

## 1. INTRODUCTION

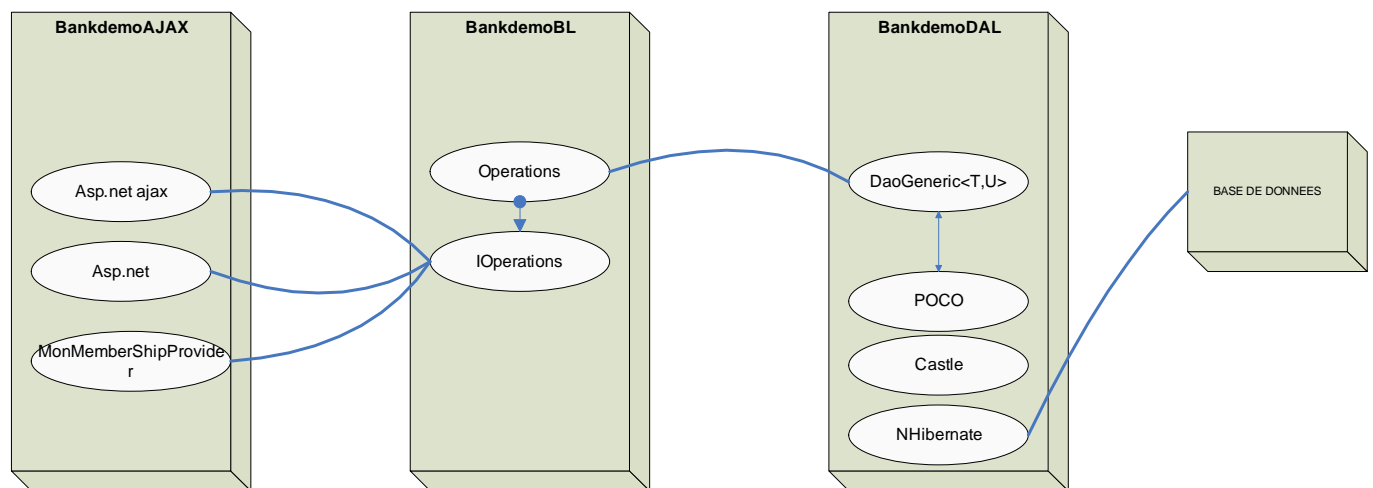
Cet article présente une architecture n-tiers Ajax autour des technologies .NET

## 2. OUTILS NECESSAIRES

Pour le développement :

- 2.1. Visual Studio 2005 (Edition pro)
- 2.2. Framework Castle (<http://www.castleproject.org/>) pour le pattern Active Record
- 2.3. Framework TestDriven.net (<http://www.testdriven.net/>) pour les tests unitaires
- 2.4. DSL tools : Active Writer pour la modélisation des objets de domaine et la génération du code ( <http://altinoren.com/activewriter/> )
- 2.5. Asp.net Ajax pour le client Ajax (<http://ajax.asp.net/> )

## 3. ARCHITECTURE APPLICATIVE



L'architecture est une architecture n-tiers classique comportant 3 couches :

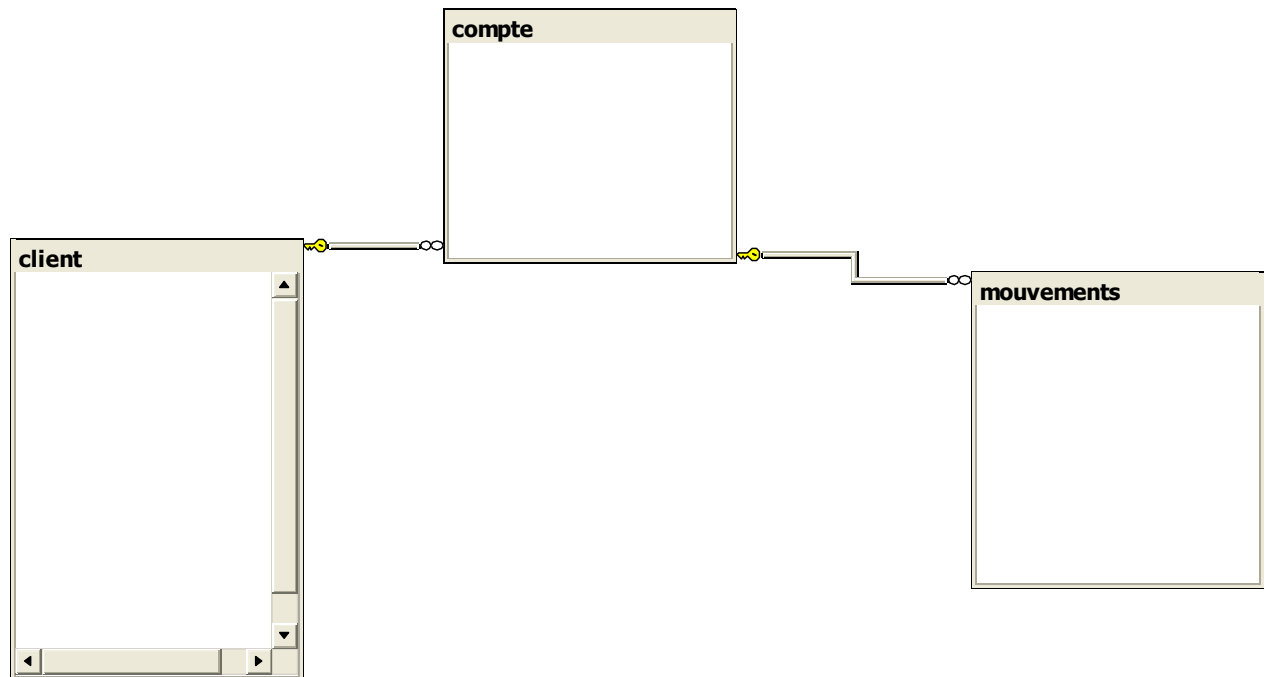
- une couche DAL (Data Access Layer)
- une couche BL (Business Layer)
- une couche AJAX (UI)

**Principe :** les couches interagissent via des contrats (des interfaces) et non des implémentations.

La couche de données est dans mon cas hébergée dans SQL Server 2005 sp1.

### 3.1. Les données

Le modèle de données est donné dans la figure suivante et représente un modèle ultra simplifié pour la consultation des comptes bancaires (3 tables).



### 3.2. La couche d'accès aux données (DAL)

La DAL sera composée des objets du domaine (trois objets, un par table). Ces objets du domaine seront des POCO (Plain Old Clr Object). Le pattern Active Record sera implémenté à travers le framework Castle qui est une des seules implémentations disponible pour .NET. Les objets du domaine dériveront donc tous d'une classe ActiveRecordBase qui implémente les fonctions CRUD.

Une dao générique sera mise en place pour accéder aux objets du domaine. Cette classe utilise les Generics c# 2.0 et permet de factoriser considérablement le code. Cette classe (daoGeneric) dérive d'une interface IdaoGeneric.

Castle utilise Nhibernate pour le mapping OR et log4net pour les traces.

#### Code de l'objet client

```

namespace bankdemoDAL {
    using System;
    using System.Collections;
    using Castle.ActiveRecord;

    [ActiveRecord("client", Schema="dbo")]
    public partial class client : ActiveRecordBase {
        private string _nom;
        private string _prenom;
        private string _civilite;
        private System.DateTime _datenaiss;
        private string _lieunaiss;
        private string _tel1;
        private string _tel2;
        private string _email;
    }
  
```

```

private string _mobile;
private System.DateTime _dateent;
private long _id;
private string _numcli;
private string _password;
private IList _comptes;

[Property("nom", ColumnType="String", NotNull=true)]
public string nom {
    get {
        return this._nom;
    }
    set {
        this._nom = value;
    }
}

[Property("prenom", ColumnType="String")]
public string prenom {
    get {
        return this._prenom;
    }
    set {
        this._prenom = value;
    }
}

[Property("civilite", ColumnType="String")]
public string civilite {
    get {
        return this._civilite;
    }
    set {
        this._civilite = value;
    }
}

[Property("datenaiss", ColumnType="DateTime", NotNull=true)]
public System.DateTime datenaiss {
    get {
        return this._datenaiss;
    }
    set {
        this._datenaiss = value;
    }
}

[Property("lieunaiss", ColumnType="String")]
public string lieunaiss {
    get {
        return this._lieunaiss;
    }
    set {
        this._lieunaiss = value;
    }
}

[Property("tel1", ColumnType="String")]
public string tel1 {
    get {
        return this._tel1;
    }
    set {
        this._tel1 = value;
    }
}

[Property("tel2", ColumnType="String")]
public string tel2 {
    get {
        return this._tel2;
    }
}

```

```

        set {
            this._tel2 = value;
        }
    }

    [Property("email", ColumnType="String")]
    public string email {
        get {
            return this._email;
        }
        set {
            this._email = value;
        }
    }

    [Property("mobile", ColumnType="String")]
    public string mobile {
        get {
            return this._mobile;
        }
        set {
            this._mobile = value;
        }
    }

    [Property("dateent", ColumnType="DateTime")]
    public System.DateTime dateent {
        get {
            return this._dateent;
        }
        set {
            this._dateent = value;
        }
    }

    [PrimaryKey(PrimaryKeyType.Native, "id", ColumnType="Int64")]
    public long id {
        get {
            return this._id;
        }
        set {
            this._id = value;
        }
    }

    [Property(ColumnType="String")]
    public string numcli {
        get {
            return this._numcli;
        }
        set {
            this._numcli = value;
        }
    }

    [Property(ColumnType="String")]
    public string password {
        get {
            return this._password;
        }
        set {
            this._password = value;
        }
    }

    [HasMany(typeof(compte), ColumnKey="idclient", Table="compte")]
    public IList comptes {
        get {
            return this._comptes;
        }
    }

```

```

    }
    set {
        this._comptes = value;
    }
}
}
}

```

## CODE DE L'OBJET COMPTE

```

namespace bankdemoDAL {
    using System;
    using System.Collections;
    using Castle.ActiveRecord;

    [ActiveRecord("compte", Schema="dbo")]
    public partial class compte : ActiveRecordBase {

        private string _libcompte;

        private System.DateTime _datecreat;

        private long _id;

        private string _numcompte;

        private IList _mouvements;

        private client _client;

        [Property("libcompte", ColumnType="String")]
        public string libcompte {
            get {
                return this._libcompte;
            }
            set {
                this._libcompte = value;
            }
        }

        [Property("datecreat", ColumnType="DateTime")]
        public System.DateTime datecreat {
            get {
                return this._datecreat;
            }
            set {
                this._datecreat = value;
            }
        }

        [PrimaryKey(PrimaryKeyType.Native, "id", ColumnType="Int64")]
        public long id {
            get {
                return this._id;
            }
            set {
                this._id = value;
            }
        }

        [Property(ColumnType="String")]
        public string numcompte {
            get {
                return this._numcompte;
            }
        }
    }
}

```

```

        this._numcompte = value;
    }
} set {

```

```

[HasMany(typeof(mouvements), ColumnKey="idcompte", Table="mouvements")]
public IList mouvements {
    get {
        return this._mouvements;
    }
    set {
        this._mouvements = value;
    }
}

[BelongsTo("idclient", NotNull=false)]
public client client {
    get {
        return this._client;
    }
    set {
        this._client = value;
    }
}
}
}

```

## Code de l'objet mouvements

```

namespace bankdemoDAL {
    using System;
    using System.Collections;
    using Castle.ActiveRecord;

    [ActiveRecord("mouvements", Schema="dbo")]
    public partial class mouvements : ActiveRecordBase {

        private System.DateTime _date;

        private System.DateTime _dateval;

        private string _typeop;

        private decimal _somme;

        private string _tiers;

        private string _rem;

        private long _id;

        private compte _compte;

        [Property("date", ColumnType="DateTime")]
        public System.DateTime date {
            get {
                return this._date;
            }
            set {

```

```
    }  
  }  
  this._date = value;  
}
```

```
[Property("dateval", ColumnType="DateTime")]
```

```
public System.DateTime dateval {  
  get {  
    return this._dateval;  
  }  
  set {  
    this._dateval = value;  
  }  
}
```

```
[Property("typeop", ColumnType="String")]
```

```
public string typeop {  
  get {  
    return this._typeop;  
  }  
  set {  
    this._typeop = value;  
  }  
}
```

```
[Property("somme", ColumnType="Decimal")]
```

```
public decimal somme {  
  get {  
    return this._somme;  
  }  
  set {  
    this._somme = value;  
  }  
}
```

```
[Property("tiers", ColumnType="String")]
```

```
public string tiers {  
  get {  
    return this._tiers;  
  }  
  set {  
    this._tiers = value;  
  }  
}
```

```
[Property("rem", ColumnType="String")]
```

```
public string rem {  
  get {  
    return this._rem;  
  }  
  set {  
    this._rem = value;  
  }  
}
```

```
[PrimaryKey(PrimaryKeyType.Native, "id", ColumnType="Int64")]
```

```
public long id {  
  get {  
    return this._id;  
  }  
  set {  
    this._id = value;  
  }  
}
```

```
[BelongsTo("idcompte", NotNull=false)]
```

```
public compte compte {  
  get {  
    return this._compte;  
  }  
}
```

```

        this._compte = value;
    }
} set {

}

}
}

```

## Code de la DAO

```

namespace bankdemoDAL
{
    using System;
    using System.Collections.Generic;
    using System.Text;
    using Castle.ActiveRecord;
    using Castle.ActiveRecord.Queries;
    using NHibernate;

    public interface IdaoGenerics<T, U, V>
    {
        bool deleteAllObjects();
        bool deleteObject(T objet);
        T[] executeQuery(Castle.ActiveRecord.Queries.SimpleQuery q);
        V[] executeQuery2(Castle.ActiveRecord.Queries.SimpleQuery q);
        T getObject(U ID);
        IList<T> getObjects();
        IList<T> getObjectsByPage(int x, int y);
        T newObject(T objet);
        T updateObject(T objet);
    }

    public class daoGenerics<T, U, V> : bankdemoDAL.IdaoGenerics<T,U,V>
    {
        public daoGenerics()
        {
        }
        ///
        ///Recherche un objet par clé
        public T getObject(U ID)
        {
            try
            {
                return ((T)(ActiveRecordMediator.FindByPrimaryKey(typeof(T), ID)));
            }
            catch(Exception e)
            {
                return default(T);
            }
        }
        ///
        ///Récupère la collection d'un type donné (max 100 en dur)
        public IList<T> getObjects()
        {
            try
            {
                return (IList<T>)(ActiveRecordMediator.SlicedFindAll(typeof(T),0,100));
            }
        }
    }
}

```

```
    {  
        return null;  
    }  
    catch (Exception e)
```

```
}
```

```
///
```

```
///Récupère x objets dans la collection d'un type donné à partir de l'enregistrement y
```

```
public IList<T> getObjectsByPage(int x, int y)  
{  
    try  
    {  
        return (IList<T>)(ActiveRecordMediator.SlicedFindAll(typeof(T), y, x));  
    }  
    catch (Exception e)  
    {  
        return null;  
    }  
}
```

```
///
```

```
///Supprime un objet
```

```
///
```

```
public bool deleteObject(T objet)  
{  
    try  
    {  
        ActiveRecordMediator.Delete(objet);  
        return true;  
    }  
    catch (Exception e)  
    {  
        return false;  
    }  
}
```

```
///
```

```
///Vide une table
```

```
///
```

```
public bool deleteAllObjects()  
{  
    try  
    {  
        ActiveRecordMediator.DeleteAll(typeof(T));  
        return true;  
    }  
    catch (Exception e)  
    {  
        return false;  
    }  
}
```

```
///
```

```
///Mise à jour d'un objet
```

```
///
```

```
public T updateObject(T objet)  
{  
    try  
    {  
        ActiveRecordMediator.Save(objet);  
        return objet;  
    }  
    catch (Exception e)
```

```

        return default(T);
    }
}

///
///Création d'un nouvel objet
///
public T newObject(T objet)
{
    try
    {
        ActiveRecordMediator.Create(objet);
        return objet;
    }

    catch (Exception e)
    {
        return default(T);
    }
}

///
///Exécute une requête de sélection (HQL)
public T[] executeQuery(SimpleQuery q)
{
    return (T[])ActiveRecordMediator.ExecuteQuery(q);
}

///
///Exécute une requête de sélection (HQL)
public V[] executeQuery2(SimpleQuery q)
{
    return (V[])ActiveRecordMediator.ExecuteQuery(q);
}
}
}

```

### 3.3. La couche métier (BL)

La couche métier contient les objets métier directement manipulés par l'IHM (choix d'une architecture simplifiée qui n'implémente pas de couche services). Dans notre exemple il n'y a qu'un objet métier Operations qui dérive de l'interface IOperations.

La couche métier interagit avec la DAL à travers l'interface IdaoGeneric.

Code de l'objet métier Operations

```
namespace bankdemoBL
```

```

using System;
using System.Collections.Generic;
using System.Collections;
using System.Text;
using bankdemoDAL;
using Castle.ActiveRecord.Framework;
using Castle.ActiveRecord.Queries;

```

```

public interface IOperations
{

```

```

    long authentifie(string user, string password);

```

```

        System.Collections.IList getAllMouvements(long idcompte);
        System.Collections.IList getComptesForClient(long idclient);
        bankdemoDAL.client getFicheClient(long idclient);
        bankdemoDAL.compte getFicheCompte(long idcompte);
        bankdemoDAL.mouvements getMouvement(long idmouv);
    }
}

public class Operations : bankdemoBL.IOperations
{
    public Int64 authentifie(string user, string password)
    {
        IdaoGenerics<client, Int64, compte> dao = new daoGenerics<client, Int64, compte>();
        SimpleQuery query = new SimpleQuery(typeof(client), @"from client c where c.numcli = ? and c.password = ?", user, password);
        client[] rep = (client[])dao.executeQuery(query);
        if (rep.Length == 1)
        {
            if (rep[0].numcli == user && rep[0].password == password)
                return rep[0].id;
            else
                return 0;
        }
        else
            return 0;
    }

    public IList getAllMouvements(long idcompte)
    {
        IdaoGenerics<compte, Int64, mouvements> dao = new daoGenerics<compte, Int64, mouvements>();
        compte cmp = dao.getObject(idcompte);
        if (cmp != null)
            return cmp.mouvements;
        else
            return null;
    }

    public IList getComptesForClient(long idclient)
    {
        IdaoGenerics<client, Int64, compte> dao = new daoGenerics<client, Int64, compte>();
        client cl = dao.getObject(idclient);
        if (cl != null)
            return cl.comptes;
        else
            return null;
    }

    public mouvements getMouvement(long idmouv)
    {
        IdaoGenerics<mouvements, Int64, mouvements> dao = new daoGenerics<mouvements, Int64, mouvements>();
        return dao.getObject(idmouv);
    }

    public compte getFicheCompte(long idcompte)
    {
        IdaoGenerics<compte, Int64, mouvements> dao = new daoGenerics<compte, Int64, mouvements>();
        return dao.getObject(idcompte);
    }
}

```

```

    {
        IdaoGenerics<client, Int64, compte> dao = new daoGenerics<client, Int64, compte>();
        return dao.getObject(idclient);
    }
    public client getFicheClient(long idclient)
}
}

```

### 3.4. Couche UI (IHM)

L'IHM comporte cinq pages asp.net:

- Login.aspx : page de connexion à l'application
- Accueil.aspx : page d'accueil contenant des informations
- Comptes.aspx : liste des comptes du client
- Mouvements.aspx : liste des opérations pour les comptes du client
- Client.aspx : fiche client

#### 3.4.1. Gestion de la sécurité dans l'application

La sécurité sera gérée de façon standard asp.net. La page de connexion comportera un contrôle login qui utilisera un membership provider personnalisé. Celui-ci appellera la méthode authentifie de l'objet métier Operations.

Le fichier Web.config sera modifié pour prendre en compte ces principes :

```

<authentication mode="Forms">
  <forms loginUrl="login.aspx" timeout="20">
  </forms>
</authentication>
<authorization>
  <deny users="?" />
</authorization>

<roleManager defaultProvider="AspNetWindowsTokenRoleProvider" />
<membership defaultProvider="MonMembershipProvider">
  <providers>
    <add name="MonMembershipProvider"
      passwordFormat="Hashed"
      minRequiredPasswordLength="2"
      minRequiredNonalphanumericCharacters="0"
      requiresQuestionAndAnswer="False"
      requiresUniqueEmail="True"
      type="bankdemoAJAX.MonMembershipProvider"/>
  </providers>
</membership>

```

Le membershipProvider derive de MembershipProvider. La seule méthode derive avec implementation et la méthode ValidateUser qui permet de valider le couple user et password. Ces données sont stockées dans la base de données (table client) et dans l'objet client. Une fois le client authentifié, on place son id dans la session http ce qui permettra de récupérer cette valeur dans les écrans et permettre le filtrage des comptes par exemple.

```

IOperations op = new Operations();
Int64 res = op.authentifie(username, password);
if (res != 0)
{
    HttpContext.Current.Session.Add("idclient", res.ToString());
}

```

```
        return true;
    }
    else
        return false;
}
```

## Code de MonMembershipProvider

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using bankdemoBL;
```

```
/// <summary>
/// Description résumée de MonMembershipProvider
/// </summary>
///
namespace bankdemoAJAX
{
    public class MonMembershipProvider : MembershipProvider
    {
        public MonMembershipProvider()
        {
            //
            // TODO : ajoutez ici la logique du constructeur
            //
        }

        public override void Initialize(string name, System.Collections.Specialized.NameValueCollection config)
        {
        }

        public override string ApplicationName
        {
            get
            {
                throw new Exception("The method or operation is not implemented.");
            }
            set
            {
                throw new Exception("The method or operation is not implemented.");
            }
        }

        public override bool ChangePassword(string username, string oldPassword, string newPassword)
        {
            throw new Exception("The method or operation is not implemented.");
        }

        public override bool ChangePasswordQuestionAndAnswer(string username, string password, string newPasswordQuestion, string
newPasswordAnswer)
        {
            throw new Exception("The method or operation is not implemented.");
        }

        public override MembershipUser CreateUser(string username, string password, string email, string passwordQuestion, string
passwordAnswer, bool isApproved, object providerUserKey, out MembershipCreateStatus status)
        {
            throw new Exception("The method or operation is not implemented.");
        }

        protected override byte[] DecryptPassword(byte[] encodedPassword)
        {
            return base.DecryptPassword(encodedPassword);
        }

        public override bool DeleteUser(string username, bool deleteAllRelatedData)
        {
            throw new Exception("The method or operation is not implemented.");
        }

        public override string Description
        {

```

```

    get
    {
        return base.Description;
    }
}
public override bool EnablePasswordReset
{
    get { throw new Exception("The method or operation is not implemented."); }
}
public override bool EnablePasswordRetrieval
{
    get { throw new Exception("The method or operation is not implemented."); }
}
public override MembershipUserCollection FindUsersByEmail(string emailToMatch, int pageIndex, int pageSize, out int totalRecords)
{
    throw new Exception("The method or operation is not implemented.");
}
public override MembershipUserCollection FindUsersByName(string usernameToMatch, int pageIndex, int pageSize, out int
totalRecords)
{
    throw new Exception("The method or operation is not implemented.");
}
public override MembershipUserCollection GetAllUsers(int pageIndex, int pageSize, out int totalRecords)
{
    throw new Exception("The method or operation is not implemented.");
}
public override int GetNumberOfUsersOnline()
{
    throw new Exception("The method or operation is not implemented.");
}
public override string GetPassword(string username, string answer)
{
    throw new Exception("The method or operation is not implemented.");
}
public override MembershipUser GetUser(object providerUserKey, bool userIsOnline)
{
    throw new Exception("The method or operation is not implemented.");
}
public override MembershipUser GetUser(string username, bool userIsOnline)
{
    throw new Exception("The method or operation is not implemented.");
}
public override string GetUserNameByEmail(string email)
{
    throw new Exception("The method or operation is not implemented.");
}
}
public override bool ValidateUser(string username, string password)
{
    IOperations op = new Operations();
    Int64 res = op.authenticate(username, password);
    if (res != 0)
    {
        HttpContext.Current.Session.Add("idclient", res.ToString());
        return true;
    }
    else
        return false;
}
public override int MaxInvalidPasswordAttempts
{
    get { throw new Exception("The method or operation is not implemented."); }
}
public override int MinRequiredNonAlphanumericCharacters
{
    get { throw new Exception("The method or operation is not implemented."); }
}
public override int MinRequiredPasswordLength
{
    get { throw new Exception("The method or operation is not implemented."); }
}
public override string Name
{
    get
    {
        return "MonMembershipProvider";
    }
}
}

```

```

    }
    public override int PasswordAttemptWindow
    {
        get { throw new Exception("The method or operation is not implemented."); }
    }
    public override bool RequiresQuestionAndAnswer
    {
        get { throw new Exception("The method or operation is not implemented."); }
    }
    public override bool RequiresUniqueEmail
    {
        get { throw new Exception("The method or operation is not implemented."); }
    }
    public override MembershipPasswordFormat PasswordFormat
    {
        get { throw new Exception("The method or operation is not implemented."); }
    }
    public override string PasswordStrengthRegularExpression
    {
        get { throw new Exception("The method or operation is not implemented."); }
    }
    public override string ResetPassword(string username, string answer)
    {
        throw new Exception("The method or operation is not implemented.");
    }
    public override void UpdateUser(MembershipUser user)
    {
        throw new Exception("The method or operation is not implemented.");
    }
    public override bool UnlockUser(string userName)
    {
        throw new Exception("The method or operation is not implemented.");
    }
}
}
}

```

### 3.4.2. Initialisation du framework castle

Le fichier Web.config doit contenir les paramètres d'initialisation des frameworks utilisée (castle, nhibernate).

```

<configSections>
    <section name="activerecord" type="Castle.ActiveRecord.Framework.Config.ActiveRecordSectionHandler,
Castle.ActiveRecord"/>
</configSections>
<activerecord>
    isWeb="true"
    <config>
        <add key="hibernate.connection.driver_class" value="NHibernate.Driver.SqlClientDriver"/>
        <add key="hibernate.dialect" value="NHibernate.Dialect.MsSql2000Dialect"/>
        <add key="hibernate.connection.provider" value="NHibernate.Connection.DriverConnectionProvider"/>
        <add key="hibernate.connection.connection_string" value="Data Source=Server;Initial
Catalog=BANKDEMO;Integrated Security=True"/>
    </config>
</activerecord>

```

Pour respecter le principe des sessions hibernate et active record, il faut créer une classe d'application dérivant de HttpApplication qui prennent en charge (sur l'événement start\_application) l'initialisation du framework.

```

using System;
using System.Web;
using Castle.ActiveRecord;
using Castle.ActiveRecord.Framework;
using Castle.ActiveRecord.Framework.Config;
using bankdemoDAL;

```

```
namespace bankdemoAJAX
```

```

{
  /// <summary>
  /// Description résumée de Init
  /// </summary>
  public class Init : HttpApplication
  {
    public Init()
    {
      //
      // TODO : ajoutez ici la logique du constructeur
      //
    }
    protected void Application_Start(Object sender, EventArgs e)
    {
      // Replace the code below as you want to match your
      // preference about ActiveRecord initialization

      IConfigurationSource source = System.Configuration.ConfigurationManager.GetSection("activerecord") as
      Castle.ActiveRecord.Framework.IConfigurationSource;

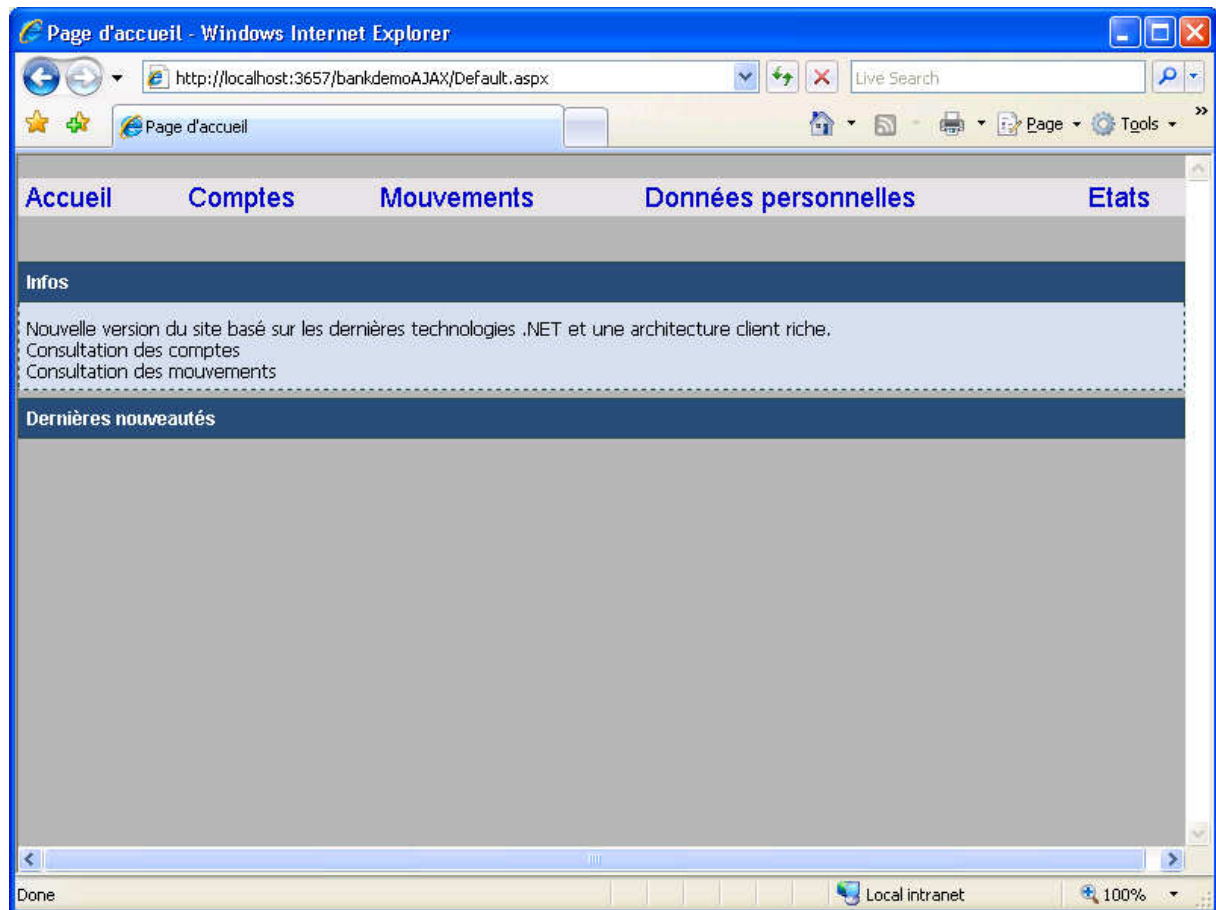
      ActiveRecordStarter.Initialize(source,
        typeof(client),
        typeof(compte),
        typeof(mouvements));
    }
  }
}

```

### 3.5. Page d'accueil

La page d'accueil est une page asp.net ajax qui contient les contrôles :

- Script manager
- UpdatePanel
- Menu
- 2 accordions qui permettent d'afficher les infos et les nouveautés du site



### 3.6. Page de consultation des comptes

La page de consultation des comptes est une page asp.net ajax qui contient les contrôles :

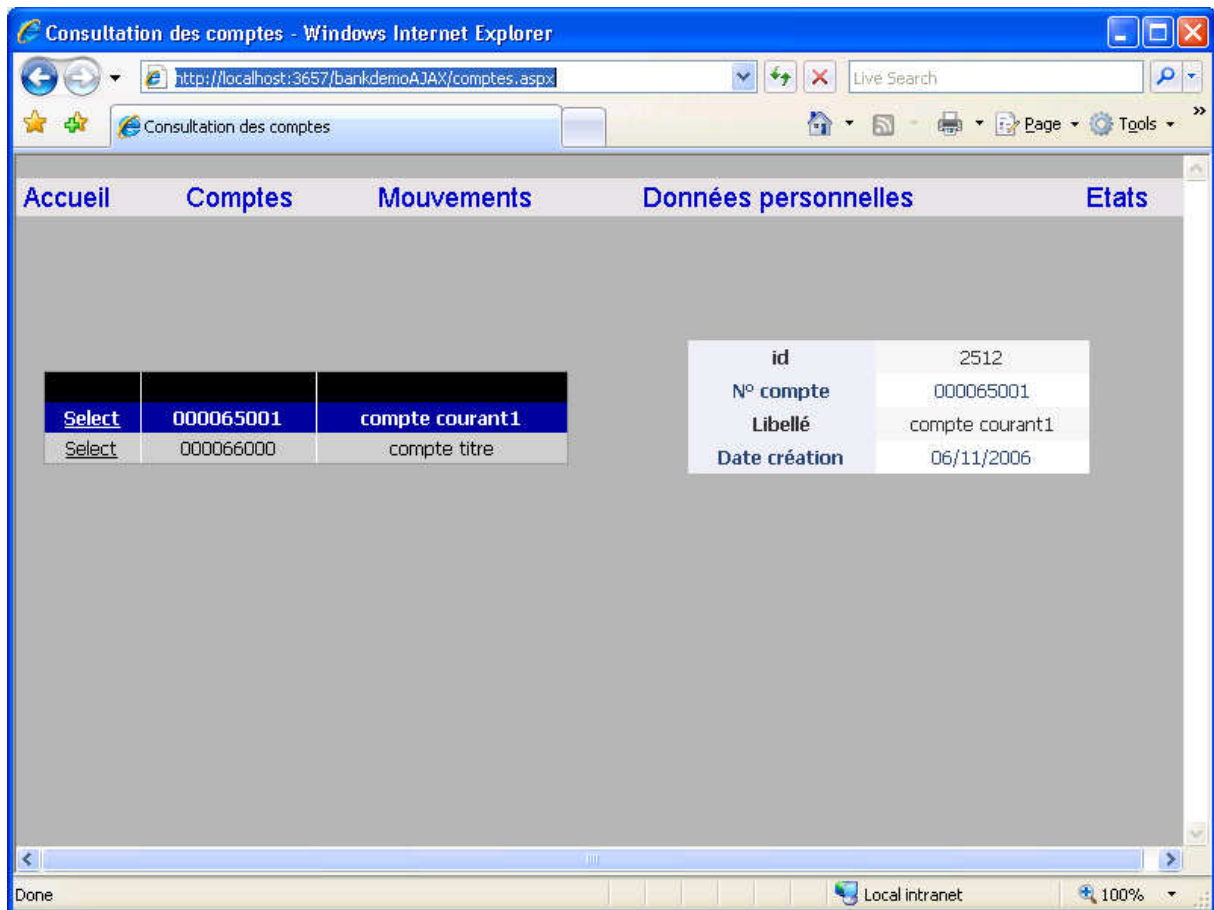
- Script manager
- Menu
- Update Panel
- Gridview
- DetailView

L'intérêt de placer les contrôles Gridview et detailView dans un updatePanel ajax (Control toolkit) est de profiter de la pagination et du tri côté client.

Le gridview récupère ses données via un binding sur l'objet métier Operations et appel de la méthode getComptesForClient(long idclient). L'idclient en session est passée en paramètre de cette méthode. La déclaration du filtre et du binding est réalisée uniquement en visual sans écriture de code.

Le detailview récupère ses données via un binding sur l'objet métier et appel de la méthode getCompte(long idcompte). L'idcompte est récupéré sur la valeur de la ligne sélectionnée dans le gridView sans aucune écriture de code.

La sélection d'un nouveau compte entraîne **le rafraîchissement du detailView uniquement (magie d'Ajax).**

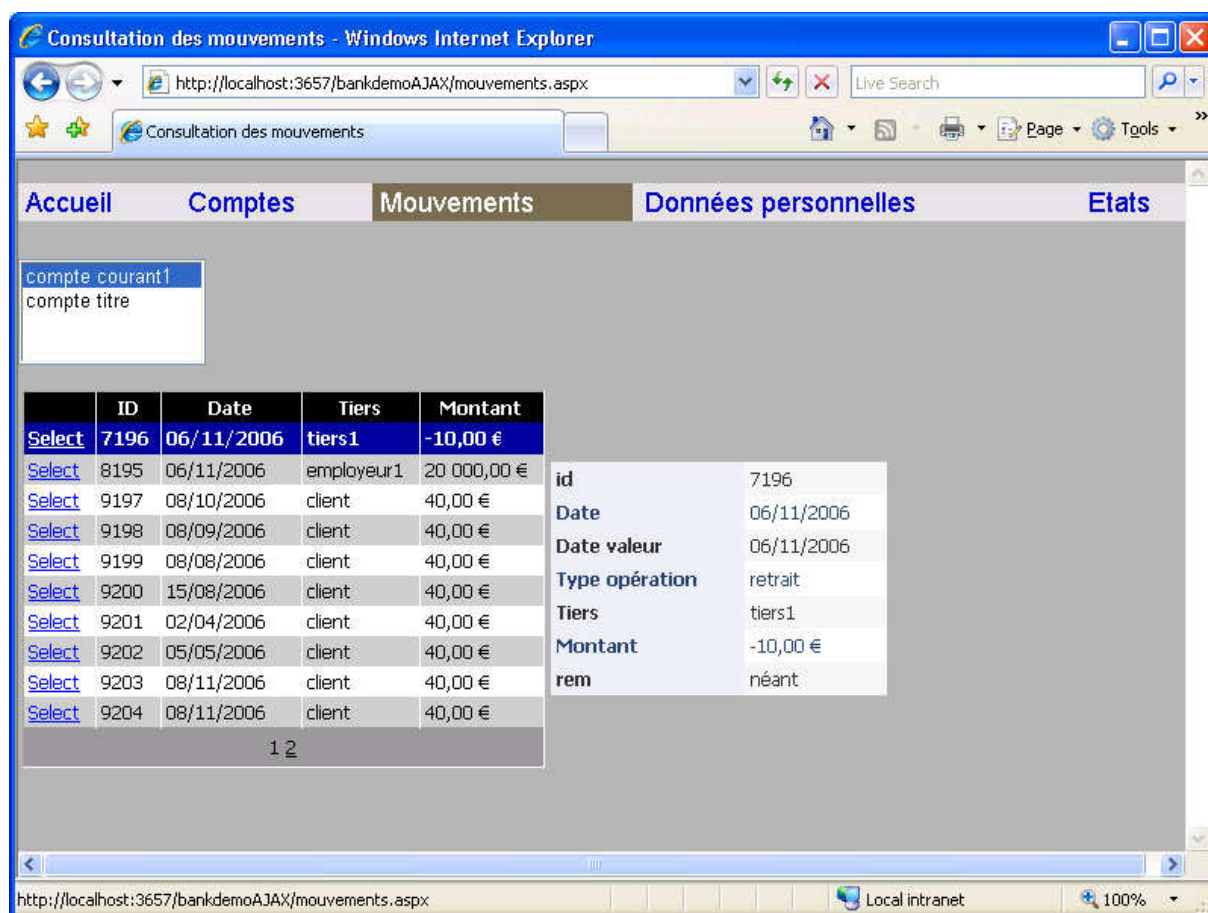


### 3.7. Page de consultation des opérations

Cette page est une page asp.net ajax qui contient les contrôles suivants :

- Script manager
- Menu
- Update panel
- Listbox (liste des comptes)
- GridView (liste des opérations du compte sélectionné)
- DetailView (détail de l'opération sélectionnée)

Le binding des données et le passage des paramètres est réalisé via les assistants visual studio 2005.

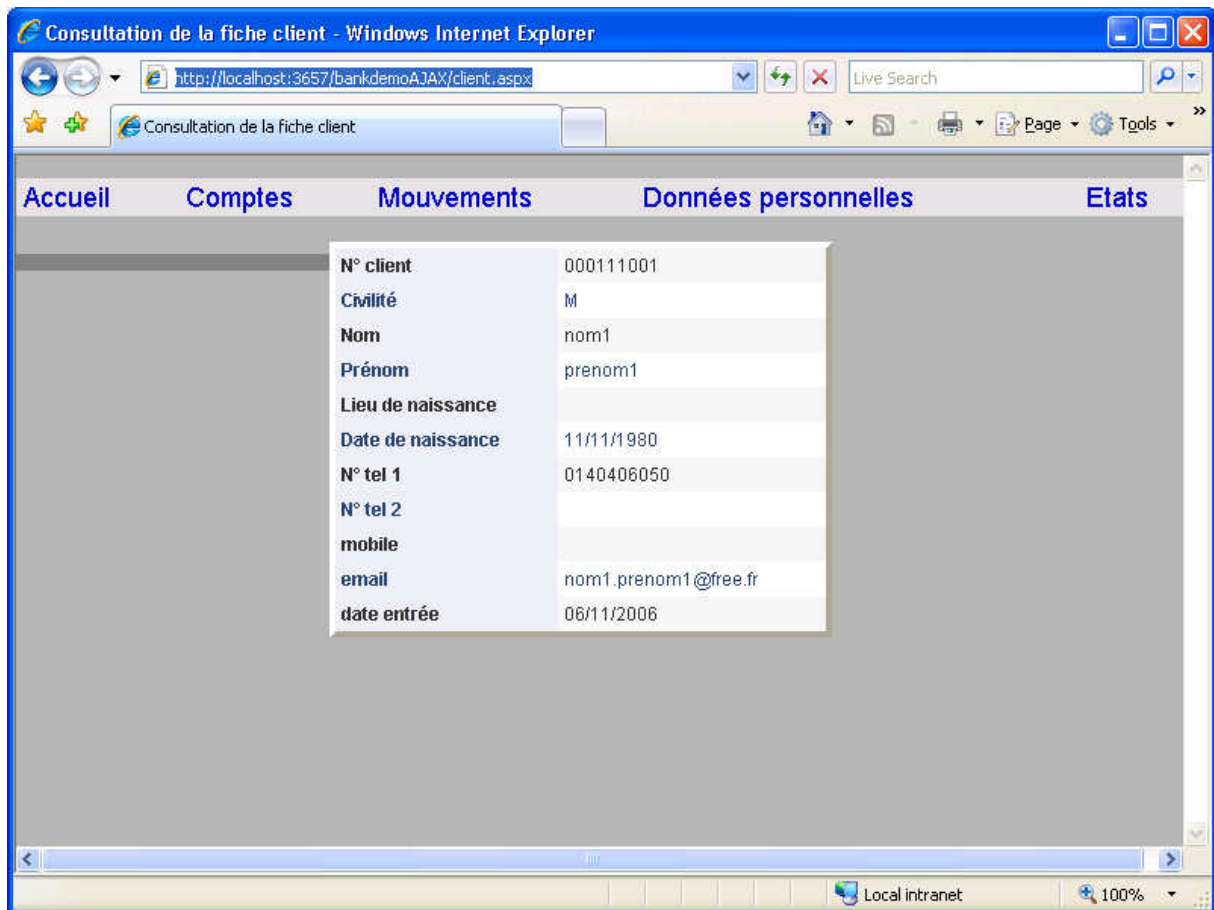


### 3.8. Page de consultation de la fiche client

Cette page est une page asp.net ajax qui contient les contrôles suivants :

- Script manager
- Menu
- 2 update panel
- detailview
- dragpanelExtender

L'originalité de cette page est la présence du dragPanelExtender qui permet de faire un drag-and-drop du detailView dans les 2 update Panel.



#### 4. CONCLUSIONS

L'utilisation d'un outil de mapping OR et du pattern Active Record permet de gagner un temps considérable par rapport à un développement de type dataset/datatable. De plus, l'utilisation d'un DSL (Active writer) pour générer les objets de domaine au format compatible avec le framework Castle entraîne une productivité impressionnante.

Asp.net ajax (ex Atlas) a beaucoup mûri et l'adjonction du control toolkit permet de construire des interfaces Ajax sans écrire du code.

Le modèle d'architecture choisi (3 couches sans couche service) est très simple et n'implémente pas tous les patterns à la mode mais permet de développer des applications très rapidement en restant propre sur certains concepts (couplage lâche, separation of concern ...).

De plus cette architecture permet de changer de type de base de données à la volée (test réalisé entre sql server, Oracle et Access) juste en modifiant les paramètres dans Web.config.