FluentNHibernate Conventions II

리얼웹	배성혁	2011-12-15 오후 10:59

FluentNHibernate Convention 에 대한 소개를 했습니다만, 실제로 사용하기에 복잡하다던가, Convention 관련된 정보를 모두 학습해야 하는 애로점이 있었습니다.
뭐 항상 하듯이, 많은 부분 공부하지 않아도 되도록 정리를 했습니다.

우선 대표적인 두 가지 Conventions 을 정의하였습니다.

1. PascalNamingConvention : RDBMS의 Name을 Pascal Naming 규칙으로 정의
    a. 예: Company, CompanyId, CompanyCode, Description

2. OracleNamingConvention : RDBMS의 Name을 대문자와 단어 구분값으로 '_' 를 사용
    a. 예: COMPANY, COMPANY_ID, COMPANY_CODE, DESCRIPTION

자 보시면, Oracle 용과 그 외 버전으로 생각할 수 있겠죠?
FluentNHibernate 를 이용하여 Mapping 을 수행하는데, 기껏 Magic String을 제거하고자 노력했는데, 테이블명, 컬럼명 등의 명명규칙의 변화에 따라 Mapping 파일이 달라진다면, 도로아미타불입니다. 그래서, 위와 같은 Convention 을 정의하고, 옵션으로 처리할 수 있도록 ConventionOptions 를 같이 활용하여, 유연하면서, 최대한 Magic String을 쓰지 않고, 환경설정으로만 RDBMS 및 Naming 규칙 변경에 따른 코드 수정을 최소화할 수 있도록 했습니다.

우선 실제 사용하기 위해 Windsor Configuration 설정 정보를 보시게 되면 다음과 같습니다.

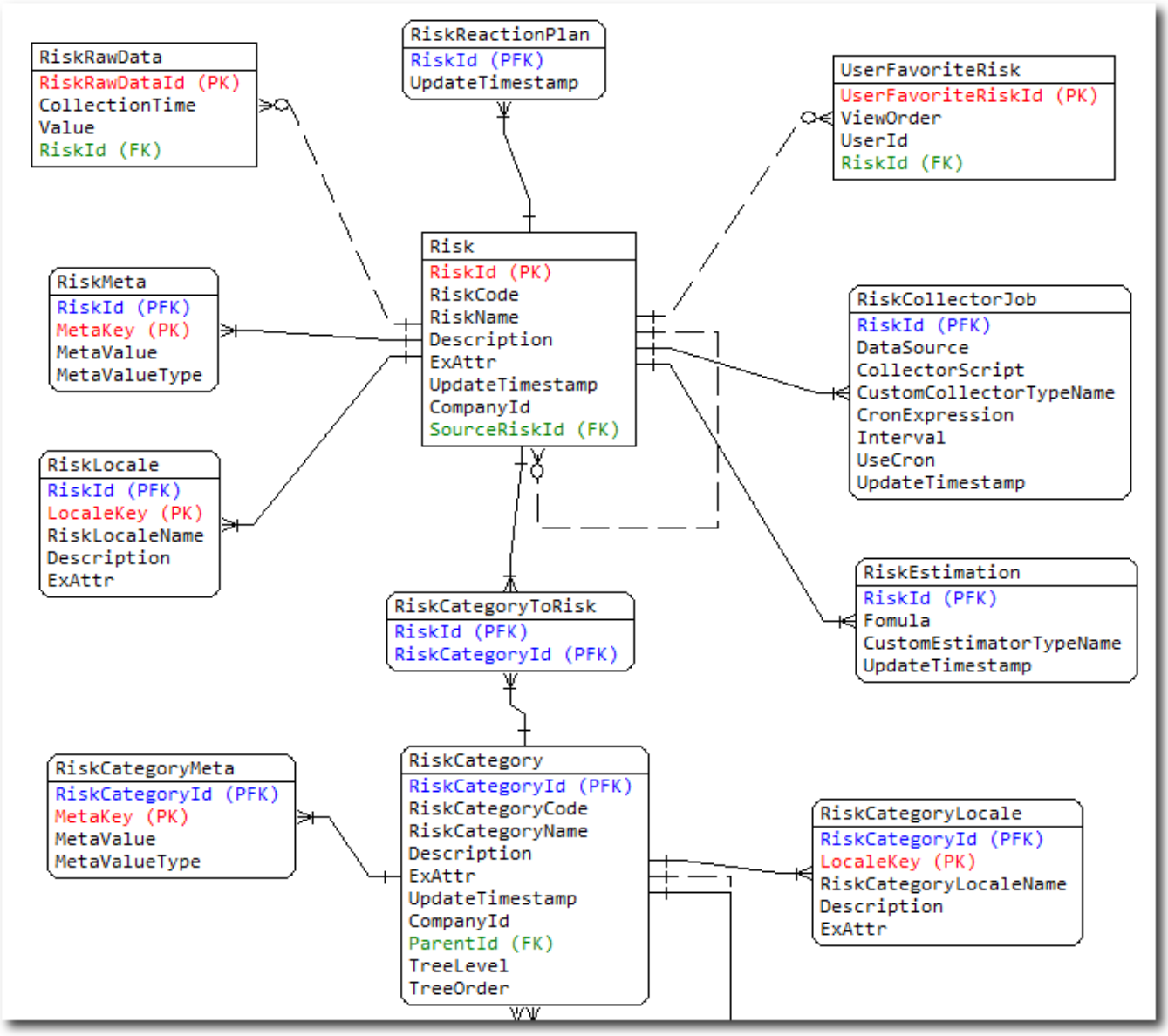
```
<component id="NHibernate.UnitOfWorkFactory.Fluent"
                   service="RCL.Data.NH.IUnitOfWorkFactory, RCL.Data"
                   type="RCL.Data.NH.FluentNHUnitOfWorkFactory, RCL.Data">
        <parameters>
                <assemblyNames>
                        <array>
                                <item>RCL.Data.NH.DomainModel.Fluent</item>
                        </array>
                </assemblyNames>
                <Convention>${FluentConvention.PascalNaming}</Convention>
        </parameters>
</component>
<component id="FluentConvention.PascalNaming"
                   service="RCL.Data.NH.Fluents.IFluentConvention, RCL.Data"
                   type="RCL.Data.NH.Fluents.PascalNamingConvention, RCL.Data">
        <parameters>
                <Options>${FluentNHibernate.ConventionOptions}</Options>
                <PropertyWithClassName>
                        <list>
                                <item>Code</item>
                                <item>Name</item>
                        </list>
                </PropertyWithClassName>
        </parameters>
</component>
<component id="FluentNHibernate.ConventionOptions"
           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>
```

우선 UnitOfWorkFactory 컴포넌트의 속성으로 Convention 을 지정할 수 있습니다. 이렇게 되면, 해당 Convention으로 Schema 를 생성할 수 있습니다. 이 예에서는 PascalNamingConvention을 지정하였고, ConventionOptions으로 Naming 규칙과 별 상관없이 공통적으로 적용되는 부분을 정의합니다.
새롭게 도입된 부분은 PropertyWithClassName 인데, 일반적으로 Class의 속성명이 컬럼명이 되지 않고, ClassName + PropertyName 조합을 컬럼명으로 지정하고자 할때 지정하게 됩니다.
즉 위의 Code 라는 속성명에 대해서는 컬럼명은 ClassName + Code 가 된다는 뜻입니다.

단위 테스트 시에는 다음과 같이 초기화 시에 원하는 Convention 클래스를 지정해 주면 됩니다.

```
protected virtual void OnTestFixtureSetUp()
{
    InitializeNHibernateAndIoC(ContainerFilePath,
                           GetDatabaseEngine(),
                           GetDatabaseName(),
                           GetMappingInfo(),
                           GetNHibernateProperties(),
                           cfg =>
                           {
            cfg.SetListener(NHibernate.Event.ListenerType.PreInsert, new UpdateTimestampEventListener());
            cfg.SetListener(NHibernate.Event.ListenerType.PreUpdate, new UpdateTimestampEventListener());
                           },
           new PascalNamingConvention(ConventionOptions.Default, new List<string> { "Code", "Name" }));

    CurrentContext.CreateUnitOfWork();
}
```

그럼 일반적인 RealERM 에 대해 PascalNamingConvention을 사용하면 다음과 같습니다.



Schema 생성 결과는 위의 이미지와 같습니다만, FluentNHibernate Mapping 파일을 보시면, 기존과는 다르게 굳이 특정적으로 지정하지 않아도, 자동으로 구성이 된다는 데 의미가 있습니다.

우선 Risk (RiskLocale, RiskMeta 포함)에 대한 매핑을 보면

```
public RiskMap()
{
	Cache.Region(ErmConst.ProductCode).ReadWrite().IncludeAll();

	Id(x => x.Id).GeneratedBy.Assigned();

	References(x => x.Company).Index("IX_Risk_Company").Fetch.Select().Cascade.None();

	Map(x => x.Code).Not.Nullable();
	Map(x => x.Name).Not.Nullable();
	Map(x => x.Description);
	Map(x => x.ExAttr);

	HasManyToMany(x => x.Categories)
		.Table(MappingContext.AsNamingText("RiskCategoryToRisk")) // RISK_CATEGORY_TO_RISK로 변환됨
		.Access.CamelCaseField(Prefix.Underscore)
		.Cascade.SaveUpdate()
		.LazyLoad()
		.AsSet();

	HasMany(x => x.DependentRisks)
```

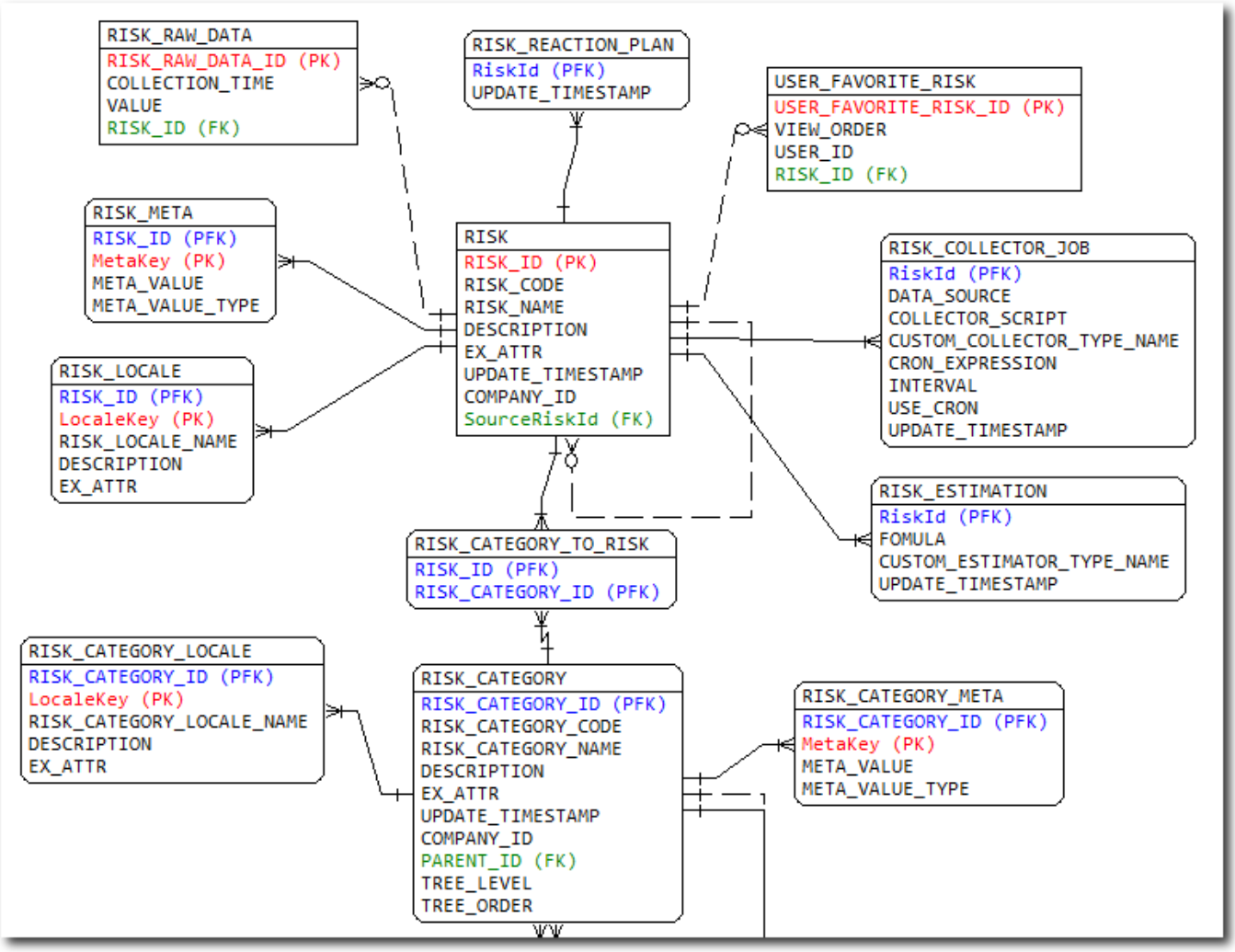
```
                .Access.CamelCaseField(Prefix.Underscore)
                .KeyColumn("SourceRiskId")
                .Inverse()
                .LazyLoad()
                .AsSet();
        // Localed
        //
        HasMany<RiskLocale>(x => x.LocaleMap)
                .Table(MappingContext.AsNamingText("RiskLocale")) // RISK_LOCALE로 변환됨
                .Access.CamelCaseField(Prefix.Underscore)
                .Cascade.AllDeleteOrphan()
                .AsMap<CultureUserType>("LocaleKey")
                .Component(loc =>
                          {
                          loc.Map(x => x.Name).Not.Nullable();
                          loc.Map(x => x.Description).Length(MappingContext.MaxStringLength);
                          loc.Map(x => x.ExAttr).Length(MappingContext.MaxStringLength);
                          });
        // Metadata
        HasMany<MetadataValue>(x => x.MetadataMap)
                .Table(MappingContext.AsNamingText("RiskMeta"))  // RISK_META로 변환됨
                .Access.CamelCaseField(Prefix.Underscore)
                .Cascade.AllDeleteOrphan()
                .AsMap<string>("MetaKey")
                .Component(m =>
                          {
                     m.Map(x => x.Value).Column("MetaValue".AsNamingText()).Length(MappingContext.MaxStringLength);
                     m.Map(x => x.ValueType).Column("MetaValueType".AsNamingText());
                          });

        Map(x => x.UpdateTimestamp).CustomType("Timestamp");
}
```

보시다시피 Id 및 Code, Name 속성에 대해서도 컬럼명을 지정하지 않았음에도 원하는 값이 되었습니다.
One-to-many 에 대해서도, Key column 값을 지정하지 않아도 "RiskId" 로 지정되었습니다.

자 이제 Mapping 정보는 건드리지 않고, Convention 과 옵션만을 변경하여, Oracle Naming 규칙으로 변경해 보겠습니다.
우선 Oracle naming으로 지정하는 테스트 코드를 보면

```
protected virtual void OnTestFixtureSetUp()
{
// 매핑 Class에서 지정한 특정 Magic String도 변경할 수 있도록 하기 위해
MappingContext.NamingRule = NamingRuleKind.Oracle;
InitializeNHibernateAndIoC(ContainerFilePath,
                           GetDatabaseEngine(),
                           GetDatabaseName(),
                           GetMappingInfo(),
                           GetNHibernateProperties(),
                           cfg =>
                           {
            cfg.SetListener(NHibernate.Event.ListenerType.PreInsert, new UpdateTimestampEventListener());
            cfg.SetListener(NHibernate.Event.ListenerType.PreUpdate, new UpdateTimestampEventListener());
                           },
            new OracleNamingConvention(ConventionOptions.Oracle, new List<string> { "Code", "Name" }));
            // new PascalNamingConvention(ConventionOptions.Default, new List<string> { "Code", "Name" }));
CurrentContext.CreateUnitOfWork();
}
```

과 같이 Convention Class 를 Oracle 대응으로 변경하고, Schema를 생성했습니다.



보시면, 대부분의 컬럼명이 Oracle Naming 규칙이 되었음을 아실 수 있습니다.