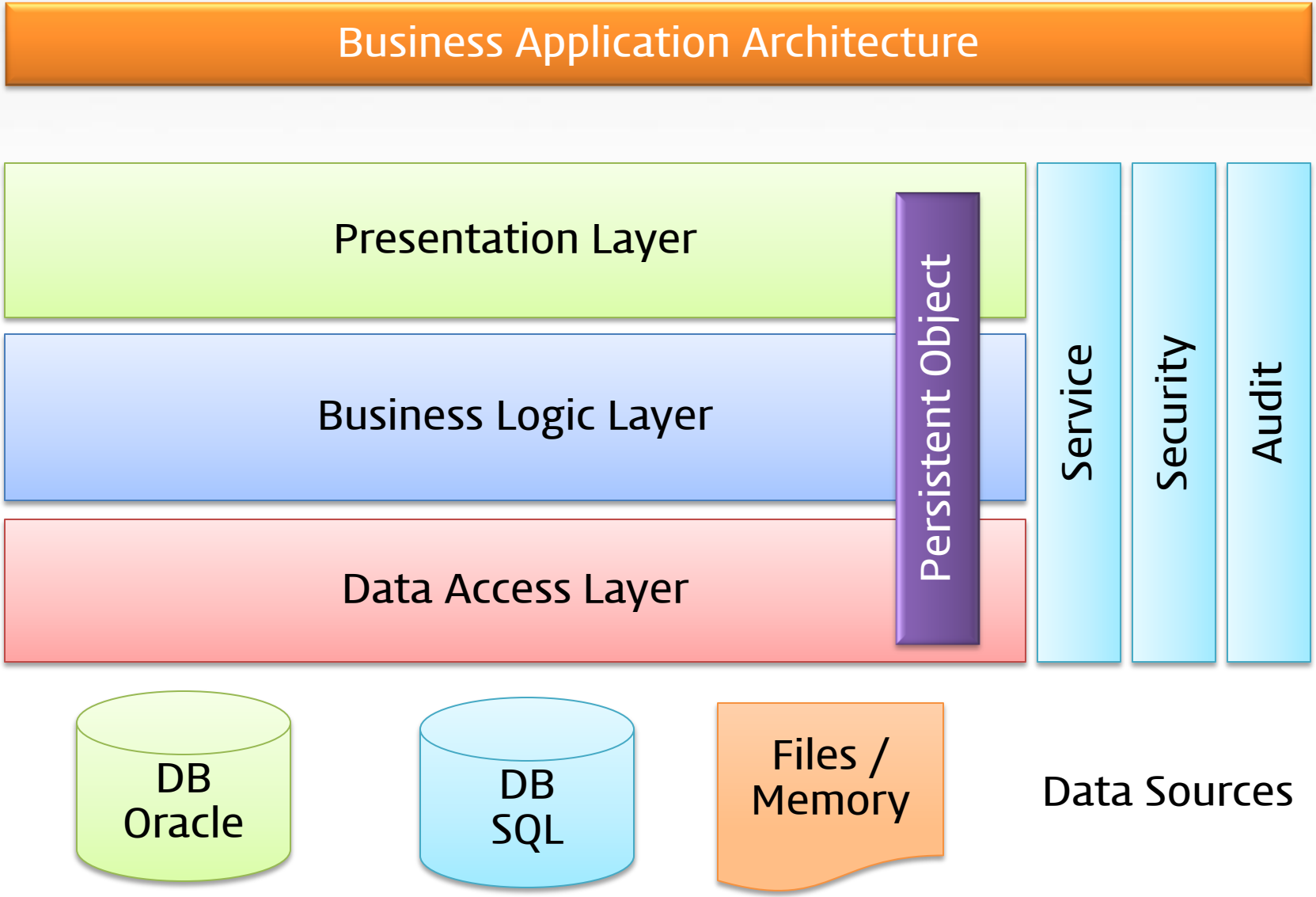
RCL.Data Repository Pattern
AdoRepository

2009.01

Realweb

목차

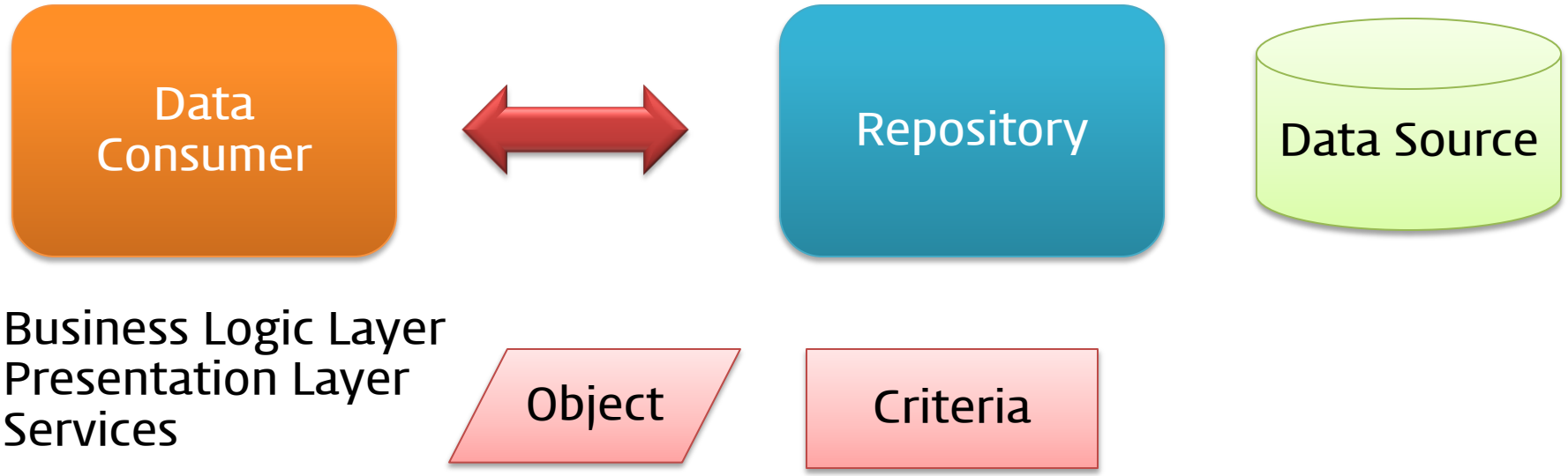
1. Repository Patten 소개



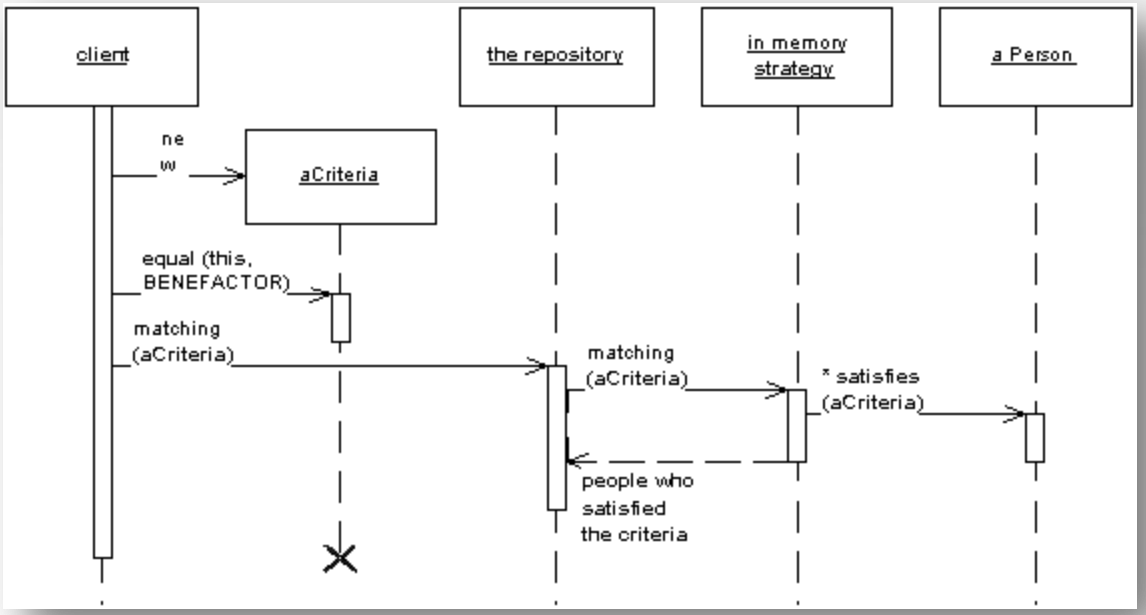
1. Repository Patten 소개

복잡한 도메인 모델로 이루어진 시스템은 Layer 개념 도입을 하면, 많은 효과를 볼 수 있다. 특히 많은 Persistent Object (Domain Object)를 다루는 시스템에서는 object를 얻기 위한 질의(query)가 집중되어 있는 mapping layer 를 따로 두면, 질의 로직 (query logic)을 재활용할 수 있고, 집중관리 할 수 있다.
Data 저장소로부터 원하는 정보를 Domain object 로 제공하고, domain object를 data 저장소에 저장하는 기능을 제공하는 Layer 를 구현하는 것을 Repository Pattern 이라 한다.

Repository Pattern 은 Domain Driven Design (DDD) 에서 가장 중추적인 Pattern 이다.



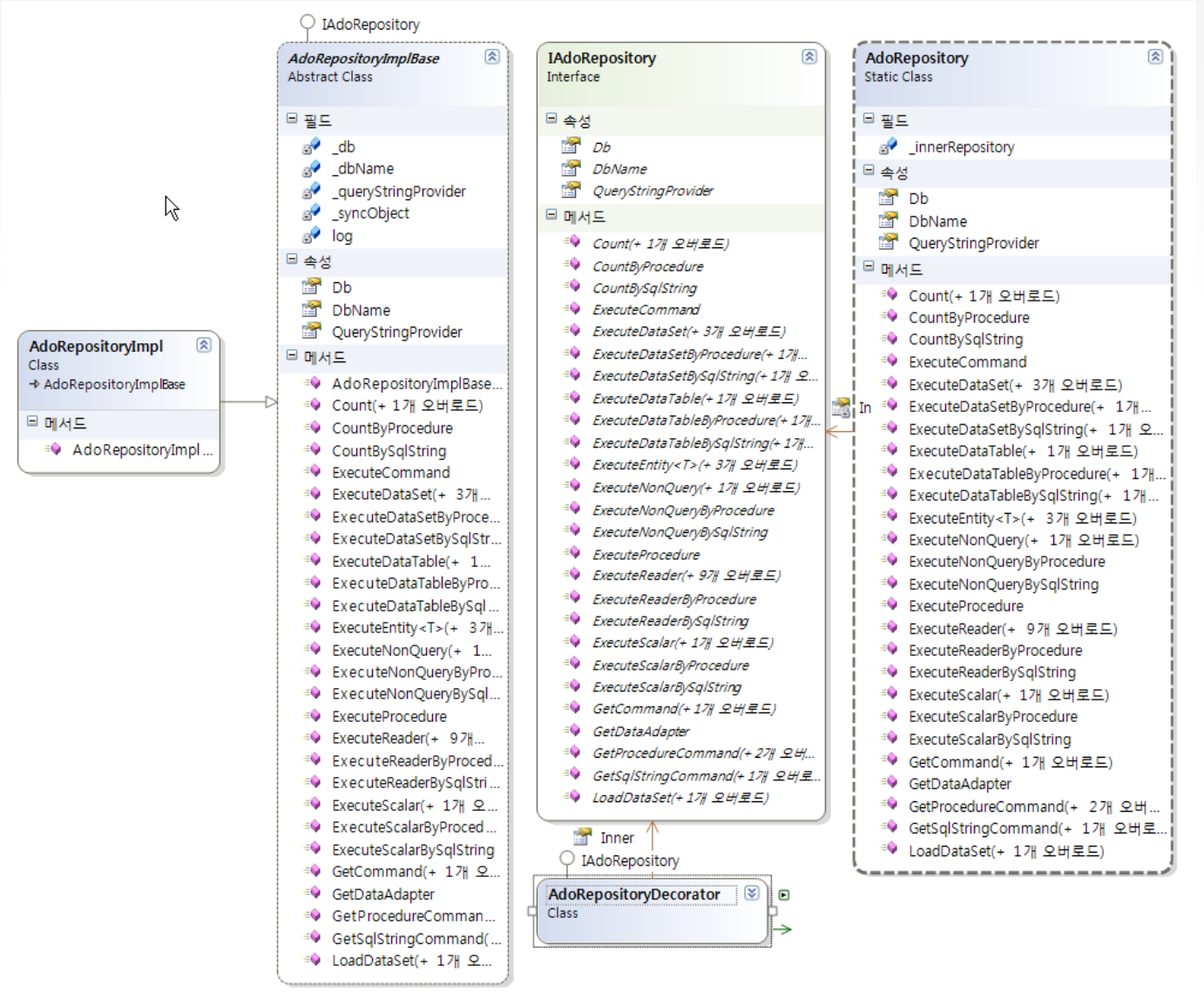
1. Repository 소개



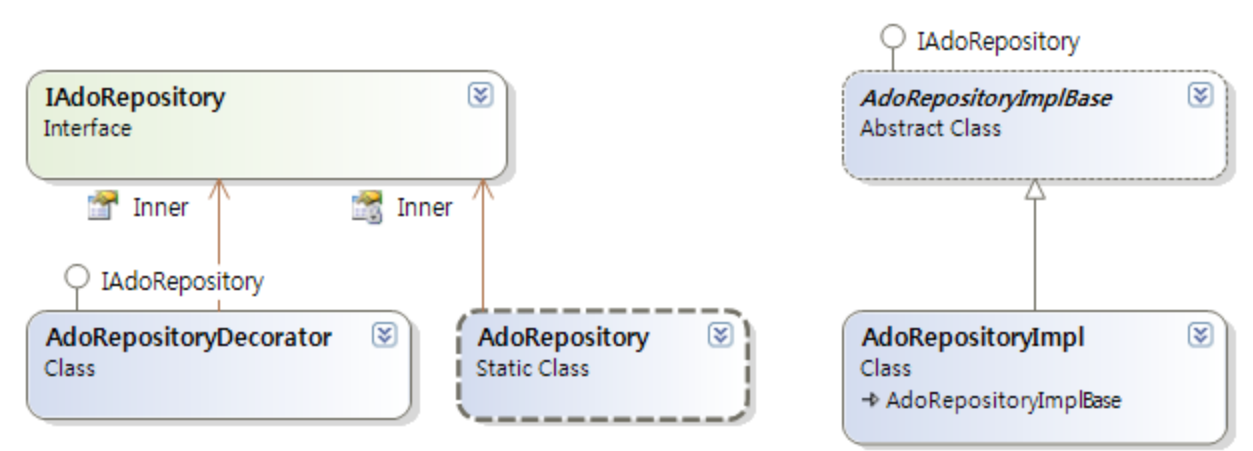
A system with a complex domain model often benefits from a layer, such as the one provided by Data Mapper (165), that isolates domain objects from details of the database access code. In such systems it can be worthwhile to build another layer of abstraction over the mapping layer where query construction code is concentrated. This becomes more important when there are a large num ber of domain classes or heavy querying. In these cases particularly, adding this layer helps minimize duplicate query logic.

A Repository mediates between the domain and data mapping layers, acting like an in-memory domain object collection. Client ob jects construct query specifications declaratively and submit them to Repository for satisfaction. Objects can be added to and remo ved from the Repository, as they can from a simple collection of objects, and the mapping code encapsulated by the Repository wil l carry out the appropriate operations behind the scenes. Conceptually, a Repository encapsulates the set of objects persisted in a d ata store and the operations performed over them, providing a more object-oriented view of the persistence layer. Repository also supports the objective of achieving a clean separation and one-way dependency between the domain and data mapping layers.

2. AdoRepository Classe Diagram



2. AdoRepository Class Diagram

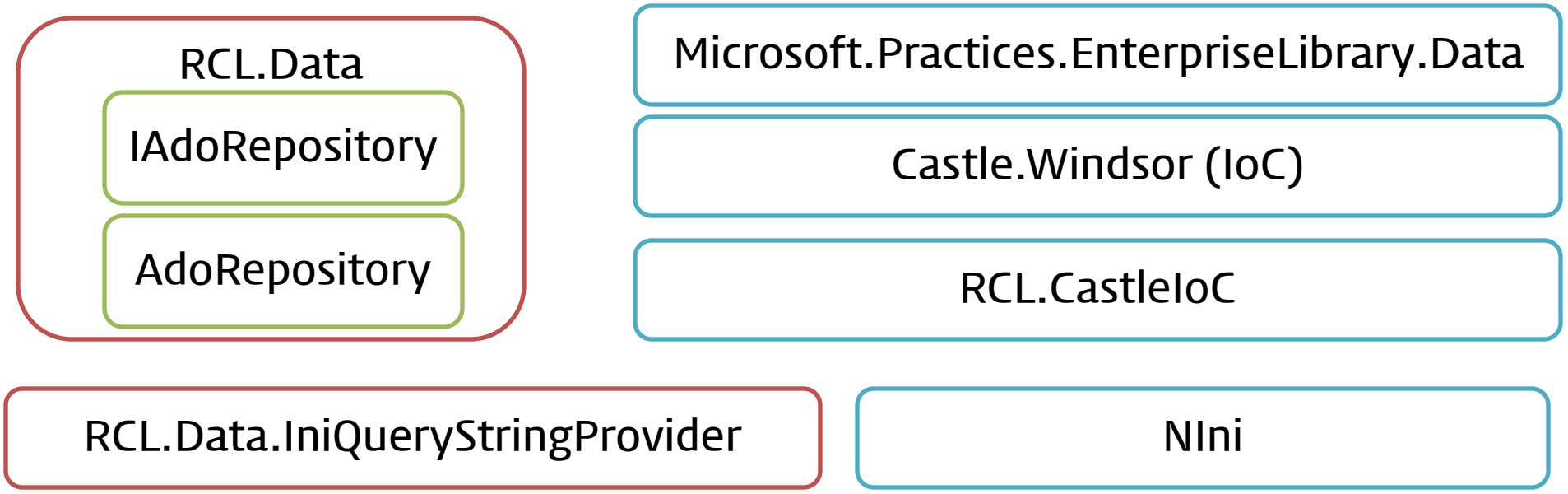


Classes	설명
IAdoRepository	ADO.NET을 이용한 기본적인 Repository 기능을 정의한 Interface
AdoRepositoryImplBase	IAdoReposiotry를 구현한 추상화 클래스 (모든 메소드 및 속성은 virtual 이다.)
AdoRepository	IoC 를 통해 실제 IAdoRepository를 구현한 클래스를 이용하여, 작업을 대신 해주는 Helper class이다.
AdoRepositoryDecorator	Repository에 대해 Decorator patten을 적용할 수 있도록 해주는 기본 class이다. IoC를 이용하면, 유연한 decorator pattern을 환경설정에서 지정할 수 있고, 이를 통해 확장성을 확보할 수 있다.
AdoUtils	IAdoRepository 구현 시, 공통된 작업을 구현한 Helper class

3. AdoRepository 구성

RCL.Data.Ado.AdoRepository는 기본적으로 MS Ent Lib와 Castle IoC를 이용합니다.
MS EntLib는 ADO.NET을 직접 이용하는 것보다 DB의 종속적이지 않도록 Method를 만들 수 있고,
Castle IoC는 환경설정에 따라 여러가지 IAdoRepository 구현 Class를 변경할 수 있는 확장성을 제공합니다.

또한 Query 문장 또는 Procedure Name등의 변경을 환경설정에서 가능토록 하기 위해 Nini을 이용한 RCL.Data.IniQueryStringProvider를 제공합니다. 이는 Query 문의 집중관리 및 DB에 따른 변경을 지원해 시스템의 확장성을 보장합니다.



3. AdoRepository 구성 – Execute DataSet

```
104         /// <summary> ...
112         void LoadDataSet(DbDataAdapter da, string tableName, DataSet targetDataSet, int firstResult, int maxResults);
113
114         /// <summary> ...
120         void LoadDataSet(DbDataAdapter da, string tableName, DataSet targetDataSet);
```

```
125
126         /// <summary> ...
134         DataSet ExecuteDataSet(DbDataAdapter da, string tableName, int firstResult, int maxResults);
135         /// <summary> ...
141         DataSet ExecuteDataSet(DbDataAdapter da, string tableName);
142         /// <summary> ...
150         DataSet ExecuteDataSet(DbCommand cmd, int firstResult, int maxResults, params IAdoParameter[] parameters);
151         /// <summary> ...
157         DataSet ExecuteDataSet(DbCommand cmd, params IAdoParameter[] parameters);
158         /// <summary> ...
166         DataSet ExecuteDataSetBySqlString(string sqlString, int firstResult, int maxResults, params IAdoParameter[] parameters);
167         /// <summary> ...
173         DataSet ExecuteDataSetBySqlString(string sqlString, params IAdoParameter[] parameters);
174         /// <summary> ...
182         DataSet ExecuteDataSetByProcedure(string spName, int firstResult, int maxResults, params IAdoParameter[] parameters);
183         /// <summary> ...
189         DataSet ExecuteDataSetByProcedure(string spName, params IAdoParameter[] parameters);
```

```
195         /// <summary> ...
203         DataTable ExecuteDataTable(DbCommand cmd, int firstResult, int maxResults, params IAdoParameter[] parameters);
204         /// <summary> ...
210         DataTable ExecuteDataTable(DbCommand cmd, params IAdoParameter[] parameters);
211         /// <summary> ...
219         DataTable ExecuteDataTableBySqlString(string sqlString, int firstResult, int maxResults, params IAdoParameter[] parameters);
220         /// <summary> ...
226         DataTable ExecuteDataTableBySqlString(string sqlString, params IAdoParameter[] parameters);
227         /// <summary> ...
235         DataTable ExecuteDataTableByProcedure(string spName, int firstResult, int maxResults, params IAdoParameter[] parameters);
236         /// <summary> ...
242         DataTable ExecuteDataTableByProcedure(string spName, params IAdoParameter[] parameters);
```

3. AdoRepository 구성 – Execute Reader

```
248         /// <summary> ...
254         IDataReader ExecuteReader(DbCommand cmd, params IAdoParameter[] parameters);
255         /// <summary> ...
261         IDataReader ExecuteReader(string query, params IAdoParameter[] parameters);
262         /// <summary> ...
268         IDataReader ExecuteReaderBySqlString(string sqlString, params IAdoParameter[] parameters);
269         /// <summary> ...
275         IDataReader ExecuteReaderByProcedure(string spName, params IAdoParameter[] parameters);
```

```
281         /// <summary> ...
287         int Count(DbCommand cmd, params IAdoParameter[] parameters);
288         /// <summary> ...
294         int Count(string query, params IAdoParameter[] parameters);
295         /// <summary> ...
301         int CountBySqlString(string sqlString, params IAdoParameter[] parameters);
302         /// <summary> ...
308         int CountByProcedure(string spName, params IAdoParameter[] parameters);
```

Count Method는 Paging 을 위해 필요한데, 주어진 Command 를 실행하여 얻은 IDataReader의 Record Count를 반환한다.
실제 SQL 문장의 count(column) 함수보다는 느리지만, DataSet / DataTable 을 이용한 Paging 처리시에는 모든 데이터를 메모리에 적재해야 하는 반면, Count 함수는 Record 개수만을 계산하는 기능만 있으므로 훨씬 빠르다.
장점은 count를 위한 별도의 쿼리 문장이 필요없다는 것이다.

3. AdoRepository 구성 – Execute Reader

ADO.NET 의 ExecuteReader 메소드를 통해 얻은 IDataReader 로부터 Persistent Object 를 빌드한다.
자세한 내용은 Fluent ADO.NET을 참고.

```
312  #region << DataReader To PersistentObject >>
313
314  ///<summary> ...
322  IList<T> ExecuteReader<T>(INameMapper nameMapper, DbCommand cmd, params IAdoParameter[] parameters) where T : new();
323  /// <summary> ...
331  IList<T> ExecuteReader<T>(INameMap nameMaps, DbCommand cmd, params IAdoParameter[] parameters) where T : new();
332  /// <summary> ...
340  IList<T> ExecuteReader<T>(IReaderPersister<T> persister, DbCommand cmd, params IAdoParameter[] parameters) where T : new();
341  /// <summary> ...
349  IList<T> ExecuteReader<T>(Converter<IDataReader, T> converter, DbCommand cmd,
350                                      params IAdoParameter[] parameters) where T : new();
351  #endregion
352
353  #region << DataReader To Persistent Object with Paging >>
354
355  /// <summary> ...
365  IPagingList<T> ExecuteReader<T>(INameMapper nameMapper, DbCommand cmd, int pageIndex, int pageSize, params IAdoParameter[] parameters) where
366  /// <summary> ...
376  IPagingList<T> ExecuteReader<T>(INameMap nameMaps, DbCommand cmd, int pageIndex, int pageSize, params IAdoParameter[] parameters) where T :
377  /// <summary> ...
387  IPagingList<T> ExecuteReader<T>(IReaderPersister<T> persister, DbCommand cmd, int pageIndex, int pageSize, params IAdoParameter[] parameters
388  /// <summary> ...
398  IPagingList<T> ExecuteReader<T>(Converter<IDataReader, T> converter, DbCommand cmd, int pageIndex, int pageSize, params IAdoParameter[] para
399
400  #endregion
```

3. AdoRepository 구성 – Execute Scalar, NonQuery

```
402     #region << Execute Scalar >>
403
404     /// <summary> ...
410     object ExecuteScalar(DbCommand cmd, params IAdoParameter[] parameters);
411     /// <summary> ...
417     object ExecuteScalar(string query, params IAdoParameter[] parameters);
418     /// <summary> ...
424     object ExecuteScalarBySqlString(string sqlString, params IAdoParameter[] parameters);
425     /// <summary> ...
431     object ExecuteScalarByProcedure(string spName, params IAdoParameter[] parameters);
432
433     #endregion
434
435     #region << Execute Non Query >>
436
437     /// <summary> ...
443     int ExecuteNonQuery(DbCommand cmd, params IAdoParameter[] parameters);
444     /// <summary> ...
450     int ExecuteNonQuery(string query, params IAdoParameter[] parameters);
451     /// <summary> ...
457     int ExecuteNonQueryBySqlString(string sqlString, params IAdoParameter[] parameters);
458     /// <summary> ...
464     int ExecuteNonQueryByProcedure(string spName, params IAdoParameter[] parameters);
465
466     #endregion
```

3. AdoRepository 구성 – Execute Procedure

기존 DAAB는 DbCommand 의 Parameter 를 조작하는 경우가 많은데, 초급 개발자의 경우 이 부분에 많은 버그를 양산한다.
이를 방지하기 위해서, IAdoParameter 를 제공하여, 쉽게 Command의 Parameter값을 설정하고, 조회할 수 있도록 했다.

```
468   #region << Execute Command / Procedure >>
469
470   /// <summary>
471   /// 지정된 Command를 수행하고, RETURN_VALUE를 반환합니다.
472   /// </summary>
473   /// <param name="cmd">DbCommand 인스턴스</param>
474   /// <param name="parameters">인자</param>
475   /// <returns>Procedure인 경우 return value를 반환한다. 반환값이 없으면 0을 반환한다.</returns>
476   object ExecuteCommand(DbCommand cmd, params IAdoParameter[] parameters);
477
478   /// <summary>
479   /// Stored Procedure를 실행하고, Parameter의 Direction이 INPUT이 아닌 Parameter들을 반환한다. (InputOutput, Output, ReturnValue)
480   /// </summary>
481   /// <param name="spName">실행할 Procedure 이름</param>
482   /// <param name="parameters">인자</param>
483   /// <returns>INPUT을 제외한 Oupput, InputOutput, ReturnValue에 해당하는 Parameter 값을 반환한다.</returns>
484   IAdoParameter[] ExecuteProcedure(string spName, params IAdoParameter[] parameters);
485
486   #endregion
```

3. AdoRepository 구성 – Execute Entity

Domain Driven Development (쉽게 DataSet 같은 것을 쓰지 않고, Persistent Object - Domain Object - 를 사용하는 패턴) 에서는 Persistent Object의 Create, Update, Delete를 쉽게 할 수 있도록 제공하는 메소드입니다.

일반적으로 SaveUser(string userid, string userName, string email, …) 를 많이 사용하는데, ExecuteEntity<User>(command, user, …)를 사용할 수 있도록 제공되는 메소드입니다.

```
488         #region << Execute Entity >>
489
490         /// <summary>
491         /// 지정된 Entity의 속성 값을 이용하여 Command의 Parameter 값을 설정하고, 실행시킨다.
492         /// 일반적으로 Save / Update시에 활용하면 좋다.
493         /// </summary>
494         /// <typeparam name="T">Persistent object 수형</typeparam>
495         /// <param name="cmd">수행할 Command 객체</param>
496         /// <param name="entity">처리할 Persistent object</param>
497         /// <param name="nameMaps">ParameterName of Procedure = Property Name of Persistence object 매핑 정보</param>
498         /// <returns>Command 의 RETURN VALUE</returns>
499         object ExecuteEntity<T>(DbCommand cmd, T entity, INameMap nameMaps);
500         /// <summary> ...
509         object ExecuteEntity<T>(string spName, T entity, INameMap nameMaps);
510         /// <summary>
511         /// 지정된 Procedure를 수행한다. 인자로 entity의 속성값을 이용한다.
512         /// </summary>
513         /// <typeparam name="T"></typeparam>
514         /// <param name="cmd">실행할 DbCommand</param>
515         /// <param name="entity">실행할 Entity</param>
516         /// <param name="nameMapper">Name Mapping Class</param>
517         /// <returns>Count of RowAffected</returns>
518         object ExecuteEntity<T>(DbCommand cmd, T entity, INameMapper nameMapper);
519         /// <summary> ...
527         object ExecuteEntity<T>(string spName, T entity, INameMapper nameMapper);
528
529         #endregion
```

3. AdoRepository 구성 - IQueryStringProvider

```
14          #region << Properties >>
15
16          /// <summary> ...
19          Database Db { get; }
20
21          /// <summary> ...
24          string DbName { get; }
25
26          /// <summary> ...
29          IQueryStringProvider QueryStringProvider { get; }
30
31          #endregion
```

일반적으로 시스템에서 사용하는 Query 문이나 Procedure 명을 코드상에 정의하고 사용한다. (const, static readonly 형식)
또한 검색을 위한 Query Builder 시에는 더욱 복잡한 Logic을 구현해야 하고, 유지관리가 어렵게 된다.
ORM을 사용하게 되면 Criteria를 자동 생성해주지만 (LINQ To SQL만 하더라도) 일반적으로 그렇게 사용하지 않는다.

좌측과 같이 Query 문장을 정의하는 것을 외부에 정의해 놓고, 시스템에서 불러와 사용한다면, 어느 정도 확장성을 확보 할 수 있고 (Order 변경, DB 변경, DB Schema 변경 등 - DAAB를 100%활용할 수 있는), Query 문도 집중 관리할 수 있다.

복잡한 Query 문장도 '\' 를 이용하여 Multiline으로 표현 가능하다.

```
1  ; ###############################################################################
2  ;
3  ; RCL.Data.IniQueryStringProvider 예제용
4  ;
5  ; 주의사항 :
6  ;
7  ;   1. 다른 섹션의 Key를 참조하려면 ${SECTION!KEY} 형태를 취하고,
8  ;    | 같은 섹션의 Key를 참조하려면 ${KEY} 형태를 취하면 된다.
9  ;
10 ; ###############################################################################
11
12 [Common]
13 DbName = Northwind
14
15 [Customer]
16 GetAll = SELECT * FROM Customers with (nolock)
17 GetById = ${GetAll} where CustomerId=@CustomerId
18 GetMatchCompanyName = ${GetAll} where CompanyName like @CompanyName
19
20 [Order]
21 GetAll = SELECT * FROM Orders with (nolock)
22 GetByCustomer = ${GetAll} where CustomerId=@CustomerId
23 GetByEmployee = ${GetAll} where EmployeeId=@EmployeeId
24
25 CheckTable = DBCC CheckTable( 'Orders' )
26
27 CustomerOrderHistory = CustOrderHist
28 CustomerOrdersDetail = CustOrdersDetail
29
30 [Order Details]
31 GetAll = SELECT * FROM [Order Details] with (nolock)
32 GetByOrder = ${GetAll} where OrderId = @OrderId
33
34 [Products]
35 GetAll = SELECT * FROM Products with (nolock)
36 GetMatchProductName = ${GetAll} WHERE ProductName like @ProductName
37
38 SalesByCategory = [SalesByCategory]
39 TenMostExpensiveProduct = [dbo].[Ten Most Expensive Products]
```

4. Examples - DataSet

Paging을 먼저 수행해서 DataSet 얻기.
firstResult, maxResults = (0, 0)이면 Paging 안함. (5, 0)면 처음 5개의 레코드는 제외한 모든 레코드 반환

```
42          [RowTest]
43          [Row(0, 0)]
44          [Row(5, 10)]
45          [Row(5, 0)]
46          public void ExecuteDataSet(int firstResult, int maxResults)
47          {
48              DataSet ds = null;
49
50              using (DbCommand cmd = NorthwindAdoRepository.GetCommand("SELECT * FROM [Order Details]"))
51              using (ds = NorthwindAdoRepository.ExecuteDataSet(cmd, firstResult, maxResults))
52              {
53                  Assert.AreEqual(ds.Tables.Count, 1);
54                  Assert.Greater(ds.Tables[0].Rows.Count, 1);
55
56                  Console.WriteLine("RowCount: " + ds.Tables[0].Rows.Count);
57              }
58
59
60              using (ds = NorthwindAdoRepository.ExecuteDataSetBySqlString(
61                              @"SELECT * From Orders where OrderDate < @OrderDate and Freight < @Freight",
62                              firstResult,
63                              maxResults,
64                              new AdoParameter("OrderDate", DateTime.Today, DbType.DateTime, ParameterDirection.Input),
65                              new AdoParameter("Freight", 2, DbType.Int32, ParameterDirection.Input)))
66              {
67                  Assert.AreEqual(ds.Tables.Count, 1);
68                  Assert.Greater(ds.Tables[0].Rows.Count, 1);
69
70                  Console.WriteLine("RowCount: " + ds.Tables[0].Rows.Count);
71              }
72          }
```

4. Examples - DataTable

Paging을 먼저 수행해서 DataTable 얻기.

일반적인 작업 Pattern 시 - Releation 정보, Constraint 정보를 활용하지 않을 시 - DataSet 을 사용하는 것보다 DataTable을 사용하는 것이 성능상의 잇점이 더 많음.

```
110         [RowTest]
111         [Row(5, 10)]
112         [Row(0, 0)]
113         public void ExecuteDataTable(int firstResult, int maxResults)
114         {
115             using (DbCommand cmd = NorthwindAdoRepository.GetCommand("SELECT * FROM [Order Details]"))
116             using (DataTable orderDetails = NorthwindAdoRepository.ExecuteDataTable(cmd, firstResult, maxResults))
117             {
118                 Assert.GreaterEqualThan(orderDetails.Rows.Count, maxResults);
119             }
120         }
```

4. Examples - IDataReader

일반적인 IDataReader 얻기.

AdoDataReader는 IDataReader의 Wrapper class 이고, 개발자가 많이 실수하는 형변환, IsNull 검사 등을 해준다.
또한 IDataReader가 제공하지 않는 Nullable 형으로 변환도 제공해준다. (Column 이 nullable 일 경우, .NET의 Nullable<T> 로 변환해 주는 것이 좋다.
예 : DateTime? StartedDate { get; set; }

```
165         [Test]
166         public void ExecuteReaderByQuery()
167         {
168             using (IDataReader reader = NorthwindAdoRepository.ExecuteReaderBySqlString("SELECT * FROM [Order Details]"))
169             {
170                 if (log.IsDebugEnabled)
171                     log.Debug(reader.ToAdoDataReader().ToString(true));
172             }
173         }
174
175         [Test]
176         public void ExecuteReaderByProcedure()
177         {
178             using (AdoDataReader reader
179                 = NorthwindAdoRepository.ExecuteReaderByProcedure("CustOrderHist", base.CustomerTestParameter).ToAdoDataReader())
180             {
181                 if (log.IsDebugEnabled)
182                     log.Debug(reader.ToString(true));
183             }
184         }
```

4. Examples – Fluent

ResultSet을 얻는 Query 문으로 Paging 된 Persistent Object의 컬렉션을 얻는다.

Paging을 하기 위해
1. Query문을 수행해서, IDataReader로부터 Record 개수를 얻고,
2. Query문을 수행해서, IDataReader로 부터 Paging되는 Record 들만 Persistent Object로 빌드한다.

장점은 하나의 Query문으로 Paging이 가능하므로, 단순 DataSet Paging 기능보다 성능이 좋다.
단점은 Counting Query 문과 Select Query 문 두 개를 따로 지정하는 것보다는 느리다.

```
204         [RowTest]
205         [Row(0, 10)]
206         [Row(3, 20)]
207         [Row(5, 10)]
208         public void ExecuteReaderWithPaging(int pageIndex, int pageSize)
209         {
210             using(DbCommand cmd = NorthwindAdoRepository.GetCommand(OrderDetailSql))
211             {
212                 var orderDetails
213                     = NorthwindAdoRepository.ExecuteReader<OrderDetail>(TrimMapper, cmd, pageIndex, pageSize);
214
215                 Console.WriteLine(orderDetails);
216             }
217         }
```

4. Examples - Fluents

```
371         [Test]
372         public void FluentByNameMapper_Load()
373         {
374             INameMapper nameMapper = new CapitalizeNameMapper();
375
376             using (DbCommand cmd = AdoRepository.GetCommand("CustOrderHist2"))
377             {
378                 var orderHistories =
379                     AdoRepository.ExecuteReader<CustomerOrderHistory>(nameMapper,
380                                                                       cmd,
381                                                                       base.CustomerTestParameter);
382
383                 Assert.IsTrue(orderHistories.Count > 0);
384                 foreach (var order in orderHistories)
385                     Console.WriteLine("Order History: " + order.ObjectToString());
386             }
387         }
388
389         [Test]
390         public void FluentByNameMapper_Save()
391         {
392             INameMapper nameMapper = new CapitalizeNameMapper();
393
394             Category category = new Category { CategoryName = "Test", Description = "FluentUtil" };
395
396             // delete exist category
397             NorthwindAdoRepository.ExecuteNonQueryBySqlString(
398                 string.Format("DELETE FROM Categories where CategoryName = {0}",
399                               category.CategoryName.QuotedStr()));
400
401             // insert
402             object result = NorthwindAdoRepository.ExecuteEntity("SaveOrUpdateCategory", category, nameMapper);
403
404             category.CategoryId = RwConvert.DefValue(result, -1);
405             Assert.IsTrue(category.CategoryId > -1);
406
407             // update
408             result = NorthwindAdoRepository.ExecuteEntity("SaveOrUpdateCategory", category, nameMapper);
409             Assert.IsTrue((int)result > 0);
410         }
```

4. Examples - Outputs

Command 실행 후의 Parameter 정보를 얻기 위해 사용되는 함수

Return Value를 얻기 위해 MS SQL Server의 “RETURN_VALUE” 를 사용하면 안되고,
Parameter 의 Direction이 ParameterDirection.ReturnValue 인 넘을 찾아야 한다.

예)
```
var return = outputs
                    .Where(p=>p.Direction ==ParameterDirection.ReturnValue)
                    .Select(p=>p.Value)
                    .FirstOrDefault(0);
```

```
309         /// <summary>
310         /// Output Parameter, Return Value 얻기
311         /// </summary>
312         [Test]
313         public void ExecuteProcedure()
314         {
315             IAdoParameter[] outputs = NorthwindAdoRepository.ExecuteProcedure("CustOrderHist", base.CustomerTestParameter);
316
317             if (log.IsDebugEnabled)
318                 outputs.ForEach(output => log.Debug(RwReflection.ObjectToString(output)));
319
320         }
```

4. Examples – IniQueryStringProvider

```
20          [Test]
21          public void ExecuteDataSet()
22          {
23              using (DataSet ds
24                  = NorthwindAdoRepository.ExecuteDataSetBySqlString(
25                      NorthwindAdoRepository.QueryStringProvider.GetQuery("Order Details, GetAll")))
26              {
27                  if (log.IsDebugEnabled)
28                      log.Debug(ds.GetXml());
29              }
30          }
31
32          [Test]
33          public void ExecuteDataSetWithParameter()
34          {
35              using (DataSet ds = NorthwindAdoRepository.ExecuteDataSetBySqlString(
36                  NorthwindAdoRepository.QueryStringProvider.GetQuery("Order Details, GetByOrder"),
37                  base.OrderTestParameter))
38              {
39                  if (log.IsDebugEnabled)
40                      log.Debug(ds.GetXml());
41              }
42          }
```

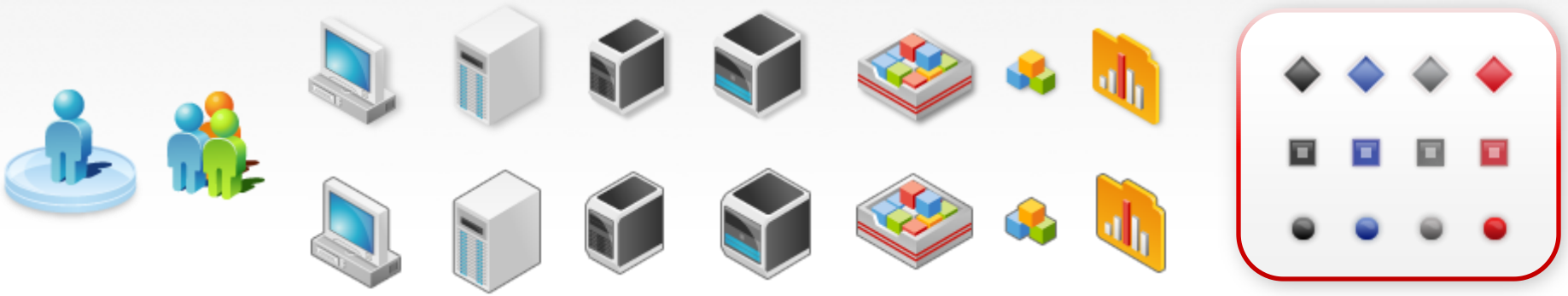
```
ryPath>
SessionFactoryName>
```

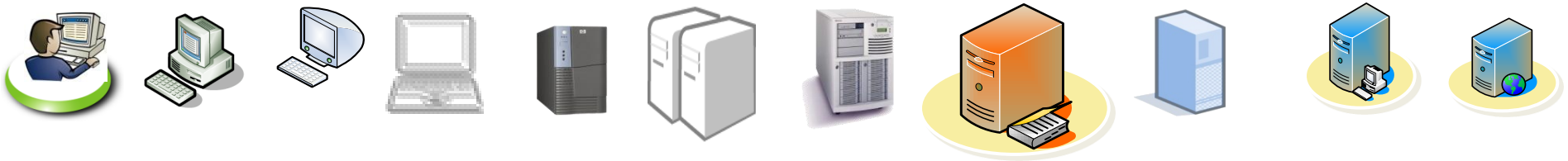
```
19              <!--<dbName>Northwind</dbName>-->
20              <QueryStringProvider>${IniAdoQueryStringProvider.Northwind}</QueryStringProvider>
21            </parameters>
22          </component>
```

```
34          <component id="IniAdoQueryStringProvider.Northwind"
35                     service="RCL.Data.IQueryStringProvider, RCL.Data"
36                     type="RCL.Data.IniQueryStringProvider.IniAdoQueryStringProvider, RCL.Data.IniQueryStringProvider">
37            <parameters>
38              <queryFilePath>.\QueryFiles\Northwind.Ado.MsSql.ini</queryFilePath>
39            </parameters>
40          </component>
41
```

2. ICON SET

Task Groups Computer Server Sub Process Bullet





감사합니다

Realweb