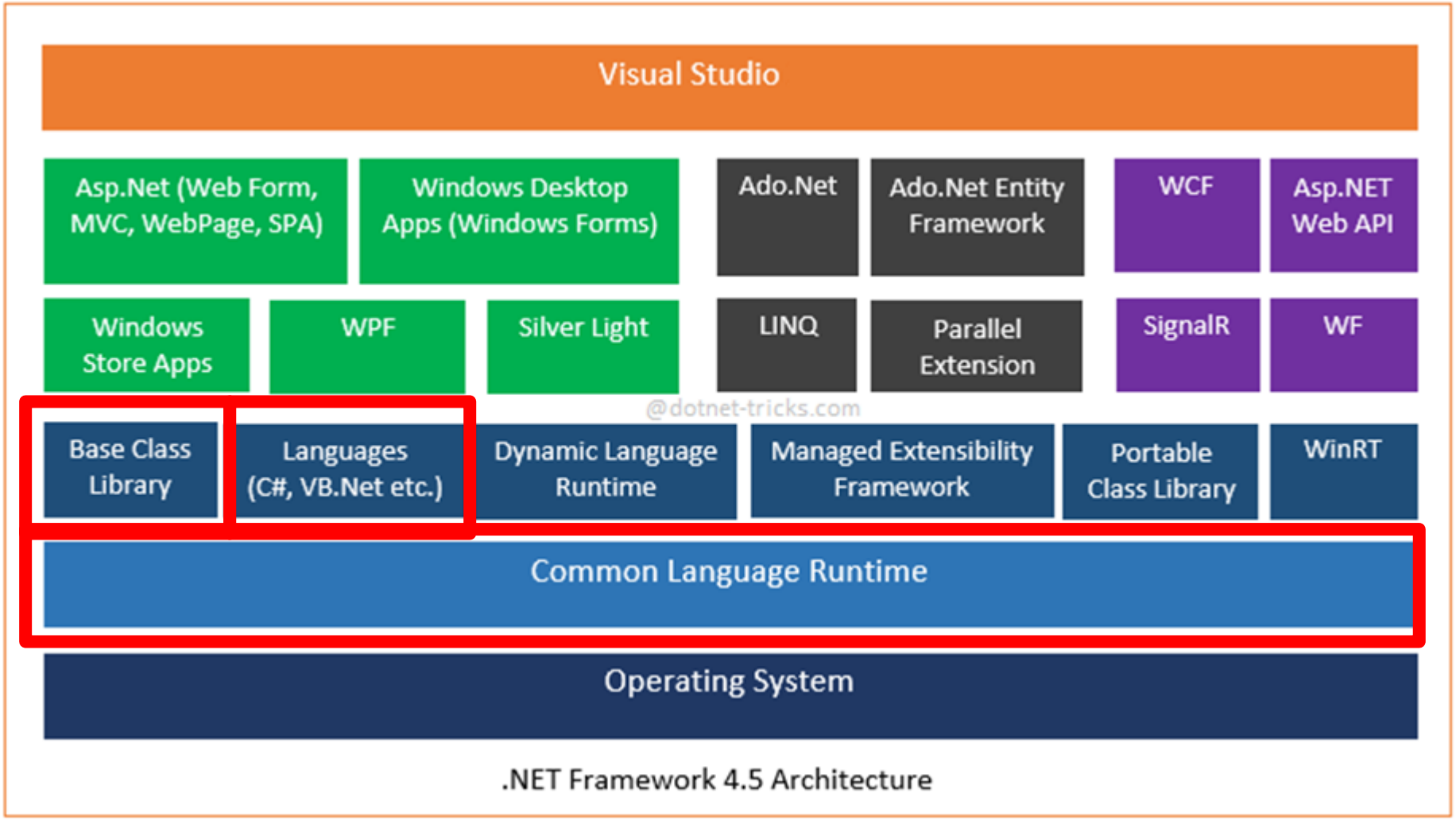
HEINSOHN
BUSINESS TECHNOLOGY

Acceso a Datos con ADO.NET

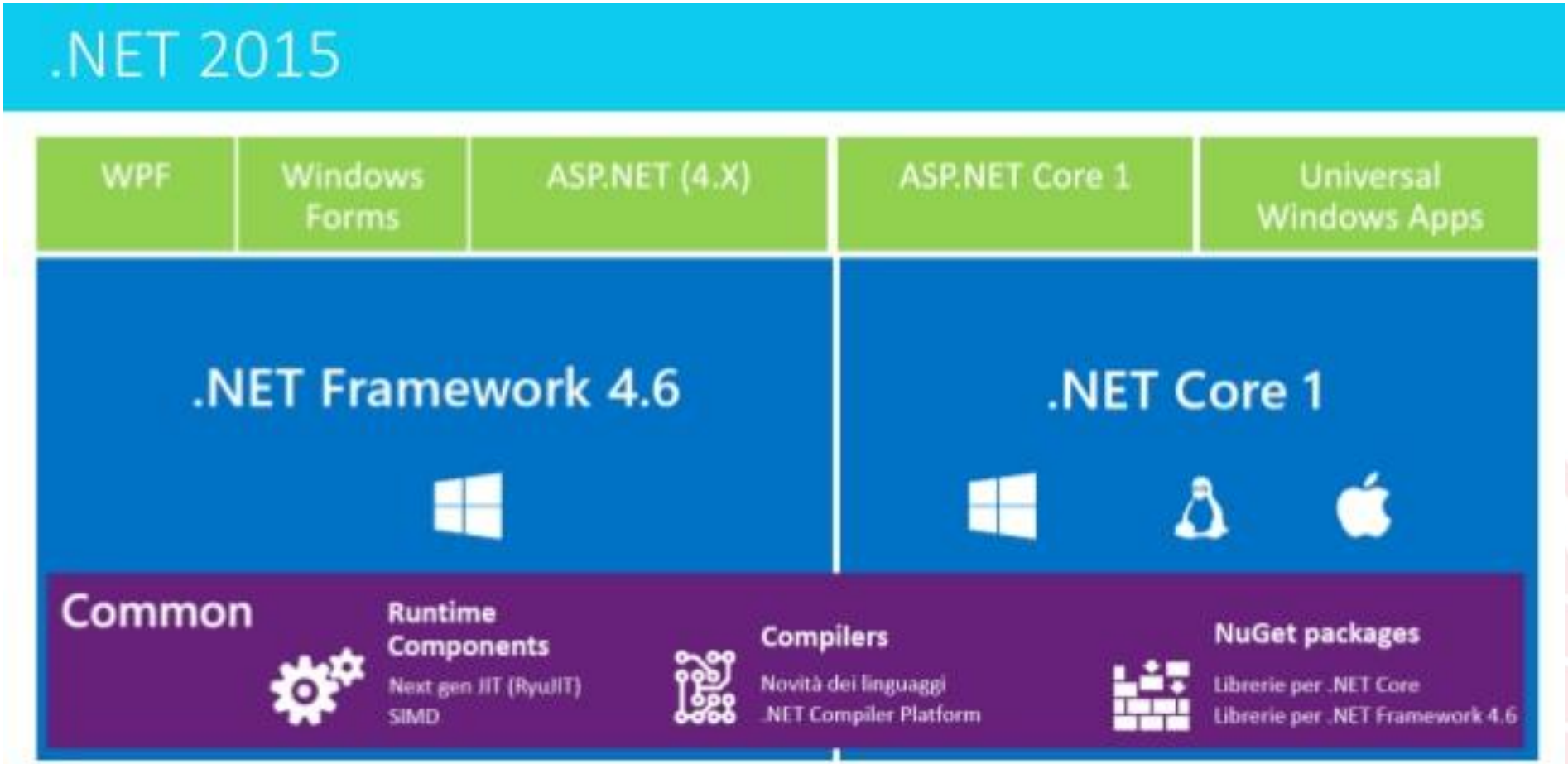
Microsoft Partner
Gold Software Development
Gold Web Development
Gold Software Asset Management

www.heinsohn.com.co

.Net Framework 4.5 Architecture



.Net Framework 4.6 & .Net Core 1



Repaso de POO

- Que es una clase
- Que es un objeto
- Conceptos Fundamentales:

  Herencia, Sobrecarga, Poliformismo, Abstracción, Encapusulamiento, Clases Abstractas, Interface

https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class

Revisando lo visto - RETO

- Cargar BD Northwind en SQL
- Realizar un diagrama de clases.

¿Que objetos vemos?

¿Cómo se relacionan?

https://www.microsoft.com/en-us/sql-server/sql-server-editions-express

Install SQL Server Express

http://www.howtosolutions.net/2013/07/solving-install-northwind-database-on-sql-server-problem/

Could not find stored procedure 'sp_dboption'

HEINSOHN
BUSINESS TECHNOLOGY

Agenda

- ADO.NET
  - Que es ADO.NET?
  - Usando namespaces
  - Que es un DataTable?
  - Que es un DataSet?
  - SQLConnection
  - TransactionScope

ADO.NET

ADO.NET Provee:

un conjunto de clases para trabajar con datos

ADO.NET es:

Una evolución más flexible de ADO y ADO.net 1

Un sistema diseñado para entornos desconectados

ADO.NET provee:

Un modelo de programación con soporte de XML

Un conjunto de clases, interfaces, estructuras, y numeraciones que manejan el acceso a datos dentro del .NET Framework

ADO.NET

Es: Una tecnología de acceso a datos que se basa en los objetos ADO (Objetos de Datos ActiveX).

Proporciona: Un conjunto variado de componentes utiliza un modelo de acceso pensado para entornos conectados o desconectados.

Utiliza: XML como el formato para transmitir datos desde y hacia la base de datos y la aplicación

Usando NameSpaces

Use la instrucción using(c#) o imports(vb.net) para importar namespaces:

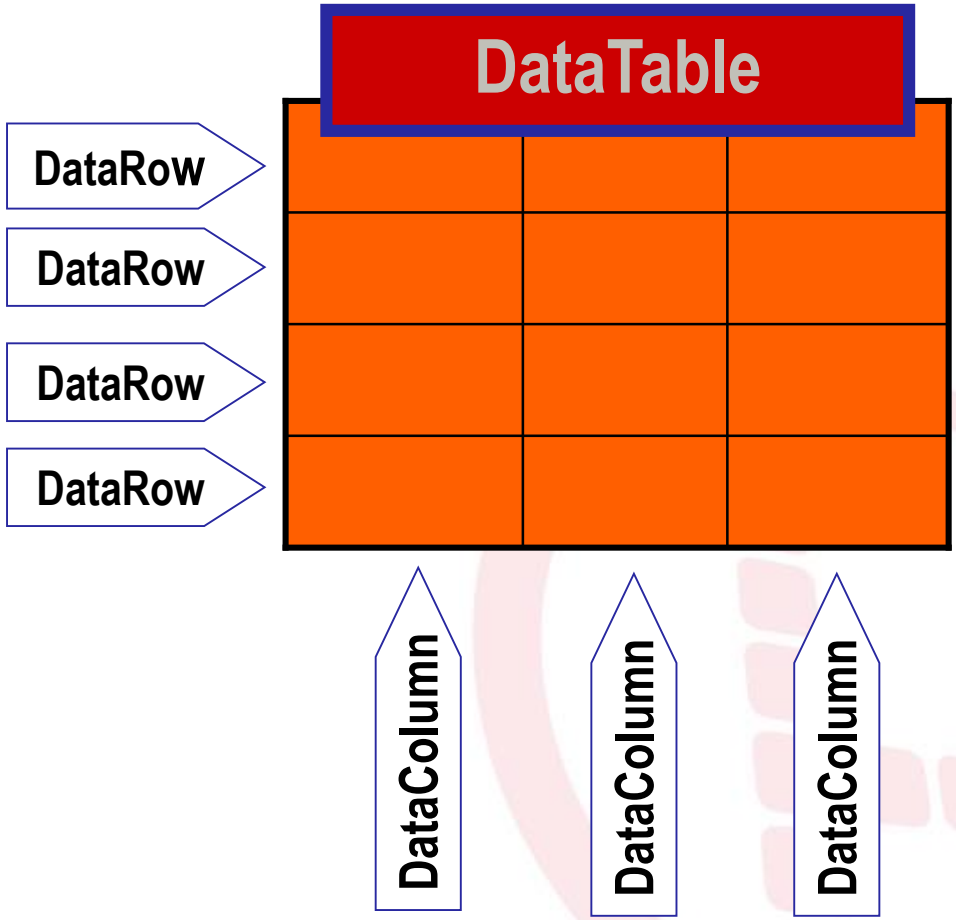
- System.Data
- System.Common
- System.SqlClient
- System.OleDB
- System.ODBC
- System.OracleClient

DataTable

- Es el objeto central de la biblioteca ADO.NET
- El esquema esta definido por DataColumnCollection.
- Mantiene la integridad de los datos por medio de Constraints
- Por medio de sus eventos podemos controlar los diferentes estados de los registros.
- Desconoce su origen de datos, por lo que funciona como una entidad independiente

Objeto DataTable

System.Data.DataTable
System.Data.DataRow
System.Data.DataColumn



Objeto DataTable

- DataTable es iXMLSerializable
- Método DataTable/DataSet.Load()
- Método Dataview.ToTable()
- Rowstate.SetAdded/SetModified
- API de Proveedor Independiente

Objeto DataTable – Algunos Miembros

.NewRow	Devuelve un objeto DataRow vacio con el esquema del DataTable
.ReadXMLSchema	Establece el Esquema del DataTable en base al contenido de un archivo XML
.ReadXML	Carga el contenido del DataTable en base a un archivo XML o Objetos Stream, Objets, etc
.Rows	Colección de Rows contenidos dentro del DataTable
.Select	Método del cual podemos por medio de expresiones realizar consultas sobre los datarows cargados.
.WriteXML	Escribe un archivo .xml con el contenido del DataTable

Objeto DataTable – Algunos Miembros

.WriteXMLSchema	Escribe en un archivo .xml con el esquema utilizado en el DataTable
.Columns	Colección de objetos DataColumn
.Add	Inserta un Objeto DataColumn o bíen indica el nombre y el tipo
.Remove	Elimina un objeto DataColumn del DataTable
.Load	Carga de datarows en base a un origen especificado (DataReader,…)
.Merge	Combina los rows entre múltiples DataTables

DataTable - Ejemplo (C#)

```
//Declaro el objeto DataTable y lo instancio
System.Data.DataTable tblTable = new System.Data.DataTable("MiTabla");

//Declaro un objeto DataColumn, le especicifico el nombre y el tipo de
//datos que almacenerá
System.Data.DataColumn colNombre =
    new System.Data.DataColumn("Nombre", typeof(System.Data.SqlTypes.SqlString));

//Adjunto a mi objeto Tabla el objeto Column que eh creao
tblTable.Columns.Add(colNombre);

//Declaro un objeto DataRow y le asigno un valor al campo Nombre
System.Data.DataRow rowData = tblTable.NewRow();
rowData["Nombre"] = "Jaime";

//Otra manera de asignar valores es indicando el indice del campo
System.Data.DataRow rowData2 = tblTable.NewRow();
rowData2[0] = "Christian";

//Adjunto los 2 Rows creados a la tabla
tblTable.Rows.Add(rowData2);
tblTable.Rows.Add(rowData);
```

DataSet

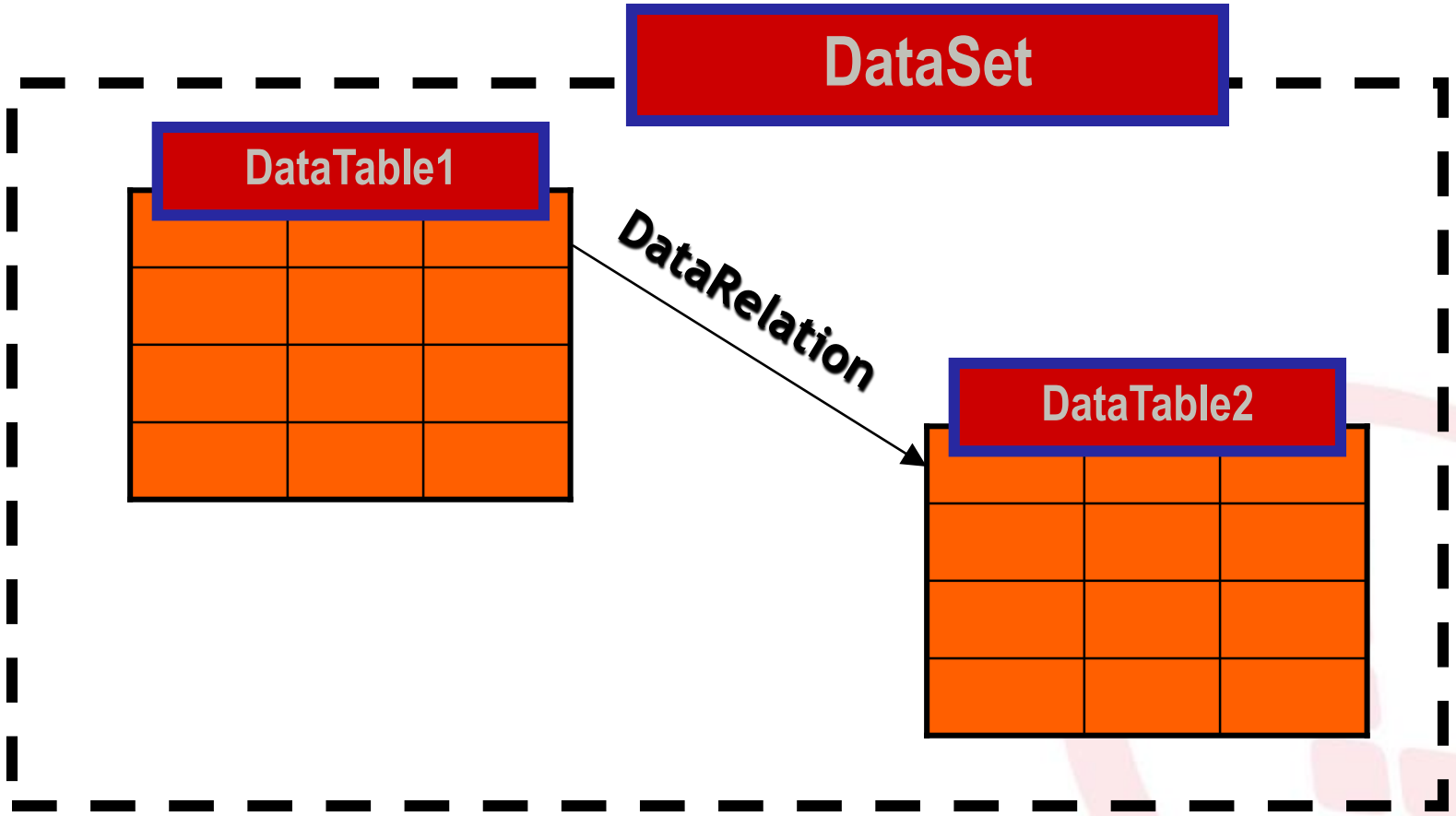
- Representación de datos en memoria
- Consiste en una Colección de objetos DataTables
- Mantiene la integridad entre los DataTables por medio del objeto DataRelation
- Desconoce el origen de los datos

DataSet

- DataSet/DataTable.Load
- Cargar un DataTable/DataSet desde un DataReader
  - Cargue desde dbDataReaders
    - OLEleDbDataReader
    - SqlDataReader
    - DataTableReader*
    - Etc.
- Permite un control más especifico de los datos

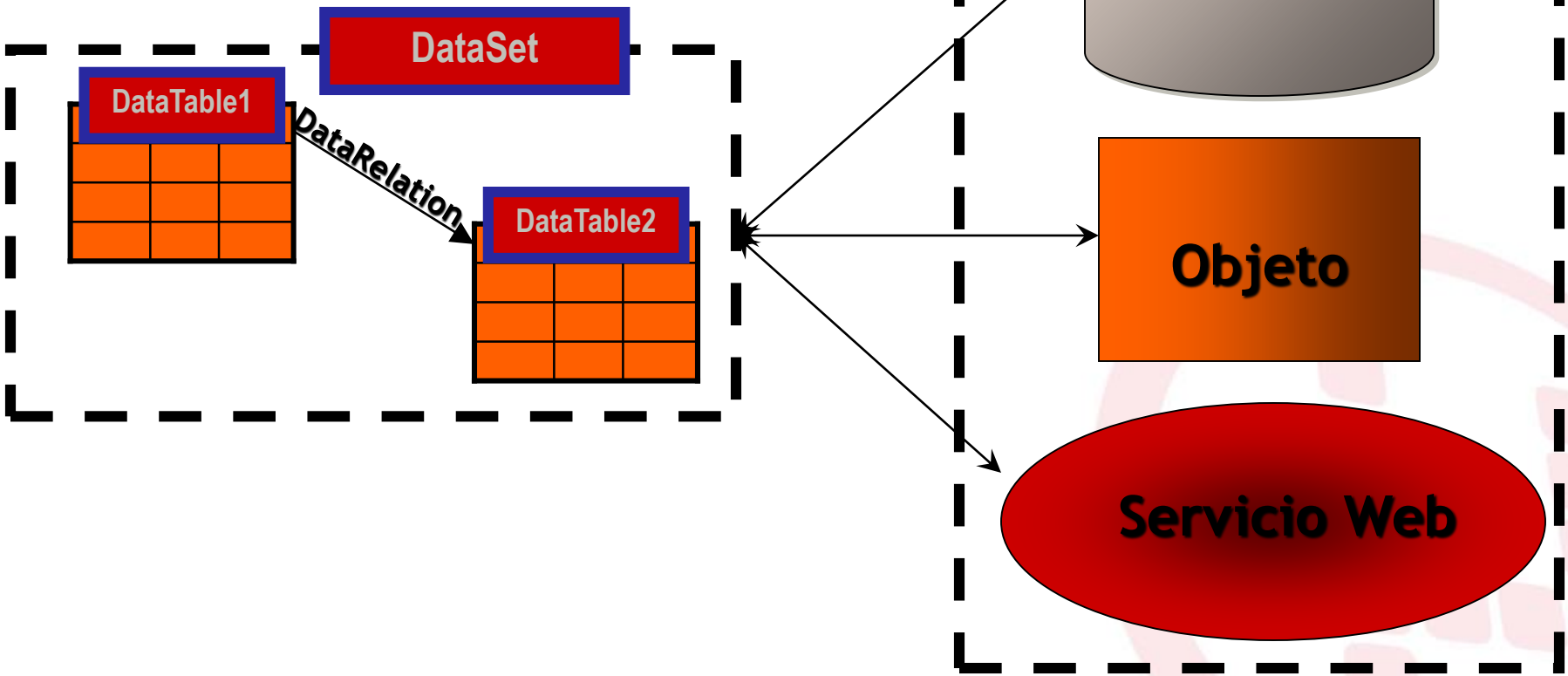
HEINSOHN
BUSINESS TECHNOLOGY

DataSet

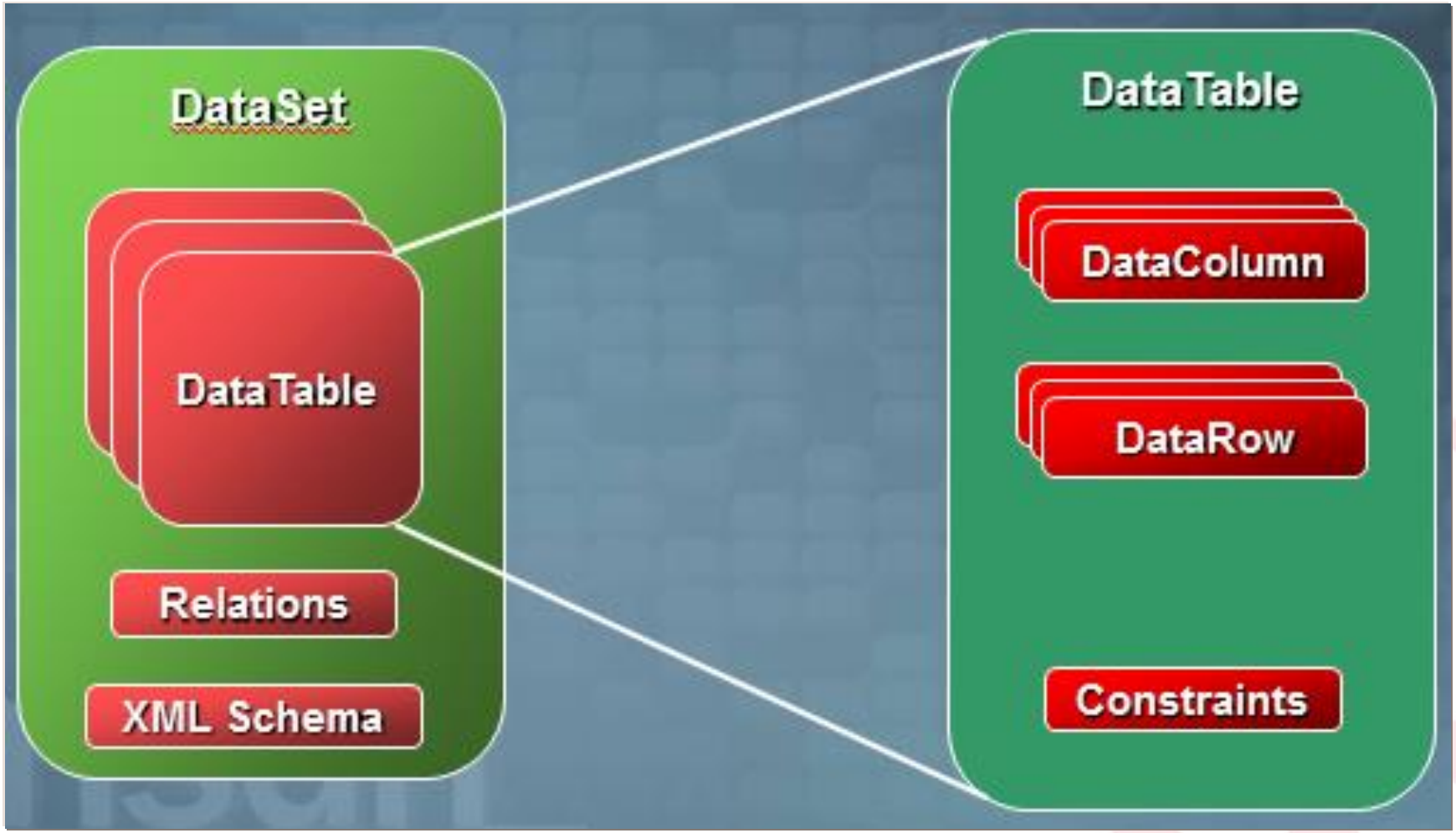


HEINSOHN
BUSINESS TECHNOLOGY

DataSet



ADO.NET


DataSet
DataTable
Relations
XML Schema
DataTable
DataColumn
DataRow
Constraints

DataSet - Ejemplo (C#)

```
//Creo 2 tablas
DataTable tblAlumnos = new DataTable("Alumnos");
tblAlumnos.Columns.Add("IDAlumno", typeof(int));
tblAlumnos.Columns.Add("Nombre", typeof(string));

DataTable tblCursos = new DataTable("Alumno_Cursos");
tblCursos.Columns.Add("IDAlumno", typeof(int));
tblCursos.Columns.Add("Curso", typeof(string));

//Creo un DataSet que contendrá las dos tablas
DataSet dsUniversidad = new DataSet("UniversidadAcme");
dsUniversidad.Tables.Add(tblAlumnos);
dsUniversidad.Tables.Add(tblCursos);

//Creo la relacion entre las tablas
DataRelation drRelacion =
    new DataRelation("AlumnCourse",
                     tblAlumnos.Columns["IDAlumno"],
                     tblCursos.Columns["IDAlumno"]);
dsUniversidad.Relations.Add(drRelacion);
```

DataSet - Ejemplo

```
/*Por medio del método Select me traigo una colección de
 * DataRows del cual solo utilizo uno solo (precisamente porque
 * se que IDAlumno es un valor único)
*/
DataRow rowAlumno = tblAlumnos.Select("IDAlumno = 1")[0];
Console.Write(rowAlumno["Nombre"]);

/*
 * Me traigo la lista de cursos del alumno seleccionado
 * y recorro la colección de objetos DataRows obtenidos para
 * representarlos en pantalla mostrando el curso al cual se
 * encuentra
 */
DataRow[] rowCursos = rowAlumno.GetChildRows("AlumnoCursos");
foreach (DataRow rowCurso in rowCursos)
{
    Console.Write(rowCurso["Curso"]);
}
```

Objeto Connection

- Representa una conexión al Data Source
- En una conexión, puedes …
  - Personalizar la conexión a la base de datos
  - Begin, commit, y abortar transacciones

Objeto Connection

System.Data.SqlClient.**SqlConnection**
System.Data.ODBC.**ODBCConnection**
System.Data.OleDBConnection.**OLEDBConnection**
System.Data.OracleClient.**OracleConnection**

Objeto Connection

Clases xxxConnection heredan de System.Data.Common.DbConnection

Propiedades:

- ConnectionString: Cadena de conexión

Métodos:

- Open: Abre la conexión con el origen especificado
- Close: Cierra la conexión
- BeginTransaction: Inicia una transacción con el origen

Objeto Connection - Ejemplo

```
System.Data.SqlClient.SqlConnection myconn =
    new System.Data.SqlClient.SqlConnection();
myconn.ConnectionString =
    "Data Source=MiPC;Initial Catalog=Northwind;User ID=sa;Password=123";
myconn.Open();
//
//Realizo las operaciones necesarias
//
myconn.Close();
//Por último me DESCONECTO
```

Objeto Command

- Representa una Instrucción SQL o un procedimiento almacenado que ejecuta en un origen de datos
- Expone 4 métodos importantes para devolver datos:
  - ExecuteReader()
  - ExecuteScalar()
  - ExecuteNonQuery()
  - ExecuteXMLReader()
- Llamada a StoresProcedures utilizando Parameters
- Objeto Command especifico para cada proveedor:
  - SQLCommand
  - ODBCCommand
  - OLEDBCommand
  - OracleCommand

Objeto Command - Ejemplo 1 de SQLCommand

```
//Creo un objeto SQLCommand
System.Data.SqlClient.SqlCommand myComand = new SqlCommand();
//Indico la conección a utilizar
myComand.Connection = myconn;
//Escribo la consulta T-SQL que me devolverá la cantidad de
//regitros en la tabla Customers
myComand.CommandText = "SELECT Count(*) FROM Customers";
myconn.Open();
//El valor devuelto lo guardo en una variable tipo Entero
int CantFilas = Convert.ToInt32(myComand.ExecuteScalar());
//Cierro la Conecció
myconn.Close();
Console.WriteLine("Cantidad de Registros: " + CantFilas.ToString());
```

Objeto Command - Ejemplo 2 de SQLCommand

```
//Creo un objeto SQLCommand
System.Data.SqlClient.SqlCommand myUpdate = new SqlCommand();
//Indico la conección a utilizar
myUpdate.Connection = myconn;
//Indico el tipo de llamada que realizaré (por default es Text)
myUpdate.CommandType = CommandType.StoredProcedure;
//Agrego los parámetros que necesita el StoreProcedure
myUpdate.Parameters.Add("@ID", SqlDbType.Int).Value = "1";
myUpdate.Parameters.Add("@Nombre", SqlDbType.Text).Value = "Juan";
//Indico el nombre del StoreProcedure
myUpdate.CommandText = "UpdateAlumnos";
//Abro la conección
myconn.Open();
//Ejecuto la operación
myUpdate.ExecuteNonQuery();
//Cierro la conección
myconn.Close();
```

Objeto DataReader

- Forward-only / Read-only
- Acceso rápido a los datos
- Conectado al origen
- La conexión la maneja usted mismo
- Los datos se manejan por código o a través de controles enlazados
- Usa pocos recursos

Objeto DataReader - Ejemplo 1

```
//Declaro e instancio un objeto SqlCommand
System.Data.SqlClient.SqlCommand comand =
    new System.Data.SqlClient.SqlCommand();
//Indico el objeto Conextion que utilizará
comand.Connection = myconn;
//Indico la cadena TSQL utilizando la propiedad CommandText
comand.CommandText = "SELECT * FROM Customers";
//Abro la conección
myconn.Open();
//Declaro un objeto SqlDataReader e invoco el método
//ExecuteReader del objeto SqlCommand
System.Data.SqlClient.SqlDataReader dr = comand.ExecuteReader();
//Recorro la los registros moviendo el cursor a la siguiente posición
while (dr.Read())
{
    Console.WriteLine(dr["CustomerName"].ToString());
}
//Cierro la coneccion
myconn.Close();
```

Objeto DataReader - Ejemplo 2

```
//Creo un objeto DataTable Tipado
DataSet1.tblAlumno_CursosDataTable tblAlumnos =
    new DataSet1.tblAlumno_CursosDataTable();

//Cargo el DataTable con el contenido obtenido
//de la base de datos
myconn.Open();
tblAlumnos.Load(dr);
myconn.Close();

//Creo otro objeto DataReader especifico para recorrer el contenido
//de un DataTable
System.Data.DataTableReader drAlumno = tblAlumnos.CreateDataReader();
while (drAlumno.Read())
{
    //.....
    Console.WriteLine(drAlumno["CustomerName"].ToString());
    //....
}
```

HEINSOHN
BUSINESS TECHNOLOGY

Transacciones

- Agrupas operaciones combinadas en una unidad lógica de trabajo
- Controla y conserva la coherencia e integridad de todas las acciones
- Permite contener transacciones locales o distribuidas
- Encerrar en funcion Using()
- Agregar Referencia: System.Transaction
- NameSpace: System.Transaction

HEINSOHN
BUSINESS TECHNOLOGY

Ejemplo

Atomic **C**onsistent **I**solated **D**urable

http://www.codeproject.com/Articles/522039/A-Beginners-Tutorial-for-Understanding-Transaction

DEMO

- Obtenga todos los clientes con una consulta y luego muéstrelos en pantalla.

HEINSOHN
BUSINESS TECHNOLOGY

Laboratorio

- Objetos SQLConnetion & SQLDataReader

HEINSOHN
BUSINESS TECHNOLOGY