Cours n°4 : Accès aux données

SERVICES RESEAUX

Hélène CHASSAGNE
helene.chassagne@orange.fr

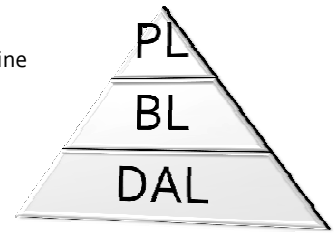
Frédéric CHASSAGNE
frederic.chassagne@atosorigin.com

Plan du cours

- L 'accès aux données
  - 1 bonne DAL
  - La connexion
  - La manipulation
  - Le stockage

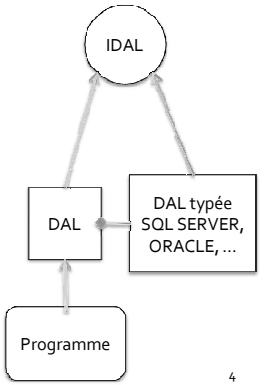
Architecture 3 tiers

- Couches du modèle
  - Presentation Layer
    - Interface Homme Machine
  - Business Layer
    - Noyau métier
  - Data Access Layer
    - Accès aux données



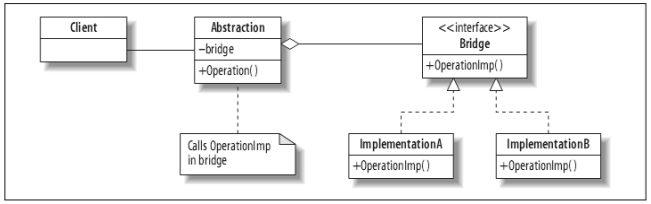
L'accès aux données

- Data Access Layer
  - Flexibilité
  - Interopérabilité
- Interface = ∑ d'assesseurs
- Factory = Implémentation Interface
- Programme = Utilisation Factory



L'accès aux données

- Dal = Implémentation d'un Bridge Pattern
- But = Support multi SGBD



5

L'accès aux données

- Exemple

```
public interface IDAL
  {
    DataTable getAuthors();
  }

public enum DALProvider
  {
    ORACLE = 1,
    SQLSERVER = 2
  }
```

```
public class DAL : IDAL
  {
    private IDAL _dal;
    public DAL(DALProvider provider, DBParams p)
    {
      switch (provider)
      {
        case DALProvider.ORACLE: _dal = new DALORACLE(p);
break;
        case DALProvider.SQLSERVER: _dal = new
DALSQLSERVER(p); break;
      }
    }

public DataTable getAuthors()
    {
      if (_dal != null) return _dal.getAuthors();
      else return null;
    }
}
```

6

L'accès aux données

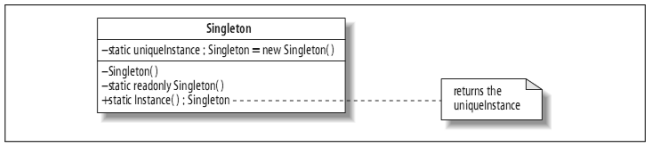
```
public class DALORACLE : IDAL
  {
    public DALORACLE(DBParams p)
    {
    // … Instanciation de la connexion …
    }

public DataTable getAuthors()
    {
    //   .. Implementation spécifique ORACLE de la requete
    }
}

// Idem avec la class DALSQLSERVER
```

Tip :
Clic droit sur IDAL dans VS
=
Implémentation des signatures de l'interface

7

L'accès aux données

- Accès à la Dal dans l'application par un singleton



8

L'accès aux données

- Implémentation

```
public class DataProvider
{
  private static DataProvider _dalProvider;
  private static readonly object padlock =
new object();

  public static DataProvider Instance
  {
    get
    {
      if (_dalProvider == null)
      {
        lock (padlock)
```

```
{
        if (_dalProvider ==
null)
        {
          _dalProvider = new
DataProvider();
        }
      }
    }
    return _dalProvider;
  }
}
```


L'accès aux données

- Api utilisées différentes en fonction des SGBD
- Même façade
- Api utilisée pour SQL SERVER = System.data.sqlClient

L'accès aux données

- Les objets à utiliser pour 1 SQL Server
  - SQLConnection : Connexion physique à la base
  - SQLCommand & SQLDataAdapter : Récupération des données dans le SGBD à partir d'1 connexion existante
  - Datatable & Dataview : manipulation des données récupérées
  - DataGridView : Présentation des données à l'utilisateur
  - SQLCommandBuilder : Sauvegarde des données

L'accès aux données

- La connexion à une base

```
private void Connexion(string bdd, string server, string user, string pass)
    {
        _chaineConnexion = string.Format("Data Source={1};User ID={2};Password={3};
Initial Catalog={0}« , bdd, server, user, pass)        _sqlConnexion = new
SqlConnection(_chaineConnexion);
      _sqlConnexion.Open();
    }
```

L'accès aux données

- La récupération des données

```
public DataTable Select(string strRequete)
    {
      if (_sqlConnexion != null)
      {
     DataTable dt = new DataTable();
      SqlTransaction myTrans =
_sqlConnexion.BeginTransaction();

        SqlCommand sqlCommande = new
SqlCommand(strRequete, _sqlConnexion, myTrans);
        SqlDataAdapter sqlAdapte = new
SqlDataAdapter(sqlCommande);
```

```
try
        {
          sqlAdapte.Fill(dt);
          myTrans.Commit();
          return dt;
        }
        catch
        {
          myTrans.Rollback();
          return null;
        }
      }
      return null;
    }
```


L'accès aux données

- La manipulation des données
- Datatable : données issues de la base
- Datarow : ligne d'une datatable
  - Accès aux cellules par index ou nom de colonnes
- Dataview : Vue sur une datatable
  - Propriétés intéressantes
    - RowFilter
    - Sort

L'accès aux données

- La manipulation des données
  - Exemple

```
DataTable t = Select("SELECT * FROM Authors");
DataView v = new DataView(t);

v.RowFilter = "au_lname like 'r*'";
for (int i = 0; i < v.Count; i++)
{
  DataRow row = v[i].Row;
  Console.WriteLine("{0} {1}", row["au_fname"], row["au_lname"]);
}
```


L'accès aux données

- La présentation des données
- Interaction avec l'utilisateur
  - Datarow = câblage champs à champs
    - Textbox, combobox, etc.
  - DatagridView = Interfaçage de l'ensemble de la source de données dans un composant

L'accès aux données

- Le stockage des données

```
public int Update(string strRequete, DataTable dttable)
    {
      try
      {
        if (_sqlConnexion != null)
        {
          //Transaction
          SqlTransaction myTrans = _sqlConnexion.BeginTransaction();

          SqlCommand sqlComm = new SqlCommand(strRequete, _sqlConnexion, myTrans);
          SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(sqlComm);
          SqlCommandBuilder sqlCommBuilder = new SqlCommandBuilder(sqlDataAdapter);
          sqlDataAdapter.UpdateCommand = sqlCommBuilder.GetUpdateCommand();
          sqlDataAdapter.InsertCommand = sqlCommBuilder.GetInsertCommand();
          sqlDataAdapter.DeleteCommand = sqlCommBuilder.GetDeleteCommand();

          sqlDataAdapter.MissingSchemaAction = MissingSchemaAction.AddWithKey;
```


L'accès aux données

```
try
      {
       int iResult = sqlDataAdapter.Update(dttable);
       myTrans.Commit();
       return iResult;
      }
      catch(DBConcurrencyException e)
            {
            myTrans.Rollback();
            }
      }
            return 0;
}
      catch(Exception e)
      {
            return -1;
      }
}
```


L'accès aux données

- Alternative au DataAdapter = DataReader
- Avantages
  - Lecture rapide
  - Peu de place prise en mémoire
  - Gestion par List Generic à la place de DataRow et DataTable
- Inconvénient
  - Sauvegarde difficilement gérable

L'accès aux données

```
public User LogUser(string username, string password)
    {
      User res = null;

      using (SqlConnection connexion = new
SqlConnection(_connexionString))
      {
        if (connexion != null)
        {
          connexion.Open();
          SqlCommand sqlCommand = new
SqlCommand(string.Format(SQL_LOG_USER_QUERY,
username), connexion);
          sqlCommand.CommandType =
System.Data.CommandType.Text;
          SqlDataReader sr = null;
          try
          {
            sr =
sqlCommand.ExecuteReader(System.Data.CommandBehav
ior.CloseConnection);
```

```
while (sr.Read())
            {
              string userPassword = sr.GetString(0);
              User u = new User(sr.GetString(1),
sr.GetString(2), sr.GetString(3), sr.GetString(4));
              if (userPassword ==
Cryptage.HashDataToText(password,
Cryptage.codage.UTF16, Cryptage.AlgoHash.SHA1))
              {
                res = u;
              }
            }
          }
          catch ...
Finally ...

      return res;
    }
```