FluentNHibernate Conventions

리얼웹	배성혁	2011-12-11 오전 12:58

FluentNHibernate의 Convention (규약) 기능은 수 많은 엔티티와 관련된 명명 규칙, 설정을 한번에 수행할 수 있도록 해줍니다. http://wiki.fluentnhibernate.org/Conventions 에 보면, HBM에서도 제공하는 기능도 포함하지만, 다음과 같은 예제를 보면 쉽게 이해가 될 것입니다.

```
Table.Is(x => x.EntityType.Name + "Table")  // 테이블명을 엔티티명+"Table" 로 규약
PrimaryKey.Name.Is(x => "ID")               // PrimaryKey 컬럼명을 "ID" 로 규약
AutoImport.Never()
DefaultAccess.Field()
DefaultCascade.All()
DefaultLazy.Always()
DynamicInsert.AlwaysTrue()
DynamicUpdate.AlwaysTrue()
OptimisticLock.Is(x => x.Dirty())
Cache.Is(x => x.AsReadOnly())
ForeignKey.EndsWith("ID")
```

과 같이 엔티티 정의에 대해 공통된 규약을 정의하게 되면,

1. Mapping 작업의 반복 작업을 줄이고,
2. Mapping 규칙이 변경되었을 시에, 변경이 쉽고,
3. Pascal Naming 에서 Oracle Naming 으로의 변경

등이 쉽게 적용됩니다.

이에 RCL-4.2.0 부터는 FluentNhibernate의 Convention 기능을 이용하여, 쉽게 Mapping 규약을 정의하고, 사용할 수 있게 했습니다.

우선 기존 Mapping 코드가 어떻게 변경되는지 알아봅시다.

기존 Mapping

```
public class RiskMap : ClassMap<Risk>
{
        public RiskMap()
        {
                Table("Risk");
                DynamicInsert();
                DynamicUpdate();
                LazyLoad();

                Cache.Region(ErmConst.ProductCode).ReadWrite().IncludeAll();
                  Id(x => x.Id).Column("RiskId").GeneratedBy.Assigned();
                  Map(x => x.Code).Column("RiskCode").Not.Nullable();
                  Map(x => x.Name).Column("RiskName").Not.Nullable();
                  Map(x => x.Description).Length(MappingContext.MaxStringLength);
                  Map(x => x.ExAttr).Length(MappingContext.MaxStringLength);
        ...
```

Convention 사용 후

```
public class RiskMap : ClassMap<Risk>
{
        public RiskMap()
        {
                Cache.Region(ErmConst.ProductCode).ReadWrite().IncludeAll();
                  Id(x => x.Id).GeneratedBy.Assigned();
```

```
                Map(x => x.Code).Column("RiskCode").Not.Nullable();
                Map(x => x.Name).Column("RiskName").Not.Nullable();
                Map(x => x.Description).Length(MappingContext.MaxStringLength);
                Map(x => x.ExAttr).Length(MappingContext.MaxStringLength);

                ...
```

기존 Mapping에서 변경된 점은 Class 와 관련된 기본 옵션인 정보가 사라졌고, 가장 중요한 Primary Key 인 Id 의 컬럼명이 사라졌습니다. 일반적으로 Primary Key 컬럼명을 지정하지 않으면, "Id" 가 됩니다만, Convention을 어떻게 정의하냐에 따라, 달라질 수 있습니다.

그럼 어떻게 Conventions을 지정하는지 Unit Test 코드를 봅시다.
이 코드는 FluentNHibernate 용의 Base Class 입니다.

```
protected virtual void OnTestFixtureSetUp()
{
      var options = ConventionOptions.Default;
      var conventions = ConventionOptions.ToConventions(options).ToList();
      conventions.Add(ConventionBuilder.HasMany.Always(x => x.Key.Column(x.EntityType.Name + options.ForeignKeySurfix)));
      conventions.Add(ConventionBuilder.Reference.Always(x => x.Column(x.Property.Name + options.ForeignKeySurfix)));
      conventions.Add(ConventionBuilder.HasManyToMany.Always(x =>
      {
          x.Key.Column(x.EntityType.Name + options.ForeignKeySurfix);
          x.Relationship.Column(x.ChildType.Name + options.ForeignKeySurfix);
      }));

      InitializeNHibernateAndIoC(ContainerFilePath,
                                 GetDatabaseEngine(),
                                 GetDatabaseName(),
                                 GetMappingInfo(),
                                 GetNHibernateProperties(),
                                 cfg =>
                                 {
                cfg.SetListener(NHibernate.Event.ListenerType.PreInsert, new UpdateTimestampEventListener());
                cfg.SetListener(NHibernate.Event.ListenerType.PreUpdate, new UpdateTimestampEventListener());
                                 },
                                 conventions.ToArray());

      CurrentContext.CreateUnitOfWork();
}
```

위에서 보면 RCL-4.2.0 에서 제공하는 ConventionOptions 에 의해 IConvention[] 을 빌드하여, NHibernate 테스트 초기화 시에 제공하면 됩니다.
RealERM 에서는 기본 Convention에 추가로, Convention을 추가하여 사용할 수 있도록 했습니다.

실 프로그램에서는 IoC를 이용하여 다음과 같이 정의하면 됩니다.

```
<!-- UnitOfWork 사용을 위한 Factory -->
<component id="nhibernate.unit-of-work.factory"
           service="RCL.Data.NH.IUnitOfWorkFactory, RCL.Data"
           type="RCL.Data.NH.FluentNHUnitOfWorkFactory, RCL.Data">
        <parameters>
                <cfgFileName>hibernate.fluent.cfg.xml</cfgFileName>
                <ConventionOptions>${RealERM.Pascal.Conventions}</ConventionOptions>
        </parameters>
</component>
<component id="RealERM.Pascal.Conventions"
                   type="RCL.Data.NH.Fluents.ConventionOptions, RCL.Data">
        <parameters>
                <DefaultLazy>true</DefaultLazy>
                <DynamicInsert>true</DynamicInsert>
                <DynamicUpdate>true</DynamicUpdate>
                <PrimaryKeySurfix>Id</PrimaryKeySurfix>
                <ForeignKeySurfix>Id</ForeignKeySurfix>
        </parameters>
</component>
```

IUnitOfWorkFactory 는 FluentNHUnitOfWorkFactory로 설정하고, ConventionOptions 속성을 지정해 주면 됩니다.

이렇게 되면, 기본적으로 one-to-many, many-to-one 의 컬럼명이 Entity Name + "Id" 형태가 됩니다. 일반적으로 원하는 형태입니다. 물론 Oracle 형태가 되려면 "_ID" 가 좋겠죠. 그렇게 설정하려면 ConventionOptions 인스턴스의 속성값을 조정하면 됩니다.

ConventionOptions 의 Property 입니다.

```
public class ConventionOptions
{
	/// <summary>
	/// LazyLoad를 기본으로 할 것인가?
	/// </summary>
	public bool DefaultLazy { get; set; }
	/// <summary>
	/// Entity 등록 시, 값이 있는 속성만으로 Query를 생성할 것인가를 설정하는 값의 기본 값
	/// </summary>
	public bool DynamicInsert { get; set; }
	/// <summary>
	/// Entity 갱신 시, 값이 있는 속성만으로 Query를 생성할 것인가를 설정하는 값의 기본 값
	/// </summary>
	public bool DynamicUpdate { get; set; }
	/// <summary>
	/// 테이블 명의 접두사 (예: "TBL_", "RAT_" 등)
	/// </summary>
	public string TableNamePrefix { get; set; }
	/// <summary>
	/// 테이블 명의 접미사 (예: "_TABLE" 등)
	/// </summary>
	public string TableNameSurfix { get; set; }
	/// <summary>
	/// Primary Key의 기본 값 (예: "Id"), 보통 <see cref="PrimaryKeySurfix"/>를 많이 사용한다.
	/// </summary>
	public string PrimaryKeyName { get; set; }
	/// <summary>
	/// Primary Key 의 명칭을 EntityName + PrimaryKeySurfix 로 설정하게 합니다.
	/// (예: Surfix가 "Id" 일 경우, User의 Primary Key 는 "UserId" 가 되고, Company의 Primary Key 는 "CompanyId" 가 됩니다)
	/// </summary>
	public string PrimaryKeySurfix { get; set; }
	/// <summary>
	/// Foreign Key의 접미사를 지정합니다. (예: "_ID", "Id")
	/// </summary>
	public string ForeignKeySurfix { get; set; }
}
```

위의 속성 정보들을 기준으로, 다음과 같이 Convention 들을 빌드합니다.

```
/// <summary>
/// Fluent NHibernate 에서 제공하는 Convention 설정에 따른 <see cref="IConvention"/> 인스턴스를 빌드하여 제공합니다.
/// </summary>
/// <param name="options">ConventionOptions 인스턴스</param>
/// <returns>Convention 설정 정보를 기초로 만든 <see cref="IConvention"/>의 인스턴스 배열</returns>
public static IList<IConvention> ToConventions(ConventionOptions options)
{
	options.ShouldNotBeNull("optioons");
	if(IsDebugEnabled)
	  log.Debug("ConventionOptions 정보로 IConvention[]을 빌드합니다...  " + options);

	var conventions = new List<IConvention>();
	conventions.Add((options.DefaultLazy) ? LazyLoad.Always() : LazyLoad.Never());

	if(options.DynamicInsert)
		conventions.Add(FluentNHibernate.Conventions.Helpers.DynamicInsert.AlwaysTrue());
	if(options.DynamicUpdate)
		conventions.Add(FluentNHibernate.Conventions.Helpers.DynamicUpdate.AlwaysTrue());

	if(options.TableNamePrefix.IsNotWhiteSpace())
		conventions.Add(Table.Is(x => options.TableNamePrefix + x.EntityType.Name));
	else if(options.TableNamePrefix.IsNotWhiteSpace())
		conventions.Add(Table.Is(x => x.EntityType.Name + options.TableNameSurfix));
	else
		conventions.Add(Table.Is(x => x.EntityType.Name));

	if(options.PrimaryKeyName.IsNotWhiteSpace())
		conventions.Add(PrimaryKey.Name.Is(x => options.PrimaryKeyName));
	else if(options.PrimaryKeySurfix.IsNotWhiteSpace())
		conventions.Add(PrimaryKey.Name.Is(x => x.EntityType.Name + options.PrimaryKeySurfix));
	if(options.ForeignKeySurfix.IsNotWhiteSpace())
	{
	    conventions.Add(ForeignKey.EndsWith(options.ForeignKeySurfix));
```

```
            conventions.Add(ConventionBuilder.HasMany.Always(x => x.Key.Column(x.EntityType.Name + options.ForeignKeySurfix)));
            conventions.Add(ConventionBuilder.Reference.Always(x => x.Column(x.Property.Name + options.ForeignKeySurfix)));
            conventions.Add(ConventionBuilder.HasManyToMany.Always(x =>
            {
                x.Key.Column(x.EntityType.Name + options.PrimaryKeySurfix);
                x.Relationship.Column(x.ChildType.Name + options.ForeignKeySurfix);
            }));
        }

        return conventions;
}
```

FluentNHibernate의 아주 좋은 기능들 (FluentMapping, Conventions, Perspecification Testing 등)을 잘 활용한다면, 기존 HBM으로는 상상할 수 없는 개발 생산성 및 유연성을 확보할 수 있습니다.
현재, RCL에서는 아주 기본적인 옵션으로만 Convention을 만들 수 있도록 되어 있습니다만, FluentNHibernate Conventions 을 참고하면, 더욱 유연하게 Conventions을 활용할 수 있습니다.