Remondo

Developer's Brain Repository in Plain C#

The Repository Pattern Example in C#

Posted on **April 13, 2012**

The Repository Pattern is a common construct to avoid duplication of data access logic throughout our application. This includes direct access to a database, ORM, WCF dataservices, xml files and so on. The sole purpose of the repository is to hide the nitty gritty details of accessing the data. We can easily query the repository for data objects, without having to know how to provide things like a connection string. The repository behaves like a freely available in-memory data collection to which we can add, delete and update objects.

The Repository pattern adds a separation layer between the data and domain layers of an application. It also makes the data access parts of an application better testable.

> You can download or view the solution sources on GitHub:
> LINQ to SQL version (the code from this example)
> Entity Framework code first version (added at the end of this post)

The example below show an interface of a generic repository of type T, which is a LINQ to SQL entity. It provides a basic interface with operations like Insert, Delete, GetById and GetAll. The SearchFor operation takes a lambda expression predicate to query for a specific entity.

```
using System;
using System.Linq;
using System.Linq.Expressions;

namespace Remondo.Database.Repositories
{
    public interface IRepository<T>
    {
        void Insert(T entity);
        void Delete(T entity);
        IQueryable<T> SearchFor(Expression<Func<T, bool>> predicate);
        IQueryable<T> GetAll();
        T GetById(int id);
    }
}
```

The implementation of the IRepository interface is pretty straight forward. In the constructor we retrieve the repository entity by calling the datacontext GetTable(of type T) method. The resulting Table(of type T) is the entity table we work with in the rest of the class methods. e.g. SearchFor() simply calls the Where operator on the table with the predicate provided.

```
using System;
using System.Data.Linq;
using System.Linq;
using System.Linq.Expressions;

namespace Remondo.Database.Repositories
{
    public class Repository<T> : IRepository<T> where T : class, IEntity
    {
        protected Table<T> DataTable;

        public Repository(DataContext dataContext)
        {
            DataTable = dataContext.GetTable<T>();
        }

        #region IRepository<T> Members

        public void Insert(T entity)
        {
            DataTable.InsertOnSubmit(entity);
        }

        public void Delete(T entity)
        {
            DataTable.DeleteOnSubmit(entity);
        }

        public IQueryable<T> SearchFor(Expression<Func<T, bool>> predicate)
        {
            return DataTable.Where(predicate);
        }

        public IQueryable<T> GetAll()
        {
            return DataTable;
        }

        public T GetById(int id)
        {
            // Sidenote: the == operator throws NotSupported Exception!
            // 'The Mapping of Interface Member is not supported'
            // Use .Equals() instead
            return DataTable.Single(e => e.ID.Equals(id));
        }

        #endregion
    }
}
```

The generic GetById() method explicitly needs all our entities to implement the IEntity interface. This is because we need them to provide us with an Id property to make our generic search for a specific Id possible.

```
namespace Remondo.Database
{
    public interface IEntity
    {
        int ID { get; }
    }
}
```

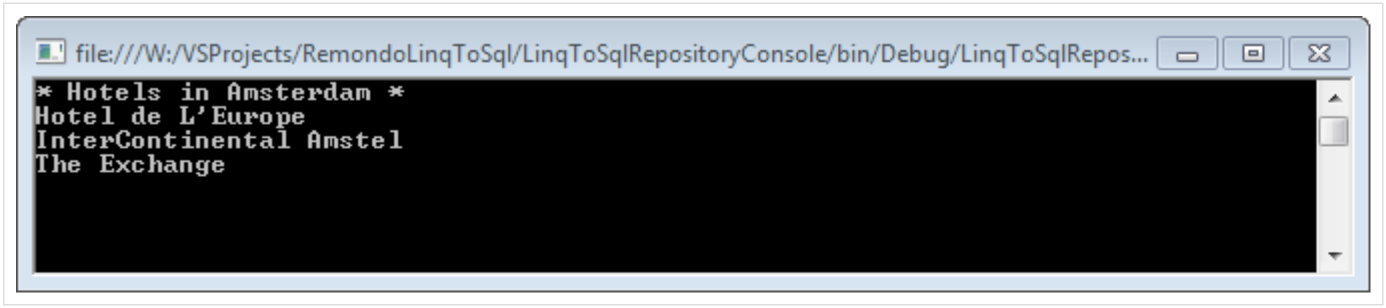
Since we already have LINQ to SQL entities with an Id property, declaring the IEntity interface is sufficient. Since these are partial classes, they will not be overridden by LINQ to SQL code generation tools.

```
namespace Remondo.Database
```

```
{
    partial class City : IEntity
    {
    }

    partial class Hotel : IEntity
    {
    }
}
```

We are now ready to use the generic repository in an application.

```
using System;
using System.Collections.Generic;
using System.Linq;
using Remondo.Database;
using Remondo.Database.Repositories;

namespace LinqToSqlRepositoryConsole
{
    internal class Program
    {
        private static void Main()
        {
            using (var dataContext = new HotelsDataContext())
            {
                var hotelRepository = new Repository<Hotel>(dataContext);
                var cityRepository = new Repository<City>(dataContext);

                City city = cityRepository
                    .SearchFor(c => c.Name.StartsWith("Ams"))
                    .Single();

                IEnumerable<Hotel> orderedHotels = hotelRepository
                    .GetAll()
                    .Where(c => c.City.Equals(city))
                    .OrderBy(h => h.Name);

                Console.WriteLine("* Hotels in {0} *", city.Name);

                foreach (Hotel orderedHotel in orderedHotels)
                {
                    Console.WriteLine(orderedHotel.Name);
                }

                Console.ReadKey();
            }
        }
    }
}
```


file:///W:/VSProjects/RemondoLinqToSql/LinqToSqlRepositoryConsole/bin/Debug/LinqToSqlRepos...

```
* Hotels in Amsterdam *
Hotel de L'Europe
InterContinental Amstel
The Exchange
```

Once we get of the generic path into more entity specific operations we can create an implementation for that entity based on the generic version. In the example below we construct a HotelRepository with an entity specific GetHotelsByCity() method. You get the idea. ;-)

```
using System.Data.Linq;
using System.Linq;

namespace Remondo.Database.Repositories
{
    public class HotelRepository : Repository<Hotel>, IHotelRepository
    {
        public HotelRepository(DataContext dataContext)
            : base(dataContext)
        {
        }

        public IQueryable<Hotel> FindHotelsByCity(City city)
        {
            return DataTable.Where(h => h.City.Equals(city));
        }
    }
}
```

[Update juli 2012] Entity Framework version

The code below shows a nice and clean implementation of the generic repository pattern for the Entity Framework. There's no need for the IEntity interface here since we use the convenient Find extension method of the DbSet class. Thanks to my co-worker Frank van der Geld for helping me out.

```
using System;
using System.Data.Entity;
using System.Linq;
using System.Linq.Expressions;

namespace Remondo.Database.Repositories
{
    public class Repository<T> : IRepository<T> where T : class
    {
        protected DbSet<T> DbSet;

        public Repository(DbContext dataContext)
        {
            DbSet = dataContext.Set<T>();
        }

        #region IRepository<T> Members

        public void Insert(T entity)
        {
            DbSet.Add(entity);
        }

        public void Delete(T entity)
        {
            DbSet.Remove(entity);
        }

        public IQueryable<T> SearchFor(Expression<Func<T, bool>> predicate)
        {
            return DbSet.Where(predicate);
        }

        public IQueryable<T> GetAll()
        {
            return DbSet;
        }

        public T GetById(int id)
        {
            return DbSet.Find(id);
        }

        #endregion
    }
}
```

Articles you might like

- Using Autofac to Resolve Dependencies in a MVC Controller
- Memento Pattern Example in C#
- Visitor Pattern Example in C#
- Simple Factory Pattern Example in C#
- The Command Pattern Example in C#
- Building a Simple IoC Container in C#
- The Strategy Pattern Example in C#
- Null Object Design Pattern Example in C#
- Observer Pattern Example in C# with IObservable
- Thread-safe Singleton Pattern Example in C#

Like 49 Tweet Share Pin it

This entry was posted in **Design Patterns** and tagged **gof**, **pattern** by **Leon van Bokhorst**. Bookmark the **permalink [http://web.archive.org/web/20150404154203/http://www.remondo.net/repository-pattern-example-csharp/]** .

94 THOUGHTS ON "THE REPOSITORY PATTERN EXAMPLE IN C#"

Jan
on **April 23, 2012 at 11:49 am** said:

Can you provide an example to use this with EF?

Btw, awesome blog. I really enjoy it. Found it today and bookmarked. :-)

Jan

Leon van Bokhorst
on **April 23, 2012 at 1:08 pm** said:

You're welcome sir ;-)

I added an Entity Framework Based example to the repository.

Great design on http://janhartmann.dk/ Your work I presume?

~~The generic repository pattern for EF looks a lot like this one.~~
~~I'll try to post it tonight.~~

I added an update to the post juli 2012 with a generic repository pattern for Entity Framework.