C# ORM with Oracle

클래스 선언으로 DATA CRUD 가능하게 하는 library

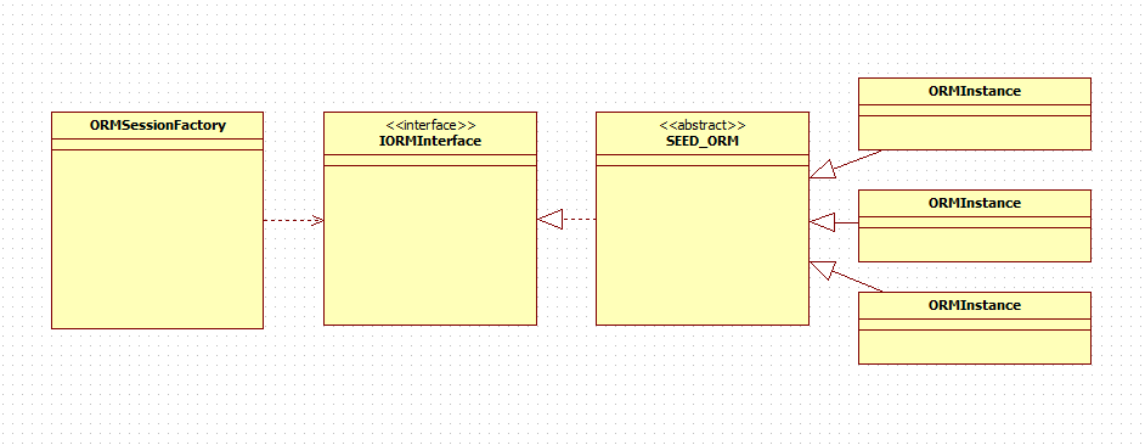
dkkim

2016-1-20

내용

1. C# Oracle ORM framework 소개

- Java spring 은 iBatis, hibernate 와 같이 ORM 기술 지원이 용이한 것과 달리 C#은 ORM 관련 기술 지원이 부족하다. Asp.net MVC 는 mssql DB 에 한하여 ORM 을 지원하고 있다. Oracle DB 에 관해서는 지원이 미약한 상황이다. C# MVC framework 를 효과적으로 사용하기 위하여 C# - Oracle ORM framework 인 ORMSessionFactory 를 개발하였다. 웹 개발뿐만 아니라 향 후 C#과 Oracle 연동에도 활용 할 수 있다.



- ORMSessionFactory 제네릭 클래스가 실질적인 ORM 역할을 수행한다. 각 ORM Instance 클래스들은 추상 클래스 SEED 를 상속 받는다. SEED 클래스는 인터페이스 클래스 IORMInterface 상속 받는다.

ORM Instance Class 들은 추상 클래스 SEED 를 상속 받음으로써 필요한 설정에 따라 Overriding 하여 선언할 수 있다. **결국 ORM instance class 는 IORMInterface 구현함으로써 ORMSessionFactory 클래스가 ORM 수행할 수 있도록 클래스 선언 하는 것이다.**

상용 ORM 프레임워크와 다른 점은 XML 에 설정하는 것이 아니라 각 instance 클래스를 선언함으로써 ORM 설정을 동시에 할 수 있다. 좀 더 객체지향적인 프로그래밍이 가능하다.

2. 개발환경

O/S	Window server
Web server	IIS 7.0
DB	Oracle 10g
Server side script	C#, Net framework 4 , .Net MVC
Client side script	HTML5, Java-script, jQuery

3. ORM framework 클래스 설명

3.1 ORM Instance Class

- DB data table 를 ORM 형식화한 Class
IORMInterface 를 상속 받아 ORMSessionFactory class 를 사용하여 ORM 구현

- Class 선언 방법

Ex) DB Table ABC 을 ORM 형식화한 ABC_ORM class

```
public class ABC_ORM: IORMInterface
{
        public string _COL1 { get; set; }
```

```
        public string _COL2 { get; set; }
        public string COL3_DATE { get; set; }
         public string COL4_INT { get; set; }

        public ABC_ORM()
        {
        }
}
```

1) IORMInterface 인터페이스 상속 받는다.

2) 멤버 변수는 실제 DB table 의 컬럼 명과 동일하게 한다.

2-1) 컬럼 명은 대문자로 지정한다.

2-2) varchar2 와 같이 문자열 데이터 타입의 컬럼 명의 멤버 변수는 _(언더바)로 시작한다.

2-3) Date Time, Number 타입의 컬럼의 멤버 변수는 _(언더바) 없이 선언한다

2-4) _(언더바)는 차이는 insert 시 **'(싱글쿼테이션)** 사용 유무이다.

_(언더바)의 사용 유무의 차이 이유는 Data insert, update 시 varchar2 타입은 반드시 싱글쿼테이션을 사용하고 Date time 와 같은 타입은 싱글쿼테이션이 필요하지 않는다. 이것을 구분하기 위해서 _(언더바)를 사용하였다.

3) IORMInterface 를 상속 받아 ORMSessionFactory 를 통해 ORM 기능을 구현할 수 있다.

3.2 IORMInterface 인터페이스 클래스

IORMInterface 인터페이스 클래스 - > 제네릭 클래스 ORMSessionFactory 사용하기 위한 인터페이스 클래스	
Type	메소드명
설명	
string	SetDBconnectionInfo ()

DB Connection String 정보 설정

string	SetPrimaryMemberVariable()

DB table 의 primary key 에 해당되는 ORM Class 의 멤버 변수명 셋업

Ex) Oracle DB 의 경우 ROWID 칼럼이 항상 unique 함으로 primary 값이 ROWID 일때

```
public string SetPrimaryMemberVariable()
{
        string sPrimariyField = "_ROWID";
        return sPrimariyField;
}
```

string	SetPrimaryValue()

DB table 의 primary key 에 해당되는 ORM Class 의 멤버변수의 값 지정

Ex) Oracle DB 의 경우 ROWID 칼럼이 항상 unique 함으로 primary 값이 ROWID 일때

```
public virtual string SetPrimaryValue()
{
        return _ROWID;
}
```

bool	isCombinedPrimaryKey()

DB table 의 primary key 값이 두개 이상의 컬럼이 조합되는 지 여부

Ex) return 값이 true 이면 primary key 값이 두 개 이상의 컬럼을 조합,

false 면 primary 값이 단일 컬럼

string	SetWhereQueryForCombinedPrimaryKey()

DB table 의 primary key 값이 두 개 이상의 컬럼일 때, upadate, delete 트랜잭션 시행할 때 사용할 Where 절 셋업

Ex) Table A 의 primary key 값의 EMP_NO, VOTE 일 경우

```
public string SetWhereQueryForCombinedPrimaryKey()
{
        string sWhere = " EMP_NO ='" + _EMP_NO + "'" + " and VOTE ='" + _VOTE + "'";
        return sWhere;
}
```

string	SetTableName()

ORM Class 의 DB table 명

ex) Table 명이 ABC 인 ORM class 일 때

```
public string SetTableName()
{
        string sTableName = "ABC";
        return sTableName;
}
```

string	SetQueryOption()

ORMSessionFactory 가 ORM class 를 Load 할 때 기본적으로 사용할 where 절 지정
Ex) 기본적으로 조회할 query 가 그룹명이 IT 일 때

```
public string SetQueryOption()
{
    string sWhere= "";
    sWhere += " where GROUP='IT' ";
    return sWhere;
}
```

string[]	SetFieldforNotEdit()

Update query 시 update 제외 시킬 칼럼
Ex) 테이블 A 의 칼럼 B,C,D 가 있을 때 D 는 update 를 제외시키려고 할 때

```
public string[] SetFieldforNotEdit()
{
        string[] arNotEditFd = { "D" };
        return arNotEditFd;
}
```

string[]	SetVirtualField()

실제 DB 에 존재 하지 않는 컬럼. View 목적이나 가상의 필드가 필요할 때
Ex) 사원 정보 테이블 USER 의 사원사진 컬럼이 필요할 때

사원 테이블 USER 는 EMP_NO, NAME 두개의 컬럼만 존재할 때, 사진 정보는 테이블 정보와 별도로 IMAGE 를 class 에 멤버로 추가한다.
그런 다음 명시적으로 SetVirtualField 함수를 통해 지정해준다.
IMAGE 칼럼은 Create, Delete, Update 트랜잭션에서 제외가 된다.

```
public string[] SetVirtualField()
    {
        string[] sVirtual = {"IMAGE"};
        return sVirtual;
    }
```

string[]	SetFieldforDatetime()

DB 테이블의 칼럼 중 Date 타입의 컬럼 지정

Date 타입은 기본적으로 yyyy-MM-dd HH24:MI:SS 형태입니다.

```
public string[]SetFieldforDatetime ()
        {
            string[] sDateTime = {"DateTime1","DateTime2"};
            return sDateTime;
        }
```

string[]	SetFieldforByteType()

DB 컬럼에 CLOB 나 BLOB 과 같이 binary 타입으로 저장되는 컬럼 변수가 있을 경우
아래와 같이 명시적으로 해당 멤버변수명을 지정한다.

Ex) BLOB 속성의 FILECONTENT 컬럼을 가지고 있는 테이블을 ORM 객체를 선언할 때

* 상기와 같은 속성이 있는 클래스를 DB insert 하거나 update 할 때는 반드시 parameterized (binding) query 를 사용해야 한다. (_가 붙은 _Create, _Edit, ... 함수를 사용할 것)

- Class 선언

```
public class CASAPPENDFILE: SEED_ORM
{
    public string _WEBREQUESTID {get;set;}
    public string _FILEID       {get;set;}
    public string _FILENAME     {get;set;}
    public string _SEQ          {get;set;}
    public string _FILETYPE     {get;set;}
    public string _FILELOCATION {get;set;}
    public string _FILEFORMAT   {get;set;}
    public byte[] _FILECONTENT { get; set; }
```

- 메소드 override

```
public override string[] SetFieldforByteType()
        {
            string[] byteFields = { "_FILECONTENT" };

            return byteFields;
        }
```

bool	BeforeAdd()

실제 DB insert 문 하기 전에 실행되는 함수

false 를 return 할 경우 DB insert 하지 않음

bool	BeforeEdit()

실제 DB update 문 하기 전에 실행되는 함수

false 를 return 할 경우 DB update 하지 않음

bool	BeforeLoad(string inPara)

실제 DB select 하기 전에 검색 조건을 변환할 때 사용
Ex) 검색 조건이 code 로 되어 있을 경우

```
public override bool BeforeLoad(string inPara)
{
        if (inPara == "GRADE") {
                _GRADE = _GRADE.Replace("고급", "2");
                _GRADE = _GRADE.Replace("중급", "1");
        }
            return true;
 }
```

bool	BeforeDelete()

실제 DB delete 문 하기 전에 실행되는 함수
false 를 return 할 경우 DB delete 하지 않음

bool	AfterAdd()

DB insert 이후 실행되는 함수

bool	AfterEdit()

DB update 이후 실행되는 함수

bool	AfterLoad()

DB select query 이후 실행되는 함수

bool	AfterDelete()

DB delete 이후 실행되는 함수

bool	ValidationAdd(out string sMsg)

DB insert 이전에 validation 함수
false 를 return 할 경우 insert 수행하지 않음,
sMsg 는 사용자에게 보여 줄 validation 이후 메시지

bool	ValidationEdit(out string sMsg)

DB Update 이전에 validation 함수
false 를 return 할 경우 Update 수행하지 않음,
sMsg 는 사용자에게 보여 줄 validation 이후 메시지

bool	ValidationDelete(out string sMsg)
DB Delete 이전에 validation 함수 false 를 return 할 경우 Delete 수행하지 않음, sMsg 는 사용자에게 보여 줄 validation 이후 메시지	
void	OnError(Exception e,string iSQL)
ORMClass 를 통해서 DB 작업을 할 때 Error 가 발생했을 때 실행되는 함수 실제 DB 에 실행된 query 를 iSQL 인자로 받아온다	
void	OnComplete(string resultSQL);
ORMClass 를 통해서 DB 작업이 정상적으로 진행되었을 때 실행되는 함수 실제 DB 에 실행된 query 를 resultSQL 인자로 받아온다	

3.3 SEED_ORM Class

- 인터페이스 클래스 IORMInterface 를 상속받아 구현한 추상 클래스

IORMinterface 클래스를 상속받아 일반적으로 구현한 다음 ORM instance Class 가 이를 다시 상속 받는 다. 재정의가 필요한 메소드는 ORM instance Class 가 overriding 을 통해 재정의하여 사용한다

<소스>

```
public abstract class SEED_ORM : IORMInterface
    {
        public string _ROWID { get; set; }

        public SEED_ORM()
        {
        }

        public abstract string SetDBconnectionInfo ();

        public abstract string SetTableName();
        public virtual string SetPrimaryField()
        {
            string sPrimariyField = "ROWID";
            return sPrimariyField;

        }
```

```
        public virtual string SetPrimaryMemberVariable()
        {
            string sPrimariyField = "_ROWID";
            return sPrimariyField;

        }
```

…

* SetTableName, SetDBprovider 와 같이 ORM instance Class 에서 반드시 재정의 해야 하는 부분은 추상메소드로 구현되었다.

*** 멤버 변수 _ROWID 는 Oracle DB 사용시 반드시 존재하는 ROWID 필드를 뜻한다.**

3.4 ORMSessionFactory

ORM 형식화한 Class 를 ORM 기능을 제공해주는 제네릭 클래스
C#의 reflection 기술을 활용하여 ORM 기능 구현

3.4.1 예제 통한 주요 메소드 설명

i) DB 테이블 정보

DB table 명 : USER	
varchal2	EMPNO
varchal2	NAME

ii) ORM instance Class 선언

```
public class JqgridDemo : SEED
    {
        public string _EMPNO { get; set; }
        public string _NAME { get; set; }

        public override string SetDBconnectionInfo()
        {
            return clsConst.DBPROVIDER.DB_USER;
        }
```

```
        public override string SetTableName()
        {
            return "USER";
        }
    }
```

- SEED 추상 클래스를 상속 받음
- SetDBprovider, SetTableName 구현
- SetDBprovider 는 DB 정보(DB 서버명, DB 계정정보)
- SetTableName 은 DB 테이블명 입력
- SEED 추상 클래스를 상속 받아 SetPrimaryMemberVariable, SetPrimaryField 재정의 하지 않으므로 Primary key 는 자동적으로 rowid 가 된다.

iii) CRUD 메소드 설명 및 예제

Create(T target) – Data insert 함수

Data insert 하는 메소드

Ex) JqgridDemo 라는 ORM 형식화 class 를 insert 예시

JqgridDemo 클래스는 Table 명이 USER 이고, EMPNO, NAME 인 두 개의 컬럼을 가진 DB table 을 ORM 한 class 이다.

<소스>

```
JqgridDemo toCreate = new JqgridDemo();
                toCreate._EMPNO = "123";
                toCreate._NAME ="홍길동";

ORMSessionFactory ifactory = new ORMSessionFactory("ConnectionSring");
                ifactory.Create(toCreate);
```

<결과>

DB table USER 에 EMPNO 가 123 이고, NAME 이 홍길동인 데이터가 insert 된다.

Edit(T target) - Data update 함수

Update 시 update 조건절은 IORMInterface 의 SetPrimaryMemberVariable, SetPrimaryValue 구현을 통해서 만들어 짐

Ex) 직번이 123 인 사원의 이름을 김길동으로 변경하는 예시

```
JqgridDemo toEdit = new JqgridDemo();
 toEdit._EMPNO = "123";
 toEdit._NAME ="김길동";
ORMSessionFactory ifactory = new ORMSessionFactory("ConnectionSring");
```

```
                Ifactory.Edit (toEdit);
```

Delete(T target) – Data delete 함수

Delete 시 update 조건절은 IORMInterface 의 SetPrimaryMemberVariable, SetPrimaryValue 구현을 통해서 만들어 짐

Ex) 직번이 123 인 사원의 이름을 김길동으로 삭제하는 예시

```
JqgridDemo toDelete = new JqgridDemo();
 toDelete._EMPNO = "123";
 toDelete._NAME ="김길동";
ORMSessionFactory ifactory = new ORMSessionFactory("ConnectionSring");
                Ifactory.Delete (toDelete);
```

CreateListData(List<T> targetList) – 일괄 데이터 insert

```
JqgridDemo toCreate1 = new JqgridDemo();
                toCreate1._EMPNO = "123";
                toCreate1._NAME ="홍길동";

JqgridDemo toCreate2 = new JqgridDemo();
                toCreate2._EMPNO = "456";
                toCreate2._NAME ="김길동";

List listData = new List();
listData.Add(toCreate1);
listData.Add(toCreate2);

ORMSessionFactory ifactory = new ORMSessionFactory("ConnectionSring");
                ifactory.CreateListData (listData);
```

EditListData(List<T> targetList) – 일괄 데이터 update

```
JqgridDemo toEdit1 = new JqgridDemo();
 toEdit1._EMPNO = "123";
 toEdit1._NAME ="홍길동";

JqgridDemo toEdit2 = new JqgridDemo();
 toEdit2._EMPNO = "456";
```

```
 toEdit2._NAME ="김길동";

List listData = new List();
listData.Add(toEdit1);
listData.Add(toEdit2);

ORMSessionFactory ifactory = new ORMSessionFactory("ConnectionSring");
                Ifactory.EditListData(listData);
```

DeleteListData(List<T> targetList) – 일괄 데이터 delete

```
JqgridDemo toDelete = new JqgridDemo();
 toDelete1._EMPNO = "123";
 toDelete1._NAME ="홍길동";

JqgridDemo toDelete = new JqgridDemo();
 toDelete2._EMPNO = "456";
 toDelete2._NAME ="김길동";

List listData = new List();
listData.Add(toDelete1);
listData.Add(toDelete2);

ORMSessionFactory ifactory = new ORMSessionFactory("ConnectionString");
                Ifactory.DeleteListData(listData);
```

List<T> Load(string sWhere = "") – Data query

Select query 문 받아오기
인터페이스 IORMInterface 의 SetQueryOption() 구현함으로써 select 구문 완성

Ex) select EMPNO,NAME from USER where emp_no ='31640'

```
                ORMSessionFactory jqFactory = new ORMSessionFactory("ConnectionSring");
                List<JqgridDemo> IList = new List<JqgridDemo>();
                IList = jqFactory.Load("and emp_no ='" +"31640" + "'");
```

List<T> LoadSQL(string sSql = "") – Data SQL query

ORM 방식이 아니라 실제 SQL 작성하여 query 결과를 List 방식으로 받아오기

주로 조인문이나 복잡한 Query 문 작성할 때 사용

* query 작성시 결과 컬럼 순서는 ORM instance 클래스 멤버 변수 선언순서와 동일해야한다.

Ex)

```
ORMSessionFactory jqFactory = new ORMSessionFactory("ConnectionSring");
List<JqgridDemo> IList = new List<JqgridDemo>();
list = jqFactory.LoadSQL("select empno, name from USER");
```

Bool EditOnly(T target)

클래스의 특정 필드만 update 할 경우

Ex)

```
ORMSessionFactory ormASE = new ORMSessionFactory("ConnectionSring");

JqgridDemo toEdit = new JqgridDemo();
 toEdit._EMPNO = "456";
 toEdit._NAME ="김길동";

// DB에 저장 NAME 필드만 update
ormASE.AddUpdateOnlyField("_NAME");
ormASE.EditOnly(toEdit2);
```

Bool EditOnlyListData(List<T> target)

복수의 인스턴스의 특정 필드만 update 할 때

```
JqgridDemo toEdit1 = new JqgridDemo();
 toEdit1._EMPNO = "123";
 toEdit1._NAME ="홍길동";

JqgridDemo toEdit2 = new JqgridDemo();
 toEdit2._EMPNO = "456";
 toEdit2._NAME ="김길동";

List listData = new List();
listData.Add(toEdit1);
listData.Add(toEdit2);
```

```
ORMSessionFactory ifactory = new ORMSessionFactory("ConnectionSring");
Ifactory.AddUpdateOnlyField("_NAME");
Ifactory.EditListData(listData);
```

_Create, _CreateListData , _Edit , _EditOnly , _EditOnlyListData

Parameterize query 를 사용하여 DB transaction

- inline 으로 DB query 하지 않고 Parameter 를 만들어 DB Transaction 한다.

함수에 **'_'를** 붙여서 기본 inline query 와 구분한다.