

version 1.0



Formations à l'informatique

Découvrez la différence ENI

Persistance des données

Support de cours

Ressources Informatiques: C# 5 – Les fondamentaux du langage

www.eni-ecole.fr

Persistance des données

Module 0

A propos de ce cours



www.eni-ecole.fr

n° 2

Programme

- Module 1 : Rappel sur le modèle en couches des applications d'entreprise
- Module 2: XML
 - Structure et règles applicables (DTD, XSD)
 - Manipulation avec les librairies du Framework .NET
- Module 3: Sérialisation
 - Sérialisation binaire et SOAP
- Module 4: Accès aux données stockées dans une base de données relationnelles
 - Mise en œuvre du mode connecté
- Module 5: LINQ
 - Langage de requête intégré
- Module 6: Entity Framework
 - Framework d'accès aux données de .NET
- Module 7: Génération de rapports/d'états
- Module 8: Déploiement d'une application
 - Mise en œuvre de la technologie *Click once*



www.eni-ecole.fr

n° 3

Développement en couches

Module 1

Rappel sur le modèle en couches des applications d'entreprise



www.eni-ecole.fr

n° 4

Présentation des couches

- Une application d'entreprise est découpée classiquement en quatre niveaux d'abstraction distincts :
 - La **couche présentation** (IHM, Interface Homme Machine)
 - La **couche logique applicative** (BLL, Business Logic Layer)
 - La **couche métier** (BO, Business Objects)
 - La **couche d'accès aux données** (DAL, Data Access Logic)



www.eni-ecole.fr

n° 5

Présentation des couches

- La couche présentation (IHM) :
 - Elle permet à l'utilisateur de dialoguer avec l'application.
 - Elle utilise les entités (BO) et la logique métier (BLL) afin de représenter graphiquement le résultat à l'utilisateur.
- La couche logique applicative (BLL) :
 - Elle décrit ce que fait l'application et comment elle le fait en s'appuyant sur les entités (BO) et les modes d'accès aux données à sa disposition (DAL).
 - Reçoit et analyse les demandes de l'utilisateur.
 - Retrouve et modifie les données via la couche données.
 - Renvoie les résultats à la couche présentation.



www.eni-ecole.fr

n° 6

Présentation des couches

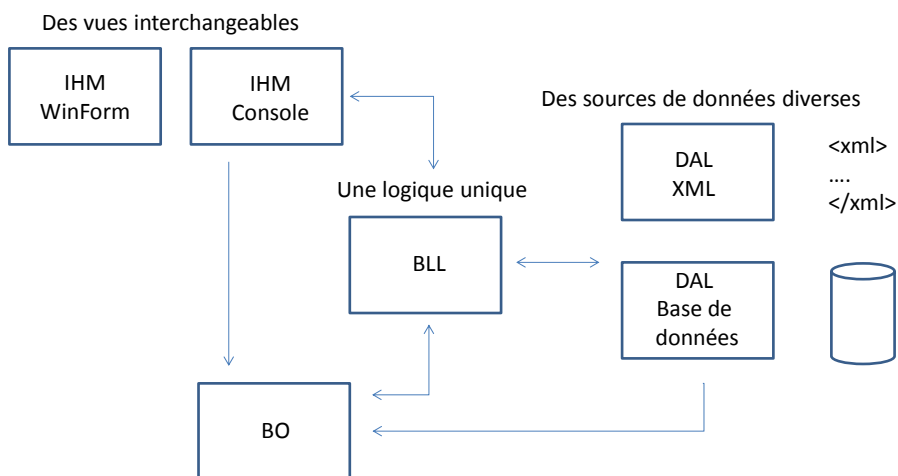
- La couche métier (BO) :
 - Elle décrit les entités qui structurent le domaine de l'application.
 - Elle structure les données montées en mémoire.
 - Elle assure la sécurité et l'intégrité des données en mémoire.
- La couche d'accès aux données (DAL) :
 - Elle s'occupe de prendre les entités (BO) et de les persister dans une source de données et réalise l'opération inverse.



www.eni-ecole.fr

n° 7

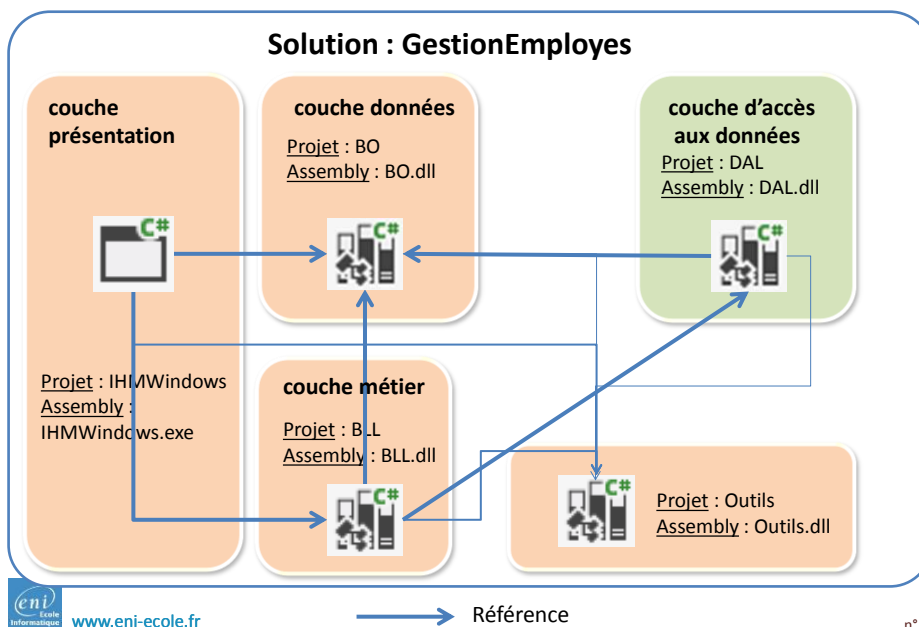
Présentation des couches



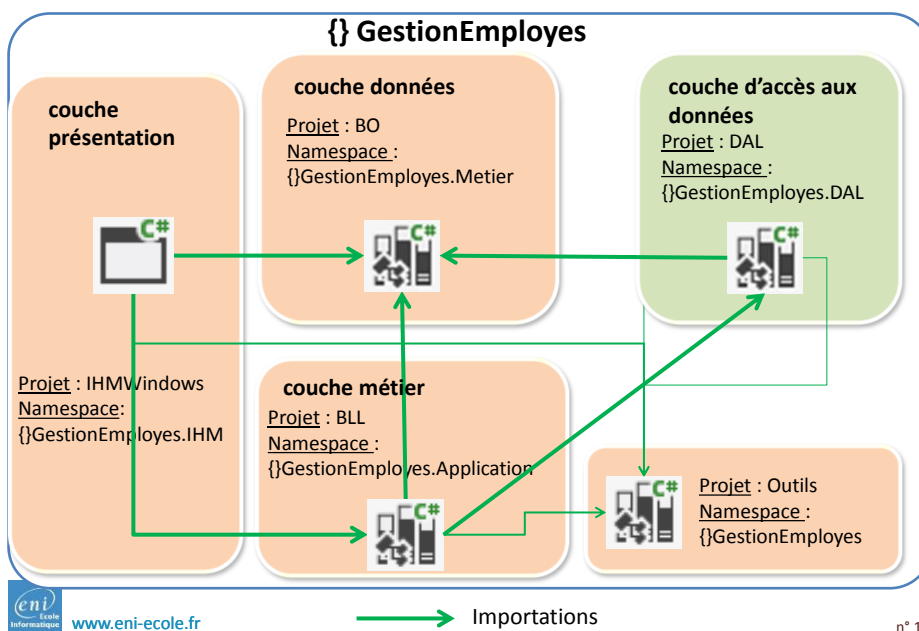
www.eni-ecole.fr

n° 8

Architecture retenue : organisation physique



Architecture retenue : organisation logique



Persistence des données

Module 2

XML



www.eni-ecole.fr

n° 11

Contenu du module

- Origine
- Grammaire: document bien formé
- Grammaire: document valide
 - Principes
 - DTD (Document type Définition)
 - XSD (XML Schema Definition)
- Manipulation avec le Framework .NET
 - Utilisation de DOM (Document Object Model)
 - Utilisation de XPath dans le DOM
- TP



www.eni-ecole.fr

n° 12

Origine

- XML pour *eXtensible Markup Language* (ou langage balise extensible)
- Pourquoi le XML?
 - Le HTML est uniquement un langage de description de l'affichage des données
 - Les données sont donc mélangées avec la manière dont on souhaite les présenter
 - Aucune possibilité de sauvegarde, traitement des données n'est possible au travers d'un fichier HTML
 - XML est alors apparu afin de pouvoir structurer les données indépendamment de leur présentation
- La puissance du XML
 - Le XML est utilisé comme un langage de balises définies par le développeur
 - Le XML structure et décrit les données qu'il contient



www.eni-ecole.fr

n° 13

Grammaire: document bien formé 1/3

- Un document XML qui obéit aux règles syntaxiques du langage XML est un document bien formé
- Exemple

```
<?xml version="1.0" encoding="utf-8" ?>
<!--un commentaire-->
<biblio>
  <livre lang="fr">
    <titre>Les rivières pourpres</titre>
    <auteur>Jean-Christophe Grangé</auteur>
  </livre>
  <livre>
    <titre>Visual Basic 2010</titre>
    <auteur>Thierry Groussard</auteur>
  </livre>
</biblio>
```

Prologue:

- C'est un fichier xml
- Utilisant les règles de la version 1.0
- Et l'encodage utf-8

Élément racine (pointe vers <biblio>)

Attribut (pointe vers lang="fr")

Élément fils (pointe vers <livre> à l'intérieur de <biblio>)



www.eni-ecole.fr

n° 14

Grammaire: document bien formé 2/3

- Règles à respecter:
 - Un document XML est sensible à la casse:


```
<auteur>Thierry Groussard</Auteur>
```
 - Pas d'entrelacement de balises:


```
<titre>Visual Basic 2010
          <auteur>Thierry Groussard
          </titre>
          </auteur>
```
 - Les caractères autorisés
 - Tous les caractères alphanumériques accentués ou pas
 - Les caractères suivants : « - », « : », « _ », « . »
 - Une balise ne commence jamais par un chiffre



www.eni-ecole.fr

n° 15

Grammaire: document bien formé 3/3

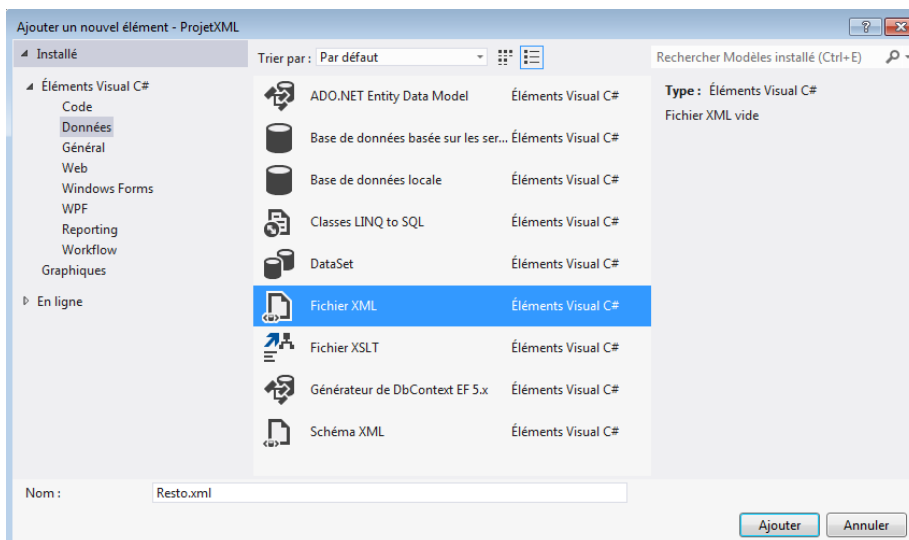
- En définitive, un document XML est bien formé si:
 - Le document possède un prologue valide
 - Toutes les balises sont fermées
 - Les éléments sont convenablement imbriqués
 - Les valeurs des attributs sont placées entre guillemets
 - Les noms des éléments ne contiennent que des caractères autorisés
 - Il y a un et un seul élément racine



www.eni-ecole.fr

n° 16

Ajouter un fichier XML au projet



www.eni-ecole.fr

n° 17

Mise en œuvre

- Réalisez un fichier XML listant les menus d'un restaurant
- Consignes:
 - Un menu peut être composé:
 - D'une entrée au choix
 - D'un plat au choix
 - D'un fromage au choix
 - D'un dessert au choix
 - D'un café au choix
 - Un menu est caractérisé par son type et son prix
 - Chaque composant du menu est caractérisé par un nom, un nombre de calories et éventuellement un parfum (pour les glaces) et un fruit (pour les tartes)



www.eni-ecole.fr

n° 18

Grammaire: document valide / Principes

- Un document XML qui obéit aux règles de construction définies dans le schéma associé est un document valide
- Il existe 2 méthodes pour définir ces règles:
 - Le DTD (*Document Type Definition*)
 - Le XSD (*Xml Schema Definition*)
- Le DTD
 - C'est un fichier texte avec l'extension .dtd
 - Ce n'est pas un document XML
 - Utilise une grammaire spécifique permettant de décrire la structure attendue d'un document XML
- Le XSD
 - C'est un fichier texte avec l'extension .xsd
 - C'est un document XML
 - Sa grammaire est définie dans l'espace de nom: <http://www.w3.org/2001/XMLSchema>
 - Apport: permet de typer les éléments



www.eni-ecole.fr

n° 19

Grammaire: document valide / DTD 1/4

- Exemple

```

<!ELEMENT biblio (livre*)>
<!ELEMENT livre (titre, auteur+)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT auteur (#PCDATA)>

<!--
  Structure d'un élément complexe
  Structure d'un élément simple
  Structure d'un attribut
-->

<!--
  lang CDATA "fr"
  type (roman|nouvelles|bd) #IMPLIED
-->

```

- La balise <!ELEMENT> permet de déclarer chaque élément d'un document XML
- La balise <!--> permet de déclarer l'ensemble des attributs d'un élément
- L'exemple n'est pas exhaustif dans les possibilités offertes par le DTD
 - Pour plus d'informations, vous pouvez consulter:
 - Le RI XML par la pratique – Bases indispensables et cas pratiques



www.eni-ecole.fr

n° 20

Grammaire: document valide / DTD 2/4

■ Règles de déclaration d'un élément

■ `<!ELEMENT nom type_element>`

■ `type_element`

- Soit une liste de sous balises

```
<!ELEMENT livre (titre, auteur+)>
```

- Les opérateurs:

- `+` : 1 ou plus
- `*` : 0 ou plus
- `?` : optionnel

- Les séparateurs:

- La virgule : impose l'ordre
- Le pipe (`|`) : ordre quelconque

- Soit des données textuelles

```
<!ELEMENT auteur (#PCDATA)>
```

- Soit rien

```
<!ELEMENT elementAvecRienDessous EMPTY>
```

- Soit n'importe quoi

```
<!ELEMENT elementAvecNImporteQuoiDessous ANY>
```



www.eni-ecole.fr

n° 21

Grammaire: document valide / DTD 3/4

■ Règles de déclaration d'un attribut

■ `<!ATTLIST balise [attribut valeur option]>`

■ `attribut`: le nom de l'attribut

■ `valeur`:

- Soit une liste de valeurs autorisées séparées par un pipe (`|`)

```
type (roman|nouvelles|bd)
```

- Soit une valeur quelconque

```
lang CDATA
```

- Soit une valeur quelconque sans espace ni caractères spéciaux

```
unAttribut NMTOKEN
```

■ `option`: Obligatoire

- Soit `#IMPLIED` : l'attribut est optionnel
- Soit `#REQUIRED` : l'attribut est obligatoire
- Soit `#FIXED` : l'attribut est une valeur fixe
- Soit « uneValeur » : l'attribut a une valeur par défaut s'il n'est pas renseigné

```
<!ATTLIST livre
  lang CDATA "fr"
  type (roman|nouvelles|bd) #IMPLIED
  unAttribut NMTOKEN #REQUIRED
>
```



www.eni-ecole.fr

n° 22

Grammaire: document valide / DTD 4/4

- Le DTD peut être interne ou externe:
 - Interne: le contenu du DTD est positionné dans le fichier XML décrit
 - Externe: le contenu du DTD est dans un fichier à part.
 - Cette solution est à privilégier dans l'optique de réutilisation

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<!DOCTYPE biblio SYSTEM "Bibliotheque.dtd">
<!--un commentaire-->
<biblio>
  <livre lang="fr">
    <titre>Les rivières pourpres</titre>
    <auteur>Jean-Christophe Grangé</auteur>
  </livre>
  <livre>
    <titre>Visual Basic 2010</titre>
    <auteur>Thierry Groussard</auteur>
  </livre>
</biblio>
```

Attribut permettant de dire que le fichier XML ne se suffit pas à lui-même

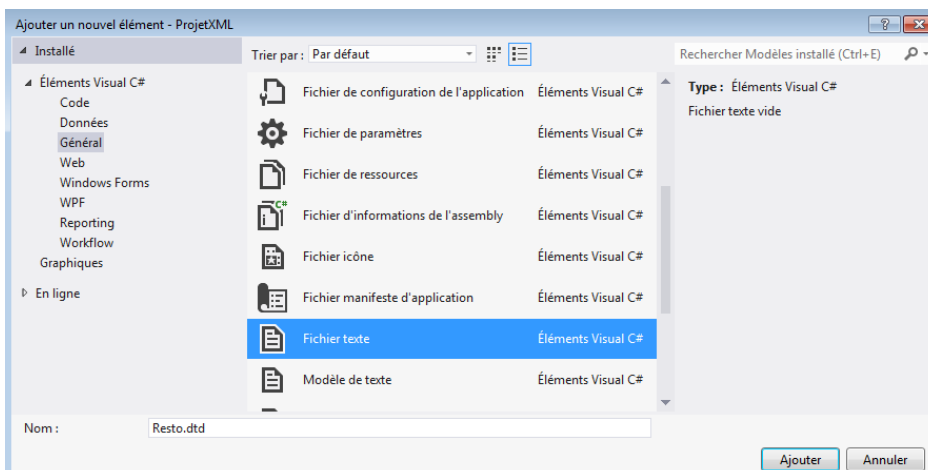
La balise <!DOCTYPE> permet d'associer une ressource externe correspondant à la grammaire de la balise biblio



www.eni-ecole.fr

n° 23

Ajouter un fichier DTD au projet



www.eni-ecole.fr

n° 24

Mise en œuvre / DTD

- TD
 - Mise en place commune d'un DTD pour le fichier xml décrivant les menus d'un restaurant
 - Correction du fichier XML afin qu'il respecte les règles mises en place dans le DTD
- Limite principale:
 - Le typage des données est faible



www.eni-ecole.fr

n° 25

Grammaire: document valide / XSD 1/3

Exemple

```
<?xml version="1.0" encoding="utf-8" ?>
<!--Created with Liquid XML Studio - FREE Community Edition 7.0.1.654 (http://www.liquid-technologies.com)-->
<xs:schema elementFormDefault="qualified" targetNamespace="http://www.eni-ecole.fr/biblio" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:attribute name="type">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="roman" />
        <xs:enumeration value="nouvelles" />
        <xs:enumeration value="bd" />
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute default="fr" name="lang" type="xs:string" />
  <xs:element name="titre" />
  <xs:element name="auteur" />
  <xs:element name="biblio">
    <xs:complexType>
      <xs:sequence minOccurs="1" maxOccurs="unbounded">
        <xs:element name="livre">
          <xs:complexType>
            <xs:sequence minOccurs="1" maxOccurs="1">
              <xs:element xmlns:q1="http://www.eni-ecole.fr/biblio" minOccurs="1" maxOccurs="1" ref="q1:titre" />
              <xs:element xmlns:q2="http://www.eni-ecole.fr/biblio" minOccurs="1" maxOccurs="1" ref="q2:auteur" />
            </xs:sequence>
            <xs:attribute xmlns:q1="http://www.eni-ecole.fr/biblio" ref="q1:type" />
            <xs:attribute xmlns:q2="http://www.eni-ecole.fr/biblio" ref="q2:lang" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Attributs

Descriptif du schéma avec le namespace associé

Eléments simples

Elément complexe

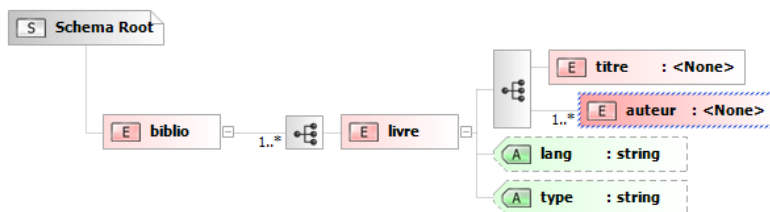


www.eni-ecole.fr

n° 26

Grammaire: document valide / XSD 2/3

- Représentation avec un éditeur WysiWyg (ici XMLStudio)



www.eni-ecole.fr

n° 27

Grammaire: document valide / XSD 3/3

- Le XSD est dans un fichier externe d'extension .xsd
- L'association se fait de cette manière si le fichier xsd est dans le même répertoire que le fichier xml:

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<!--un commentaire-->
<biblio xmlns="http://www.eni-ecole.fr/biblio">
  <livre lang="fr">
    <titre>Les rivières pourpres</titre>
    <auteur>Jean-Christophe Grangé</auteur>
  </livre>
  <livre>
    <titre>Visual Basic 2010</titre>
    <auteur>Thierry Groussard</auteur>
  </livre>
</biblio>
```

- Sinon:

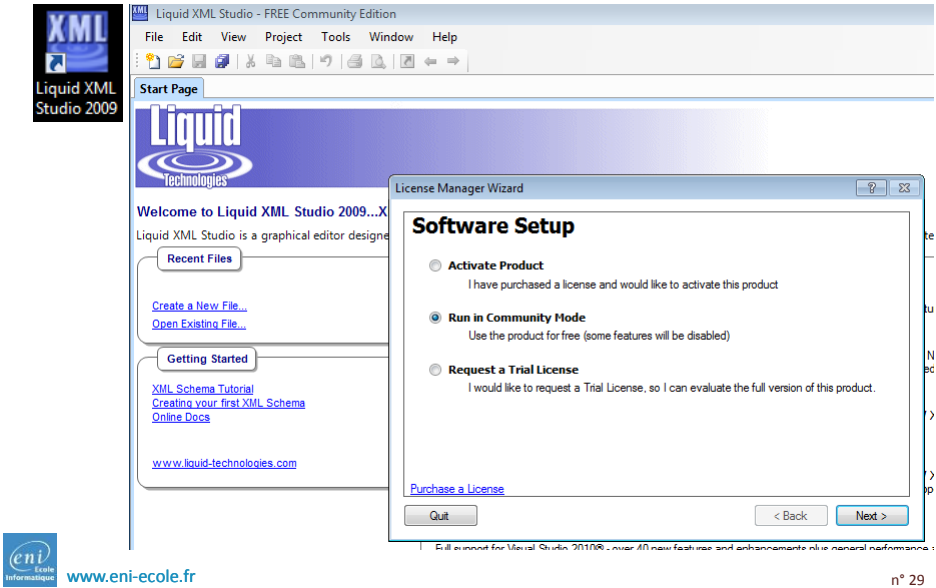
```
<biblio xmlns="http://www.eni-ecole.fr/biblio"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.eni-ecole.fr/biblio bibliotheque.xsd">
```



www.eni-ecole.fr

n° 28

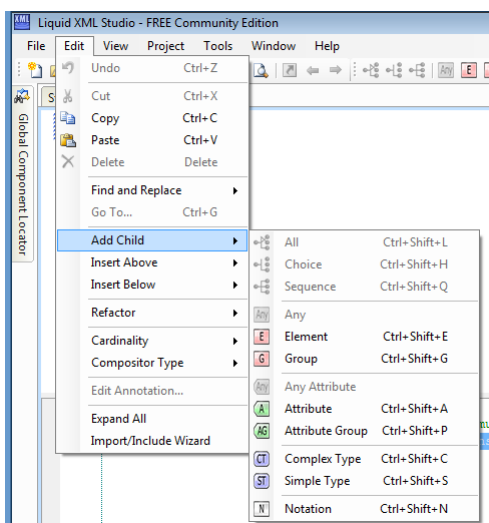
Créer un schéma avec XMLStudio



Créer un schéma avec XMLStudio



Créer un schéma avec XMLStudio



Relations

Liens

Attributs

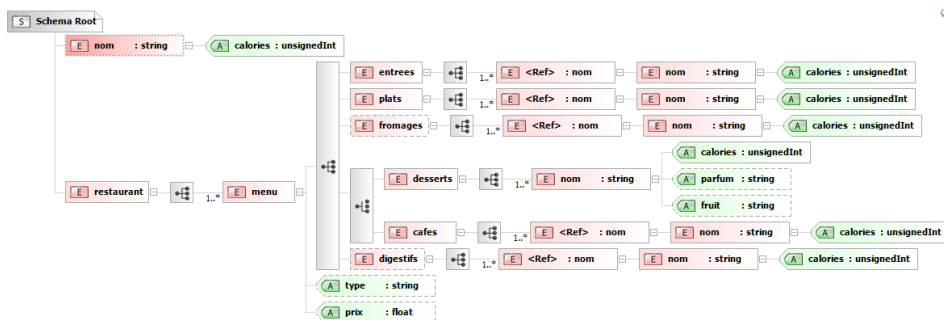


www.eni-ecole.fr

n° 31

Mise en œuvre / XSD

- TD
 - Mise en place d'un XSD pour le fichier xml décrivant les menus d'un restaurant
 - Correction du fichier XML afin qu'il respecte les règles mises en place dans le XSD



www.eni-ecole.fr

n° 32

Manipulation avec le Framework .NET

- RI 455-463
- DOM pour Document Object Model
 - `using System.Xml;`
 - C'est une librairie permettant de représenter un fichier XML sous la forme d'un graphe d'objet
 - Principales classes disponibles:
 - XmlDocument
 - XmlNode
 - XmlAttribute
- Compléments fonctionnels
 - `using System.Xml.XPath;`
 - Utilisation de XPath; Le Framework .NET met à disposition une librairie permettant de réaliser des requêtes XPath sur un XmlDocument
 - Ressources sur Xpath:
 - <http://www.w3schools.com/xpath/default.asp>



www.eni-ecole.fr

n° 33

TP

- Mettez en place l'export XML pour les services et les employés dans deux fichiers différents.
- Assurez vous de la validité de votre fichier XML en fonction du fichier xsd fourni.



www.eni-ecole.fr

n° 34

Persistence des données

Module 3 Sérialisation



www.eni-ecole.fr

n° 35

Contenu du module

- Définition
- Les différents types de sérialisation
- Le Framework .NET et la sérialisation
- La sérialisation binaire
- La sérialisation SOAP
- La différence entre les 2 types de sérialisation
- Complément d'architecture

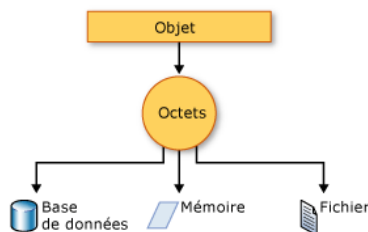


www.eni-ecole.fr

n° 36

Définition

- La sérialisation c'est quoi?
 - C'est le fait de transcrire, dans un fichier, des données présentes en mémoire



- A quoi ça sert?
 - Ca sert à rendre des données persistantes afin de pouvoir les réutiliser plus tard
 - Ca sert à transférer des données d'un endroit à un autre



www.eni-ecole.fr

n° 37

Définition

- Comment ça marche?
 - L'introspection permet la sérialisation
 - Heureusement, des librairies le font pour nous!!
- La sérialisation est toujours associée à la désérialisation
 - La désérialisation est l'opération inverse de la sérialisation
 - La désérialisation permet de mettre en mémoire des objets qui ont été préalablement transcrits dans un fichier
 - ATTENTION: Lors de la désérialisation, on ne crée pas une nouvelle instance d'un objet en utilisant son constructeur



www.eni-ecole.fr

n° 38

Les différents types de sérialisation

- Il existe différents types de sérialisation
- Le type est fonction du format d'écriture des données dans le fichier
- On peut citer:
 - La sérialisation binaire
 - La sérialisation SOAP
 - La sérialisation XML
 - La sérialisation « personnalisée »
- Les différences:
 - La sérialisation binaire est spécifique au langage utilisé pour faire la sérialisation
 - La sérialisation SOAP respecte le standard du même nom. Le format est donc interprétable par différents langages
 - Nous pouvons bien sûr créer notre propre système de sérialisation avec notre propre format de stockage



www.eni-ecole.fr

n° 39

Le Framework .NET et la sérialisation

- Le Framework .NET propose des bibliothèques permettant la sérialisation
 - Elles sont situées dans le namespace: System.Runtime.Serialization
- Un certain nombre d'éléments du Framework .NET sont déjà sérialisables
- Comment influencer sur la sérialisation?
 - Pour les classes que l'on a écrites, il faudra indiquer ce qui est sérialisable et ce qui ne l'est pas
 - Les classes sérialisables seront annotées: `[Serializable]`
 - Les attributs non sérialisables seront annotés: `[NonSerialized]`
 - Il est possible d'intégrer des traitements en cours de sérialisation:

```
[OnSerializing()]
internal void OnSerializingMethod(StreamingContext context)
{
    member2 = "This value went into the data file during serialization.";
}

[OnSerialized()]
internal void OnSerializedMethod(StreamingContext context)
{
    member2 = "This value was reset after serialization.";
}
```



www.eni-ecole.fr

n° 40

Le Framework .NET et la sérialisation

- Comment influencer sur la désérialisation?
 - Lors de la désérialisation, on pourra déclencher un traitement particulier
 - Soit d'une façon globale, en fin de désérialisation

```
IDeserializationCallback
public void OnDeserialization(object sender)
```

- Soit au cours du traitement

```
[OnDeserializing()]
internal void OnDeserializingMethod(StreamingContext context)
{
    member3 = "This value was set during deserialization";
}

[OnDeserialized()]
internal void OnDeserializedMethod(StreamingContext context)
{
    member4 = "This value was set after deserialization.";
}
```



www.eni-ecole.fr

n° 41

La sérialisation binaire

- L'objectif est de **sérialiser** une **instance de quelque-chose** dans un **fichier**
- Il y a donc 3 éléments qui entre en jeu:
 - Le quoi: la sérialisation en utilisant la classe `BinaryFormatter`
 - Le quelque-chose de n'importe quel type `Object`
 - Le fichier: utilisation arbitraire de la classe `FileStream`
- Les namespaces utilisés sont:

```
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;
```



www.eni-ecole.fr

n° 42

Comprendre l'erreur : Non sérialisable

- La sérialisation d'un objet Ville entraîne l'erreur suivante

Une erreur est survenue: Le type 'ProjetSerialisation.Ville' dans l'assembly 'ProjetSerialisation, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null' n'est pas marqué comme sérialisable.

- Pourquoi cela fonctionne-t-il avec une chaîne de caractères ?

```

C#  C++  VB
[SerializableAttribute]
[ComVisibleAttribute(true)]
public sealed class String : IComparable,
    ICloneable, IConvertible, IComparable<string>, IEnumerable<char>,
    IEnumerable, IEquatable<string>

```

- Solution

```

[Serializable]
public class Livre

```



www.eni-ecole.fr

n° 43

Mise en œuvre / Sérialisation binaire

- Comment sérialiser une chaîne de caractère?
- Comment sérialiser un objet?
- Comment sérialiser une liste d'objets?




www.eni-ecole.fr

n° 44

La sérialisation SOAP

- L'objectif est de **sérialiser une instance de quelque-chose** dans un **fichier**
- Il y a donc 3 éléments qui entre en jeu:
 - Le quoi: la sérialisation en utilisant la classe `SoapFormatter`
 - Le quelque-chose de n'importe quel type `Object`
 - Le fichier: utilisation arbitraire de la classe `FileStream`
- Les namespaces utilisés sont:


```
using System.IO;
using System.Runtime.Serialization.Formatters.Soap;
```
- Les références supplémentaires:

 `System.Runtime.Serialization.Formatters.Soap` 4.0.0.0



www.eni-ecole.fr

n° 45

Mise en œuvre / Sérialisation SOAP

- Comment sérialiser une chaîne de caractère?
- Comment sérialiser un objet?
- Comment sérialiser une liste d'objets?



www.eni-ecole.fr

n° 46

La différence entre les 2 types de sérialisation

- Voici l'erreur typique que l'on a en faisant de la sérialisation SOAP:

Une erreur est survenue: Le sérialiseur ne prend pas en charge la sérialisation des types génériques : System.Collections.Generic.List'1[ProjetSerialisation.Ville].

- Pourquoi cette sérialisation ne gère-t-elle pas les types génériques?
 - Chaque langage a sa propre implémentation des types génériques
 - Le format SOAP n'a donc pas défini une manière d'écrire les listes.
 - Il faut utiliser les tableaux pour sauvegarder des collections d'objets au format SOAP



www.eni-ecole.fr

n° 47

TP

- Mettre en place la sérialisation dans le projet « Gestion des employés ».
- Vous pouvez partir du projet de base ou du projet gérant déjà l'export au format XML
- Votre projet permet donc de sauvegarder et de récupérer les services et employés saisis.
- Option : optimiser la réaffectation du service et du chef pour chaque employé désérialisé.



www.eni-ecole.fr

n° 48

Complément d'architecture

- Le code de la sérialisation binaire et de la sérialisation SOAP est très semblable
 - Les classes BinaryFormatter et SOAPFormatter diffèrent seulement par leur nom
 - La sérialisation SOAP ne permet pas la sérialisation des génériques
- Comment modifier ce que l'on a fait pour:
 - Limiter le code redondant?
 - Faire en sorte de pouvoir réaliser de façon transparente une sérialisation binaire ou une sérialisation SOAP?

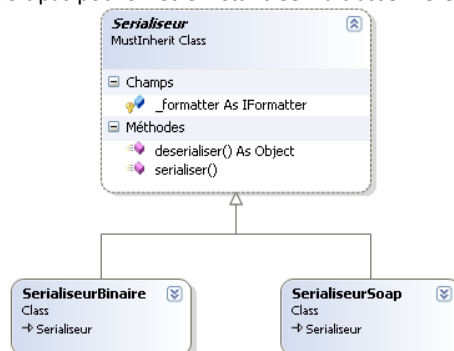


www.eni-ecole.fr

n° 49

Comment limiter le code redondant?

- Quand on souhaite limiter le code redondant, il faut garder à l'esprit les spécificités
 - L'élément IFormatter est différent selon que l'on est dans une sérialisation binaire ou une sérialisation SOAP
- La solution est la mise en place d'un héritage
 - La classe mère contient le code commun
 - Les classes filles contiennent le code spécifique
 - La classe mère ne doit pas pouvoir être instanciée. La classe mère est donc déclarée abstraite

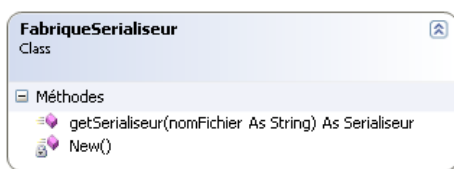


www.eni-ecole.fr

n° 50

Comment simplifier la création du bon sérialiseur?

- La difficulté est de savoir quand instancier un `SérialiseurBinaire` et quand instancier un `SérialiseurSOAP`
 - Le caractère discriminant est connu: l'extension du fichier
 - Le patron de conception **Fabrique** a pour objectif de gérer la création d'un objet dont le type est dérivé d'un type abstrait (ici `Sérialiseur`) en fonction d'un critère donné
 - C'est cette Fabrique qui possèdera le code permettant de savoir quel type concret (ici `SérialiseurBinaire` et `SérialiseurSOAP`) instancier
 - La méthode **`getSérialiseur`** est une méthode de classe

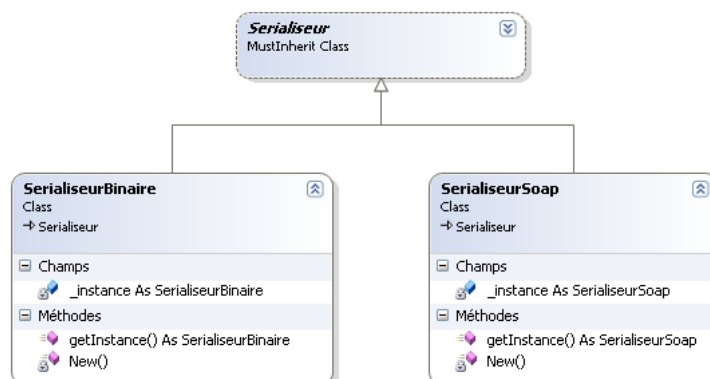


www.eni-ecole.fr

n° 51

Comment limiter le nombre d'instance à 1?

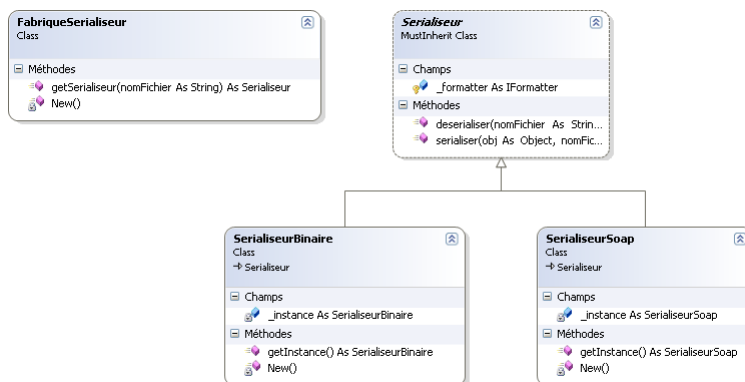
- Dans le cas d'une sérialisation, on ne maintient aucune donnée spécifique à la sérialisation en cours dans les classes `SérialiseurBinaire` et `SérialiseurSOAP`
- Il faut donc faire en sorte de pouvoir réutiliser la même instance pour les sérialisations suivantes de même type
- Le patron de conception Singleton est tout à fait approprié



www.eni-ecole.fr

n° 52

Architecture finale de la DAL



www.eni-ecole.fr

n° 53

Persistence des données

Module 4

Accès aux données stockées dans une base de données relationnelle



www.eni-ecole.fr

n° 54

Contenu du module

- RI Chapitre 8 (p. 363-385)
- Les différents modes de fonctionnement
- Le mode connecté
 - Découverte des classes
 - Les fournisseurs disponibles
 - Mise en œuvre
- Compléments d'architecture
 - Une application indépendante du SGBD
- TD
- Utiliser un procédure stockée
- TP



www.eni-ecole.fr

n° 55

Les différents modes de fonctionnement

- ADO.NET est un ensemble de classes permettant la manipulation des données
- Deux modes de fonctionnement sont disponibles dans ADO.NET:
 - Le mode connecté
 - Un accès « permanent » à la base de données est nécessaire
 - Aucune donnée n'est maintenue dans l'application; à chaque fois que c'est nécessaire, les données sont lues dans la base de données
 - Avantages: facilité de mise en œuvre, les données sont toujours à jour
 - Inconvénient: une connexion quasi permanente est nécessaire vers la base de données
 - Le mode non connecté
 - Une grande quantité de données peut être maintenue au niveau de l'application; une connexion permanente à la base de données n'est pas nécessaire
 - Avantages: utilisation possible de l'application sans nécessité d'une connexion
 - Inconvénients: les données ne sont pas forcément à jour; des risques de conflits lors de la mise à jour sont élevés



www.eni-ecole.fr

n° 56

Le mode connecté / Découverte des classes

- Le travail avec une base de données peut se résumer avec les étapes suivantes:
 - Se connecter à la base de données
 - Exécuter une requêtes (avec éventuellement des paramètres)
 - Lire le résultat de la requête
 - Se déconnecter
- A chacune de ces étapes est associée une ou plusieurs classes de ADO.NET
 - Pour SQLServer, les classes seront les suivantes:
 - Pour la connexion et la déconnexion: **SqlConnection**
 - Pour l'exécution d'une requête: **SqlCommand (SqlParameter)**
 - Pour lire le résultat d'une requête retournant un jeu de résultats: **SqlDataReader**
- Il faut aussi une chaîne de connexion vers la base de données (RI p 374-375):
 - Exemple avec authentification windows intégrée:
 - «**server**=[adresse ip ou nom serveur];**database**=[nom base de données];**integrated security=true**»
 - Exemple avec authentification auprès du SGBD:
 - « **server**=[adresse ip ou nom serveur];**database**=[nom base de données];**user**=[nom utilisateur];**password**= [mot de passe]»



www.eni-ecole.fr

n° 57

Le mode connecté / Les fournisseurs disponibles 1/3

- RI p369
- 4 fournisseurs sont disponibles par défaut:
 - Le fournisseur SQLServer
 - Le fournisseur OLE DB
 - Le fournisseur ODBC
 - Le fournisseur Oracle



www.eni-ecole.fr

n° 58

Mise en œuvre

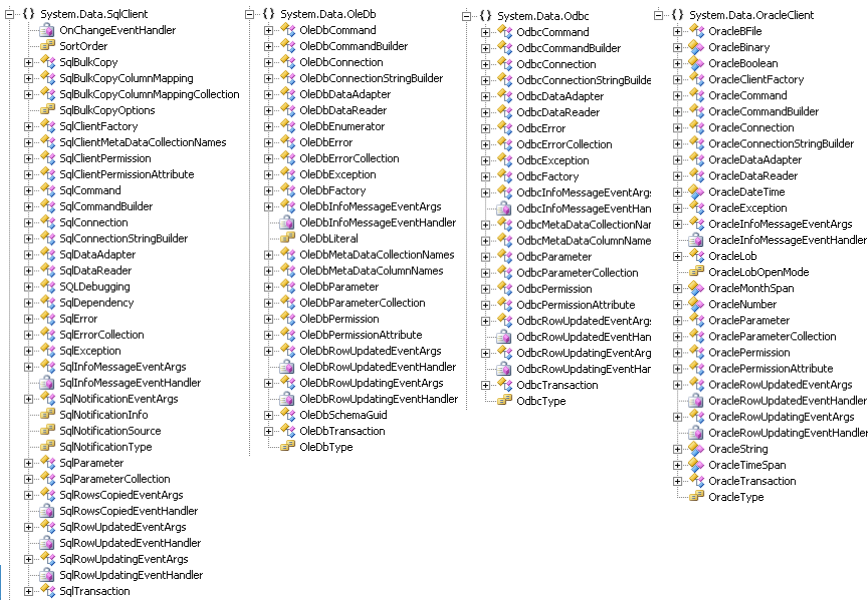
- Lister les fournisseurs disponibles.



www.eni-ecole.fr

n° 59

Le mode connecté / Les fournisseurs disponibles 2/3

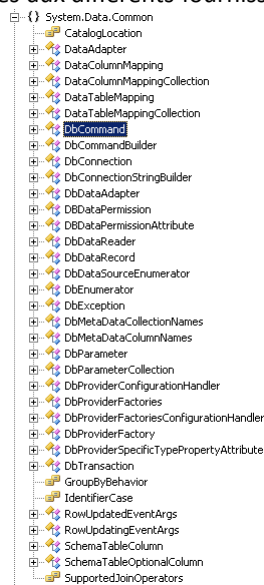
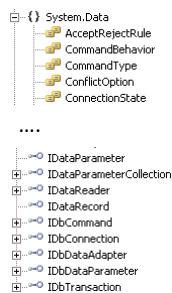


www.eni-ecole.fr

n° 60

Le mode connecté / Les fournisseurs disponibles 3/3

- Les classes et les interfaces parentes communes aux différents fournisseurs



www.eni-ecole.fr

n° 61

Mise en œuvre

- Accéder à SQLServer
 - TD « Gestion des logins »



www.eni-ecole.fr

n° 62

Mise en œuvre

- Accéder à un autre SGBD
 - TD « Gestion des logins »
- Référencer un nouveau fournisseur ADO.Net (PostgreSQL)
 - Ajouter en référence du projet les assemblys Npgsql.dll et Mono.security.dll



www.eni-ecole.fr

n° 63

Mise en œuvre

- Un premier constat ?
 - Le code est identique à l'exception des types des classes (NpgsqlConnection/SqlConnection....)
- Comment faire en sorte d'utiliser le même code quel que soit le SGBD ?
 - S'appuyer sur les interfaces de l'espace de noms **System.Data;**
- Un seul souci subsiste: l'instanciation de la connexion
 - S'appuyer sur une Fabrique de connexions.



www.eni-ecole.fr

n° 64

Mise en œuvre

- Mettre en place toutes les méthodes du CRUD (Create Read Update Delete)
 - Mise en place d'une architecture en couche
 - Namespace GUI:
 - Classe IHMLogin
 - Namespace BLL:
 - Classe MgtLogin
 - Namespace BO:
 - Classe Login
 - Namespace DAL:
 - SQLAcces
 - SQLLogin
- Utiliser une procédure stockée **RI p384**



www.eni-ecole.fr

n° 65

Compléments d'architecture 1/4

- Une application indépendante du SGBD
 - Utilisation de la fabrique de fabriques : **System.Data.Common.DbProviderFactories**

The screenshot shows the Visual Studio IDE. On the left, the 'Solution Explorer' displays a project structure with files like DbParameter, DbParameterCollection, DbProviderConfigurationHandler, DbProviderFactories, DbProviderFactoriesConfigurationHandler, DbProviderFactory, DbProviderSpecificTypePropertyAttribute, DbTransaction, GroupByBehavior, IdentifierCase, RowUpdatedEventArgs, RowUpdatingEventArgs, and SchemaTableColumn. The 'DbProviderFactories' file is selected. On the right, the 'Code' window shows the implementation of the `GetFactory` method. The method signature is `GetFactory(String) As System.Data.Common.DbProviderFactory`. The code defines a public shared function `GetFactory` that takes a `providerInvariantName` as a string and returns a `System.Data.Common.DbProviderFactory`. The summary states: 'Retourne une instance de System.Data.Common.DbProviderFactory.' The parameters section indicates: `providerInvariantName`: Nom invariant d'un fournisseur. The return values section states: 'Instance de System.Data.Common.DbProviderFactory pour un nom de fournisseur spécifié.'

- La méthode de classe **GetFactory(String)** va nous retourner une fabrique spécifique du fournisseur de données en fonction du paramètre **providerInvariantName**
 - Ce paramètre est celui affiché dans la colonne **InvariantName** lorsque l'on affiche l'ensemble des fournisseurs disponibles
 - La fabrique instanciée sera celle qui est écrite dans la colonne **AssemblyQualifiedName**
 - SQLClientFactory
 - OleDbFactory
 - OdbcFactory
 - OracleClientFactory

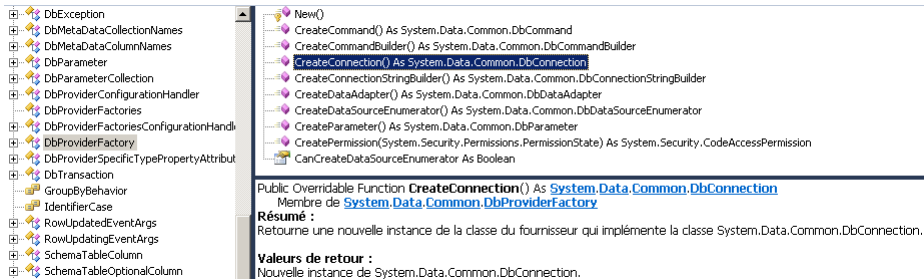


www.eni-ecole.fr

n° 66

Compléments d'architecture 2/4

- Utilisation de la fabrique abstraite `System.Data.Common.DbProviderFactory`



- Exemple pour créer une connexion SQLServer

```
private IDbConnection _cnx;
_cnx = DbProviderFactories.GetFactory("System.Data.SqlClient").CreateConnection();
```

- Exemple pour créer une connexion Oracle

```
private IDbConnection _cnx;
_cnx = DbProviderFactories.GetFactory("System.Data.OracleClient").CreateConnection();
```



www.eni-ecole.fr

n° 67

Compléments d'architecture 3/4

- Référencer les fournisseurs supplémentaires
 - Pour pouvoir utiliser avec le `DbProviderFactories` un fournisseur autre que ceux présents par défaut dans le Framework .NET, il faut le référencer dans le fichier **app.config**
 - Ce fournisseur est généralement téléchargeable sur le site de l'éditeur du SGBD
 - Sur ce site, on vous indique comment référencer ce fournisseur dans votre application .NET

```
<system.data>
  <DbProviderFactories>
    <add name="Npgsql Data Provider"
      invariant="Npgsql"
      description=".Net Framework Data Provider for PostGreSql Server"
      type="Npgsql.NpgsqlFactory, Npgsql, Version=2.0.11.91, Culture=neutral,
        PublicKeyToken=5d8b90d52f46fda7"/>
  </DbProviderFactories>
</system.data>
```

- L'utilisation de ce fournisseur se fait de cette manière

```
private IDbConnection _cnx;
_cnx = DbProviderFactories.GetFactory("Npgsql").CreateConnection();
```



www.eni-ecole.fr

n° 68

Compléments d'architecture 4/4

- Externaliser le nom du fournisseur et la chaîne de connexion dans le fichier de configuration

- Utilisation du nœud <connectionStrings> prévu à cet effet

```
<connectionStrings>
  <!--Connection à SQL Server-->
  <add name="cnxBDD"
        connectionString="server=localhost;database=TEST;user=sa;password="
        providerName="System.Data.SqlClient"/>
  <!--Connection à PostgreSQL-->
  <!--<add name="cnxBDD"
        connectionString="server=localhost;port=5432;database=TEST;user=postgres;password=mdp;"
        providerName="Npgsql"/>-->
</connectionStrings>
```

- Lecture de ce nœud
 - Importer l'assembly **System.Configuration**

```
ConnectionStringSettings settings = ConfigurationManager.ConnectionStrings["cnxBDD"];
_cnx = DbProviderFactories.GetFactory(settings.ProviderName).CreateConnection();
_cnx.ConnectionString = settings.ConnectionString;
```



www.eni-ecole.fr

n° 69

TP

- Mettre en place la persistance des données de l'application « Gestion des employés » dans une base de données SQLServer



www.eni-ecole.fr

n° 70

Persistence des données

Module 5 LINQ



www.eni-ecole.fr

n° 71

Contenu du module

- RI Chapitre 9 – p417-427
- Bilan du besoin
- Solution retenue
- Mise en œuvre
 - Les différents fournisseurs LINQ
 - Les concepts de base



www.eni-ecole.fr

n° 72

Bilan du besoin

- Voici quelques techniques pour récupérer des données:
 - Utilisation d'une boucle d'itération sur une collection
 - Utilisation d'une requête SQL sur une table dans une base de données
 - Utilisation d'une requête XPath sur un fichier XML
 - ...
- Il existe autant de techniques d'accéder aux données qu'il y a de techniques pour les stocker
- Pour certaines d'entre elles, le code s'écrit sous forme d'une chaîne de caractères donc non vérifiable par le compilateur
- L'idée est de pouvoir uniformiser au maximum ces techniques en mettant en place un langage « commun » et vérifiable par le compilateur
 - Le SQL est un langage connu par le plus grand nombre; le langage devra donc s'en approcher
 - Le compilateur doit pouvoir compiler ces instructions; le langage devra donc faire partie du VB.NET et du C#



www.eni-ecole.fr

n° 73

Solution retenue

- LINQ est la solution proposée par Microsoft répondant à ces critères
- LINQ pour *Language Integrated Query* (Langage de requête intégrée)
- Le Framework .NET propose des bibliothèques de fournisseur LINQ permettant d'utiliser LINQ avec:
 - des collections .NET Framework
 - des bases de données SQL Server
 - des groupes de données ADO.NET
 - des documents XML
- Le fonctionnement de base est similaire entre tous ces fournisseurs
- Par contre, les objets manipulés et l'exploitation qui en est faite est différente



www.eni-ecole.fr

n° 74

Les différents fournisseurs

- Les principaux fournisseurs sont
 - LINQ to Object
 - utilisable sur toutes collections implémentant l'interface `IEnumerable<T>`
 - [http://msdn.microsoft.com/fr-fr/library/vstudio/bb397919\(v=vs.100\).aspx](http://msdn.microsoft.com/fr-fr/library/vstudio/bb397919(v=vs.100).aspx)
 - LINQ to XML
 - utilisable sur n'importe quelle source XML
 - [http://msdn.microsoft.com/fr-fr/library/vstudio/bb387098\(v=vs.100\).aspx](http://msdn.microsoft.com/fr-fr/library/vstudio/bb387098(v=vs.100).aspx)
 - LINQ to Entities (Fournisseur spécifique pour Entity Framework)
 - Les requêtes sont évaluées sur le serveur de données
 - <http://msdn.microsoft.com/fr-fr/library/vstudio/bb386964.aspx>
- Une syntaxe commune pour accéder à différentes sources de données

```
using System.Linq;
```



www.eni-ecole.fr

n° 75

Les requêtes LINQ

- Elles permettent d'extraire des données d'une collection d'objets
- 2 syntaxes possibles :
 - Sous la forme d'expression de requête


```
var linq = from string prenom in liste
            where prenom.StartsWith("G")
            select prenom;
```
 - Sous la forme de méthodes exposées par `IEnumerable<T>`

```
var linqBis = liste.Where(w => w.StartsWith("G"))
                  .Select(s => s);
```

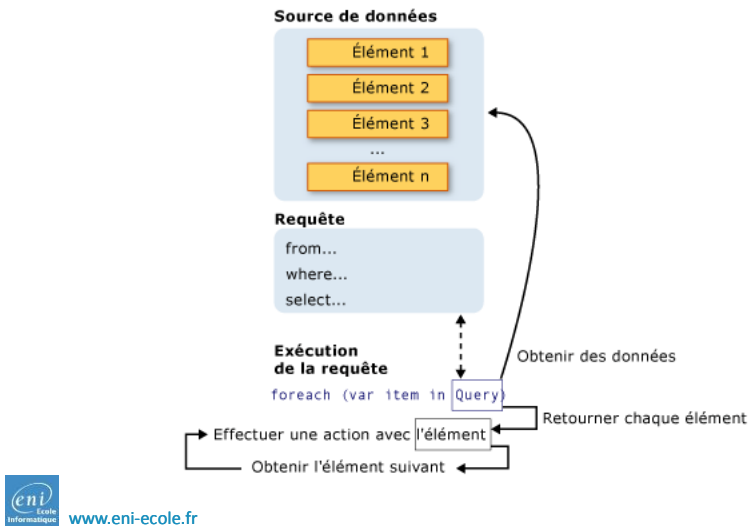


www.eni-ecole.fr

n° 76

Les requêtes LINQ

- Les requêtes LINQ utilisent des mots clés tels que FROM, WHERE, SELECT...
- Elles permettent d'extraire des données d'une collection d'objets



n° 77

Les requêtes LINQ

- Deux étapes :
 - Création de la requête
- Une requête s'exécute à chaque fois qu'on s'intéresse à sa variable

```
var linq = from string prenom in liste
           where prenom.StartsWith("G")
           select prenom;
```

```
foreach (string prenomSelectionne in linq) {
    Console.WriteLine(prenomSelectionne);
}
```

n° 78

LINQ to XML

- Classes XDeclaration, XDocument, XElement, XComment, XAttribute

```
XDeclaration xdeclaration = new XDeclaration("1.0", "utf-8", "oui");
XDocument xdoc = new XDocument(xdeclaration, new XElement("Racine",
    new XComment("Liste des utilisateurs"),
    new XElement("Utilisateur",
        new XAttribute("type", "0"),
        new XAttribute("profil", "cadre"),
        "Tom"),
    new XElement("Utilisateur",
        new XAttribute("type", "1"),
        new XAttribute("profil", "agent"),
        "Samuel"),
    new XElement("Utilisateur",
        new XAttribute("type", "1"),
        new XAttribute("profil", "cadre"),
        "Felix"),
    new XElement("Utilisateur",
        new XAttribute("type", "0"),
        new XAttribute("profil", "agent"),
        "Sandra")
    ));
```



www.eni-ecole.fr

n° 81

Mise en œuvre - Requêtes LINQ to XML

- Démonstration



www.eni-ecole.fr

n° 82

TP

- Sur l'écran FrmServices, présenter dans une DataGridView la liste des employés du service sélectionné.



www.eni-ecole.fr

n° 83

Persistence des données

Module 6 Entity Framework



www.eni-ecole.fr

n° 84

Contenu du module

- Concepts
 - Objectifs
 - Vue d'ensemble
 - Mise en œuvre
- TD
- TP



www.eni-ecole.fr

n° 85

Objectifs

- Le Framework Entity fait partie de la famille des Frameworks ORM (*Object Relational Mapping* ou Mappage Objet Relationnel)
- Dans cette famille on compte également :
 - Le Framework Hibernate (Java) et Nhibernate (.NET)
 - Le Framework Doctrine (PHP)
- L'objectif
 - Rendre transparent la liaison entre les objets métiers et leur stockage en base de données
 - Un fichier de correspondance doit suffir à faire le lien entre les 2 mondes
 - Les manipulations réalisées sur les objets au niveau applicatif doivent être répercutées sur la base de données

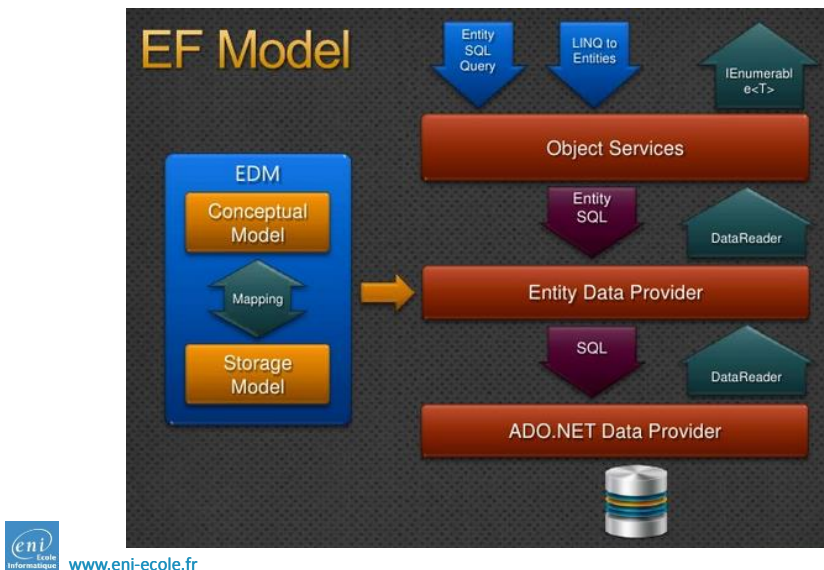


www.eni-ecole.fr

n° 86

Vue ensemble – Architecture 1/5

- Représentation détaillée du Entity Framework

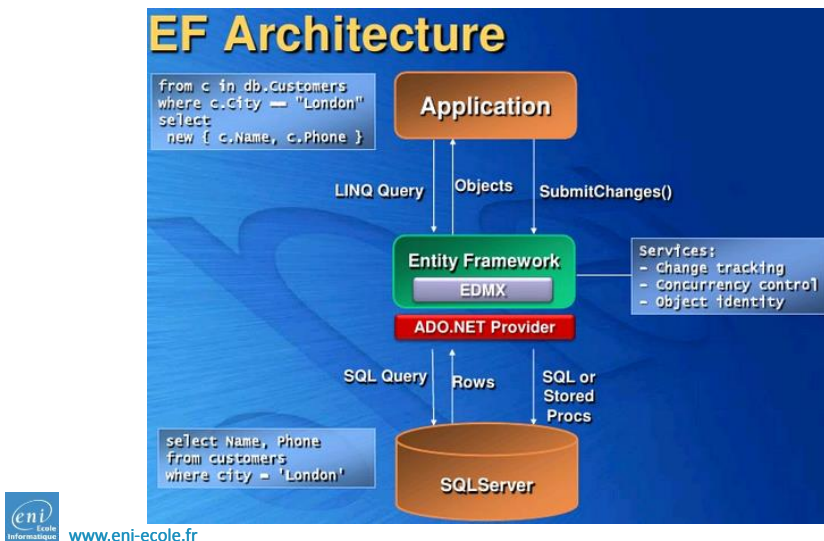


www.eni-ecole.fr

n° 87

Vue d'ensemble – Object Services 2/5

- Zoom sur l'Object Services
 - L'Object Services compte parmi ses principales classes la classe **ObjectContext**.



www.eni-ecole.fr

n° 88

Vue d'ensemble – Object Services 3/5

Object Services Functions

- **Querying**
 - Querying data as objects
 - Shaping query results
 - Composing queries
 - CRUD
 - Lazy loading
 - Inheritance
 - Navigating relationship
- **Managing**
 - Object identities
 - Concurrency
 - Transactions
- **Change tracking**
 - Saving changes
 - Attaching objects
 - Detaching objects
 - Serializing objects

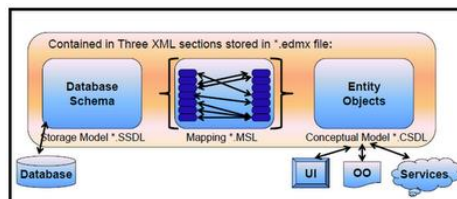


www.eni-ecole.fr

n° 89

Vue d'ensemble – Mapping 4/5

- Zoom sur le Mapping
 - Basiquement, VS 2012 génère une classe par table de la base de données considérée en s'appuyant sur la vision conceptuelle des données
 - Il est possible de modifier cet aspect par la suite
 - La relation entre les classes et les tables correspondantes se fait au travers de fichiers de mapping
 - Pour chaque ligne d'une table, une instance d'une classe sera associée
 - Pour chaque colonne de cette table, un attribut de l'instance de la classe sera associé
 - L'Entity Framework utilise 3 fichiers:
 - Le CSDL (Conceptual Schema Definition Language) pour décrire le modèle métier
 - Le SSDL (Store Schema Definition Language) pour décrire le modèle de stockage
 - Le MSL (Mapping Specification Language) pour décrire le mappage entre le modèle métier et le modèle de stockage



www.eni-ecole.fr

n° 90

Vue d'ensemble – Différentes approches 5/5

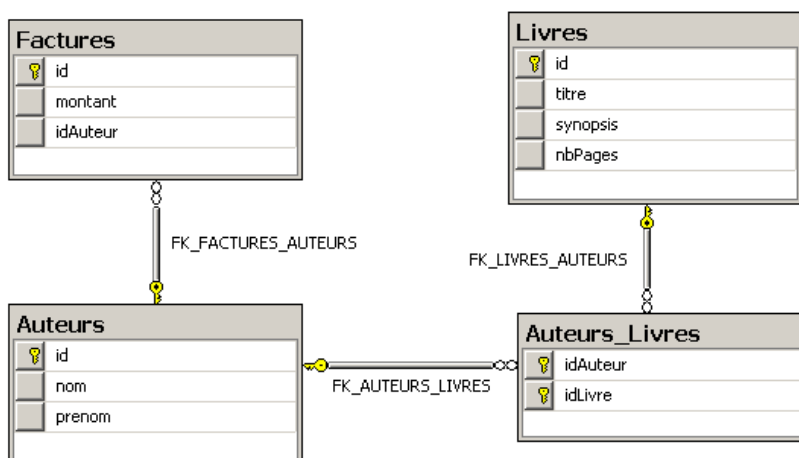
- L'utilisation d'Entity Framework peut se faire au travers de 3 approches différentes :
 - **Database First** : Créer la base de données, les tables, relations, contraintes d'intégrité dans Sql Server et en déduire le modèle objet et les classes correspondantes
 - **Model First** : Créer le modèle de données (entités, propriétés de navigation, ...) dans le designer Visual Studio et en déduire la structure de la base de données ainsi que les classes correspondantes
 - **Code First** : Créer les classes et les propriétés par code et en déduire le modèle relationnel correspondant et la structure de la base de données.
- Il n'y a pas de bon ou mauvais choix. Tout dépend de la complexité de votre modèle de données.



www.eni-ecole.fr

n° 91

Mise en œuvre – Présentation de la base de données

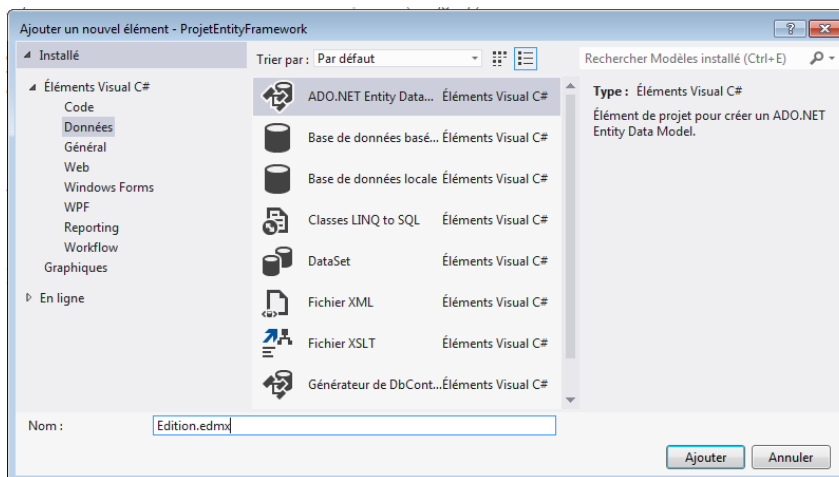


www.eni-ecole.fr

n° 92

Mise en œuvre – Couche Business Object Entity Framework

- Création du composant Entity Data Model

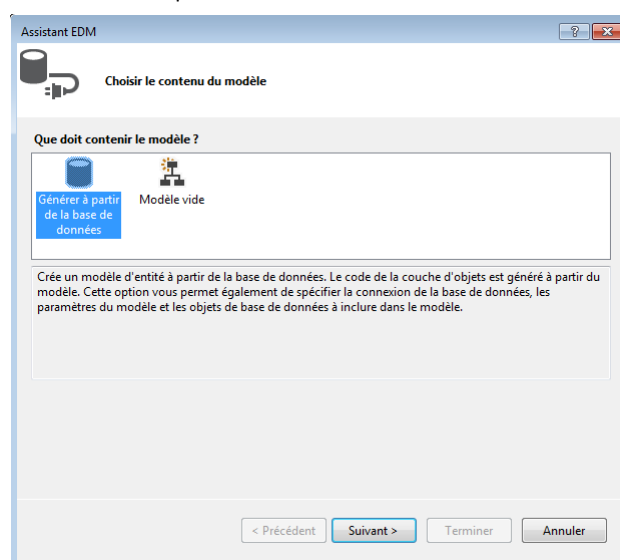


www.eni-ecole.fr

n° 93

Mise en œuvre – Couche Business Object Entity Framework

- Construire le modèle à partir de la base de données

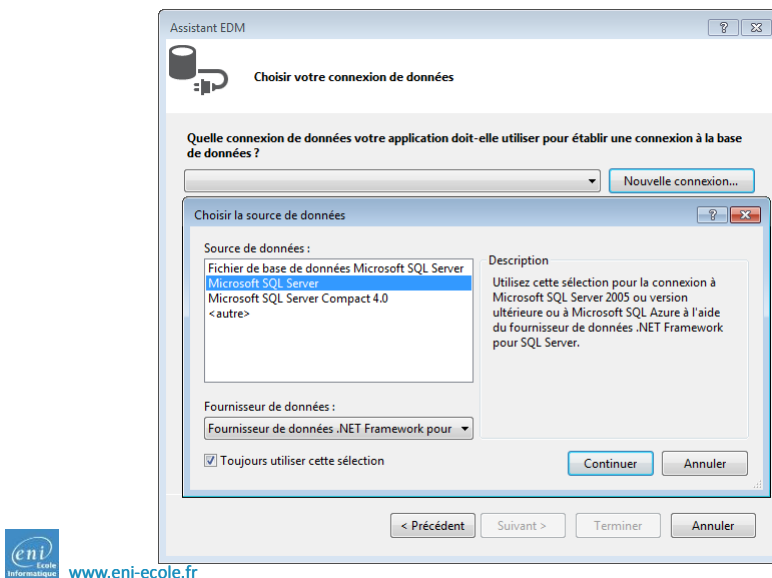


www.eni-ecole.fr

n° 94

Mise en œuvre – Couche Business Object Entity Framework

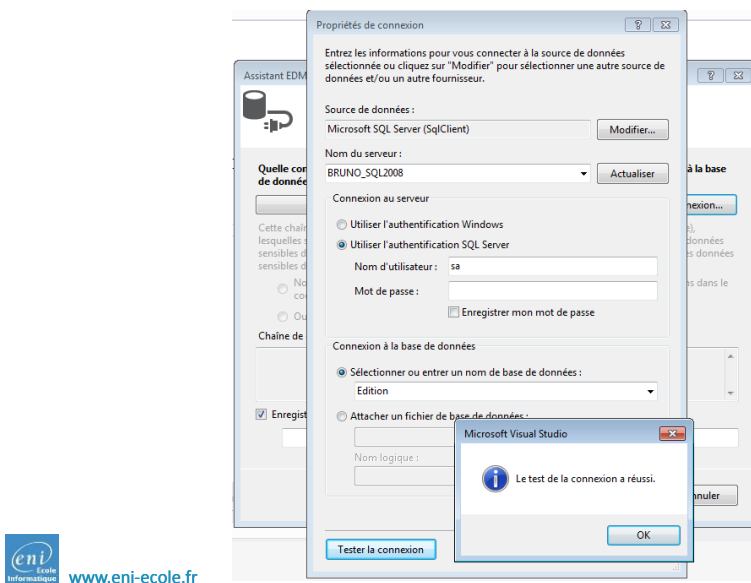
- Définir la connexion à la base 1/2



n° 95

Mise en œuvre – Couche Business Object Entity Framework

- Définir la connexion à la base 2/2



n° 96

Mise en œuvre – Couche Business Object Entity Framework

- La chaine de connexion dans App.config

Chaîne de connexion de l'entité :

metadata=res://*/BOEF.Edition.csdl|res://*/BOEF.Edition.ssdl|
res://*/BOEF.Edition.msl;provider=System.Data.SqlClient;provider connection string="data
source=BRUNO_SQL2008;initial catalog=Edition;user

☒ Enregistrer les paramètres de connexion de l'entité dans App.Config en tant que :

EditionEntities

```
<connectionStrings>  
  <add name="EditionEntities"  
        connectionString="metadata=res://*/Edition.csdl|res://*/Edition.ssdl|res://*/Edition  
</connectionStrings>
```

Mise en œuvre – Couche Business Object Entity Framework

- Transformer les tables en modèle

Assistant EDM

Choisir vos paramètres et objets de base de données

Quels objets de base de données voulez-vous inclure dans votre modèle ?

☒ Tables

☒ dbo

☒ Auteurs

☒ Auteurs_Livres

☒ Factures

☒ Livres

☐ sysdiagrams

☐ Vues

☐ Procédures et fonctions stockées

☒ Mettre au pluriel ou au singulier les noms d'objets générés

☐ Inclure les colonnes clés étrangères dans le modèle

☐ Importer les fonctions et les procédures stockées sélectionnées dans le modèle d'entité

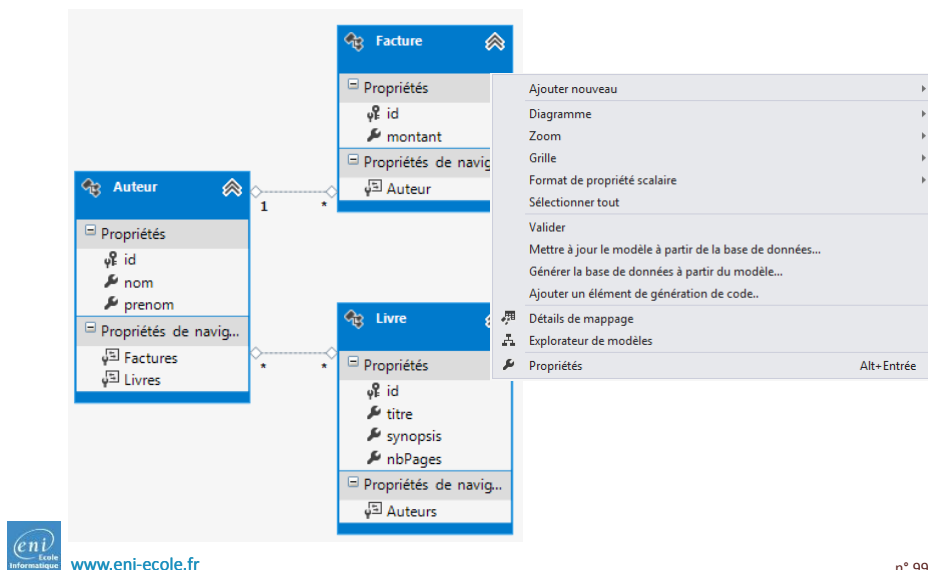
Espace de noms du modèle :

EditionModel

< Précédent Suivant > Terminer Annuler

Mise en œuvre – Couche Business Object Entity Framework

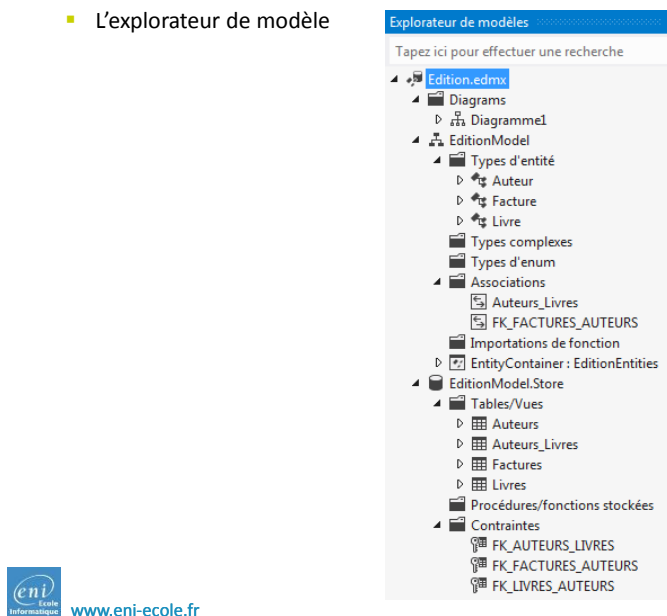
- Présentation du modèle obtenu



n° 99

Mise en œuvre – Couche Business Object Entity Framework

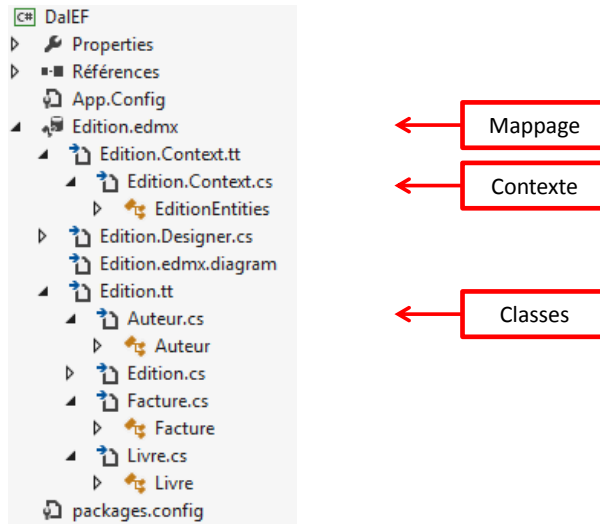
- L'explorateur de modèle



n° 100

Mise en œuvre – Couche Business Object Entity Framework

- Et des fichiers associés générés automatiquement par le Template T4



www.eni-ecole.fr

n° 101

Mise en œuvre – Classes POCO

- Classes métier :
 - Exposant la structure des données,
 - Exposant la logique métier,
 - Exposant la logique de validation ,
- Mappé sur un objet EntityEntry du contexte
 - Gestion de l'état de l'objet dans le contexte
 - Mapper à une table et à des colonnes
- Mais n'exposant aucune notion de persistance.
- Intérêt : exposer la structure des données métier sans se soucier de la manière dont ils sont persistés.

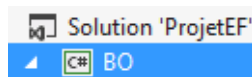


www.eni-ecole.fr

n° 102

Complément d'architecture – Dissocier les POCO de la persistance

- Le problème : Les classes BO (POCO) sont fortement associées à la couche de gestion de la persistance des données.
- L'objectif : Nous souhaiterions pouvoir les utiliser dans un projet qui ne tienne pas compte de la persistance.
- Pour cela :
 - Ajoutons un nouveau projet de type Bibliothèque de classes à notre solution



- Déplaçons les classes du nœud Edition.tt vers le projet BO
 - Couper/Coller les fichiers via l'explorateur Windows,
 - Ajouter des éléments existant au projet BO,
 - Modifier le namespace de chaque classe.

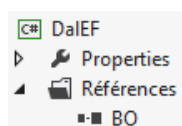


www.eni-ecole.fr

n° 103

Complément d'architecture – Dissocier les POCO de la persistance

- Au projet DaIEF, ajouter la référence au projet BO



- puis modifier le template Edition.context.tt afin de permettre au contexte d'utiliser les classes POCO

```
using System;
using System.Data.Entity;
using System.Data.Entity.Infrastructure;
using BO;
```

- Enfin supprimer l'arborescence liée au template Edition.tt du projet DaIEF si cela n'a pas été réalisé préalablement.
- Le projet de test ProjetEF n'a plus qu'à référencer le projet DaIEF et BO.



www.eni-ecole.fr

n° 104

Mise en œuvre – Personnaliser les classes POCO

- Classe partielle pouvant être enrichie fonctionnellement

```
public partial class Livre
{
    public Livre(string titre, string synopsis, List<Auteur> auteurs, int nbpages)
        : this()
    {
        this.titre = titre;
        this.synopsis = synopsis;
        foreach (Auteur auteur in auteurs)
        {
            this.Auteurs.Add(auteur);
        }
        this.nbPages = nbPages;
    }
}
```

- Nous compléterons ces classes plus loin dans le cours, lors de la mise en place de la validation.



www.eni-ecole.fr

n° 105

Mise en œuvre – Object Services

- Couche logique permettant de :
 - gérer les opérations de type CRUD sur les entités,
 - suivre les modifications des objets,
 - gérer l'accès concurrentiel,
 - générer les requêtes vers la source de données
- Matérialisée par la classeObjectContext

```
public partial class EditionEntities : DbContext
```

- Contenant la référence aux jeux d'enregistrements gérés : DbSet

```
public DbSet<Auteur> Auteurs { get; set; }
public DbSet<Facture> Factures { get; set; }
public DbSet<Livre> Livres { get; set; }
```

- Classe partielle pouvant être enrichie fonctionnellement



www.eni-ecole.fr

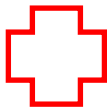
n° 106

Mise en œuvre – Object Services

```
public partial class EditionEntities : DbContext
{
    public EditionEntities()
        : base("name=EditionEntities")
    {
    }

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        throw new UnintentionalCodeFirstException();
    }

    public DbSet<Auteur> Auteurs { get; set; }
    public DbSet<Facture> Factures { get; set; }
    public DbSet<Livre> Livres { get; set; }
}
```



www.eni-ecole.fr

```
/// <summary>
/// Complément de définition du contexte
/// </summary>
public partial class EditionEntities
{
    public EditionEntities(bool modeLazyLoading, bool modeValidateOnSave)
        : this()
    {
        this.Configuration.AutoDetectChangesEnabled = true;
        this.Configuration.LazyLoadingEnabled = modeLazyLoading;
        this.Configuration.ProxyCreationEnabled = true;
        this.Configuration.ValidateOnSaveEnabled = modeValidateOnSave;
    }
}
```

n° 107

Mise en œuvre – Cycle de vie du contexte

```
EditionEntities contexte = new EditionEntities();
contexte.Auteurs.Add(a);
contexte.SaveChanges();
contexte.Dispose();
```

```
using (EditionEntities contexte = new EditionEntities())
{
    contexte.Auteurs.Add(a);
    contexte.SaveChanges();
}
```



www.eni-ecole.fr

n° 108

Mise en œuvre – Suivi des objets dans le contexte

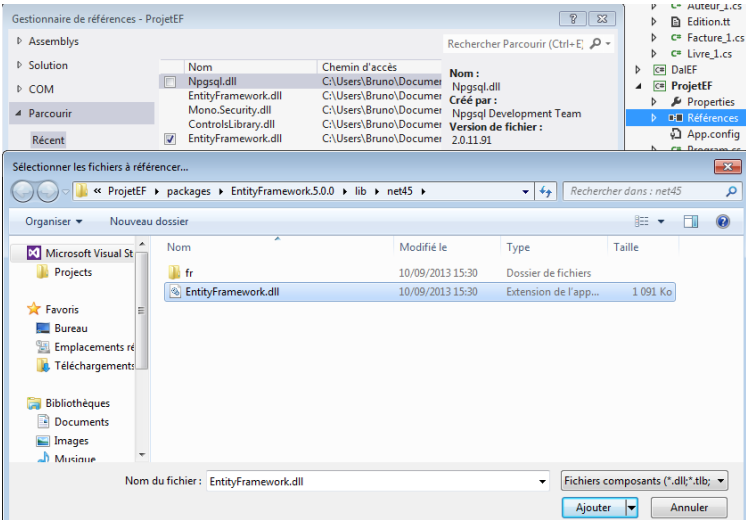
- Classe de gestion d’une entité du contexte : EntityEntry ou DbEntityEntry

Entity State	Tracking	DB
Added	Yes	Insert → Unchanged
Deleted	Yes	Delete → Detached
Modified	Yes	Update → Unchanged
Unchanged	Yes	None
Detached	No	None

```
context.Entry(a).State
```

Utiliser les classes POCO et le contexte

- Préparer le projet de test :
 - Référencer la dll de l’Entity Framework dans le projet de Test :



Utiliser les classes POCO et le contexte

- Copier la chaîne de connexion dans le fichier App.config du projet de démarrage :

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
  </startup>
  <connectionStrings>
    <add name="EditionEntities"
          connectionString="metadata=res://*/Edition.csd|res://*/Edition.ssd|
    </connectionStrings>
</configuration>
```



www.eni-ecole.fr

n° 111

Mise en œuvre – Extraction d'éléments

- Utilisation de Linq To Entities :
 - Expressions de requête

```
var rqt = from Auteur a in contexte.Auteurs
          where a.nom.StartsWith("G")
          orderby a.nom
          select a;
foreach (Auteur a in rqt)
{
  /*
   * Mode lazy loading : extraction des li
   */
  foreach (Livre l in a.Livres)
  {
    AfficherLivre(l, contexte);
  }
}
```

```
rqt
{SELECT
 [Extent1].[id] AS [id],
 [Extent1].[nom] AS [nom],
 [Extent1].[prenom] AS [prenom]
 FROM [dbo].[Auteurs] AS [Extent1]
 WHERE [Extent1].[nom] LIKE 'G%'
 ORDER BY [Extent1].[nom] ASC}
```



www.eni-ecole.fr

n° 112

Mise en œuvre – Extraction d'éléments

- Utilisation de Linq To Entities :

- Méthodes d'extension

```
var rqt = contexte.Auteurs
    .Where(auteur => auteur.nom.StartsWith("G"))
    .OrderBy(auteur => auteur.nom);
foreach (Auteur a in rqt)
{
    /*
     * Mode lazy loading : extraction des livres ici
     */
    foreach (Livre l in a.Livres)
    {
        AfficherLivre(l, contexte);
    }
}
```



www.eni-ecole.fr

n° 113

Mise en œuvre – Extraction d'éléments

- Mode Lazy Loading par défaut : les dépendances ne sont pas chargées en même temps que les entités, mais lorsqu'elles sont référencées la première fois.
 - Les + : Occupation mémoire optimisée
 - Les - : Performance (lors d'une itération, une nouvelle requête est effectuée pour chaque élément associé au foreach)
- Mode Eagerly Loading : les dépendances sont chargées en même temps que les entités grâce à la méthode **Include()**.
 - Les + : Performance (une seule requête est exécutée)
 - Les - : Occupation mémoire

```
using System.Data.Entity;

var rqt = contexte.Auteurs.Where(auteur => auteur.nom.StartsWith("G"))
    .OrderBy(auteur => auteur.nom)
    .Include(auteur => auteur.Livres);
```



www.eni-ecole.fr

n° 114

Mise en œuvre – Extraction d'éléments

- Requête de type lazy loading

```
{SELECT
[Extent1].[id] AS [id],
[Extent1].[nom] AS [nom],
[Extent1].[prenom] AS [prenom]
FROM [dbo].[Auteurs] AS [Extent1]
WHERE [Extent1].[nom] LIKE 'G%'
ORDER BY [Extent1].[nom] ASC}
```

- Requête de type Eagerly Loading

```
{SELECT
[Project1].[id] AS [id],
[Project1].[nom] AS [nom],
[Project1].[prenom] AS [prenom],
[Project1].[C1] AS [C1],
[Project1].[id1] AS [id1],
[Project1].[titre] AS [titre],
[Project1].[synopsis] AS [synopsis],
[Project1].[nbPages] AS [nbPages]
FROM ( SELECT
[Extent1].[id] AS [id],
[Extent1].[nom] AS [nom],
[Extent1].[prenom] AS [prenom],
[Join1].[id] AS [id1],
[Join1].[titre] AS [titre],
[Join1].[synopsis] AS [synopsis],
[Join1].[nbPages] AS [nbPages],
CASE WHEN ([Join1].[idAuteur] IS NULL) THEN CAST(NULL AS int) ELSE 1 END AS [C1]
FROM [dbo].[Auteurs] AS [Extent1]
LEFT OUTER JOIN (SELECT [Extent2].[idAuteur] AS [idAuteur], [Extent3].[id] AS [id], [Extent3].[titre] AS [titre], [Extent3].[synopsis] AS [synopsis], [Extent3].[nbPages] AS [nbPages]
FROM [dbo].[Auteurs_Livres] AS [Extent2]
INNER JOIN [dbo].[Livres] AS [Extent3] ON [Extent3].[id] = [Extent2].[idLivres] ) AS [Join1] ON [Extent1].[id] = [Join1].[idAuteur]
WHERE [Extent1].[nom] LIKE 'G%'
) AS [Project1]
ORDER BY [Project1].[nom] ASC, [Project1].[id] ASC, [Project1].[C1] ASC}
```



www.eni-ecole.fr

n° 115

Mise en œuvre – Ajout d'éléments

- L'objet passe à l'état **Added**

```
Auteur a = new Auteur("HITCHCOCK", "Alfred");
using (EditionEntities contexte = new EditionEntities())
{
    contexte.Auteurs.Add(a);
    contexte.SaveChanges();
}
```

```
a = new Auteur("GRANGE", "Jean Christophe");
using (EditionEntities contexte = new EditionEntities())
{
    contexte.Entry(a).State = System.Data.EntityState.Added;
    contexte.SaveChanges();
}
```

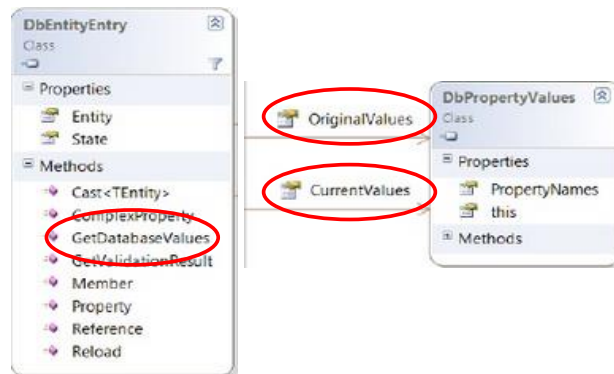


www.eni-ecole.fr

n° 116

Mise en œuvre – Modification d'éléments

- Il suffit de modifier une propriété d'un objet récupéré dans le contexte.
 - L'objet passe à l'état **Modified**.
- Pour chaque objet, le contexte conserve différents jeux d'enregistrements consultables au travers des propriétés et méthodes exposées par la classe DbEntityEntry



www.eni-ecole.fr

n° 117

Mise en œuvre – Modification d'éléments

```
Console.WriteLine("Valeurs courantes");
printValues(contexte.Entry(a).CurrentValues);
```

```
private static void printValues(DbPropertyValues dbPropertyValues)
{
    foreach (var propertyName in dbPropertyValues.PropertyNames)
    {
        Console.WriteLine("La propriété {0} = {1}",
            propertyName,
            dbPropertyValues[propertyName]);
    }
}
```

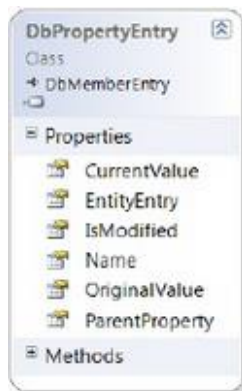


www.eni-ecole.fr

n° 118

Mise en œuvre – Modification d'éléments

- La classe DbPropertyEntry propose de consulter les valeurs courantes et originelles d'une propriété donnée et de savoir s'il s'agit d'une des propriétés modifiée.



```
Console.WriteLine(contexte.Entry(a)
                    .Property("nom")
                    .CurrentValue);
Console.WriteLine(contexte.Entry(a)
                    .Property(p=>p.nom)
                    .CurrentValue);
Console.WriteLine(contexte.Entry(a)
                    .Property(p => p.nom)
                    .IsModified);
```



www.eni-ecole.fr

n° 119

Mise en œuvre – Suppression d'éléments

- L'objet passe à l'état **Deleted**

```
a = contexte.Auteurs.Find(26);
contexte.Auteurs.Remove(a);
```



www.eni-ecole.fr

n° 120

Mise en œuvre – Mettre à jour la source de données

- Méthode SaveChanges()
 - System.Data.Entity.Infrastructure.DbUpdateException

```
catch (DbUpdateException exUpdate)
{
    AfficherAuteur(a, contexte);
    Console.WriteLine("L'auteur {0} ne peut être supprimé de la base (motif : {1} - {2})!",
        a.nom, exUpdate.Message, exUpdate.GetBaseException().Message);
    contexte.Entry(a).State = System.Data.EntityState.Unchanged;
}
```

- System.Data.Entity.Validation.DbEntityValidationException

```
catch (DbEntityValidationException ex)
{
    foreach (DbEntityValidationResult error in ex.EntityValidationErrors)
    {
        foreach (DbValidationError validationErrors in error.ValidationErrors)
        {
            Console.WriteLine("{0} - {1}",
                validationErrors.PropertyName,
                validationErrors.ErrorMessage);
        }
    }
}
```






www.eni-ecole.fr

n° 121

Mise en œuvre – Validation d'éléments

- A partir des valeurs attribuées aux propriétés des entités (validation côté base de données)

Propriétés	
EditionModel.Livre.titre Property	
  	
Clé d'entité	False
Documentation	
Getter	Public
Longueur fixe	False
Longueur max.	50
Mode d'accès concurre	None
Nom	titre
Nullable	False
Setter	Public
StoreGeneratedPattern	None
Type	String
Unicode	False
Valeur par défaut	(Aucune)



www.eni-ecole.fr

n° 122

Mise en œuvre – Validation d'éléments

- Annotation dans les classes POCO (validation personnalisée)
 - Importer l'espace de noms

```
using System.ComponentModel.DataAnnotations;
```

- Associer classe de gestion des validations à la classe POCO

```
[MetadataType(typeof(AuteurValidation))]
public partial class Auteur
```

- Utiliser les annotations pour mettre en place les validations

```
/// <summary>
/// classe de validation des données
/// </summary>
internal class AuteurValidation
{
    [Required(ErrorMessage = "Le nom de l'auteur est obligatoire")]
    public string nom;
}
```



www.eni-ecole.fr

n° 123

Mise en œuvre – Validation d'éléments

- Annotation dans les classes POCO (validation personnalisée)
 - Validation customisée d'une propriété

```
/// <summary>
/// classe de validation des données
/// </summary>
internal class FactureValidation
{
    [FactureCustomValidation]
    public decimal montant;
}

internal class FactureCustomValidation : ValidationAttribute
{
    protected override ValidationResult IsValid(object value, ValidationContext validationContext)
    {
        if (value != null && value is decimal && (decimal)value < (decimal)0)
            return new ValidationResult("Le montant de la facture doit être positif !");
        else
            return ValidationResult.Success;
    }
}
```

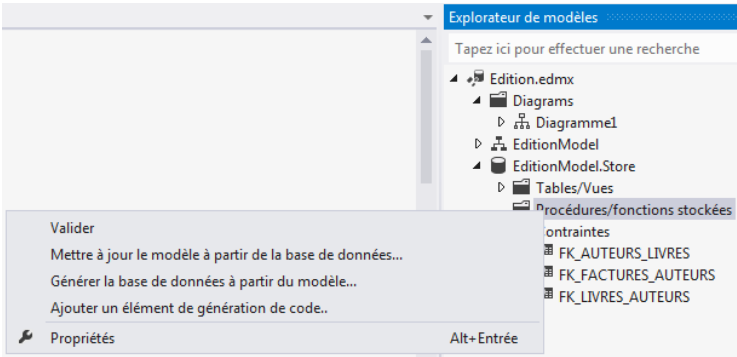


www.eni-ecole.fr

n° 124

Mise en œuvre – Importer les procédures stockées

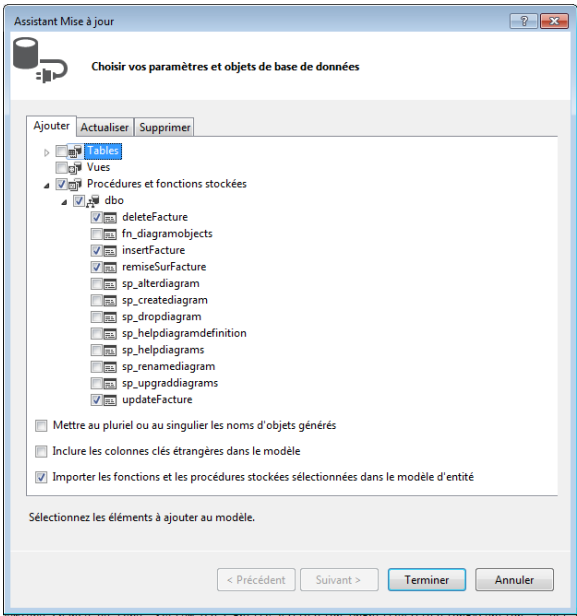
- A partir de l'explorateur de modèle, **mettre à jour le modèle à partir de la base de données**



www.eni-ecole.fr

n° 125

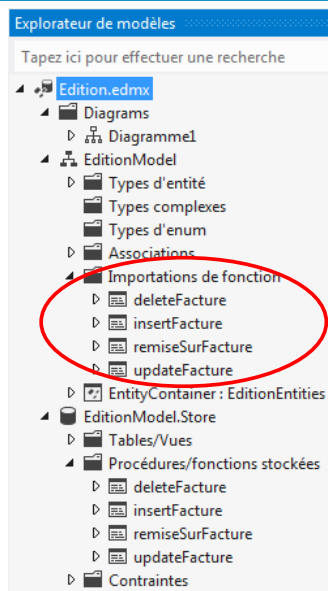
Mise en œuvre – Importer les procédures stockées



www.eni-ecole.fr

n° 126

Mise en œuvre – Importer les procédures stockées



www.eni-ecole.fr

n° 127

Mise en œuvre – Utiliser une méthode mappée à une procédure stockée

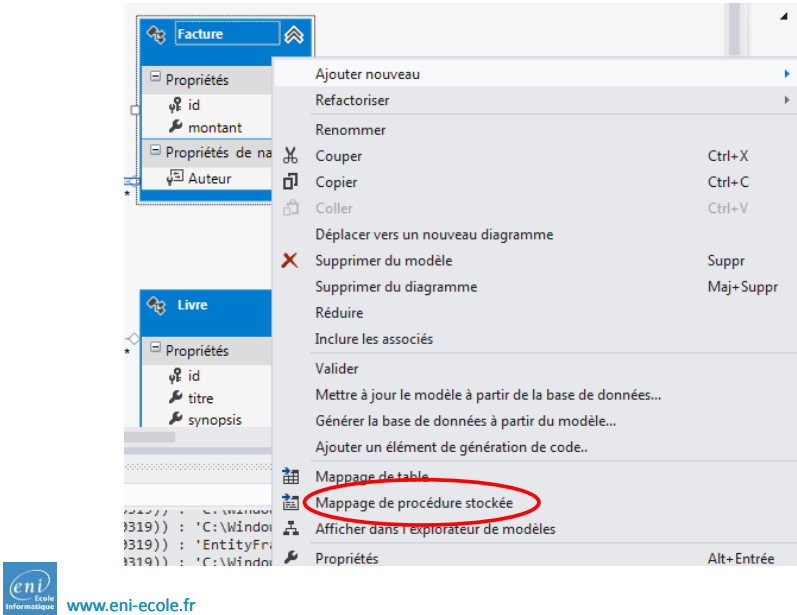
```
using (EditionEntities contexte = new EditionEntities())
{
    //appel à la procédure stockée mappée par la méthode remiseSurFacture
    contexte.remiseSurFacture(8, (decimal)0.10);
}
```



www.eni-ecole.fr

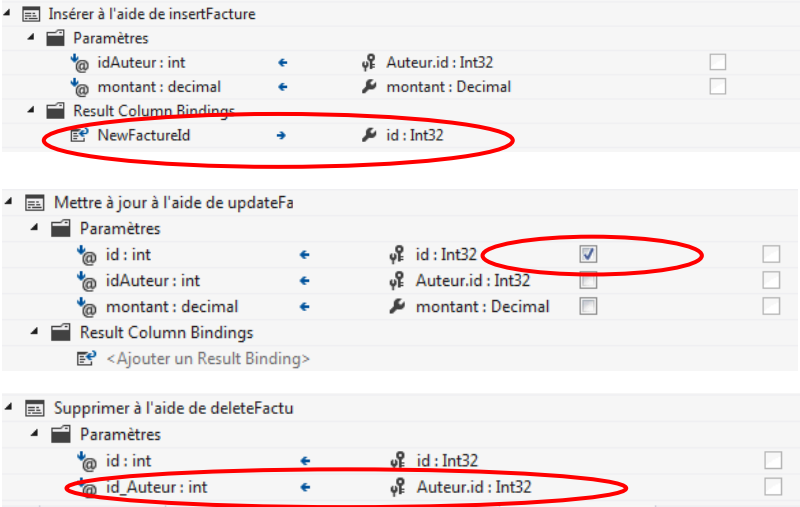
n° 128

Mise en œuvre – Mapper les entités aux procédures stockées



n° 129

Mise en œuvre – Mapper les entités aux procédures stockées



n° 130

Mise en œuvre – Mapper les entités aux procédures stockées

```
using (EditionEntities contexte = new EditionEntities())
{
    Auteur a = contexte.Auteurs.Find(25);

    Facture f = new Facture((decimal)9999.99, a);
    contexte.Factures.Add(f);
    AfficherFacture(f, contexte);
    contexte.SaveChanges();
    AfficherFacture(f, contexte);
}
```

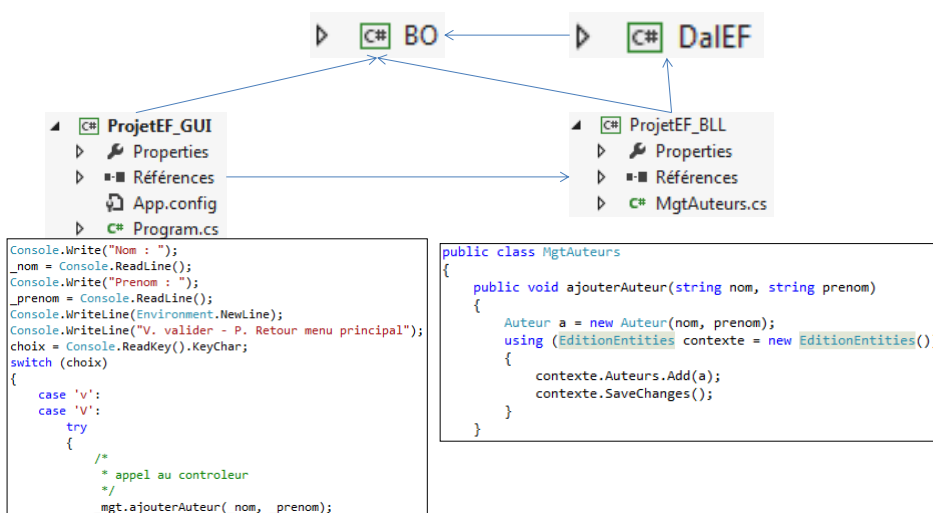


www.eni-ecole.fr

n° 131

Complément d'architecture – Pattern Repository

- 1^{ère} évolution apportée à la solution : approche MVC standard



www.eni-ecole.fr

n° 132

Complément d'architecture – Pattern Repository

- Le problème : Le contrôleur est fortement lié à la source de données.

```
public class MgtAuteurs
{
    public void ajouterAuteur(string nom, string prenom)
    {
        Auteur a = new Auteur(nom, prenom);
        using (EditionEntities contexte = new EditionEntities())
        {
            contexte.Auteurs.Add(a);
            contexte.SaveChanges();
        }
    }
}
```

L'utilisation du contexte et de Linq to Entities est en dur dans le code

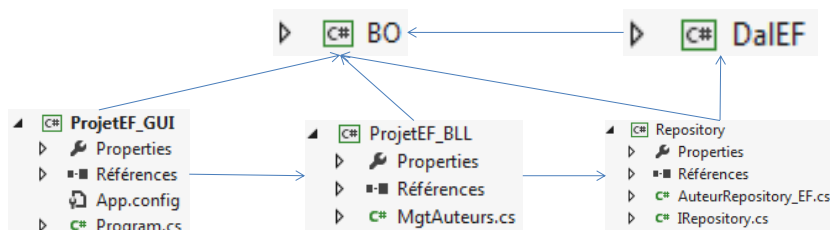


www.eni-ecole.fr

n° 133

Complément d'architecture – Pattern Repository

- Le pattern Repository
 - Guideline de programmation permettant d'éviter les dépendances.
- Ce qui change au niveau de notre architecture



- La couche BLL ne référence plus directement la Dal
- Seul le Repository connaît les sources de données

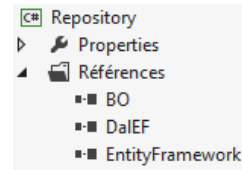
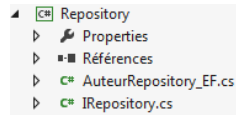


www.eni-ecole.fr

n° 134

Complément d'architecture – Pattern Repository

■ Détail du Repository



```
public interface IRepository<T> where T : class
{
    void insert(T element);
    void update(T element);
    void delete(T element);
    T getById(long id);
    T getOneBy(Func<T, Boolean> where);
    IEnumerable<T> getAll();
    IEnumerable<T> getMany(Func<T, Boolean> where);
}
```

```
public class AuteurRepository_EF : IRepository<Auteur>
{
    public void insert(Auteur element)
    {
        using (EditionEntities contexte = new EditionEntities())
        {
            contexte.Auteurs.Add(element);
            contexte.SaveChanges();
        }
    }
}
```



www.eni-ecole.fr

n° 135

Complément d'architecture – Pattern Repository

■ Utilisation dans le contrôleur

```
private IRepository<Auteur> _repository;
public MgtAuteurs()
{
    _repository = new AuteurRepository_EF();
}

public void ajouterAuteur(string nom, string prenom)
{
    Auteur a = new Auteur(nom, prenom);
    _repository.insert(a);
}
```

Seule dépendance
restante



www.eni-ecole.fr

n° 136

Pour aller plus loin – EF Code First



www.eni-ecole.fr

n° 137

TP

- Mettre en place la persistance des données de l'application « Gestion des employés » dans une base de données SQLServer au travers de Entity Framework.



www.eni-ecole.fr

n° 138

Persistence des données

Module 7

Génération de rapports/d'états ou *Reporting*



www.eni-ecole.fr

n° 139

Contenu du module

- Objectifs
- TD



www.eni-ecole.fr

n° 140

Objectifs

- Le reporting permet :
 - de sélectionner des données relatives à telle période, telle production, tel secteur, ...
 - de trier, regrouper ou répartir ces données selon les critères de leur choix
 - de réaliser divers calculs (totaux, moyennes, écarts, comparatif d'une période à l'autre, ...)
 - de présenter et de fixer les résultats d'une manière synthétique ou détaillée, le plus souvent graphique selon leurs besoins



www.eni-ecole.fr

n° 141

TD

- Suivre la fiche technique *FT_Rapports RDLC.pdf*



www.eni-ecole.fr

n° 142

Persistence des données

Module 8

Déploiement d'une application

Chapitre 11



www.eni-ecole.fr

n° 143

Click Once

- Principe **RI 477-491**
- Click Once est une technologie de déploiement qui permet de créer des applications Windows à mise à jour automatique
 - La publication de l'application et des mises à jour se fait au choix par:
 - Une page web
 - Un partage de fichier
 - Un CD-ROM
 - Les avantages sont
 - La réinstallation automatique, en cas de mise à jour, dans un nouveau dossier
 - L'autonomie de l'application par rapport aux autres (aucun composant partagé)
 - Les autorisations minimalistes à octroyer à l'utilisateur en terme de sécurité
- L'application peut fonctionner au choix en ligne ou hors connexion
- Ressources supplémentaires:
 - [http://msdn.microsoft.com/fr-fr/library/142dbbz4\(v=vs.90\).aspx](http://msdn.microsoft.com/fr-fr/library/142dbbz4(v=vs.90).aspx)
- TD
 - Déploiement sur un partage réseau



www.eni-ecole.fr

n° 144

Développement d'une application objet avec VB.NET

Fin de la formation



www.eni-ecole.fr

n° 145