Cours .Net
Windows Communication Foundation- suite

Lemettre Arnaud
Arnaud.lemettre@gmail.com

EPITA
ÉCOLE D'INGÉNIEURS EN INFORMATIQUE

Microsoft® .NET

Sommaire

- Introduction
- WCF version sécurisée ;)
- Génération de backend

Introduction

- Nous avons vu lors du dernier cours le système de communication, mais qui dit communication dit aussi risque de corruption des données. Ce à quoi WCF a répondu par la sécurisation de ces échanges.

Introduction

- Grâce à Entity framework, de nouveaux templates de projet sont apparus. Nous verrons ici comment générer un backend rapidement avec ces nouveaux templates.

Principe fondamentaux

- Tout d'abord, il faut sélectionner le bon binding.
- Pour la sécurité, il faudra choisir wsHttpBinding. De base c'est le binding par défaut. Cependant il n'est pas encore configuré

Pour éviter les problèmes de configuration de sécurité, vous pouvez passer les bindings en basicHttpBinding. Cependant vous perdrez toute sécurité sur vos communications.

Principe fondamentaux

- Bien sur, selon les bindings que vous utiliserez, vous n'aurez pas accès aux mêmes sécurités.

- Pour la sécurité, on peut utiliser

les wsHttpBinding et NetTcpBinding

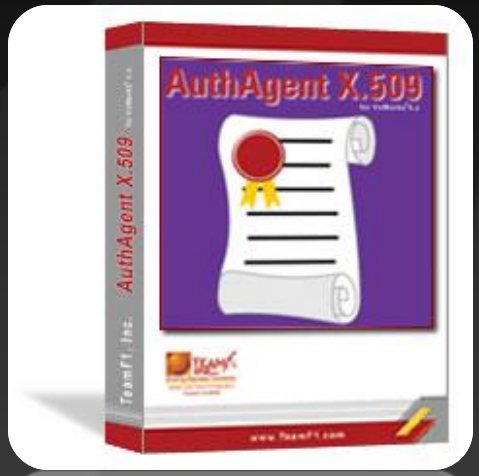
- Pour la suite du cours, nous ne verrons que wsHttpBinding, format le plus utilisé

Principe fondamentaux

- Il existe plusieurs modes de sécurité :
  - None. Désactive la sécurité.
  - Transport. Utilisé pour l'authentification et la protection des messages.
  - Message. Utilisé pour l'authentification et la protection des messages. (Celui que nous allons utiliser)
  - Both. Seulement utilisé par MSMQ
  - TransportWithMessageCredential. Les Credentials sont passés avec le message.
  - TransportCredentialOnly. Les Credentials sont passés avec la couche de transport.

Principe fondamentaux

- L'utilisation de sécurité impose l'utilisation de certificat SSL. Ce qui implique une phase compliquée sous windows … ☹

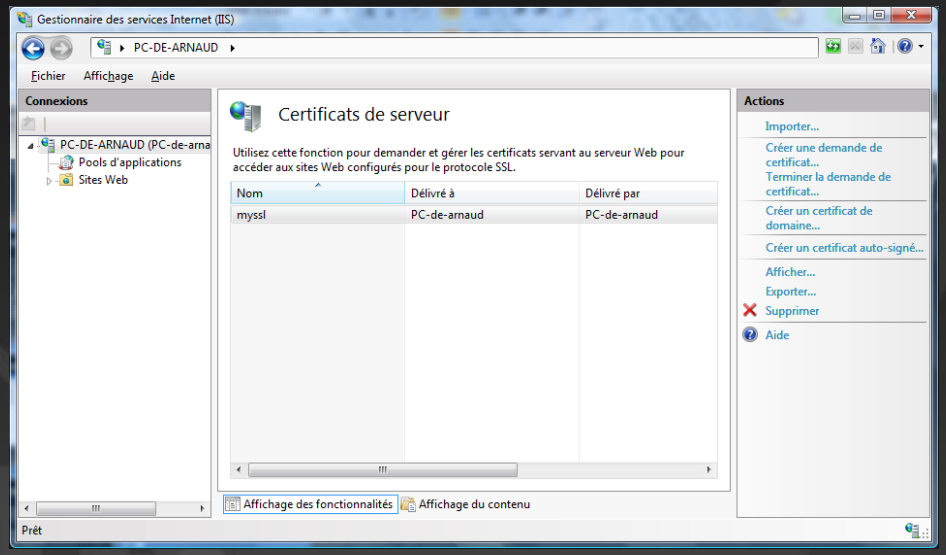


Cas Pratique

- Sécurisation d'un service par membership Provider

Console Management
IIS

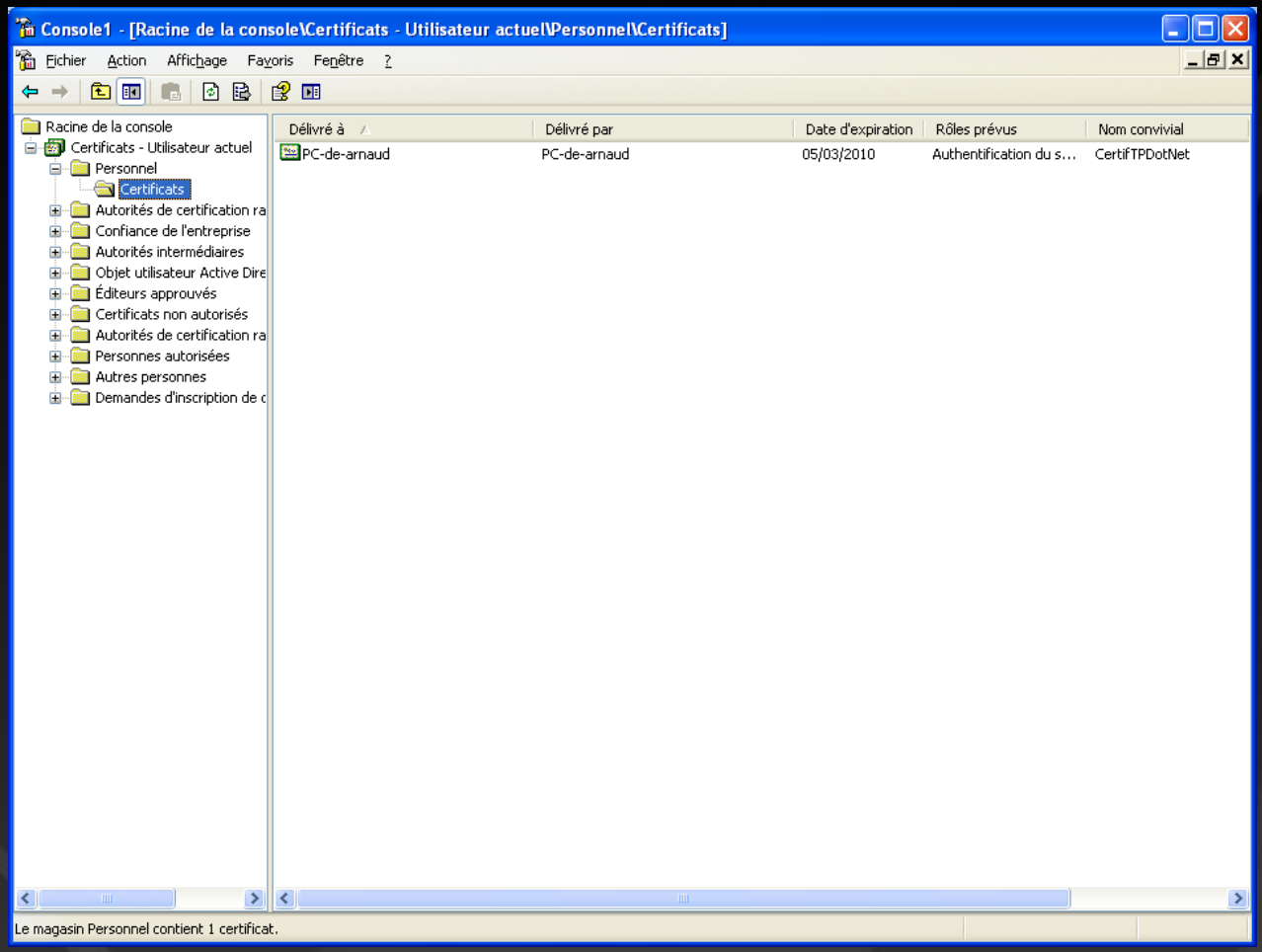
- Création d'un certificat SSL sous Vista / Seven :
  - Exécuter -> inetmgr
  - Puis Certificats de serveur
  - Créer un certificat auto signé
  - Lui donner un nom



Console MMC

- La console Microsoft Management Console (MMC) permet de voir entre autres les certificats installés sur la machine.
  - Exécuter -> mmc
  - Fichier -> ajouter / supprimer un composant enfichable
  - Sélectionner Certificats puis ajouter pour le compte utilisateur par exemple.

Console MMC


Console1 - [Racine de la console\Certificats - Utilisateur actuel\Personnel\Certificats]
Fichier Action Affichage Favoris Fenêtre ?
Racine de la console
Certificats - Utilisateur actuel
Personnel
Certificats
Autorités de certification ra
Confiance de l'entreprise
Autorités intermédiaires
Objet utilisateur Active Dire
Éditeurs approuvés
Certificats non autorisés
Autorités de certification ra
Personnes autorisées
Autres personnes
Demandes d'inscription de c
Délivré à
Délivré par
Date d'expiration
Rôles prévus
Nom convivial
PC-de-arnaud
PC-de-arnaud
05/03/2010
Authentification du s...
CertifTPDotNet
Le magasin Personnel contient 1 certificat.

Configuration serveur

- La configuration de la sécurité en WCF se fait en majorité dans le Web.config. Il y plusieurs phases.
  - Déclaration du membership provider
  - Configuration behaviors
  - Configuration Binding
  - Configuration service

Configuration serveur

Ajout du MemberShip Provider :

```
<configuration>
     <configSections/>
     <appSettings/>
     <connectionStrings/>
     <system.web>
    <!--ajout du member ship custon-->
    <membership defaultProvider="CustomMembershipProvider" userIsOnlineTimeWindow="15">
      <providers>
        <!--<clear />-->
        <add
          name="CustomMembershipProvider"
          type="WcfServiceTest.CustomMembershipProvider"

          applicationName="/"
          enablePasswordRetrieval="false"
          enablePasswordReset="false"
          requiresQuestionAndAnswer="false"
          requiresUniqueEmail="true"
          passwordFormat="Clear" />
      </providers>
    </membership>
</system.web>
</configuration>
```

Le reste de la configuration

Type custom du membership

Le nom

Endroit de L'implémentation Du membership

CONFIGURATION SERVEUR

Implémentation du membership provider

```
namespace WcfServiceTest
{
    public class CustomMembershipProvider : MembershipProvider
    {
        toutes les méthodes dont on a pas besoin

        public override bool ValidateUser(string username, string password)
        {
            if (username == null || password == null )
            {
                return false;
            }
            if (username == "" && password == "")
            {
                return false;
            }

            else
            {
                if (username == "toto" && password == "toto")
                {
                    return true;
                }
                else
                {
                    return false;
                }
            }
        }
    }
}
```

Configuration serveur

Configuration des behaviors; ils permettent de spécifier le Comportement du service.

```
<configuration>
  <!--le reste de la configuration-->
  <system.serviceModel>
    <services/>
    <bindings/>
    <behaviors>
      <serviceBehaviors>
        <!--permet de spécifier certaines options-->
        <behavior name="WcfServiceTest.ServiceCalcBehavior">
          <serviceMetadata httpGetEnabled="true"/>
          <serviceDebug includeExceptionDetailInFaults="false"/>
          <serviceCredentials>
            <userNameAuthentication userNamePasswordValidationMode ="MembershipProvider"
                membershipProviderName ="CustomMembershipProvider"/>
            <serviceCertificate storeLocation="CurrentUser" storeName="Root"
     x509FindType="FindBySerialNumber" findValue="e3 33 0e 2f 05 b1 08 82 41 39 0a 50 3a e1 ce 4d"/>
          </serviceCredentials>
        </behavior>
      </serviceBehaviors>
    </behaviors>
  </system.serviceModel>
</configuration>
```

On indique La façon de S'authentifier Ici par notre Membership provider

On indique quel certificat SSL utiliser pour assurer la protection

Configuration serveur

Configuration des bindings; ils permettent de spécifier le mode de communication.

```
<configuration>
  <!--le reste de la configuration-->
<system.serviceModel>
  <services />
  <bindings>
    <wsHttpBinding>
      <!-- Set up a binding that uses UserName as the client credential type -->
      <binding name="MembershipBinding">
        <security mode ="Message">
          <message clientCredentialType ="UserName"/>
        </security>
      </binding>
    </wsHttpBinding>
  </bindings>
  <behaviors/>
</system.serviceModel>
</configuration>
```

Nom de notre Configuration de binding

Sur quelle partie porte notre sécurité

Moyen de sécuriser L'échange

Configuration serveur

Configuration du point de connexion

```
<configuration>
  <!--le reste de la configuration-->
<system.serviceModel>
  <services>
    <!--name : nom du service doit mettre namespace.nomService-->
    <service behaviorConfiguration="WcfServiceTest.ServiceCalcBehavior"
    name="WcfServiceTest.ServiceCalc">
      <endpoint address="" binding="wsHttpBinding"
    bindingConfiguration="MembershipBinding"
    contract="WcfServiceTest.IServiceCalc"/>
    </service>
  </services>
  <bindings />
  <behaviors/>
</system.serviceModel>
</configuration>
```

Nom de notre Configuration de behavior

Type de binding

Notre configuration de binding

Configuration serveur

Exo : Changer la configuration du service pour supprimer la sécurité.

```
<configuration>
  <!--le reste de la configuration-->
<system.serviceModel>
  <services>
    <!--name : nom du service doit mettre namespace.nomService-->
    <service behaviorConfiguration="WcfServiceTest.ServiceCalcBehavior"
    name="WcfServiceTest.ServiceCalc">
      <endpoint address="" binding="basicHttpBinding"
    contract="WcfServiceTest.IServiceCalc"/>
    </service>
  </services>
  <bindings />
  <behaviors/>
</system.serviceModel>
</configuration>
```

On passe en basic

On supprime le lien vers le bindingConfiguration

Utilisation Client

- La phase d'ajout du service reference se passe comme d'habitude.
- Les différences viennent sur l'utilisation du service :
  - Ajout du certificat et des clés privées
  - Ajout du login – password

Utilisation Client

Rajout du client Credentials :

```
ServiceReference1.ServiceCalcClient client = new
   ServiceReference1.ServiceCalcClient();
client.ClientCredentials.UserName.UserName = "toto";
client.ClientCredentials.UserName.Password = "titi";
int res = client.add(1, 2);
Console.WriteLine(res);
Console.ReadKey();
```

Si les username sont faux alors une exception sera levée, de la même façon si le client n'a pas le bon certificat une exception sera également levée.

Back End

Back End

- Avec entity framework et la version 3.5 SP1 de .Net, Visual studio nous propose un nouveau template de projet : Dynamic Data Entity.

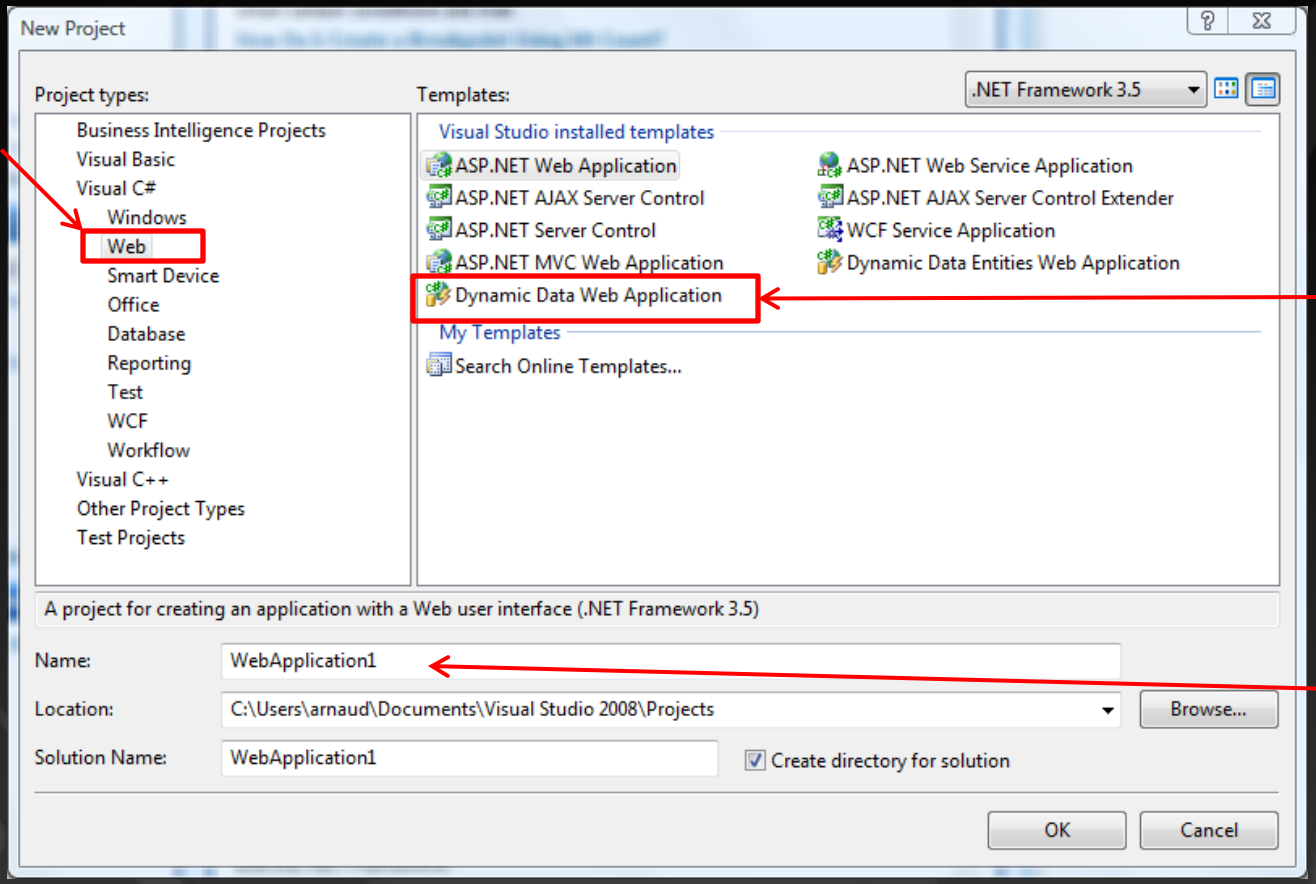
- Cela permet de mapper directement une base de données à une interface.

Un peu de théorie

- Comment cela fonctionne ?
  - Ce template utilise bien entendu la technologie de l'entity Framework.
  - Mais peut aussi être utilisé avec Linq To SQL
  - Pour accéder aux données, on utilise le rewriting d'URL d'ASP.Net

Création du Projet

Projet Web



Le template de la solution

Nom du projet

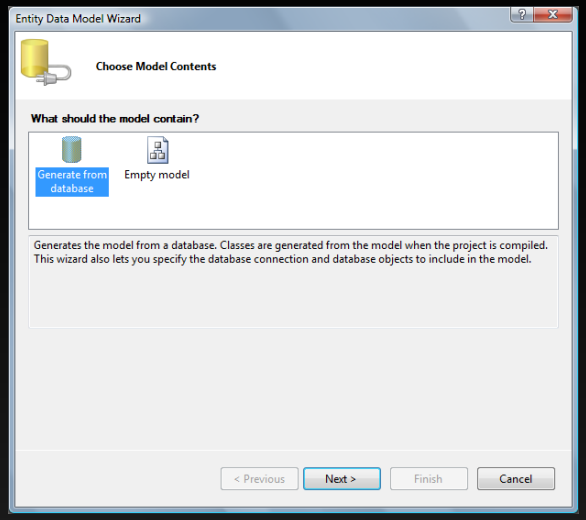
Création du Projet

Dans l'arborescence créée par visual studio, il y a un fichier important :
-global.asax

Bien sûr, actuellement on ne peut pas faire grand-chose si on lance le projet celui-ci va provoquer une exception.
Ce qui est normal, il faut lui spécifier sur quelle base il doit regarder.

Pour cela on va utiliser entity framework en spécifiant la base. Exemple :

Création du Projet



L'ajout d'un modèle se fait de la même manière que vu précédemment.

Pour que l'application puisse se lancer il faut la présence au moins d'une table accessible, sinon cela produira une erreur au moment de l'exécution.

Création du Projet

Maintenant que nous avons notre modèle, il faut spécifier dans le code qu'il faut l'utiliser. Pour cela dans le fichier Global.asax.cs, il faut rajouter cette ligne de code :

```
model.RegisterContext(typeof(testEntities), new
    ContextConfiguration() { ScaffoldAllTables = true
    });
```

Le scaffoldalltables à true laisse le dynamic data découvrir les tables et les afficher de manière automatique. En le passant à false, il faut implémenter vous-même la découverte et l'affichage.

Maintenant lors du lancement du projet, la solution sait quel modèle utiliser pour afficher les tables.

Si vous modifiez votre base de données, ajout suppression de tables, champs : pensez à mettre à jour votre modèle en faisant un update dessus.

Quoi afficher ?

Avec ce système, il y a plusieurs moyens d'afficher les informations, soit :

- sous forme de liste
- De détails
- Et permettre la suppression, l'édition et l'ajout.

```
routes.Add(new DynamicDataRoute("{table}/{action}.aspx")
{
    Constraints = new RouteValueDictionary(new { action =
  "List|Details|Edit|Insert" }),
    Model = model
});
```

Ici en complétant action, cela permet de paramétrer ce que l'on veut faire des données.

QUOI AFFICHER ?

Cas pratique - réalisation, d'un affichage en read only des données :
Dans le fichier Global.asax.cs

```
routes.Add(new DynamicDataRoute("{table}/{action}.aspx")
{
    Constraints = new RouteValueDictionary(new { action =
  "List|Details" }),
    Model = model
});
```

Ensuite il faut modifier DynamicData/PageTemplates/Details et list
Il faut supprimer dans Details :

```
<asp:DetailsView ID="DetailsView1" runat="server"
   DataSourceID="DetailsDataSource"
   OnItemDeleted="DetailsView1_ItemDeleted"
                CssClass="detailstable" FieldHeaderStyle-CssClass="bold" >
```

QUOI AFFICHER ?

Mettre en commentaire :

```
<asp:LinkButton ID="DeleteLinkButton" runat="server" CommandName="Delete"
    CausesValidation="false"
                                OnClientClick='return confirm("Are you sure
    you want to delete this item?");'
                                Text="Delete" />
```

Sur cette ligne :

```
<asp:EntityDataSource ID="DetailsDataSource" runat="server"
    EnableDelete="true">
```

Il faut mette EnableDelete à false

QUOI AFFICHER ?

Il faut aussi modifier le fichier list.aspx et mettre commentaire :

```
<asp:LinkButton ID="DeleteLinkButton" runat="server"
    CommandName="Delete" CausesValidation="false" Text="Delete"
    OnClientClick='return confirm("Are you sure you want to delete
    this item?");' />
```

Afficher Autrement

- Bien sûr, les templates par défaut peuvent ne pas vous convenir. Par exemple l'ajout en base d'un champ nécessite la création d'un id, il n'est pas utile à l'utilisateur de devoir le saisir. Pour cela, il vous faudra modifier le template pour ne pas l'afficher et changer la route pour l'amener sur votre page.

Afficher Autrement

- Il faut tout d'abord créer une arborescence spécifique pour notre affichage :
  - DynamicData
    - CustomPages
      - Skill
        - Insert.aspx

Répertoire pour les pages personnalisables

Nom de la table

Page de l'action, peut être aussi :
List, delete, ….
A ce moment on peut modifier la page d'insertion, rajouter des effets, des contrôles ….

Afficher Autrement

Modification de la route pour atteindre la nouvelle page dans le fichier global.asax, il faut rajouter la route avant la route « globale »

```
routes.Add(new DynamicDataRoute("Skill/Insert.aspx")
{
    Action = PageAction.Insert,
    ViewName = "Insert",

    Model = model
});


routes.Add(new DynamicDataRoute("{table}/{action}.aspx")
{
    Constraints = new RouteValueDictionary(new { action =
    "List|Details|Edit|Insert" }),
    Model = model
});
```

Afficher Autrement

On peut aussi ne pas vouloir afficher une colonne, dans ce cas il faut utiliser la propriété partial :

```
[ScaffoldTable(true)]
[MetadataType(typeof(SkillMetadata))]
public partial class Skill
{
}

public class SkillMetadata
{
    [ScaffoldColumn(false)]
    public Object IdSkill { get; set; }
}
```

Permet l'affichage, ou non

Peu importe le type de la propriété, il faut mettre Object

Afficher Autrement

Pour modifier un champ avant l'insertion, il faut utiliser le contexte de l'entity framework, et s'inscrire à l'événement avant l'ajout.

```
partial void OnContextCreated()
{
    this.SavingChanges += new EventHandler(APMDBEntities_SavingChanges);
}

void APMDBEntities_SavingChanges(object sender, EventArgs e)
{
    var stateManager = ((APMDBEntities)sender).ObjectStateManager;
    var changedEntities = stateManager.GetObjectStateEntries(EntityState.Added);

    foreach (ObjectStateEntry stateEntryEntity in changedEntities)
    {
        if (stateEntryEntity.Entity is Skill)
        {
            Skill skill = (Skill)stateEntryEntity.Entity;
            skill.IdSkill = Guid.NewGuid();
        }
    }
}
```


QUESTIONS ?