Fluent ADO.NET

2009.01

Realweb

목차

1. Fluent ADO.NET 개요
2. Fluent 필요성 (CBD)
3. Fluent ADO.NET 구조
    1. Mapping Method (INameMapper, IPersister, Converter<I,O>)
    2. Mapping Rules (NoChange, Trim, Capitalize, Others …)
4. Fluent ADO.NET 예제
    1. Using IDataReader (Simple, Paging)
    2. Using DataTable
    3. SaveOrUpdate Persistent Object
5. Fluent ADO.NET 활용 절차
6. Fluent ADO.NET 제한 사항
7. Persister 상세 구현 Logic

1. Fluent ADO.NET 개요

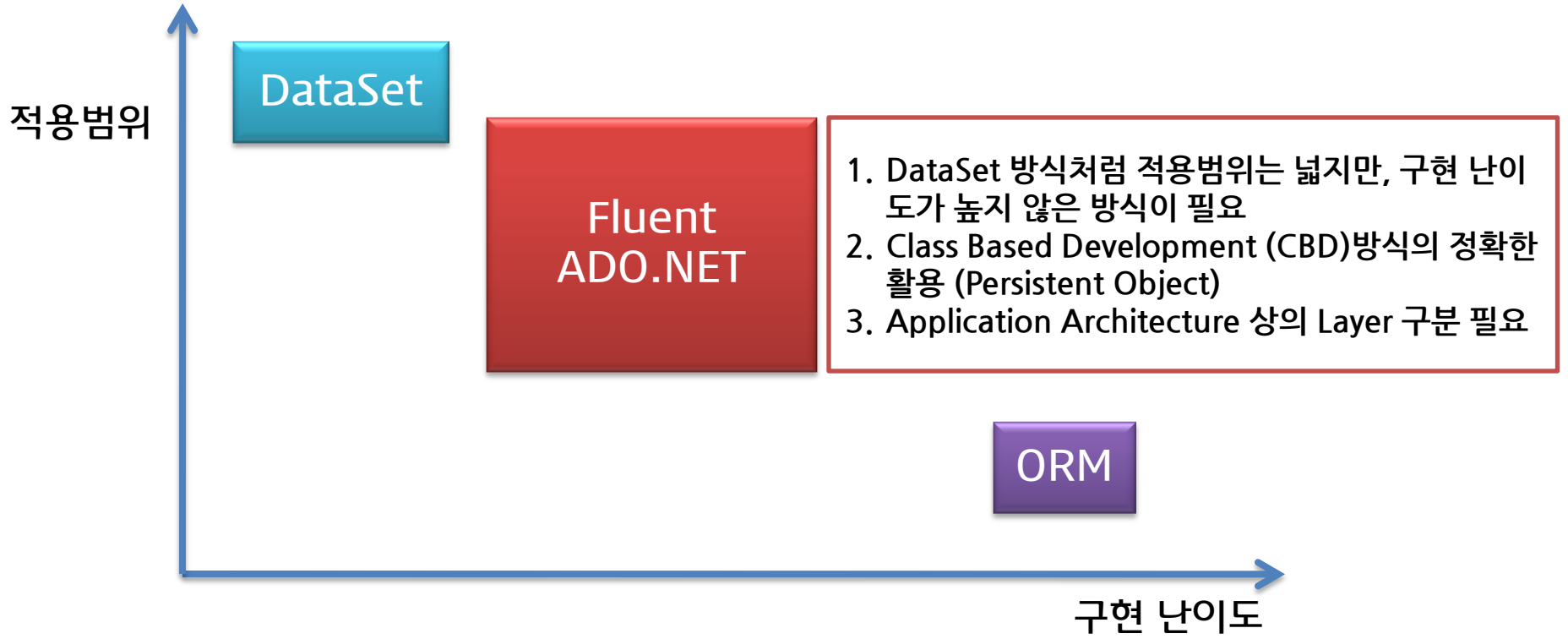
- Data Object 와 Database 간의 차이를 명시적인 코딩이 아닌, Rule에 의한 방식을 통해 Mapping을 자동화하는 것을 말함.

- Database 의 Entity 정보 (Table, View, Procedure의 결과 셋 정보)를 Neutral한 DataSet, DataTable, IDataReader 가 아닌 Class의 Instance (Object)로 자동 변환
  - 예 : IList<User> LoadUsers(), Process LoadProcess(int processId)

- Object를 Database에 적용하기 위해, 자동 변환을 수행
  - 예 : InsertUser, UpdateUser …

2. Fleunt ADO.NET 필요성

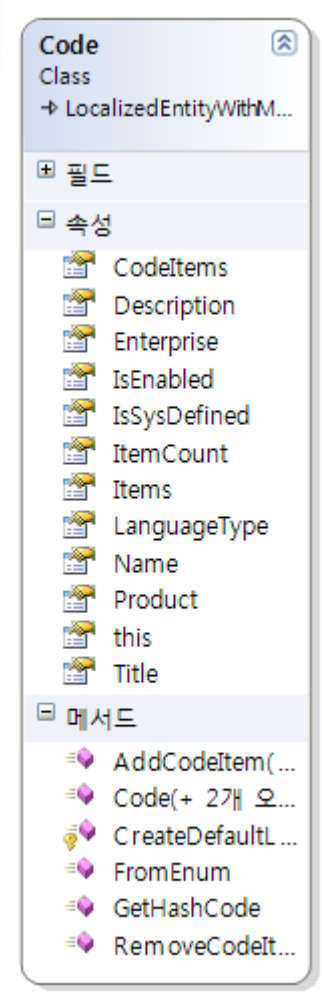
ORM 은 적용 범위가 제한적이다. (SI 성 Project 및 다양한 표현 Data에는 효율성이 떨어진다)
DataSet Binding 방식은 Business Data의 형 안정성을 떨어뜨린다. Data 기반이 아닌 Document 기반이므로, Validation 등 Business Logic 쪽의 확장된 기능을 사용할 수 없다.



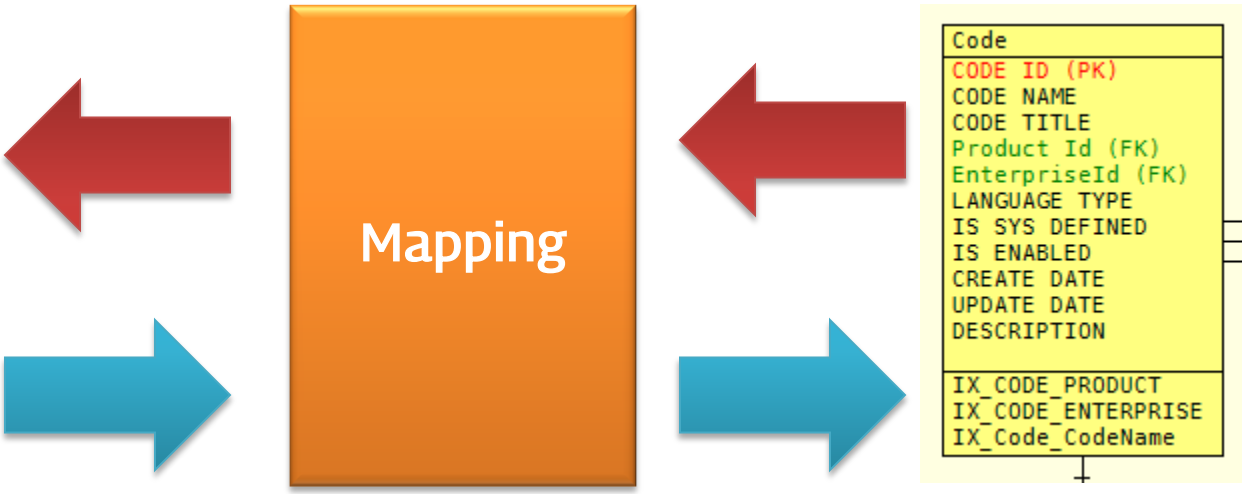
2. Fleunt ADO.NET 필요성

Mapping 방식

1. ORM Tools
   1. XML Mapping File 사용(NHibernate 등)
   2. .NET Attribute 사용 (ActiveRecord, iBATIS 등)
2. General Methods
   1. Code Generation (Modeling Tool or Scripts) (LINQ To SQL, CodeGenerator)
   2. **Simple Rule Expression**
   3. **Name Mapper 구현**

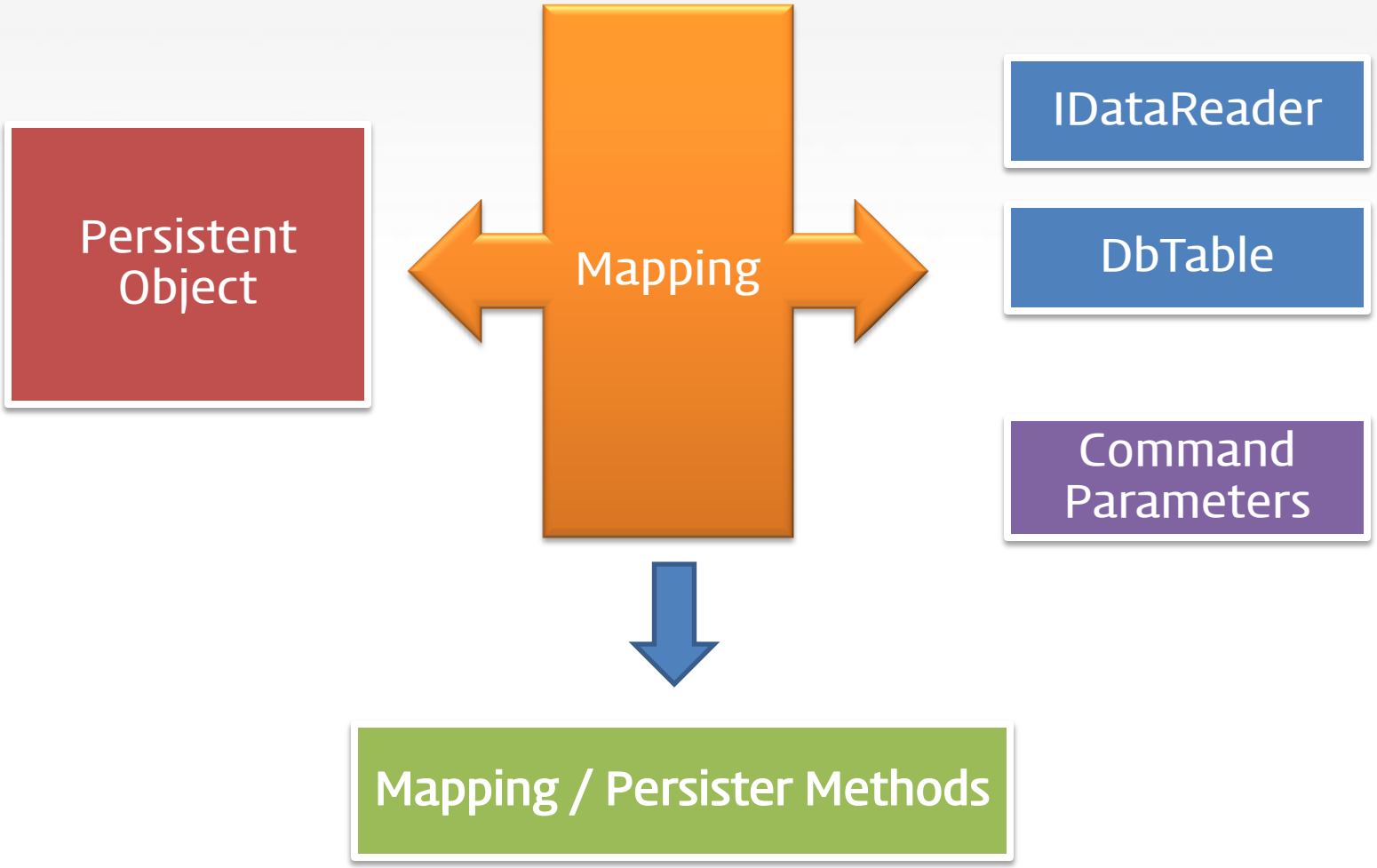


Object



Database Entity

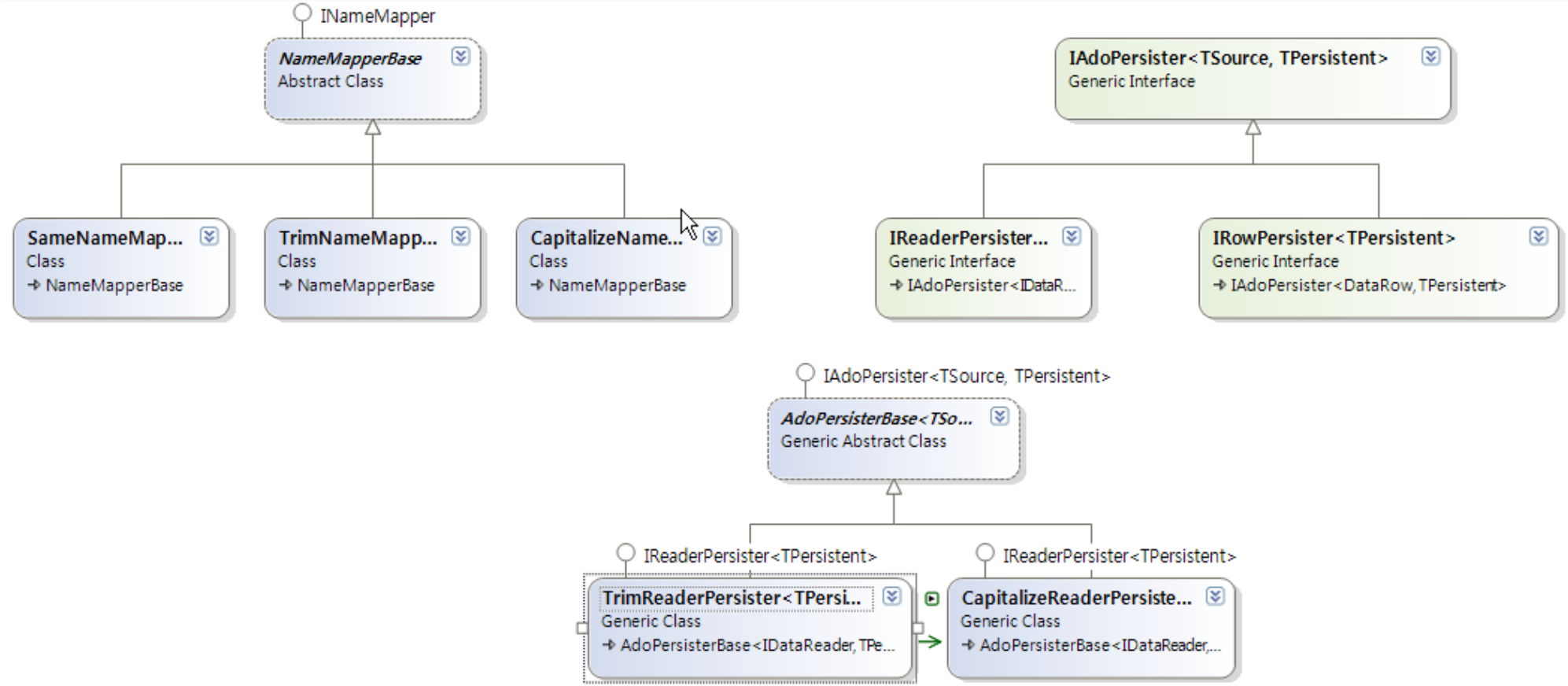
3. Fluent ADO.NET 구조



3.1 Mapping Methods

Methods	설명
INameMapper	Column name : Property name mapping 규칙을 가진 class 를 구현 또한 환경설정을 통해 특정 INameMapper 구현 Class를 지정할 수 있어서 유연한 시스템을 만들 수 있다.
INameMap	Column name : Property name mapping 정보를 가진 IDictionary 를 제공 (저장소에서 얻을 수 있다. - DB, 파일, Cache)
IAdoPersister	IDataReader, DbTable로부터 Schema 정보를 읽어, 직접 Persistent Object를 생성할 수 있다. Nhibernate DynamicTransform처럼 생성자에 모든 속성정보를 전달하게 되면, 생성자 호출->속성 값 설정 과정을 거치는 위의 두 방법보다 빠르다. 또한 환경설정을 통해 특정 IAdoPersister 구현 Class를 지정할 수 있어서 유연한 시스템을 만들 수 있다.
Converter<TIn, TOut>	Converter<IDataReader, User> 처럼 IPersister와 같은 방법이기는 하나 anonymous method 를 쓸 수 있어서 IPersister 구현을 할 필요가 없다. 코드상에서 간단히 Test나 변형된 작업 시에 구현 편의성을 제공한다.

3. Fluent ADO.NET – Class Diagram



3.1 Mapping Methods

INameMapper

```
264         [RowTest]
265         [Row("ANATR", 0, 0)]
266         [Row("ANATR", 1, 5)]
267         [Row("ANATR", 0, 0)]
268         [Row("ANATR", 5, 100)]
269         public void ConvertAllFromDataReaderWithPaging(string customerId, int firstResult, int maxResults)
270         {
271             using (AdoDataReader reader = GetCustomerOrderHistoryDataReader(customerId).ToAdoDataReader())
272             {
273                 var orderHistories = AdoUtils.ConvertAll<CustomerOrderHistory>(reader, CapitalizeMapper, firstResult, maxResults);
274
275                 if (maxResults > 0)
276                     Assert.IsTrue(orderHistories.Count <= maxResults);
277
278                 if (log.IsInfoEnabled)
279                     log.Info(orderHistories.CollectionToString());
280             }
281         }
```

IAdoPersister

```
224         [RowTest]
225         [Row(0, 0)]
226         [Row(0, 5)]
227         [Row(5, 10)]
228         public void ConvertAllFromDataReaderByPersister(int firstResult, int maxResults)
229         {
230             IReaderPersister<CustomerOrderHistory> readerPersister = new CapitalizeReaderPersister<CustomerOrderHistory>();
231
232             using(IDataReader reader = NorthwindAdoRepository.ExecuteReaderByProcedure("CustOrderHist2", base.CustomerTestParameter))
233             {
234                 var orderHistories = AdoUtils.ConvertAll(reader, readerPersister, firstResult, maxResults);
235                 Assert.IsTrue(orderHistories.Count > 0);
236             }
237         }
```

3.2 Name Mapping Rules

Mapping Rules

Rule	설명	예제	
		Property	Column
NoChange	컬럼명 = 속성명 (MS SQL Style)	ProjectID TaskID UserName	ProcessID TaskID UserName
Trim	컬럼명의 '_', 공백을 없앤다.	ProjectId TaskId UserName OrderDetails	Project_Id Task_Id User_Name Order Details
Capitalize (Pascal Naming)	컬럼명의 '_' 와 공백을 이용 단어로 규정하고, 첫 단어만 대문자로 만든 후, 모든 공백과 '_' 를 제거한다.	ProjectId TaskId UserName OrderDetails	PROJECT_ID TASK_ID USER_NAME ORDER_DETAILS
Extentions…	???		

3. Mapping Rule 예

NameMapper Sample

```
namespace RCL.Data.Ado
{
    /// <summary>
    /// Fluent ADO.NET을 위해, DB Column 명과 Class의 속성명을 매핑시키는 클래스에 대한 인터페이스
    /// </summary>
    public interface INameMapper
    {
        /// <summary>
        /// 컬럼명을 속성명으로 맵핑시킨다.
        /// </summary>
        /// <param name="columnName"></param>
        /// <returns></returns>
        string MapToPropertyName(string columnName);
    }
}
```

```
namespace RCL.Data.Ado.NameMappers
{
    /// <summary>
    /// 컬럼명에서 '_', Space 를 제외하고, Pascal 명명법에 따라 단어를 대문자로 사용한다.
    /// </summary>
    public class CapitalizeNameMapper : NameMapperBase
    {
        /// <summary>
        /// 컬럼명에서 '_', Space 를 제외하고, Pascal 명명법에 따라 단어를 대문자로 사용한다.
        /// </summary>
        /// <param name="columnName"></param>
        /// <returns></returns>
        /// <example>
        /// <code>
        ///     CapitalizeNameMapper cMapper = new CapitalizeNameMapper();
        ///     string propertyName = cMapper.MapToPropertyName("PROJECT_NAME");    // property name is ProjectName
        ///     propertyName = cMapper.MapToPropertyName("PROJECT_ID");             // property name is ProjectId
        /// </code>
        /// </example>
        public override string MapToPropertyName(string columnName)
        {
            return columnName.Capitalize().DeleteCharAny('_', ' ');
        }
    }
}
```

Realweb

4. Fluent ADO.NET 예제

```
328         /// <summary> ...
331         [Test]
332         public void DatabaseToPersistentObject()
333         {
334             // CustOrderHist2 는 컬럼명만 PROJECT_NAME, TOTAL 로 변경한 것이다.
335             using (AdoDataReader reader = NorthwindAdoRepository.ExecuteReaderByProcedure("CustOrderHist2", base.CustomerTestParameter).ToAdoDataReader())
336             {
337                 IList<CustomerOrderHistory> orderHistories =
338                     AdoUtils.ConvertAll<CustomerOrderHistory>(reader,
339                                                               NameMappingUtil.Mapping(reader,
340                                                                                       NameMappingUtil.CapitalizeMappingFunc('_', ' ')));
341
342                 Assert.IsTrue(orderHistories.Count > 0);
343                 Console.WriteLine("Order History: " + orderHistories.CollectionToString());
344             }
345         }
346
347         [Test]
348         public void PersistentObjectToDatabase()
349         {
350             Category category = new Category { CategoryName = "Test", Description = "FluentUtil" };
351
352             // delete exist category
353             NorthwindAdoRepository.ExecuteNonQueryBySqlString(
354                 string.Format("DELETE FROM Categories where CategoryName = {0}",
355                               category.CategoryName.QuotedStr()));
356
357             // insert
358             object result = NorthwindAdoRepository.ExecuteEntity("SaveOrUpdateCategory", category, CapitalizeMapper);
359
360             category.CategoryId = RwConvert.DefValue(result, -1);
361             Assert.IsTrue(category.CategoryId > -1);
362
363
364             // update
365             result = NorthwindAdoRepository.ExecuteEntity("SaveOrUpdateCategory", category, CapitalizeMapper);
366             Assert.IsTrue((int)result > 0);
367         }
```

Load / Save Persistent Objects

4. Fluent ADO.NET 예제

```
53          /// <summary>
54          /// 지정한 Entity의 Parameter 세팅없이 NameMapping으로 저장한다.
55          /// </summary>
56          [Test]
57          public void SetParameterValuesGeneric()
58          {
59              Category category = new Category { CategoryName = "Test", Description = "FluentUtil" };
60
61              // delete exist category
62              NorthwindAdoRepository.ExecuteNonQueryBySqlString(
63                  string.Format("DELETE FROM Categories where CategoryName = {0}",
64                                category.CategoryName.QuotedStr()));
65
66              // insert
67              using (DbCommand command = NorthwindAdoRepository.GetProcedureCommand("SaveOrUpdateCategory", true))
68              {
69                  AdoUtils.SetParameterValues(NorthwindAdoRepository.Db,
70                                              command,
71                                              category,
72                                              NameMappingUtil.Mapping(command,
73                                                                      NameMappingUtil.CapitalizeMappingFunc('_', ' ')));
74                  int id = RwConvert.DefValue(NorthwindAdoRepository.ExecuteCommand(command), -1);
75
76                  category.CategoryId = id;
77
78                  Assert.IsTrue(category.CategoryId != -1);
79              }
80
81              // update
82              using (DbCommand command = NorthwindAdoRepository.GetProcedureCommand("SaveOrUpdateCategory", true))
83              {
84                  AdoUtils.SetParameterValues(NorthwindAdoRepository.Db,
85                                              command,
86                                              category,
87                                              NameMappingUtil.Mapping(command,
88                                                                      NameMappingUtil.CapitalizeMappingFunc('_', ' ')));
89                  int id = RwConvert.DefValue(NorthwindAdoRepository.ExecuteCommand(command), -1);
90
91                  category.CategoryId = id;
92
93                  Assert.IsTrue(category.CategoryId != -1);
94              }
95          }
```

Save
Persistent Object
Without
Parameter settings

5. Fluent ADO.NET 활용 절차

Mapping 방식 선택	Database 설계 Table Column Procedure Parameters	Class 설계 Property
1. Same 2. Trim 3. Capitalize	1. UserName 2. User Name 3. USER_NAME	1. UserName 2. UserName 3. UserName

6. Fluent ADO.NET 제한 사항

- ORM이 아니기 때문에 Class 간의 Association은 지원 안함.
- Column : Property의 Name Mapping만을 지원하며, 기본적으로 Type conversion은 지원하지 않음. (IAdoPersister, Converter를 이용해야 함)
- Persistent Object Build시에 Paging을 지원하나, count/select 처럼 구분된 동작보다는 성능이 느림. (이 문제는 DataSet 이용보다는 빠름)
- Name Mapping 규칙이 한 제품에서는 통일되어야 함.
  (단 IoC를 이용해서 복합 제품에서 선택적으로 Mapping 방식을 이용할 수 있도록 되어 있음)

7. Persister 구현 Logic

1. DataSource 및 Name Mapping 정보를 얻는다.
2. 반환할 Persistent Object를 생성한다.
3. DataSource (IDataReader) 로부터 컬럼 정보를 얻는다.
4. Reflection을 통해 Persistent Object의 속성명을 얻는다.
5. Name Mapping에 근거 하여 DataSource로부터 값을 얻어, 대상이 되는 Persistent Object에 값을 설정한다.
6. 복수의 Persistent Object 라면, 2 ~ 5를 반복한다.

7. Persister 구현 Logic

```
335         /// <summary>
336         /// DataReader의 모든 레코드를 지정된 형식의 인스턴스로 빌드한다.
337         /// </summary>
338         /// <typeparam name="T">type of persistent object</typeparam>
339         /// <param name="instances">Collection of Persistent object</param>
340         /// <param name="dr">data source</param>
341         /// <param name="nameMaps">컬럼명 : 속성명의 Name Mapping 정보</param>
342         public static void FillPersistences<T>(IList<T> instances, IDataReader dr, INameMap nameMaps) where T : new()
343         {
344             if (log.IsDebugEnabled)
345                 log.DebugFormat("DataReader 를 읽어 {0} 인스턴스를 빌드합니다. 속성 매핑: {1}",
346                     typeof(T).FullName, nameMaps.CollectionToString());
347
348             AdoDataReader reader = dr.ToAdoDataReader();
349             IDynamicAccessor<T> accessor = DynamicAccessorFactory.CreateDynamicAccessor<T>();
350
351             int num = 0;
352             while (dr.Read())
353             {
354                 var instance = Activator.CreateInstance<T>();
355
356                 // 인스턴스의 속성값을 DataReader의 컬럼 값으로 설정한다.
357                 foreach (var columnName in nameMaps.Keys)
358                     accessor.SetPropertyValue(instance, nameMaps[columnName], reader.GetValue(columnName));
359
360                 instances.Add(instance);
361                 num++;
362             }
363
364             if (log.IsDebugEnabled)
365                 log.DebugFormat("DataReader를 읽어 {0} 의 인스턴스를 {1}개 빌드했습니다.", typeof(T).FullName, num);
366         }
```

감사합니다

Realweb