**ADO.NET - Programmation Client/Serveur C#**

|  |
| --- |
| - Intro .NET  - Recap POO  - Mise a Jour - MODE CONNECTE  + Ajouter, Supprimer, Modifier, Afficher  - Mise a Jour - MODE DECONNECTE  + Ajout – Modification – Suppression  + Consultation ( For – Select – Find )  + Save BDD parameters (Suppression – Modification – Insertion)  + Save BDD ( avec Builder)  + Button << >> Premier - Dernier  + Button < > Suivant - Précédent  + Les Procédures Stocké  + Les Fonctions  - Mise a jour avec - DATAVIEW  - Mise a jour avec - DATABINDING  - la DataRelation  - Transactions – Avec Procédures Stockés  - XML : Extensible markup language  + XML C#  + XML to HTML  + Write XML w/o Version  - Les Transactions  + Commit – RollBack  - Creation d’un Executable  + Création du setup.exe  + Création du setup.exe dans le projet  + Gestion du Connexion string depuis App.Config  - Plus+ |

|  |  |  |
| --- | --- | --- |
| .NET.jpg La Plateforme .NET  (pronon « dot net ») est le nom donné à un ensemble de produits et de technologies informatiques de l'entreprise Microsoft pour rendre des applications facilement portables sur Internet. Le but est de fournir un serveur web local permettant de gérer des services et évitant d'externaliser des données privées sur un service web de stockage ou un hébergement web tiers. | | |
| .NET.jpg Le .NET Framework  Le .NET Framework est un cadriciel (framework) pouvant être utilisé par un système d'exploitation Microsoft Windows et Microsoft Windows Mobile, s'appuie sur la norme Common Language Infrastructure (CLI), Il a pour but de faciliter la tâche des développeurs en proposant une approche unifiée à la conception d'applications Windows ou Web, tout en introduisant des facilités pour le développement, le déploiement et la maintenance d'applications. Il a besoin d'être installé sur la machine de l'utilisateur final, rendant les applications créées sous cet environnement impropres à un usage portable. | | **.NET Fram.gif**  **c#.jpeg** |
| ADO.NET.jpg ADO.NET c’est une technologie Microsoft qui prend en charge le modèle de programmation connecter est déconnecter et permet la prise en charge du langage XML.  Fourni l’accès au base de données et de l’objet DataSet. | **Modéle Ado.NET.jpg** | | |

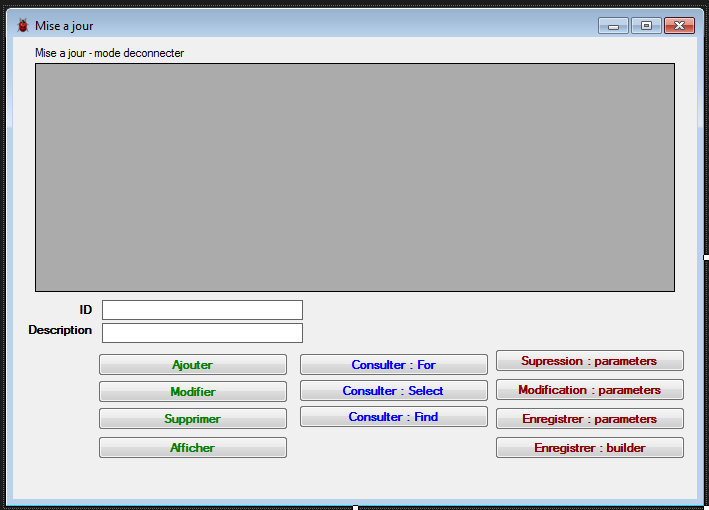
|  |  |
| --- | --- |
| Recap POO  Variable.png Les attributs fournissent des métadonnées définies par l'utilisateur supplémentaires.La plupart du temps, ils sont utilisés pour stocker les informations supplémentaires relatives à un type de l'assembly, ou pour modifier le comportement d'un membre de type dans l'environnement au moment du design ou dans l'environnement d'exécution.  Les attributs sont des classes qui héritent de [System.Attribute](https://msdn.microsoft.com/fr-fr/library/system.attribute(v=vs.110).aspx).Les langages qui prennent en charge l'utilisation d'attributs ont chacun leur propre syntaxe pour l'application d'attributs à un élément du langage.Les attributs peuvent être appliqués à presque n'importe quel élément de langage ; les éléments spécifiques auxquels un attribut peut être appliqué sont définis par l'[AttributeUsageAttribute](https://msdn.microsoft.com/fr-fr/library/system.attributeusageattribute(v=vs.110).aspx) qui est appliqué à cette classe d'attributs.   |  | | --- | | Si l'accessibilité déclarée de M est ****public****, le domaine d'accessibilité de M correspond à celui de T.  Si l'accessibilité déclarée de M est ****protected internal****, le domaine d'accessibilité de M correspond à l'intersection du domaine d'accessibilité de T avec le texte de programme de P et le texte de programme de n'importe quel type dérivé de T déclaré en dehors de P.  Si l'accessibilité déclarée de M est ****protected****, le domaine d'accessibilité de M correspond à l'intersection du domaine d'accessibilité de T avec le texte de programme de T et n'importe quel type dérivé de T.  Si l'accessibilité déclarée de M est ****internal****, le domaine d'accessibilité de M correspond à l'intersection du domaine d'accessibilité de T avec le texte de programme de P.  Si l'accessibilité déclarée de M est ****private****, le domaine d'accessibilité de M correspond au texte de programme de T. |   Classes.png Une classe est un type référence qui peut être dérivé directement d'une autre classe et qui est implicitement dérivé de [System.Object](https://msdn.microsoft.com/fr-fr/library/system.object(v=vs.110).aspx). Une classe définit les opérations qu'un objet *(qui est une instance de la classe)* peut effectuer (méthodes, événements ou propriétés) et les données que l'objet contient (champs).Bien qu'une classe inclue généralement la définition et l'implémentation (contrairement aux interfaces, par exemple, qui contiennent uniquement la définition sans l'implémentation), elle peut comporter un ou plusieurs membres dépourvus d'implémentation.  Delegate.png Les délégués sont des types référence qui jouent un rôle similaire à celui des pointeurs fonction en C++.Ils sont utilisés pour les gestionnaires d'événements et les fonctions de rappel dans le .NET Framework.Contrairement aux pointeurs fonction, les délégués sont de type sécurisé, sûr et vérifiable.Un type délégué peut représenter n'importe quelle méthode d'instance ou méthode statique ayant une signature compatible.  enum.png Une énumération (enum) est un type valeur qui hérite directement de [System.Enum](https://msdn.microsoft.com/fr-fr/library/system.enum(v=vs.110).aspx) et qui fournit des noms de remplacement pour les valeurs d'un type primitif sous-jacent.Un type énumération a un nom, un type sous-jacent qui doit être l'un des types d'entiers signés ou non signés intégrés (tels que [Byte](https://msdn.microsoft.com/fr-fr/library/system.byte(v=vs.110).aspx), [Int32](https://msdn.microsoft.com/fr-fr/library/system.int32(v=vs.110).aspx) ou [UInt64](https://msdn.microsoft.com/fr-fr/library/system.uint64(v=vs.110).aspx)) et un ensemble de champs.  EventHandler.png Un événement définit un incident auquel il est possible de répondre, et définit les méthodes permettant de s'abonner à l'événement, d'annuler cet abonnement et de déclencher l'événement.Les événements sont souvent utilisés pour signaler d'autres types de changements d'état.Pour plus d'informations, consultez [Gestion et déclenchement d'événements](https://msdn.microsoft.com/fr-fr/library/edzehd2t(v=vs.110).aspx).  Icon Class.jpeg Un constructeur est un genre de méthode particulier qui crée de nouvelles instances d'une classe ou d'une structure.Comme n'importe quelle autre méthode, un constructeur peut inclure des paramètres ; toutefois, les constructeurs n'ont pas de valeur de retour (en d'autres termes, ils retournent ****void****).  Methodes.jpeg Une méthode décrit les opérations qui sont disponibles sur le type.La signature d'une méthode spécifie les types autorisés de tous ses paramètres et de sa valeur de retour.  Bien que la plupart des méthodes définissent le nombre précis de paramètres requis pour les appels de méthode, certaines méthodes acceptent un nombre variable de paramètres.Le dernier paramètre déclaré de ces méthodes est marqué avec l'attribut [ParamArrayAttribute](https://msdn.microsoft.com/fr-fr/library/system.paramarrayattribute(v=vs.110).aspx).Les compilateurs de langage fournissent généralement un mot clé, tel que ****params**** en C# et****ParamArray**** en Visual Basic, qui rendent l'utilisation explicite de [ParamArrayAttribute](https://msdn.microsoft.com/fr-fr/library/system.paramarrayattribute(v=vs.110).aspx) inutile.  Propreties.jpeg Une propriété nomme une valeur ou un état du type et définit des méthodes pour obtenir ou définir la valeur de la propriété.Les propriétés peuvent être des types primitifs, des collections de types primitifs, des types définis par l'utilisateur ou des collections de types définis par l'utilisateur.Les propriétés sont souvent utilisées pour maintenir l'interface publique d'un type indépendante de la représentation réelle du type.Cela permet aux propriétés de refléter des valeurs qui ne sont pas directement stockées dans la classe (par exemple, lorsqu'une propriété retourne une valeur calculée) ou d'effectuer une validation avant que des valeurs soient assignées à des champs privés.L'exemple suivant illustre ce dernier modèle.  Struct.png Une structure est un type valeur qui dérive implicitement de [System.ValueType](https://msdn.microsoft.com/fr-fr/library/system.valuetype(v=vs.110).aspx) qui, à son tour, est dérivé de [System.Object](https://msdn.microsoft.com/fr-fr/library/system.object(v=vs.110).aspx).Une structure est très utile pour représenter des valeurs dont les besoins en ressources mémoire sont faibles, ainsi que pour passer des valeurs en tant que paramètres par valeur à des méthodes qui ont des paramètres fortement typés.Dans la bibliothèque de classes .NET Framework, tous les types de données primitifs ([Boolean](https://msdn.microsoft.com/fr-fr/library/system.boolean(v=vs.110).aspx), [Byte](https://msdn.microsoft.com/fr-fr/library/system.byte(v=vs.110).aspx), [Char](https://msdn.microsoft.com/fr-fr/library/system.char(v=vs.110).aspx),[DateTime](https://msdn.microsoft.com/fr-fr/library/system.datetime(v=vs.110).aspx), [Decimal](https://msdn.microsoft.com/fr-fr/library/system.decimal(v=vs.110).aspx), [Double](https://msdn.microsoft.com/fr-fr/library/system.double(v=vs.110).aspx), [Int16](https://msdn.microsoft.com/fr-fr/library/system.int16(v=vs.110).aspx), [Int32](https://msdn.microsoft.com/fr-fr/library/system.int32(v=vs.110).aspx), [Int64](https://msdn.microsoft.com/fr-fr/library/system.int64(v=vs.110).aspx), [SByte](https://msdn.microsoft.com/fr-fr/library/system.sbyte(v=vs.110).aspx), [Single](https://msdn.microsoft.com/fr-fr/library/system.single(v=vs.110).aspx), [UInt16](https://msdn.microsoft.com/fr-fr/library/system.uint16(v=vs.110).aspx), [UInt32](https://msdn.microsoft.com/fr-fr/library/system.uint32(v=vs.110).aspx) et [UInt64](https://msdn.microsoft.com/fr-fr/library/system.uint64(v=vs.110).aspx)) sont définis en tant que structures. |

|  |
| --- |
| La spécialisation représente la démarche inverse de la généralisation puisqu’elle  consiste à créer à partir d’une classe, plusieurs classes spécialisées.  L’encapsulation : les données ne sont accessibles qu’à partir des opérations définies  dans la classe. Le principe d’encapsulation renforce l’autonomie et l’indépendance de  chaque classe et donne une forte potentialité de définition de classe réutilisable  Le polymorphisme : est la capacité donnée à une opération de s’exécuter  différemment suivant le contexte de la classe où elle se trouve.  La persistance est la propriété donnée à un objet de continuer à exister après la fin de  l’exécution du programme qui l’a créé.  Par défaut dans l’approche objet, aucun objet n’est persistant. Les modèles décrivent  le système en exécution en mémoire centrale et ne tiennent pas compte a priori de  l’état du système qui doit être stocké sur disque.  L’héritage est un mécanisme qui permet d’assurer une grande variabilité dans la  réutilisation des objets. Il existe deux techniques liées à l’héritage : les classes  abstraites et l’héritage multiple. |

**Mise a Jour - MODE CONNECTE   
+ Ajouter, Supprimer, Modifier, Afficher**

|  |
| --- |
| SqlConnection cnx = new SqlConnection("Data Source=.;Initial Catalog=MAJ;Integrated Security=True");  SqlCommand cmd;  SqlDataReader dr; |
| private void btnAjouter\_Click(object sender, EventArgs e)  cnx.Open();  string insert = "insert into joueurs values  (" + textBox1.Text + ",'" + textBox2.Text + "','" + comboBox1.Text + "')";  cmd = new SqlCommand(insert, cnx);  cmd.ExecuteNonQuery();  cnx.Close(); |
| private void btnSupprimer\_Click(object sender, EventArgs e)  cnx.Open();  string delete = "delete from joueurs where CodeJ = (" + textBox1.Text + ")";  cmd = new SqlCommand(delete,cnx);  cmd.ExecuteNonQuery();  cnx.Close(); |
| private void btnModifier\_Click(object sender, EventArgs e)  cnx.Open();  string update = "update joueurs set NomJ= '" + textBox2.Text + "'  , Pays='" + comboBox1.Text + "' where CodeJ = " + textBox1.Text +;  cmd = new SqlCommand(update,cnx);  cmd.ExecuteNonQuery();  nx.Close(); |
| private void btnAfficher\_Click(object sender, EventArgs e)  cnx.Open();  string select = " select \* from joueurs";  cmd = new SqlCommand(select, cnx);  dr = cmd.ExecuteReader();  // Solution 1 : avec DataTable  DataTable dt = new DataTable();  dt.Load(dr);  dataGridView1.DataSource = dt;   // Solution 2 : Avec ListeBox  while(dr.read () )  { lbAffichage.Items.Add(dr[1] + "\_" + dr[2] ); }  dr.Close();  cnx.Close(); |

**Mise a Jour - MODE DECONNECTE  
--**

****

**+ Program.cs – Form\_Load**

|  |
| --- |
| namespace GestionEau  { static class Program  {  public static SqlConnection cnx = new SqlConnection(@"Data Source=Rlz-PC\Rlz; Initial Catalog=GestionEau;  Integrated Security=True");  public static SqlDataReader dr;  public static SqlDataAdapter da;  public static DataSet ds = new DataSet();  public static Datatable dt = new DataTable();  /// <summary>  /// The main entry point for the application.  /// </summary>  [STAThread]  static void Main()  {  Application.EnableVisualStyles();  Application.SetCompatibleTextRenderingDefault(false);  Application.Run(new Menu());  }  } } |
| private void FormulaireGA\_Load(object sender, EventArgs e)  Program.da = new SqlDataAdapter("select \* from AbonnementEau", Program.cnx);  Program.da.MissingSchemaAction = MissingSchemaAction.AddWithKey;  Program.da.Fill(Program.ds);  dt = Program.ds.Tables[0];  dataGridView1.DataSource = dt; |

**+ Ajout – Modification – Suppression**

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Threading.Tasks;  using System.Windows.Forms;  using System.Data.SqlClient;  using System.Data; |

|  |
| --- |
| private void btn\_Ajout\_Click(object sender, EventArgs e)  bool Find = true;  // Condition : si idAbonnement Déjà dans la table - Find=False; Break;  for (int i = 0; i < dt.Rows.Count; i++)  {  if (txt\_idAbonnement.Text == dt.Rows[i][0].ToString() )  {  Find = false;  MessageBox.Show("Déjà Ajouter.","infos", MessageBoxButtons.OK,MessageBoxIcon.Warning);  break;  }  }  // Condition : si idAbonnement n'existe pas dans la table - Find=true; Remplissage de la ligne (row).  if (Find == true)  {  DataRow dar = dt.NewRow();  dar[0] = Convert.ToInt16(txt\_idAbonnement.Text) ;  dar[1] = txt\_dateAb.Text;  dar[2] = Convert.ToInt16 (cmb\_idUsage.Text);  dar[3] = Convert.ToInt16 (cmb\_idAbonne.Text);  dt.Rows.Add(dar);  MessageBox.Show("Ajouter.", "infos", MessageBoxButtons.OK , MessageBoxIcon.Asterisk) ;  } |
| private void btn\_Modification\_Click(object sender, EventArgs e)  DataRow dar2;  dar2 = dt.Rows.Find(txt\_idAbonnement.Text);  dar2.BeginEdit();  dar2[0] = Convert.ToInt16(txt\_idAbonnement.Text);  dar2[1] = txt\_dateAb.Text;  dar2[2] = cmb\_idUsage.Text;  dar2[3] = cmb\_idAbonne.Text;  dar2.EndEdit(); |
| private void btn\_Suppresion\_Click(object sender, EventArgs e)  bool Find2 = false;  int pos = -1;  for(int i=0; i < dt.Rows.Count; i++)  {  if(dt.Rows[i].RowState != DataRowState.Deleted)  {  if( txt\_idAbonnement.Text == dt.Rows[i][0].ToString() )  {  pos = i;  Find2 = true;  break;  }  }  }  if (Find2 == true && pos !=-1)  { dt.Rows[pos].Delete(); } |

**+ Consultation ( For – Select – Find )**

|  |
| --- |
| private void btnConsulterFor\_Click(object sender, EventArgs e)  for (int i = 0; i < dt.Rows.Count; i++)  {  if (dt.Rows[i].RowState != DataRowState.Deleted)  {  if (dt.Rows[i][0].ToString() == txtID.Text.ToString())  {  txtD.Text = dt.Rows[i][1].ToString();  }  }  } |
| private void btnConsulterSelect\_Click(object sender, EventArgs e)  DataRow[] dar2;  dar2 = dt.Select("ID\_Machine=" + Convert.ToInt16(txtID.Text));  foreach (DataRow dar in dar2)  {txtD.Text = dar[1].ToString(); } |
| private void btnConsulterFind\_Click(object sender, EventArgs e)  DataRow dar3;  dar3 = dt.Rows.Find(txtID.Text);  ;l dar3[1].ToString(); |

**+ Save BDD parameters (Suppression – Modification – Insertion)**

|  |
| --- |
| private void btnSuppresion\_Click(object sender, EventArgs e)  SqlCommand cmd3 = new SqlCommand();  cmd3.Connection = Program.cnx;  cmd3.CommandType = CommandType.Text;  SqlParameter p5 = new SqlParameter();  p5.ParameterName = "@codeEE";  p5.DbType = DbType.Int16;  p5.SourceColumn = "ID\_Machine";  p5.SourceVersion = DataRowVersion.Original;  cmd3.Parameters.Add(p5);  cmd3.CommandText = " DELETE MACHINE WHERE ID\_Machine = @codeEE ";  Program.da.DeleteCommand = cmd3;  Program.da.Update(Program.ds.Tables[0].Select("", "", DataViewRowState.Deleted)); |
| private void btnModification\_Click(object sender, EventArgs e)  SqlCommand cmd2 = new SqlCommand();  cmd2.Connection = Program.cnx;  cmd2.CommandType = CommandType.Text;  SqlParameter p3 = new SqlParameter();  p3.ParameterName = "@codeE";  p3.DbType = DbType.Int16;  p3.SourceColumn = "ID\_Machine";  p3.SourceVersion = DataRowVersion.Original;  cmd2.Parameters.Add(p3);  SqlParameter p4 = new SqlParameter();  p4.ParameterName = "@DescE";  p4.DbType = DbType.String;  p4.Size = 50;  p4.SourceColumn = "Description\_Machine";  p4.SourceVersion = DataRowVersion.Current;  cmd2.Parameters.Add(p4);  cmd2.CommandText = "update machine set Description\_Machine=(@DescE) WHERE ID\_Machine = (@CodeE)";  Program.da.UpdateCommand = cmd2;  Program.da.Update(Program.ds.Tables[0].Select("", "", DataViewRowState.ModifiedCurrent)); |
| private void btnEnregistrer2\_Click(object sender, EventArgs e)  SqlCommand cmd1 = new SqlCommand();  cmd1.Connection = Program.cnx;  cmd1.CommandType = CommandType.Text;  SqlParameter p1 = new SqlParameter();  p1.ParameterName = "@code";  p1.DbType = DbType.Int16;  p1.SourceColumn = "ID\_Machine";  p1.SourceVersion = DataRowVersion.Current;  cmd1.Parameters.Add(p1);  SqlParameter p2 = new SqlParameter();  p2.ParameterName = "@Desc";  p2.DbType = DbType.String;  p2.Size = 50;  p2.SourceColumn = "Description\_Machine";  p2.SourceVersion = DataRowVersion.Current;  cmd1.Parameters.Add(p2);  cmd1.CommandText = "insert into Machine values (@code,@Desc)";  Program.da.InsertCommand = cmd1;  Program.da.Update(Program.ds.Tables[0].Select("","",DataViewRowState.Added)); |
| **+ Save BDD ( avec Builder)**  private void btnEnregistrer\_Click(object sender, EventArgs e)  SqlCommandBuilder cmb = new SqlCommandBuilder(Program.da);  // Case 1 :  Program.da.Update(Program.ds );  // Case 2 :  Program.da.Update(Program.dtBureau.Select("","",DataViewRowState.Added));  Program.da.Update(Program.dtBureau.Select("","",DataViewRowState.Deleted));  Program.da.Update(Program.dtBureau.Select("", "",DataViewRowState.ModifiedCurrent)); |

**+ Button << >> Premier - Dernier**

|  |
| --- |
| private void btnPremier\_Click(object sender,EventArgs e)  txt\_idAbonnement.Text=dt.Rows[0][0].ToString();  txt\_dateAb.Text = dt.Rows[0][1].ToString();  cmb\_idUsage.Text = dt.Rows[0][2].ToString();  cmb\_idAbonne.Text = dt.Rows[0][3].ToString(); |
| private void btnDernier\_Click(object sender, EventArgs e)  txt\_idAbonnement.Text = dt.Rows[dt.Rows.Count-1][0].ToString();  txt\_dateAb.Text = dt.Rows[dt.Rows.Count-1][1].ToString();  cmb\_idUsage.Text = dt.Rows[dt.Rows.Count-1][2].ToString();  cmb\_idAbonne.Text = dt.Rows[dt.Rows.Count-1][3].ToString(); |

**+ Button < > Suivant - Précedent**

|  |
| --- |
| private void btnSuivant\_Click(object sender, EventArgs e)  int pos = -1;  for (int i = 0; i < Program.dt.Rows.Count; i++)  {  if (txtIdHopital.Text == Program.dt.Rows[i][0].ToString() && i < Program.dt.Rows.Count - 1)  {  pos = i;  break;  }  if (i == Program.dt.Rows.Count - 1)  {  MessageBox.Show("vous étes sur le dernier enregistrement");  }  }  if(pos!=-1)  {  txtIdHopital.Text = Program.dt.Rows[pos + 1][0].ToString();  txtNom.Text = Program.dt.Rows[pos + 1][1].ToString();  txtAdresse.Text = Program.dt.Rows[pos + 1][2].ToString();  txtVille.Text = Program.dt.Rows[pos + 1][3].ToString();  } |
| private void btnPrecedent\_Click(object sender, EventArgs e)  for (int i = 0; i < dt.Rows.Count; i++)  {  if (txt\_idAbonnement.Text == dt.Rows[i][0].ToString() && i > 0)  {  txt\_idAbonnement.Text = dt.Rows[i - 1][0].ToString();  txt\_dateAb.Text = dt.Rows[i - 1][1].ToString();  cmb\_idUsage.Text = dt.Rows[i - 1][2].ToString();  cmb\_idAbonne.Text = dt.Rows[i - 1][3].ToString();  }  if (i == 0)  {  MessageBox.Show("vous étes sur le premier enregistrement") ;  }  } |

**+ Les Procédures Stocké**

|  |
| --- |
| // Requette Creation Procédure en SQL Create Procedure Pr1 @x char(5) , @y int output As SELECT @y = COUNT (OrderID)  FROM Orders WHERE CustomerID = @x   // Execution en SQL Declare @n int EXEC Pr1 ‘vinet’ , @n output  Print @n |
| // Code Creation Procédure en C#  cnx.Open();  cmd.Connection = cnx;  cmd.CommandType = CommandType.StoredProcedure;  cmd.CommandText = "Pr1";  SqlParameter p1 = new SqlParameter();  p1.SqlDbType = SqlDbType.Int;  p1.Direction = ParameterDirection.Input;  p1.ParameterName = "@x";  p1.Value = Convert.ToInt16(CustomerID.Text);  cmd.Parameters.Add(p1);  SqlParameter p2 = new SqlParameter();  p2.SqlDbType = SqlDbType.Int;  p2.Direction = ParameterDirection.output;  p1.ParameterName = "@y";  cmd.Parameters.Add(p2);  cmd.ExecuteScalar();  MessageBox.Show(Convert.ToString(p2.Value));  cnx.Close(); |

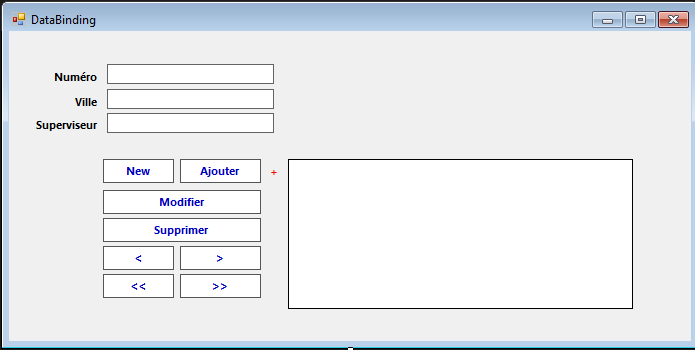
**+ Les Fonctions**

|  |
| --- |
| // Requette Fonction Procédure en SQL  Create Function F1 (@a int)  Returns int  AS  BEGIN  DECLARE @b int  SELECT @b = COUNT(ID\_Machine)  FROM MACHINE  WHERE ID\_Lot = @a    RETURN @b  END  GO  // Execution en SQL  SELECT F1(1) |
| // Code Création Fonction en C#  cnx.Open();  cmd.Connection = cnx;  cmd.CommandType = CommandType.StoredProcedure;  cmd.CommandText = "F1";  SqlParameter p1 = new SqlParameter();  p1.SqlDbType = SqlDbType.Int;  p1.Direction = ParameterDirection.Input;  p1.ParameterName = "@a";  p1.Value = Convert.ToInt16(txtIdlot.Text);  cmd.Parameters.Add(p1);  SqlParameter p2 = new SqlParameter();  p2.SqlDbType = SqlDbType.Int;  p2.Direction = ParameterDirection.ReturnValue;  cmd.Parameters.Add(p2);  cmd.ExecuteScalar();  MessageBox.Show(Convert.ToString(p2.Value));  cnx.Close(); |

**- Mise a jour avec - DATAVIEW**

|  |
| --- |
| PROGRAM.cs  public static DataView dv = new DataView(); |
| LOAD  Program.da = new SqlDataAdapter("select \* from Bureau", Program.cnx);  Program.da.MissingSchemaAction = MissingSchemaAction.AddWithKey;  Program.da.Fill(Program.dsBureaux); // Fill ds  Program.dtBureau = Program.dsBureaux.Tables[0];  Program.dv = Program.dtBureau.DefaultView; // Fill dv > avec DefaultView  DGNewBureaux.DataSource = Program.dv; // Show > en DGNewBureaux |
| MAJ  private void btnAjouter\_Click(object sender, EventArgs e)  { // MODE DV  DataRowView drv ; // DataRowView > pour contenir une ligne du DataView "dv"  drv = Program.dv.**AddNew**(); // DataRowView > ajouter la ligne "drv > a > "dv"  drv[0] = Convert.ToInt16(txtNumero.Text);  drv[1] = txtVille.Text;  drv[2] = cmbSup.Text;  }  private void btnModifier\_Click(object sender, EventArgs e)  { // MODE DV  Program.dv.**Sort** = "NumBur";  int i = Program.dv.**Find**(Convert.ToInt16(txtNumero.Text));  DataRowView drv = Program.dv[i];  drv.**BeginEdit**();  drv[0] = Convert.ToInt16(txtNumero.Text);  drv[1] = txtVille.Text;  drv[2] = cmbSup.Text;  drv.**EndEdit**();  }  private void btnSupprimer\_Click(object sender, EventArgs e)  { // MODE DV  Program.dv.**Sort** = "NumBur";  int i = Program.dv.**Find**(Convert.ToInt16(txtNumero.Text));  DataRowView drv = Program.dv[i];  drv.**Delete**();  } |
| // Affichage en Label depuis un « Combobox » event > SelectedIndexChanged if( cmbSup.Text != "")  {  Program.dt.**Clear**();  Program.da2 = new SqlDataAdapter("select NomSup from Superviseur where CINSup = ('"  + cmbSup.Text + "' ) ", Program.cnx);  Program.da2.MissingSchemaAction = MissingSchemaAction.Add;  Program.da2.Fill(Program.ds);  Program.dt = Program.ds.Tables[0];  lblNomSuperviseur.Text = Program.dt.Rows[0][0].ToString();  } |

**- Mise a jour avec – DATABINDING**

****

|  |
| --- |
| // Load :  Program.da3 = new SqlDataAdapter("select \* from Bureau", Program.cnx);  Program.da3.MissingSchemaAction = MissingSchemaAction.AddWithKey;  Program.da3.Fill(Program.ds2); // Fill ds  Program.dt2 = Program.ds2.Tables[0];  DG.DataSource = Program.dt2;  // LINK > with DataBindings  txtNumero.DataBindings.Add("Text", Program.dt2, "NumBur");  txtVille.DataBindings.Add("Text", Program.dt2, "VilleBur");  txtSuperviseur.DataBindings.Add("Text", Program.dt2, "CINSup"); |
| // MAJ:  **Nouveaux**  this.BindingContext[Program.dt2].AddNew();  **Ajouter**  this.BindingContext[Program.dt2].EndCurrentEdit();  **Suivant >**  this.BindingContext[Program.dt2].Position = this.BindingContext[Program.dt2].Position + 1;  **Précedent <**  this.BindingContext[Program.dt2].Position = this.BindingContext[Program.dt2].Position -1;  **Premier <<**  this.BindingContext[Program.dt2].Position = this.BindingContext[Program.dt2].Position = 0 ;  **Dernier >>**  this.BindingContext[Program.dt2].Position = this.BindingContext[Program.dt2].Count -1;  **Modifier**  this.BindingContext[Program.dt2].EndCurrentEdit();  **Supprimer**  this.BindingContext[Program.dt2].RemoveAt(this.BindingContext[Program.dt2].Position); |

**- la DataRelation**

|  |
| --- |
| // Create DataColumn > Child [1] + Parent [0]  DataColumn childcol = Program.dt.Columns[1];  DataColumn parentcol = Program.dt1.Columns[0];  // Create DataRelation > between Columns :> parentcol & childcol  DataRelation dre = new DataRelation('r1', parentcol, childcol);  Program.ds.Relations.Add(dre);  // Create DataView dv > Remplire par la table[0] : Client > + DefaultView  DataView dv = new DataView();  dv = Program.ds.Tables[0].DefaultView;  // Create DataRowView drv  DataRowView drv;  // Sort la colonne "idCLient"  dv.Sort = "idClient";  // Find la valeur de la selectedValue en COMBOBOX  int i = dv.Find(cmbClient.SelectedValue);  // la ligne DataRowView > N\* i >  drv = dv[i];  DataView dv1 = new DataView();  dv1 = drv.CreateChildView(dre);  // Afficher en DatagridView le DataView  DataGridView.DataSource = dv; |
| // Create DataColumn > Child [1] + Parent [0]  DataColumn childcol = Program.dt.Columns[1];  DataColumn parentcol = Program.dt1.Columns[0];  // Create DataRelation > between Columns :> parentcol & childcol  DataRelation dre = new DataRelation("R1", parentcol, childcol);  Program.ds.Relations.Add(dre);  DataTable dt3 = new DataTable();  dt3.Clone();  DataRow[] darc;  darc = Program.dt1.Rows.Find(cmbClient.SelectedValue).GetChildRows(dre);  darc.CopyToDataTable(dt3, LoadOption.OverwriteChanges); |

**- Transactions – Avec Procédures Stockés**

|  |
| --- |
| // View 1 :  dataGridView1.ColumnCount = 4;  dataGridView1.Columns[0].Name = "Order\_ID";  dataGridView1.Columns[1].Name = "Customer\_ID";  dataGridView1.Columns[2].Name = "EmployéID";  dataGridView1.Columns[3].Name = "OrderDate";  cnx.Open();  cmd.Connection = cnx;  cmd.CommandType = CommandType.Text;  cmd.CommandText = "P3";  SqlParameter pr = new SqlParameter();  pr.DbType = DbType.String;  pr.Size = 5;  pr.Direction = ParameterDirection.Input;  pr.ParameterName = "@c";  pr.Value = comboBox1.Text;  cmd.Parameters.Add(pr);  SqlDataReader dr;  dr = cmd.ExecuteReader();  // -------------------------- //  while (dr.Read())  {  dataGridView1.Rows.Add(dr[0].ToString(), dr[1].ToString(),  dr[2].ToString(), dr[3].ToString());  }  // View 2 :  dataGridView2.ColumnCount = 5;  dataGridView2.Columns[0].Name = "ProductID";  dataGridView2.Columns[1].Name = "UnitPrice";  dataGridView2.Columns[2].Name = "Quantity";  dataGridView2.Columns[3].Name = "OrderID";  dataGridView2.Columns[4].Name = "CustomerID";  dr.NextResult();  while (dr.Read())  {  dataGridView2.Rows.Add(dr[0].ToString(), dr[1].ToString(),  dr[2].ToString(), dr[3].ToString());  }    cnx.Close(); |

**- XML : Extensible Markup language**

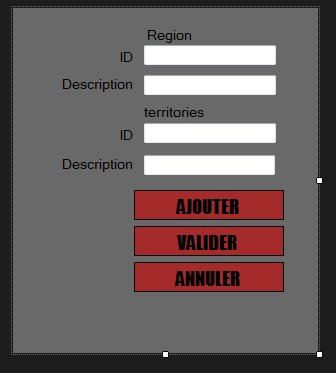
**+ XML C#**

|  |
| --- |
| public static SqlConnection cn = new SqlConnection("Data Source=(local);Initial Catalog=ComptoirAnglais;Integrated Security=True");  public static SqlCommand cmd = new SqlCommand();  public static SqlDataAdapter da = new SqlDataAdapter(cmd);  public static DataSet ds = new DataSet(); |
| private void **EcritureXML**\_Click(object sender, EventArgs e)  {  //ecriture ds--- xml  cmd.Connection = cn;  cmd.CommandType = CommandType.Text;  cmd.CommandText = "Select \* from Customers";  da.Fill(ds);  ds.WriteXml("C:\\Users\\Rlz\\Desktop\\xml\\tdi2020.xml");  }  private void **EcritureShemaXML**\_Click(object sender, EventArgs e)  {  //ecriture ds--- xsd  cmd.Connection = cn;  cmd.CommandType = CommandType.Text;  cmd.CommandText = "Select \* from Customers ";  da.Fill(ds);  ds.WriteXmlSchema("C:\\Users\\Rlz\\Desktop\\xml\\tdi.xsd");  }  private void **LectureXML**\_Click(object sender, EventArgs e)  {  //lecture xml--- ds  ds.ReadXml("C:\\Users\\Rlz\\Desktop\\xml\\tdi2020.xml");  dataGridView1.DataSource = ds.Tables[0];  }  private void **LectureShemaXML**\_Click(object sender, EventArgs e)  {  //lecture xsd--- ds  ds.ReadXml("C:\\Users\\Rlz\\Desktop\\xml\\tdi.xsd");  dataGridView1.DataSource = ds.Tables[0];  }  private void **RetourShemaXML\_EnString\_**Click(object sender, EventArgs e)  {  //lecture ds---chaine de caracteres(xml)  cmd.Connection = cn;  cmd.CommandType = CommandType.Text;  cmd.CommandText = "Select \* from Customers";  da.Fill(ds);  // to get XML  String x = ds.GetXml();  // to ge XML Shema  String y = ds.GetXmlSchema();  MessageBox.Show(x);  MessageBox.Show(y);  } |

|  |
| --- |
| **+ XML to HTML**  XslTransform xsltron = new XslTransform();  DataSet ds = new DataSet();  ds.ReadXml("C:\\Users\\Rlz\\Desktop\\xml\\Emp\\Employees1.xml");  XmlDataDocument xmldoc = new XmlDataDocument(ds);  xsltron.Load("C:\\Users\\Rlz\\Desktop\\xml\\Emp\\Employees1.xsl");  XmlTextWriter writer = new XmlTextWriter("C:\\Users\\Rlz\\Desktop\\xml\\tdi2016.html", System.Text.Encoding.UTF32);  xsltron.Transform(xmldoc, null, writer);  writer.Close(); |

|  |
| --- |
| **+ Write XML w/o Version**  **Ecriture XML avec Version** (serialiser) Program.ds.WriteXMl("C:\\Users\\Rlz\\Desktop\\xml\\tpdif.xml",XmlWriteMode.DiffGram);  **Ecriture XML sans Version**  Program.ds.WriteXMLShema("C:\\Users\\Rlz\\Desktop\\xml\\tpsans.xml");  **Diffgram : est une forme XML qui identifie la version actuelle et la version d’origine des éléments de données.** |

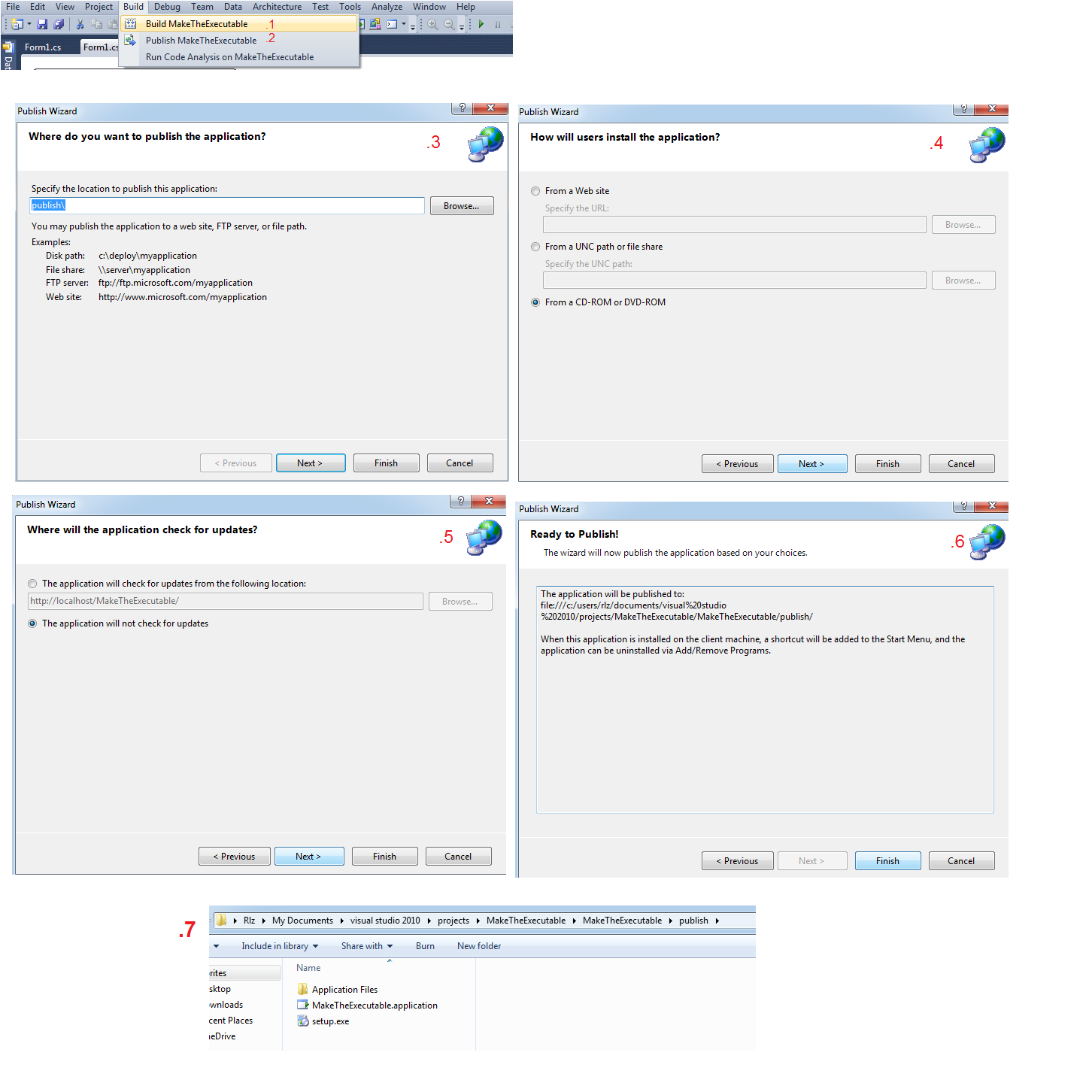
**- Les Transactions   
 + Commit - RollBack**

****

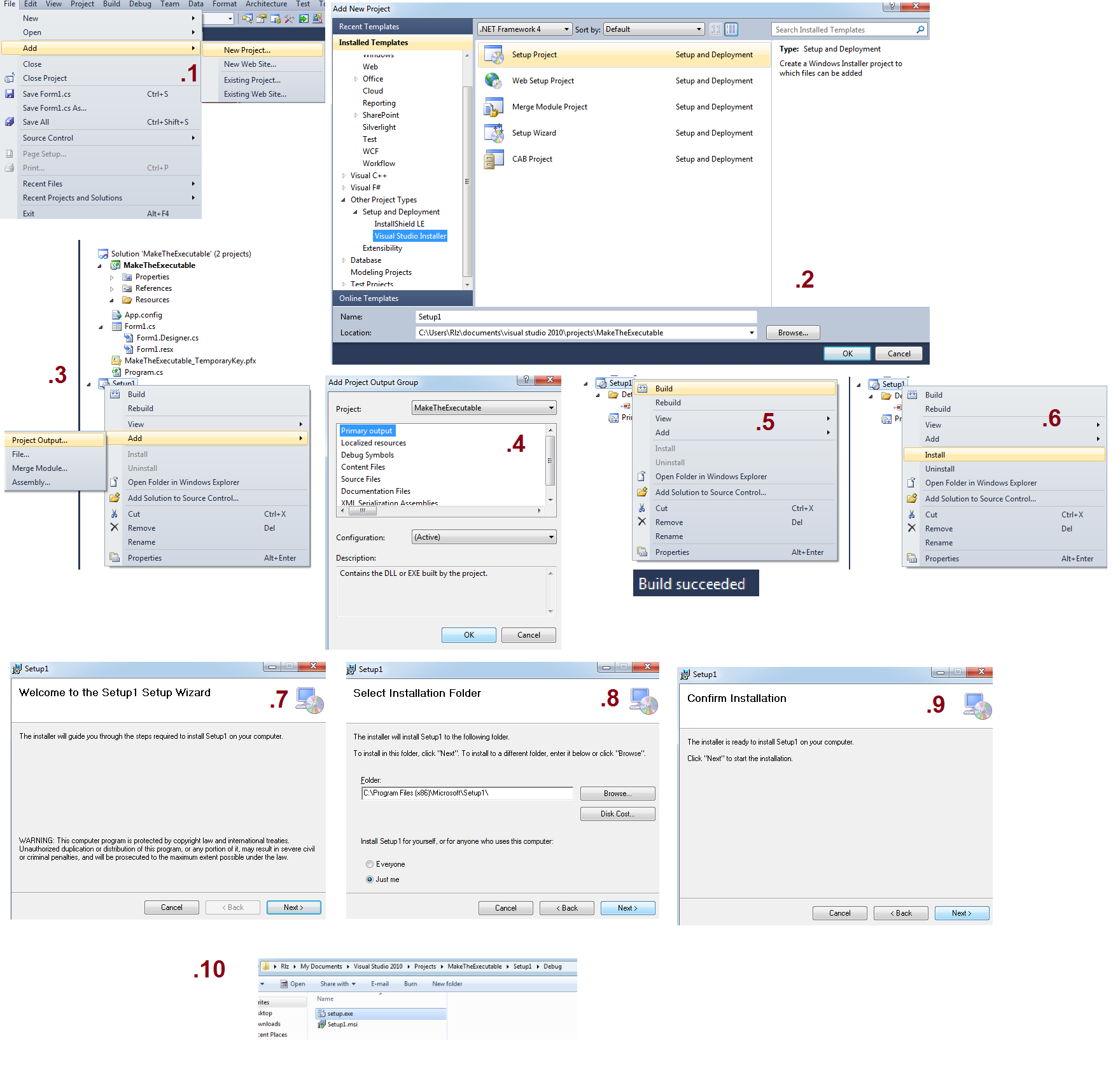
|  |
| --- |
| // DECLARE  SqlConnection cnx = new SqlConnection("Data Source=(local);Initial Catalog=comptoir\_anglais;Integrated Security=True");  SqlCommand cmd;  SqlCommand cmd2;  SqlTransaction tr;  private void btnajouterR\_Click(object sender, EventArgs e)  {  cnx.Open();  tr= cnx.BeginTransaction();  cmd = new SqlCommand("insert into region values(" +  Convert.ToInt16(txtid.Text) + ",'"  + txtdesc.Text + "')", cnx);  cmd2 = new SqlCommand("insert into territories values('"  + txttrID.Text + "','" +  txttrdesc.Text + "'," + Convert.ToInt16(txtid.Text) + ")", cnx);  cmd.Transaction = tr;  cmd2.Transaction = tr;  cmd.ExecuteNonQuery();  cmd2.ExecuteNonQuery();  }  private void btnValider\_Click(object sender, EventArgs e)  { tr.Commit(); cnx.Close(); }  private void btnAnnuler\_Click(object sender, EventArgs e)  { tr.Rollback(); cnx.Close(); } |

**Creation d’un Executable**

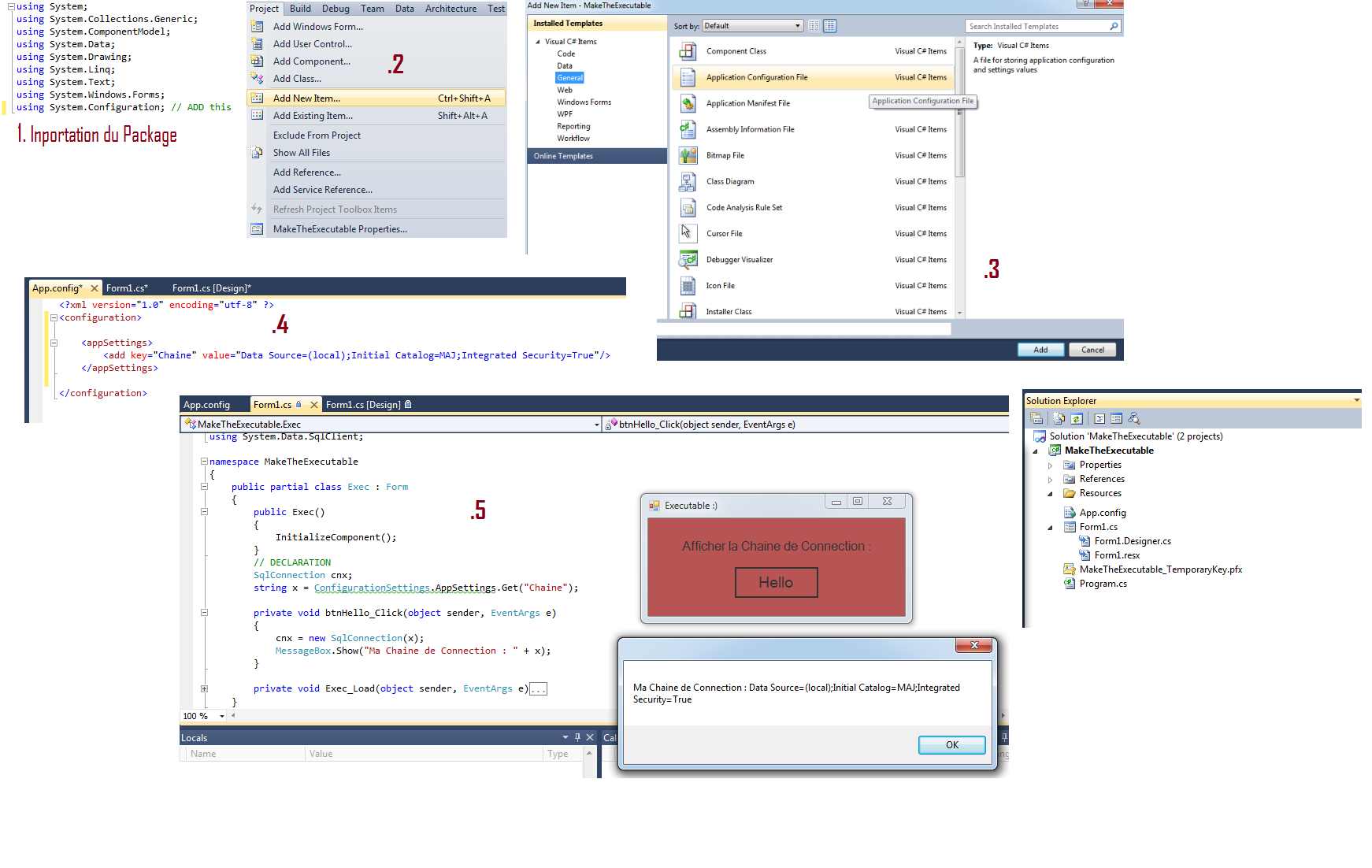
**+ Création du setup.exe**

****

**+ Création du setup.exe dans le projet**

****

**+ Gestion du Connexion string depuis App.Config**

****

**+Plus**

Tester l’entrée en TextBox :

|  |
| --- |
| try  {  int max = Convert.ToInt16(txtN.Text);  MessageBox.Show("Input Valid");  }  catch { MessageBox.Show("Input Invalid - Enter int not string", "ERROR"); } |

Utilisation DateTime :

|  |
| --- |
| spublic Form1()  {  InitializeComponent();  timer1.Start();  }  private void datetime\_tick ()  {  DateTime dtime = datetime.Now;  lblTime.Text = dtime.toString();  }  // TIMER btn:  counter = 10;  TimerBeforeShow = new Timer();  TimerBeforeShow.Tick += new EventHandler(TimerBeforeShow\_Tick);  TimerBeforeShow.Interval = 1000; // 1 second  TimerBeforeShow.Start();  lblShowT.Text = counter.ToString();  private void TimerBeforeShow\_Tick(object sender, EventArgs e)  {  counter--;  if (counter == 0)  {  TimerBeforeShow.Stop();  }  lblShowT.Text = counter.ToString();  } |

Crystal Repport Startup – AppConfig :

|  |
| --- |
| <startup useLegacyV2RuntimeActivationPolicy="true">  <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/>  </startup> |

Progress Bar :

|  |
| --- |
| progressBar1.Maximum = max;  for(int i=0; i<=max ;i++)  { progressBar1.Value = i; } |

Utilisations de Fichiers Text :

|  |
| --- |
| using System.IO;  // Charger Fichier  OpenFileDialog ofd = new OpenFileDialog();  ofd.ShowDialog();  txtFileName.Text = ofd.FileName;  // Lire Text  StreamReader sr = new StreamReader(txtFileName.Text);  RichTextBox1.Text = sr.ReadToEnd();  sr.Close();  // Ecrire Text  StreamWriter sw = new StreamWriter(txtFileName.Text,true);  sw.WriteLine(txtTextaEcrire.Text);  sw.Close(); |

Test if Null or Empty :

|  |
| --- |
| if (String.IsNullOrEmpty(txtNumProjet.Text)  { MessageBox.Show("Champ null ou vide "); |

Show / Hide Button to DGView

|  |
| --- |
| bool x = true;  ---  if (x == false)  {  dataGridView1.Hide();  x = true;  }  else if ( x == true)  { dataGridView1.Show(); x = false; } |

Fonction Load Mdi

|  |
| --- |
| public void LoadForm( Form Formtext)  var SaveFirstActiveForm = ActiveMdiChild;  // TEST > If SaveFirstActiveForm == this ActiveForm  if (SaveFirstActiveForm == Formtext)  {  // Rien.  }  else if (SaveFirstActiveForm != Board.ActiveMdiChild)  { // SAVE.  SaveFirstActiveForm = ActiveMdiChild;  // SHOW.  Formtext.MdiParent = this;  Formtext.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;  Formtext.Dock = DockStyle.Fill;  Formtext.Show();  // HIDE.  SaveFirstActiveForm.Hide();  } |

IMAGE PICTURE IMPORT TO DB:

|  |
| --- |
| // DECLARE Binary image  byte[] imageBt = null;  FileStream fstream = new FileStream(this.txtpicPath.Text, FileMode.Open, FileAccess.Read);  BinaryReader br = new BinaryReader(fstream);  imageBt = br.ReadBytes((int)fstream.Length);  // INSERT Commande  cnx.Open();    try  {  SqlCommand cmd;  string Query = "INSERT INTO USERS (ID,NOM,PRENOM,PICTURE) VALUES ( "  + txtID.Text + ",'" + txtNOM.Text + "','" + txtPRENOM.Text + "', @IMG)";  cmd = new SqlCommand( Query, cnx);  cmd.Parameters.Add(new SqlParameter("@IMG", imageBt));  cmd.ExecuteNonQuery();  MessageBox.Show("Saved");  }  catch (Exception ex)  {  MessageBox.Show("HELLO ERROR : " + ex);  }  cnx.Close();  // Import the Image  private void btnLoadImage\_Click(object sender, EventArgs e)  {  OpenFileDialog dlg = new OpenFileDialog();  dlg.Filter = "JPG Files(\*.jpg)|\*.jpg|PNG Files(\*.png)|\*.png|All Files(\*.\*)|\*.\*";  //dlg.Title = "SELECT USER PICTURE.";  if (dlg.ShowDialog() == DialogResult.OK)  {  picPath = dlg.FileName.ToString();  txtpicPath.Text = picPath;  pictureBox.ImageLocation = picPath;  }  } |