Secteur Tertiaire Informatique
Filière étude - développement

Développer des composants d'interface

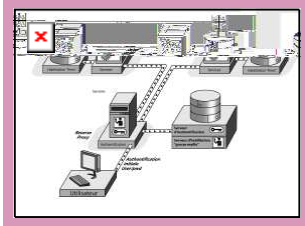
Accès aux données :
Découpler une application d'un fournisseur de données

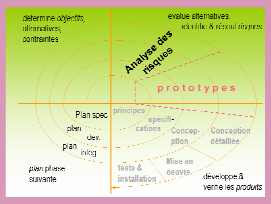
Accueil

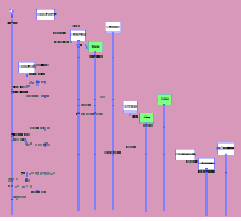
Apprentissage

Période en entreprise

Evaluation







Code barre

Lorsqu'une application exploite des données stockées dans un SGBD il est intéressant de chercher à rendre le code le plus indépendant possible (le plus abstrait possible) d'un type de SGBD.

L'intérêt de cette démarche est qu'il est courant de ne pas maîtriser le type de SGBD qui sera utilisé en production. Un client peut vouloir utiliser l'application avec un SGDB *Oracle* parce qu'il ne veut pas changer ses habitudes tandis qu'un autre client préférera un SGBD gratuit tel que *MySQL.*

Cette indépendance du type de SGBD est aisément obtenue **si ne sont exploitées que les fonctionnalités communes aux SGBD.** En effet, il suffit alors de ne **manipuler les données** qu'**au travers des classes abstraites** de l'espace de noms **System.Data Common** .

Dans ce nouvel espace de nom, ADO.Net 2 présente plusieurs classes abstraites qui définissent un modèle pour l'ensemble des fonctionnalités partagées par les SGBD.

Quelques classes de l'espace de noms **System.Data.Common**

Classe	**Description**
DbDataAdapter	Implémente un jeu de fonctions pour fournir un type fort, mais héritent de la plupart des fonctionnalités nécessaires à l'implémentation complète d'un **DataAdapter**.
DbCommand	Représente une instruction SQL ou une procédure stockée à exécuter par rapport à une source de données. Fournit une classe de base pour les classes spécifiques à la base de données qui représentent des commandes.
DbCommandBuilder	Génère automatiquement des commandes de table simple utilisées pour harmoniser les modifications apportées à **DataSet** avec la base de données associée. Cette classe est une classe abstraite qui peut uniquement être héritée.
DbConnection	Représente une connexion à une base de données.
DbConnectionStringBuilder	Fournit une classe de base pour les générateurs de chaînes de connexion fortement typées.
DbException	Classe de base pour toutes les exceptions levées au nom de la source de données.
DbParameter	Représente un paramètre de DbCommand et, éventuellement, son mappage à une colonne **DataSet**.
DbParameterCollection	La classe de base pour une collection de paramètres relatifs à **DbCommand**.
DbProviderFactories	Représente un jeu de méthodes statiques permettant de créer une ou plusieurs instances des classes DbProviderFactory.
DbProviderFactory	Représente un jeu de méthodes permettant de créer des instances de l'implémentation d'un fournisseur des classes source de données.

Par exemple , la classe **DbConnection** définit une connexion vers une base de données, alors que la classe **DbCommand** définit la notion de commande.

Néanmoins, un problème se pose : à un endroit du code, il faudra créer les objets ADO.NET manipulés. Cette opération implique de préciser explicitement la classe à utiliser.

```
using System.Data.Common;
using System.Data.SqlClient;
DbConnection oConnection;
DbCommand oCommand;
if (/* test si on travaille avec SQL server*/ true)
{
// couplage avec SqlClient
oConnection = new SqlConnection();
oCommand = new SqlCommand();
}
// exploitation des objets indépendamment du SGBD
```

ADO.Net 2 résout ce problème par l'application d'un Design Pattern d'architecture, le pattern ***fabrique abstraite***.

En effet, chaque fournisseur de données ADO.Net 2 présente une classe dérivée de la classe abstraite **System.Data.Common.DbProviderFactory.**

Par exemple, **System.Data.Common.SqlClientFactory** présente les méthodes suivantes :

Nom	Description
CreateCommand	Retourne une instance de DbCommand fortement typée.
CreateConnection	Retourne une instance de DbConnection fortement typée.
CreateDataAdapter	Retourne une instance de DbDataAdapter fortement typée.
CreateParameter	Retourne une instance de DbParameter fortement typée.

La seule manière de construire une instance de la classe **SqlClientFactory** est d'utiliser la méthode statique **GetFactory()** de la classe **System.Data.Common.DbProviderFactories.**

```
using System.Data.Common;
// using System.Data.SqlClient;
DbProviderFactory fabrique =
DbProviderFactories.GetFactory("System.Data.SqlClient");
// à partir d'ici, le code est indépendant du SGBD

DbConnection oConnection = fabrique.CreateConnection();
DbCommand oCommand = fabrique.CreateCommand();
```

Pour que notre code soit totalement indépendant du fournisseur de données, il suffit de stocker la chaîne de connexion dans un fichier de configuration, situé dans le répertoire du projet .

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
  </configSections>
  <connectionStrings>
      <add name="Test" providerName="System.Data.SqlClient"
      connectionString="Data Source=SRVSQL;Initial
      Catalog=BDTest;Integrated Security=True" />
  </connectionStrings>
</configuration>
```

Il suffit de prévoir dans le code de récupérer les éléments du fichier de configuration : L'utilisation de la classe **ConnectionStringSettings** de l'espace de noms **System**.**Configuration**.permet par le biais de ses deux propriétés, **ConnectionString** et **Name**, de récupérer la chaîne de connexion , par son nom à travers la classe **ConfigurationManager**, qui accède aux informations de configuration machine et de l'application.

```
using System.Data.Common;
// using System.Data.SqlClient;
using System.Configuration;            // ajouter la référence
au projet

System.Configuration.ConnectionStringSettings oconfig;
oconfig = ConfigurationManager.ConnectionStrings["Test"];

DbProviderFactory fabrique =
DbProviderFactories.GetFactory(oconfig.Providername)
// à partir d'ici, le code est indépendant du SGBD

DbConnection oConnection = fabrique.CreateConnection();
oConnection.ConnectionString = oconfig.ConnectionString;

DbCommand oCommand = fabrique.CreateCommand(); ;
```

Etablissement référent
Marseille Saint Jérôme

Equipe de conception
Elisabeth Cattanéo

Remerciements :

Date de mise à jour 05/05/2008
afpa © Date de dépôt légal mai 08

afpa / Direction de l'Ingénierie13 place du Générale de Gaulle / 93108 Montreuil Cedex
association nationale pour la formation professionnelle des adultes
Ministère des Affaires sociales du Travail et de la Solidarité