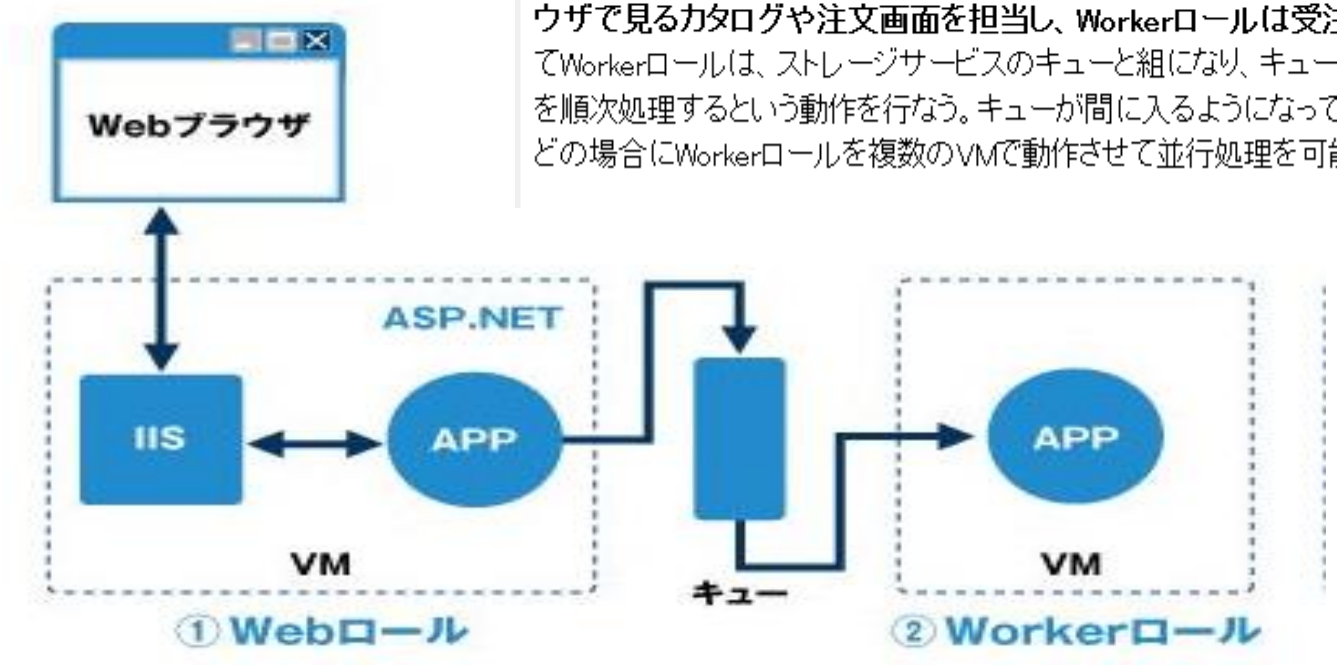
- マイクロソフトが提供するクラウド プラットフォーム。2014/3/26にWindows AzureからMicrosoft Azureに名称が変わった。
- クラウドサービスなので、使った分だけ払う。アクセスが多いサイトだと金額が大変なことになると思う。レンタカーやホテルと同じで毎日使うと割高。AmazonのAWSも同じ。
- ロールとストレージというのがある。
- アプリを載せるマシン環境(= ロール)
  - Webロール
  - Workerロール
- 外部記憶(ストレージ)
  - Blob(Binary Large Object。ブロブ)・・・画像や動画などのバイナリデータの保存
  - Queue(キュー)・・・アプリ間連携のためのメッセージキュー
  - Table(テーブル)・・・Key-Value型でデータを保存
- SQLDatabase
  - SQL Server 2008をベースに開発されたクラウド上の関係データベースエンジン

Microsoft Azureについて(2/2)

WebロールとWorkerロール

Webロールは、IISの上で提供されるWebページとその処理機能だ。簡単にいえば、ASP.NETによるページ表示機能に相当する。2つ目のWorkerロールは、Webロールと対にして利用するもので、サーバー側のバックエンド処理ルーチンとなる。

たとえばオンラインショッピングサイトを考えた場合、**Webロールはユーザーが Webブラウザで見るカタログや注文画面を担当し、Workerロールは受注処理を担当する。**そしてWorkerロールは、ストレージサービスのキューと組になり、キューに投入されたアイテムを順次処理するという動作を行なう。キューが間に入るようになっているのは、大量処理などの場合にWorkerロールを複数のVMで動作させて並行処理を可能にするためだ。



http://ascii.jp/elem/000/000/596/596541/より引用

おまけ

他にも

・LINQ　　・・・案件によっては使ったことないと面談で落ちる。

・PowerShell・・・例外処理(try-catch)がある。戻り値でコレクション返せる

・JSON処理

・ZIP圧縮／解凍・・・.Net4.5からZIPアーカイブの作成・展開・編集を行うためのクラスが追加されてます

・非同期メソッド(async／await)

・Visual Studioのコード分析

コード分析ツールを使用したアプリケーション品質の分析

http://msdn.microsoft.com/ja-jp/library/dd264897.aspx