|  |  |
| --- | --- |
| Geo Parking | DiffGrapph (Actualización de relaciones muchas - muchas o muchas - una)  2014 |

# Control de la documentación

### Control de la Configuración.

|  |  |
| --- | --- |
| Título: | DiffGrapph (Actualización de relaciones muchas - muchas o muchas - una) |
| Referencia: | GeoP\_Proyecto\_Doc\_DiffGraph\_1.0.docx |
| Autores: | Leonel Romero |
| Fecha: | 25/08/2014 |

### Histórico de Versiones.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Versión | Fecha | Estado | Responsable | Cambios |
| 1.0\_DraftA | 25/08/2014 | Pendiente de Revisión | Leonel Romero[autor] |  |
|  |  |  |  |  |

Contenido

[Control de la documentación 2](#_Toc390887892)

[Control de la Configuración. 2](#_Toc390887893)

[Histórico de Versiones. 2](#_Toc390887894)

[1 Introducción **¡Error! Marcador no definido.**](#_Toc390887895)

[1.1 Propósito **¡Error! Marcador no definido.**](#_Toc390887896)

[1.2 Alcance **¡Error! Marcador no definido.**](#_Toc390887897)

[1.3 Acrónimos **¡Error! Marcador no definido.**](#_Toc390887898)

[1.4 Definiciones **¡Error! Marcador no definido.**](#_Toc390887899)

[2 Organización **¡Error! Marcador no definido.**](#_Toc390887900)

[2.1 Sistema de Gestión de la Configuración **¡Error! Marcador no definido.**](#_Toc390887901)

[2.2 Personal, Roles y Responsabilidades **¡Error! Marcador no definido.**](#_Toc390887902)

[2.3 Herramientas **¡Error! Marcador no definido.**](#_Toc390887903)

[3 Procedimientos **¡Error! Marcador no definido.**](#_Toc390887904)

[3.1 Estimación de tiempo para identificación de Elementos **¡Error! Marcador no definido.**](#_Toc390887905)

[3.2 Identificar elementos de configuración **¡Error! Marcador no definido.**](#_Toc390887906)

[3.3 Establecer un sistema de administración de configuración **¡Error! Marcador no definido.**](#_Toc390887907)

[3.4 Crear o liberar las líneas base **¡Error! Marcador no definido.**](#_Toc390887908)

[3.5 Seguir las peticiones de cambio **¡Error! Marcador no definido.**](#_Toc390887909)

[3.6 Controlar los elementos de configuración **¡Error! Marcador no definido.**](#_Toc390887910)

[3.7 Realizar auditorías de configuración **¡Error! Marcador no definido.**](#_Toc390887911)

# Introducción

DiffGraph es una librería de clases para utilizar en conjunto con Entity Framework, el propósito de esta librería es poder actualizar las relaciones que entity framework por sí solo no maneja, estas son las que de algún lado de la relación se tenga muchos objetos de cierto tipo, una restricción (o no) de la librería, es que solo maneja el estado del contexto desasociado (detached), es decir que se debe validar este estado antes de llamar al método UpdateGraph(), o bien asegurarnos de que este será el caso.

Para realizar esto nos valemos del paquete de Nuget llamado DiffGraph, el cual se descarga y agrega a nuestro proyecto, desde que se realiza esto, se pueden utilizar las funciones disponibles

# Forma de Uso

Para utilizar las distintas funciones, DiffGraph nos provee un método, el cual está definido en el contexto de entity framework. Este método es el UpdateGraph(), el cual actualiza las tablas e la base de datos de acuerdo a ciertos parámetros que veremos más adelante.

Para realizar las actualizaciones, al método UpdateGraph le debemos indicar como son estas relaciones, es decir definirle que objeto contiene otros objetos, si es simplemente una referencia o es una pertenencia, etc. Se deben incluir todas las relaciones necesarias para la actualización.

Para definir la estructura de referencias, DiffGraph tiene 2 conceptos importantes, estos son Owned y Associated, el primero indica que un objeto es dueño de otro, por lo tanto cuando se actualice la relación se deberá hacer en ambas tablas, es decir en ambos lados de la relación (izquierdo y derecho); el segundo, indica que un objeto tiene referenciado uno de otro tipo, y solo se debe actualizar la referencia, es decir el lado izquierdo de la relación.

Estos 2 conceptos se utilizan con colecciones y entidades, una colección es para definir una relación que de algún lado de la misma tiene muchos objetos, mientras que una entidad sirve para definir una relación simple.

Para aclarar un poco lo antes mencionado, vamos a mostrar los distintos métodos que tiene diffgraph, estos son 4 ya que son las combinaciones de los conceptos dados anteriormente.

OwnedEntity()

OwnedCollection()

AsociatedEntity()

AssociatedCollection()

Una vez definida la estructura de las relaciones, debemos llamar al método SaveChanges() de Entity Framework.

A continuación muestro unos ejemplos del proyecto GeoParking en donde se utiliza para una relación muchos – muchos y uno – muchos.

Relación Muchos a Muchos:

public override int Update(Rol t)

{

RepositorioPermiso repoPermisos = new RepositorioPermiso();

using (var contexto = new ContextoBD())

{

contexto.UpdateGraph(t, map => map

.AssociatedCollection(r => r.Permisos));

foreach (var permiso in t.Permisos)

{

contexto.UpdateGraph(permiso, map => map

.AssociatedCollection(p => p.Roles));

}

return contexto.SaveChanges();

}

}

Se puede ver que al rol llamado t, se le define la estructura de relación, se indica que cada rol tiene una colección de permisos asociados, y luego por cada permiso que tiene t, se le indica que tiene una colección de roles asociados, así es como queda definida la relación muchos a muchos y solo queda llamar al método SaveChanges para grabar los cambios.

Relación uno a muchos:

public override int Update(PlayaDeEstacionamiento t)

{

var entry = contexto.Entry(t);

if (entry.State == System.Data.Entity.EntityState.Detached)

{

contexto.UpdateGraph(t, map => map

.OwnedCollection(p => p.Direcciones, with => with

.AssociatedEntity(d => d.PlayaDeEstacionamiento))

.OwnedCollection(p => p.Horarios, with => with

.AssociatedEntity(h => h.PlayaDeEstacionamiento))

.OwnedCollection(p => p.Precios, with => with

.AssociatedEntity(p => p.PlayaDeEstacionamiento))

.OwnedCollection(p => p.Servicios, with => with

.AssociatedEntity(s => s.PlayaDeEstacionamiento))

);

return contexto.SaveChanges();

}

else

{

return base.Update(t);

}

}

Aquí la situación es distinta, cada playa de estacionamiento tiene una colección de direcciones, otra de horarios, de precios, y de servicios, en donde cada dirección, horario, precio y servicio tiene, a su vez una playa de estacionamientos asociada.

Luego se llama al método SaveChanges() y queda actualizada la base de datos.