Danysoft

Primeros pasos con ECO

Germán Benito | Danysoft

Este artículo tiene como objetivo iniciarle en ECO, presentándole los principales componentes, clases e interfaces que intervienen en ECO.

1. - Introducción.

A medida que crece el nivel de complejidad de un gran proyecto software, se hace necesaria la planificación y diseño del objetivo que quiere alcanzarse.

Para tal fin surgió en la década de los noventa el lenguaje UML (Unified Modeling Languaje) cuyo propósito era aportar un conjunto de elementos que facilitasen el diseño de software. UML especifica las entidades visuales que se utilizarán en la creación del diseño, así como el significado de cada una de ellas, sus atributos, su interacción con los demás etc. A los esquemas creados usando cada una de estos elementos es a lo que se le conoce habitualmente como *modelo*. Los modelos se utilizan para generar clases y código de soporte que representan el núcleo de la lógica de su aplicación. Estos modelos son independientes de la implementación final.

La tecnología ECO de Borland es un novedoso marco de trabajo guiado por el diseño para .NET Framework que utiliza diagramas UML como el motor que guía el desarrollo de aplicaciones. Esta visión del desarrollo de aplicaciones se conoce frecuentemente como Arquitectura Dirigida por Modelos (Model Driven Architecture – MDA). Al utilizar MDA, siempre se parte de un modelo independiente de la plataforma, lo que se conoce como PIM (Plataform Independt Model), que mediante un proceso de transformación dará lugar a un PSM (Plataform Specific Model) o modelo específico de la plataforma.

Cuando se utiliza ECO, los modelos UML no son solamente una guía para el desarrollo; están íntimamente ligados al propio proceso de desarrollo. Cuando es necesario hacer cambios en la aplicación, usted retorna al modelo, modificando sus atributos, asociaciones y restricciones, a partir de lo cual el código de la aplicación es actualizado. Como el modelo es el foco central de sus esfuerzos de desarrollo, existe una sincronización inherente entre el modelo UML y la aplicación creada a partir de él, evitando de esta forma que el modelo tienda a quedar obsoleto como ocurre en otros entornos de desarrollo en los que el diagrama UML solo representa una guía para el desarrollo de nuestra aplicación.

ECO realmente es equivalente a un desarrollo rápido guiado por modelos; reduce drásticamente la cantidad de código que es necesario teclear manualmente, acelerando el despliegue y mejorando la mantenibilidad general de sus aplicaciones. Además de utilizar un subconjunto del lenguaje UML, ECO emplea OCL (Object Constraint Language), el lenguaje de restricciones sobre objetos, un estándar del OMG (Object Management Group) para definir expresiones sobre los modelos UML. OCL se utiliza para crear reglas declarativas que calculan o controlan los valores de los atributos de sus objetos. Al igual que ocurre con UML, el código OCL que usted emplea en sus aplicaciones ECO reduce la cantidad de código que es necesario escribir y mantener.

ECO está desarrollado utilizando tecnología Together, un producto de Borland dirigido al modelado de aplicaciones, y una biblioteca de clases especifica para la plataforma .NET. Por una parte contamos con un diseñador de diagramas de clases y paquetes UML que nos servirán para crear nuestro modelo y por otra una serie de componentes capaces de aportar los servicios necesarios que hagan posible la implementación de ese modelo. Los distintos servicios con que cuenta ECO se encuentran repartidos en una serie de espacios de nombres, que tienes todos ellos como nombre raíz *Borland.Eco* En todos ellos encontraremos interfaces, clases y atributos que servirán como base para la implementación de nuestros modelos, facilitando la sincronización entre el código Delphi y el propio modelo.

2. - Principales componentes, clases e interfaces que intervienen en ECO

Centrándonos ya en la propia herramienta del que es objeto de este artículo, vamos a analizar los componentes y clases necesarias para el diseño y desarrollo de una pequeña aplicación que nos servirá de ejemplo para ver lo sencillo que resultaría trabajar con la tecnología ECO.

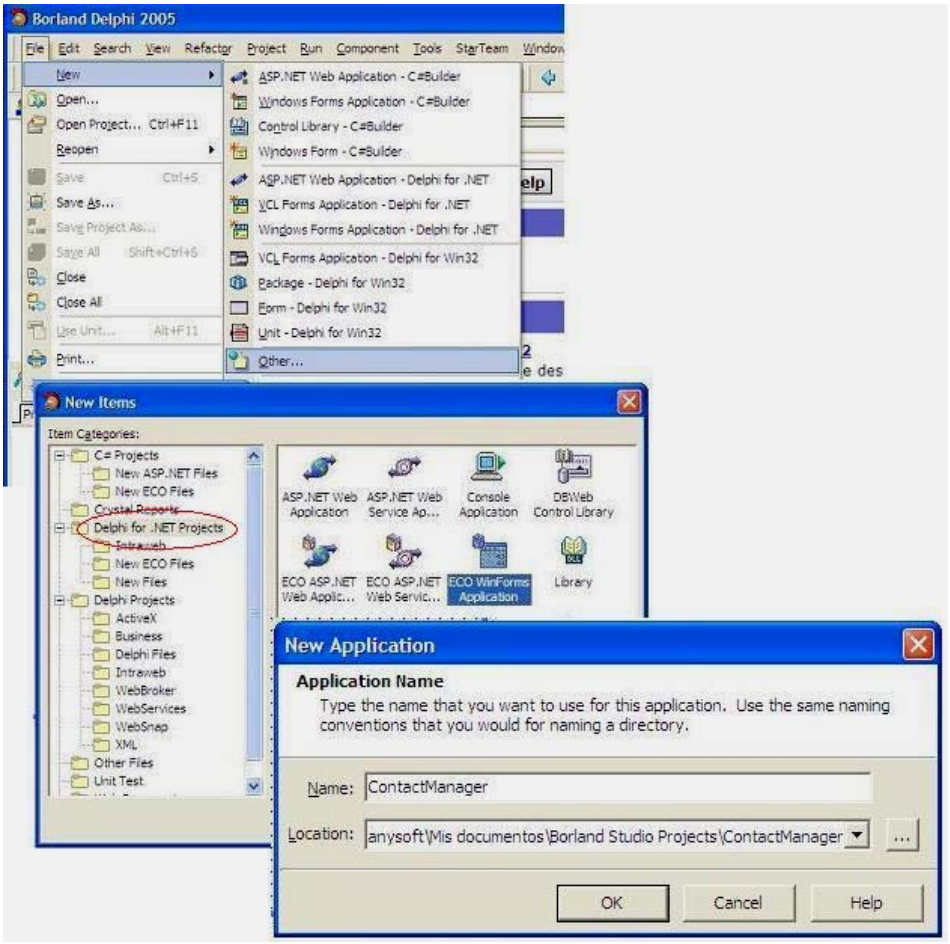

Fig.1 – Asistente para la creación de un nuevo proyecto ECO

Para nuestro caso particular utilizaremos un ECO Windows Form Application como muestra la imagen superior. Después de seleccionarlo el asistente nos pedirá un nombre para el proyecto y una ruta para la carpeta donde queremos ubicar los archivos del proyecto. Una vez aceptado, en el Gestor de proyectos aparecerán tres módulos.

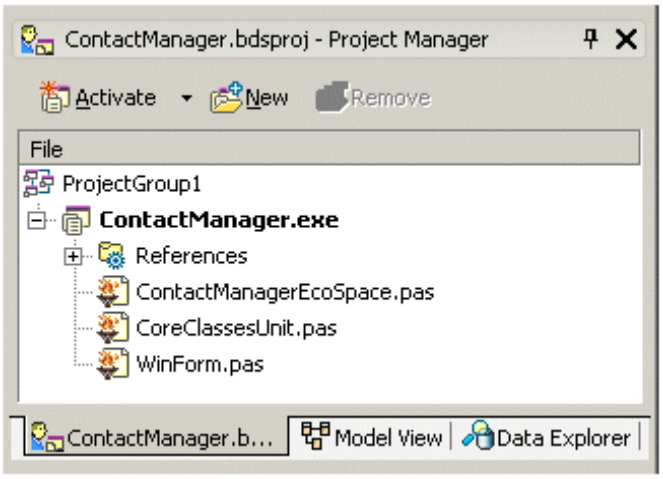

Fig. 2 – Unidades creadas por el asistente cuando iniciamos un proyecto ECO

De estos, hay dos unidades que son diferentes a los que se generan habitualmente cuando creamos una aplicación Delphi normal. Estos son <NombreProjecto>EcoSpace.pas y CoreClassesUnit.pas

Pasamos a analizar con detalle cada una de estás unidades añadidas por el asistente al "Project Manager".

CoreClassesUnit.pas:

Esta unidad contendrá las definiciones de paquetes, clases, interfaces y sus asociaciones del proyecto generadas a partir de nuestro modelo. Vamos a ver como podríamos generar el diagrama de nuestro modelo.

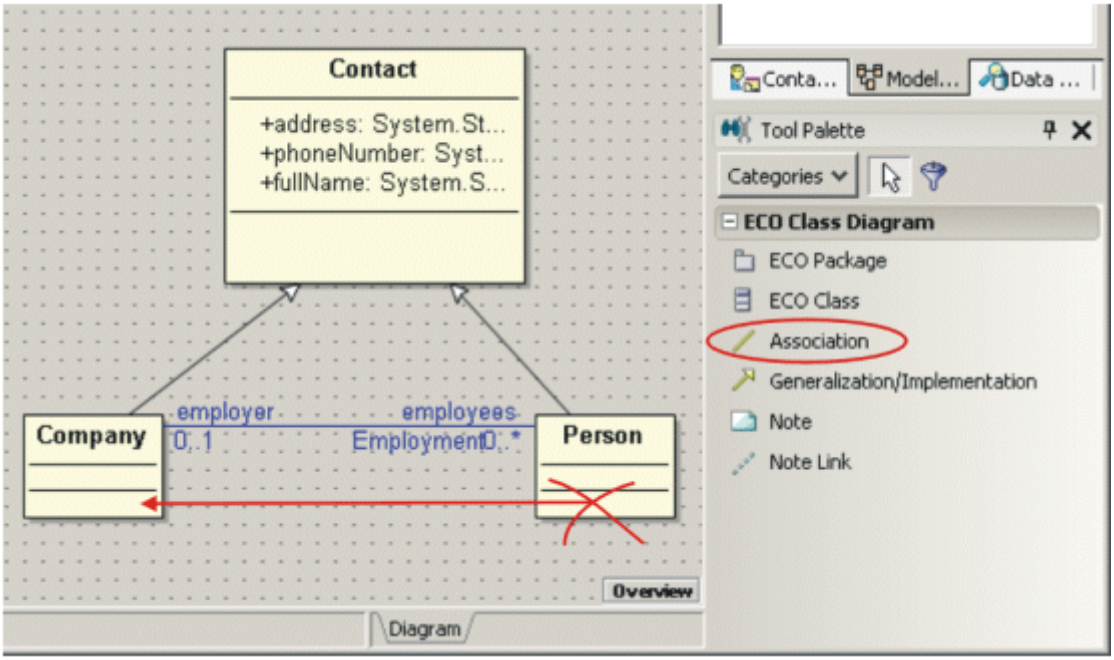

Fig. 3 – Ventana desde donde podremos definir nuestro diseño del modelo

Desde el panel **Model View** haciendo doble clik en el item CoreClasses accederíamos al editor de diagramas. Cuando dicho editor se abre, la paleta de herramientas cambia su contenido con aquellos componentes que pueden ser utilizados en nuestro modelo. Podríamos crear tres clases sin más que arrastrar dentro del editor tres componentes ECO Class quedando un diagrama tal y como muestra la figura 3. En la imagen de arriba hemos creado un diagrama donde la clase "Contact" se ha definido como abstracta configurando su propiedad en el Inspector de Objetos. Las otras dos clases son descendientes de la primera heredando así sus propiedades y atributos. Esto queda reflejado en el diagrama al utilizar dos componentes "Generalization/Implementation". Debemos configurar las propiedades de estas dos últimas clases para que no sean abstractas y así poder crear objetos a partir de ellas. Por otra parte la relación que existe entre compañías y personas es de una a muchos y esto lo hemos implementado utilizando el componente **Association** y configurando sus propiedades en el Inspector de Objetos. Este modelo se puede hacer todo lo complejo que se necesite de forma que se ajuste al objetivo que perseguimos. A modo de ejemplo, podríamos añadir un atributo en la clase Contact que podemos llamar "*/Descripción: string"*. En el Inspector de Objetos aparecerá en su propiedad **Derived** True y hay que utilizar su propiedad **Derivation OCL** para escribir: fullName + '-' + address de forma que pudiésemos obtener en un mismo atributo los datos del nombre y su dirección.

Podríamos haber creado este modelo en un "**ECO UML Package**" con la finalidad de poderlo utilizar en múltiples aplicaciones clientes (WinForm y ASP.NET)

Si inspeccionamos la unidad CoreClassesUnit.pas podemos comprobar como todas las clases definidas en el modelo así como las relaciones entre ellas han sido implementadas por el propio ECO es decir es "*Autogenerated ECO code*"

<NombreProjecto>EcoSpace.pas

Cuando creamos un proyecto con ECO utilizando cualquiera de las opciones que ofrece el deposito de objetos de Delphi, podemos ver que nuestro proyecto es una clase derivada de **DefaultEcoSpace** la cual a su vez es una clase derivada de **EcoSpace**. Estas dos últimas clases se encuentran en el espacio de nombres **Borland.Eco.Handles**. Un EcoSpace en tiempo de ejecución actúa como una caché de objetos creados a partir de nuestro modelo definido en la clase de diseño, ocupándose de su persistencia y de la gestión de transacciones. Los objetos en el EcoSpace son accedidos y manipulados a través de una serie de servicios disponibles en el EcoSpace a través de los servicios de la interfaz IEcoService. Las clases **EcoSpace** y **DefaultEcoSpace** no pueden ser utilizadas directamente. En particular esta última es una clase abstracta, y es utilizada por el asistente de la aplicación ECO para crear una subclase a partir de ella.

La unidad <NombreProjecto>EcoSpace.pas contiene la clase <NombreProjecto>EcoSpace que es como hemos dicho subclase de Borland.Eco.Handles.DefaultEcoSpace. La clase <NombreProjecto>EcoSpace contiene propiedades de solo lectura que te permiten acceder a cada uno de los servicios ofrecidos por la plataforma ECO. En ejecución ECOSpace contiene una instancia del modelo que hemos diseñado en el paso anterior.

Entre todos los miembros que nuestra clase "<NombreProjecto>EcoSpace" hereda de DefaultEcoSpace nos interesan especialmente tres propiedades, todas ellas relacionas con la persistencia de datos (concepto que introduciremos en breve). A saber:

PersistenceMapper: contiene una referencia a un objeto que es el encargado de la persistencia del EcoSpace, almacenandolo y recuperandolo de un fichero XML o bien de una base de datos.
OptimisticLocking: Si se pone a True, activa el bloqueo optimista a la hora de actualizar el depósito en el que reside el EcoSpace.
UpdateWholeObjects: Determina si los objetos enteros o solo los miembros que hayan sufrido modificaciones son escritos a la base de datos.

Una aplicación generalmente ofrece la opción de que los datos introducidos o editados por el usuario final sean grabados, con el objeto de que puedan ser recuperados posteriormente. Esto es lo que conocemos como persistencia de los datos. Con ECO hay dos métodos básicos de soportar la

persistencia. Grabando dichos valores en un fichero XML o bien utilizando un sistema gestor de bases de datos relacionales.

En el espacio de nombres **Borland.Eco.Persistence** existen tres clases que podemos utilizar como gestores de persistencia para un EcoSpace: **PesistenceMapperXML**, **PersistenteMapperBdp**, **PersistenceMapperSqlServer**. Cada uno de ellos utiliza o bien un fichero XML como deposito del EcoSpace, una base de datos Sql Server o una base de datos basada en una conexión BDP respectivamente.

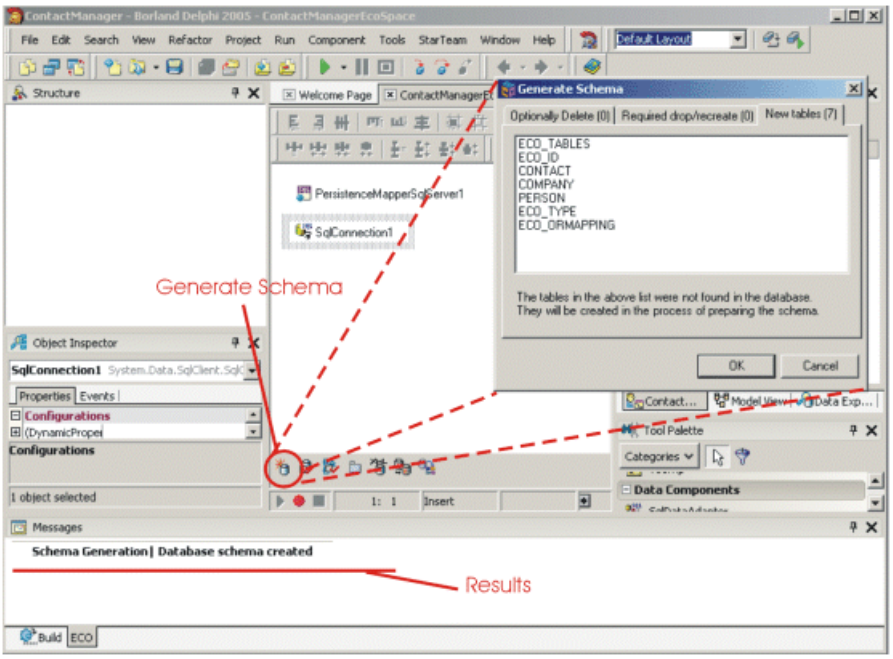

Fig. 4 - Pestaña Design del ECOSpace

ECO tiene la característica de generar un esquema de la base de datos para el modelo que hemos diseñado. Más aún, ECO puede crear un modelo a partir de la existencia de una base de datos utilizando herramientas de "ingeniería inversa".

El ECOSpace contiene, no sólo nuestro modelo, sino también los objetos o instancias (en tiempo de ejecución) de las clases del modelo. La pestaña "Design" del ECOSpace como muestra la figura 4 contiene todas las herramientas para:

- Configurar la persistencia de la aplicación ECO (RDBMS o fichero XML).
- Crear o rectificar el esquema de la base de datos.
- Seleccionar un paquete UML ECO del modelo para el que deseamos realizar la persistencia.
- Validar el modelo.

A continuación explicamos brevemente los pasos a seguir para configurar nuestro ECOSpace:

1.- Seleccionar el paquete UML que contiene las clases del modelo para el cual queremos establecer la persistencia. Esto se realiza haciendo clic en el botón "**Select packages**" en la barra de herramientas de la ventana del diseñador del ECOSpace

2.- Seleccionar un método cualquiera de persistencia. Para ello vamos a crear una base de datos en blanco a la que conectaremos el ECOSpace. Creada la base de datos en blanco, con el correspondiente RDBMS, introduciremos en el ECOSpace un componente para la persistencia de datos. Arrastraremos p.e. un componente **PersistenteMapperBdp** desde la categoría Enterprise Core Objects de la paleta de

herramientas y un componente de conexión a la base de datos, yendo a la pestaña Data Explorer y arrastrando la base de datos que acabamos de crear. Debemos configurar las siguientes propiedades:

```
Type: persistenceMapperBdp
  Name: persistenceMapperBdp1
  Connection: BdpConnection1

Type: BdpConnection
  Name: BdpConnection1

Type: <NombreProjecto>EcoSpace
  PersistanceMapper: persistenceMapperBdp1
```

3.- Validar el modelo. (botón "**Validate Model**") Esto provoca que nuestro ECOSpace realice una serie de comprobaciones para asegurarse que nuestro modelo es correcto. Por ejemplo, se asegura que las expresiones OCL definidas previamente son válidas

4.- Si estamos creando una aplicación ECO debemos hacer clic en el botón "**Create Database Schema"** del diseñador. Esto creará las tablas necesarias para soportar las clases y sus relaciones establecidas en el modelo. Si ya estamos trabajando con una aplicación ECO y hemos realizado cambios al modelo, debemos hacer clic sobre el botón "**Evolve Database"** del diseñador para rectificar la base de datos y realizar cambios en las columnas de las tablas correspondientes.

Todo este trabajo desarrollado hasta ahora de nada serviría, si no disponemos de mecanismos que permitan al usuario final interactuar con los objetos que son instancias de nuestras clases del modelo definido anteriormente. A continuación explicamos los componentes que intervienen en el formulario principal (WinForm.dfm) el cual va a constituir la interfaz de usuario.

WinForm.pas

La unidad WinForm.pas que aparece en el Project Manager, contiene el código fuente de la ventana principal WinForm de la aplicación ECO. La unidad WinForm.pas para una aplicación ECO te ofrece una propiedad que almacenará una instancia del ECOSpace de nuestra aplicación. Tal y como muestran las siguientes líneas de código:

```
type
  TWinForm = class (System.Windows.Forms.Form)
   (...)
  strict private
    fEcoSpace: TProject5EcoSpace;
  strict protected
   (...)
  Public
    constructor Create;
    property EcoSpace: TProject5EcoSpace read fEcoSpace;
  end;
```

El formulario asociado a la unidad Winform.pas constituye el interfaz de usuario a través de la cual podremos acceder a los objetos que se generan en tiempo de ejecución como instancias de las clases creadas en el diseño del modelo.

Vamos a analizar los cinco componentes que figuran en la ventana Design del formulario principal WinForm y que han sido añadidos por el asistente cuando hemos creado la aplicación ECO.

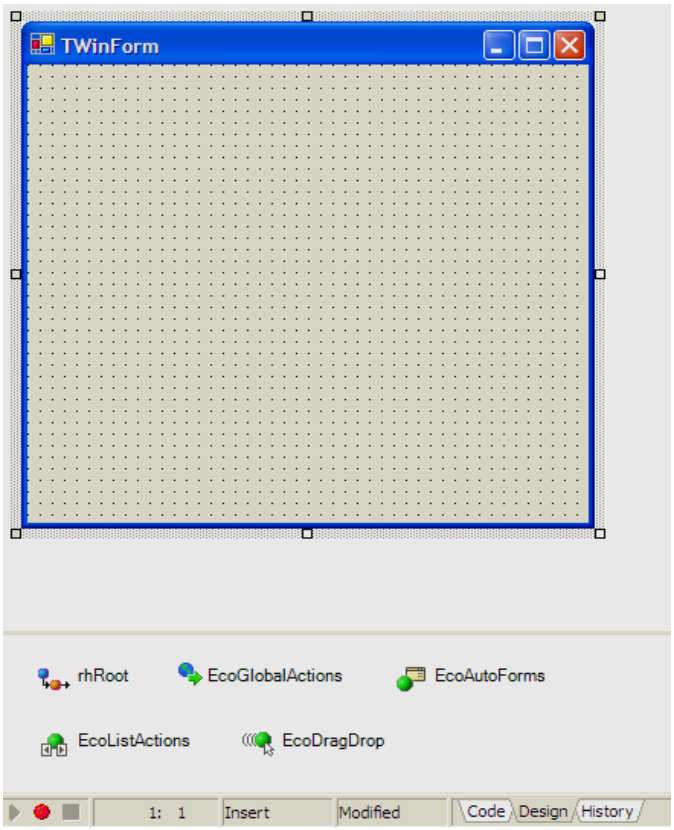

Fig. 5 - Componentes añadidos por el asistente a la aplicación ECO

Comenzaremos estudiando el componente "*rhRoot*" para lo cual tenemos que introducir un nuevo concepto, el de **Manejadores** que pasamos a explicar en el siguiente punto.

Manejadores:

Cada aplicación ECO debe tener una instancia de un EcoSpace. El EcoSpace contiene tanto la definición del modelo como el objeto que es creado mientras la aplicación se está ejecutando. Los manejadores son mecanismos que te permiten obtener los objetos asociados al EcoSpace en tiempo de ejecución. Un manejador puede representar un objeto simple, una lista de objetos o un valor calculado cualquiera.