REVISTA DIGITAL DE LAS TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES
TELEM@TICA
Revista Telem@tica. Vol. 11. No. 2, mayo-agosto, 2012, p. 1-9 ISSN 1729-3804

ACCESO A DATOS CON LINQ TO SQL

Jimy Sánchez Mena

División Territorial de Ciudad de la Habana, DESOFT, Calle 24 # 408 e/ 23 y 25, Vedado, Plaza, La Habana.
[1]e-mail: jimy.sanchez@hab.desoft.cu

RESUMEN

Linq to SQL es una tecnología de Linq que le permite a los desarrolladores de software de gestión la posibilidad de abstraerse de la capa de acceso a datos de sus aplicaciones, así como definir consultas a la base de datos en lenguaje Linq. En el presente artículo se explicarán y ejemplificarán los pasos para la utilización de esta tecnología así como una escueta descripción de la arquitectura de la misma.

PALABRAS CLAVES: Linq, Linq to SQL, ORM, framework 3.5, Transfer Object.

ABSTRACT

Linq to SQL is a technology of Linq that allows to the developers of management software the possibility to absorb from the data access layer of its applications, as well as to define consultations to the database in language Linq. Presently article will be explained and they will exemplify the steps for the use of this technology as well as a concise description of the architecture of the same one.

KEY WORDS: Linq, Linq to SQL, ORM, framework 3.5, Transfer Object.

1. INTRODUCCIÓN

Linq to SQL es un ORM (Object Relational Mapping, Mapeador de Objetos Relacionales) que incorpora el .NET framework 3.5, el cual es una componente de Linq [1] (Language Integrated Query / Lenguaje Integrado de Consultas) dirigida al tratamiento de bases de datos relacionales creadas en MSSQL 2000 o superior. El principio del mismo consiste en mapear el esquema de una base de datos [3] en clases, métodos y propiedades, tratando las tablas como colecciones de datos y las columnas como campos de las clases generadas, permitiendo definir consultas sobre los objetos mapeados en lenguaje Linq, brindando esto la ventaja de que el código de la consulta ya esté pre compilado y que el mismo forme parte del código de la aplicación [1].

Esta tecnología le brinda a los desarrolladores de software de gestión la posibilidad de abstraerse de la capa de acceso a datos de sus aplicaciones, ya que al ser mapeado el esquema de la base de datos [3] como se explicó anteriormente, no es necesario crear los Dao (Data Access Object / Objeto de Acceso a Datos) por parte de los desarrolladores, ni enfrentarse a las declaraciones Sql para consultar las tablas de la base de datos, por otra parte, permite abstraerse de la creación de los Transfer Object (Objetos de transferencia), que viajan entre capas con la información a tratar, ya que estos son generados en el mapeo.

Actualmente Microsoft ya no está desarrollando Linq to Sql, pues se ha dedicado al desarrollo del ORM Entity Framework, que permite una amplia definición de modelos de dominio de objetos y sus relaciones con muchos proveedores diferentes de datos, pero Linq to Sql además de ser un desarrollo más sencillo, es una gran API para la construcción rápida de acceso a datos bien diseñada para bases de datos en SQL Server, proporcionando en si mejoras en cuanto a rendimiento.

DESARROLLO

A continuación se muestra, paso a paso, lo antes expuesto en la introducción, para lo cual se requiere de Microsoft Visual Studio 2008 (VS 2008) o superior y MSSQL 2000 o superior.

En el siguiente ejemplo se empleó VS 2008 y MSSQL 2000, la base de datos se llama “EjLinqToSql” y está formada por las tablas “tbEquipos” y “tbJugadores” como se muestra en la Figura 1.

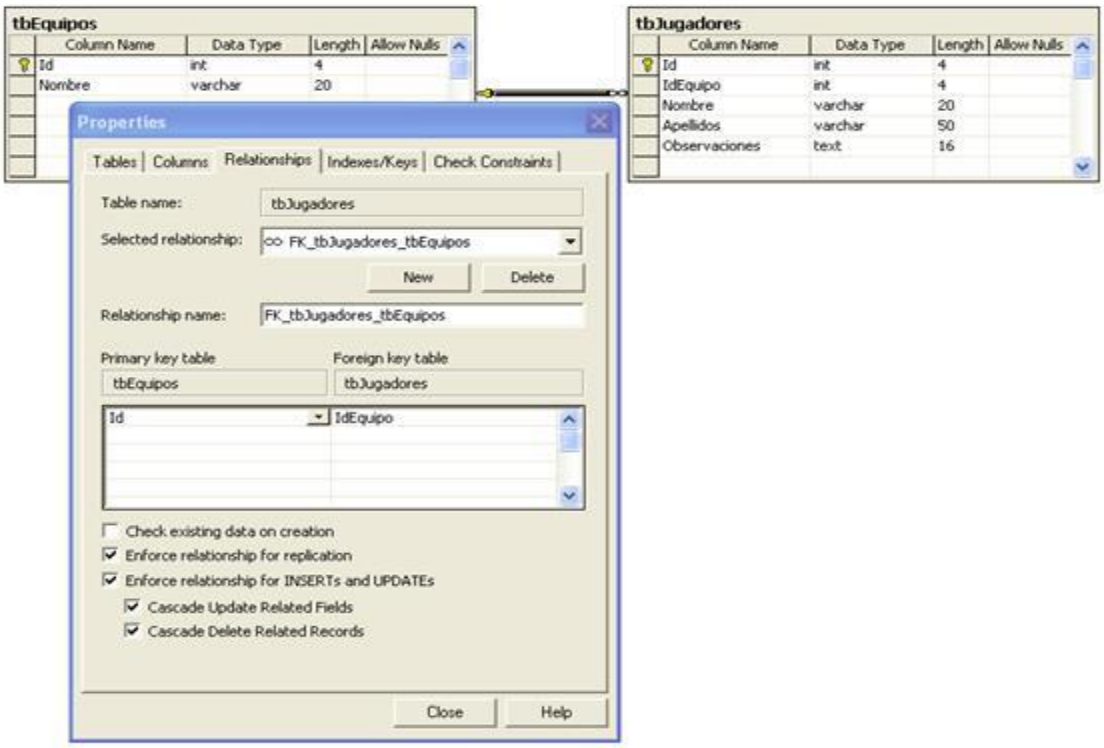

Figura 1: Relación entre tablas involucradas en el ejemplo.

El campo Id de ambas tablas es de tipo autonumérico.
Se crea un proyecto en VS 2008 como se muestra en la Figura 2.

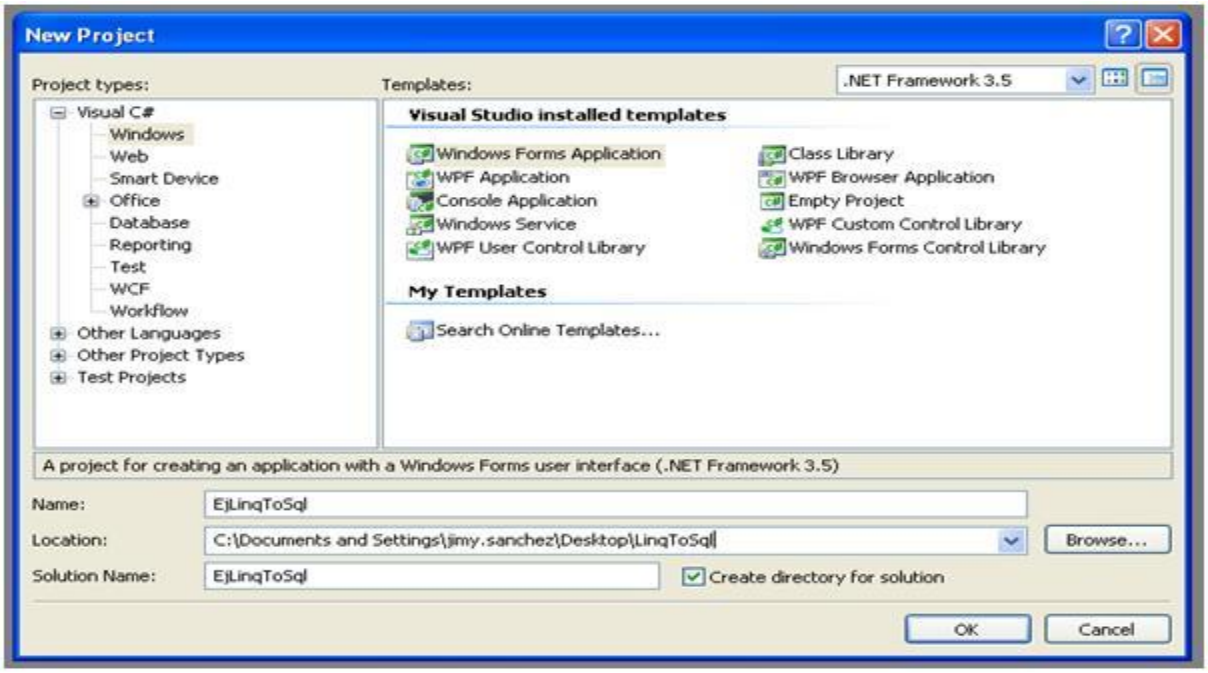

Figura 2: Creación de un nuevo proyecto en VS 2008

Una vez creado el proyecto se le incorpora un nuevo elemento tal como se muestra en la Figura 3, en este caso sería el diseñador de LINQ to SQL.

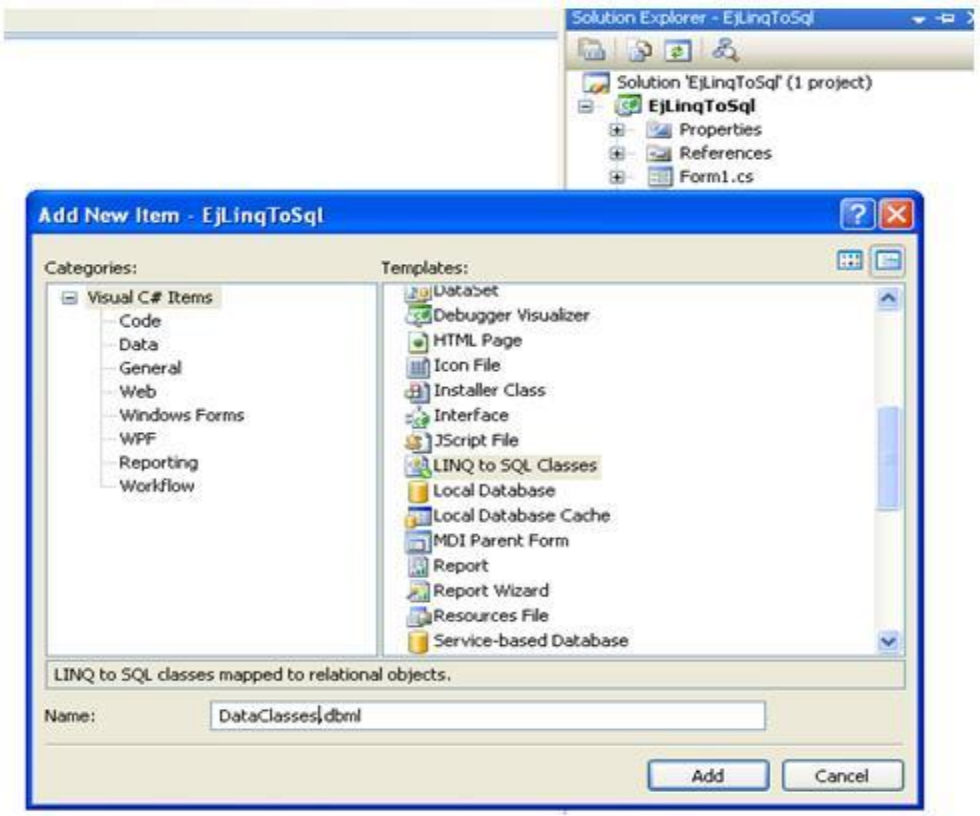

Figura 3: Incorporación del diseñador de LINQ to SQL al proyecto.

Ya conectados al origen de datos desde el explorador de servidores que trae integrado VS 2008, hacia el diseñador de LINQ to SQL, se arrastran las tablas de la base de datos, quedando como se muestra la Figura 4.

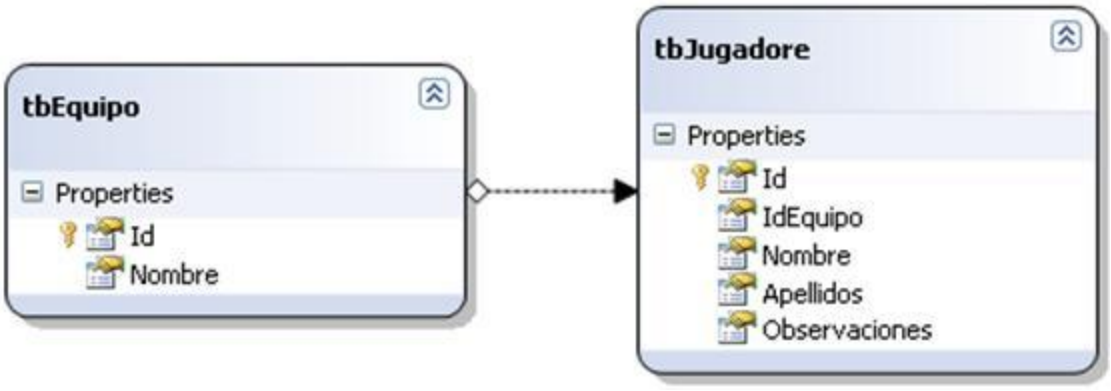

Figura 4: Incorporación de las tablas de la base de datos al diseñador de LINQ to SQL del proyecto.

Seguidamente pulsado el botón “Save” del Ide, se generarán las clases que representan a cada una de las entidades y relaciones del modelo de datos definido, así como la clase DataContext, mediante la cual podremos consultar las entidades, ya que ésta contendrá propiedades que representan cada tabla modelada, así como métodos para cada procedimiento de almacenado que se le añada.

A partir de este momento solo es necesario enfocarse en la capa de lógica de negocio, basándose principalmente en los métodos extensores de C# 3.0 que tributan a Linq [1] y en expresiones lambda.

Los métodos extensores son métodos estáticos definidos dentro de la clase estática Queryable y estos contienen estructuras que implementan la Interfaz genérica IEnumerable <> disponible en System.Collections.Generic [2], facilitando de esta manera el intercambio con colecciones de datos.
Las expresiones lambda son la simplificación de delegados anónimos [1] y estas se aplican en los métodos extensores en el momento de filtrar información.

A continuación se muestra un ejemplo de cómo realizar las consultas basadas básicamente en los elementos antes mencionados, creando para ello una clase llamada CtrlEquipos.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace EjLinqToSql
{
 class CtrlEquipos
  {
 public CtrlEquipos()
    {
     }
public int AddEquipo(tbEquipo nuevoEquipo)
  {
DataClassesDataContext datCtext = new DataClassesDataContext();
if (datCtext.tbEquipos.FirstOrDefault(e => e.Nombre == nuevoEquipo.Nombre) == null)
  {
   datCtext.tbEquipos.InsertOnSubmit(nuevoEquipo);
   datCtext.SubmitChanges();
   return nuevoEquipo.Id;
   }
return -1; }
}
public bool ModificarEquipo(int idEquipo, tbEquipo nuevoEquipo)
   {
    DataClassesDataContext datCtext = new DataClassesDataContext();
    tbEquipo equipo = datCtext.tbEquipos.FirstOrDefault(e => e.Id == idEquipo);
    if (equipo == null) return false;
    equipo.Nombre = nuevoEquipo.Nombre;
   datCtext.SubmitChanges();
    return true;
}
```

```
public bool EliminarEquipo(int idEquipo)
  {
  DataClassesDataContext datCtext = new DataClassesDataContext();
  tbEquipo equipo = datCtext.tbEquipos.FirstOrDefault(e => e.Id == idEquipo);
  if (equipo == null) return false;
  datCtext.tbEquipos.DeleteOnSubmit(equipo);
  datCtext.SubmitChanges();
  return true;
  return true;
   }
public List<tbEquipo> SeleccionarEquipos()
   {
DataClassesDataContext datCtext = new DataClassesDataContext();
return datCtext.tbEquipos.ToList();
  }
public tbEquipo SeleccionarEquipo(int idEquipo
  {
   DataClassesDataContext datCtext = new DataClassesDataContext();
   tbEquipo equipo = datCtext.tbEquipos.FirstOrDefault(e => e.Id == idEquipo);
   return equipo;
  }
public List<tbJugadore> SeleccionarJugadoresPorEquipo(int idEquipo)
  {
   DataClassesDataContext datCtext = new DataClassesDataContext();
   tbEquipo equipo = datCtext.tbEquipos.FirstOrDefault(e => e.Id == idEquipo);
   return equipo != null ? equipo.tbJugadores.ToList() : null;
   }
 }
}
```

En el método public int AddEquipo(tbEquipo nuevoEquipo) lo primero es verificar que no exista ningún equipo con el mismo nombre que el que se va a registrar, para ello la idea es instanciar la clase DataContext, pidiéndole luego a la instancia la colección de equipos, devolviendo los que están registrados como un IEnumerable<>, luego se llama al extensor FirstOrDefault a fin de seleccionar si existe algún equipo que su nombre coincida con el del equipo a registrar, basándose para ello en expresiones lambda, quedando de la siguiente manera: datCtext.tbEquipos.FirstOrDefault(e => e.Nombre == nuevoEquipo.Nombre), en caso de que el resultado no sea nulo, se retorna -1 indicando de esta manera que ya hay registrado un equipo con ese nombre y en caso contrario se llama el extensor InsertOnSubmit para registrar el equipo, retornando finalmente el Id asignado al mismo.

Como se puede apreciar, en el resto de los métodos igualmente se usan los extensores y expresiones lambda, siempre y cuando se esté frente a una colección que implementa IEnumerable<> [3], pero acotado a la finalidad del método.

Los casos de los métodos public List<tbEquipo> SeleccionarEquipos() y public List<tbJugadore> SeleccionarJugadoresPorEquipo(int idEquipo), devuelven la lista de equipos y de jugadores por equipo respectivamente, además de dar la posibilidad de ser utilizados como ObjectDataSource para otros objetos que requieran de los mismos [4].

Como se puede observar en los métodos public int AddEquipo(tbEquipo nuevoEquipo), public bool ModificarEquipo(int idEquipo, tbEquipo nuevoEquipo) y public bool EliminarEquipo(int idEquipo), después de aplicársele la lógica correspondiente a cada uno, se llama al método datCtext.SubmitChanges(), el cual se encuentra en la clase DataContext, y el mismo se encarga de calcular el conjunto de objetos modificados que se va a insertar, actualizar o eliminar y ejecuta los comandos adecuados para implementar los cambios en la base de datos.

CONCLUSIONES

1. Es muy fácil desarrollar aplicaciones empleando esta tecnología, garantizando de esta manera la reducción considerable del tiempo de desarrollo de las aplicaciones y lograr un producto final eficaz y con el nivel de robustez requerido.

2. A pesar de que Microsoft no le dio continuidad al desarrollo de Linq To Sql, este ORM constituye una gran API para la construcción rápida de acceso a datos, bien diseñada para bases de datos en SQL Server.

REFERENCIAS

1. M. Katrib, M. Valle, L. Paneque, R. Fresneda, T. Fuentes, I. Sierra, Y. Hernández, G. Som, Visual Studio .NET 2008 desafía todos los retos. Ed. Capitán San Luis, Ciudad de La Habana, 2008.

2. O. Hernández, Lo que nos traerá Orcas: novedades en C# 3.0. dotNetManía, n. 24, España, 2006.

3. O. Al Zabir, Building a Web 2.0 Portal with ASP.NET 3.5. Ed. O'Reilly, USA, 2008.

4. D. Espósito, Programming Microsoft® ASP.NET 3.5, Ed. Microsoft Press, USA, 2008.