Dapper를 이용한 ORM 활용

발표자: 안현모
github.com/bluepope

목차

1. Dapper 소개

1. Stack Exchange에서 만든 OpenSource Mirco ORM 모듈
2. Nuget의 1.0.0 버전 등록일은 2011-04-14 이며 현재 최신버전은 2.0.30 버전
3. Query시 Generic으로 받은 Class의 Instance를 생성하여 Property에 값을 저장한 후 IEnumerable 열거형 반환함 (Select Query)
4. Execute 시 실행된 숫자를 반환함 (Insert, Update, Delete 등)

참고자료

1) https://stackoverflow.com/tags/dapper/info
2) https://docs.microsoft.com/ko-kr/azure/sql-database/sql-database-elastic-scale-working-with-dapper
3) https://dapper-tutorial.net/dapper

1. Dapper 소개

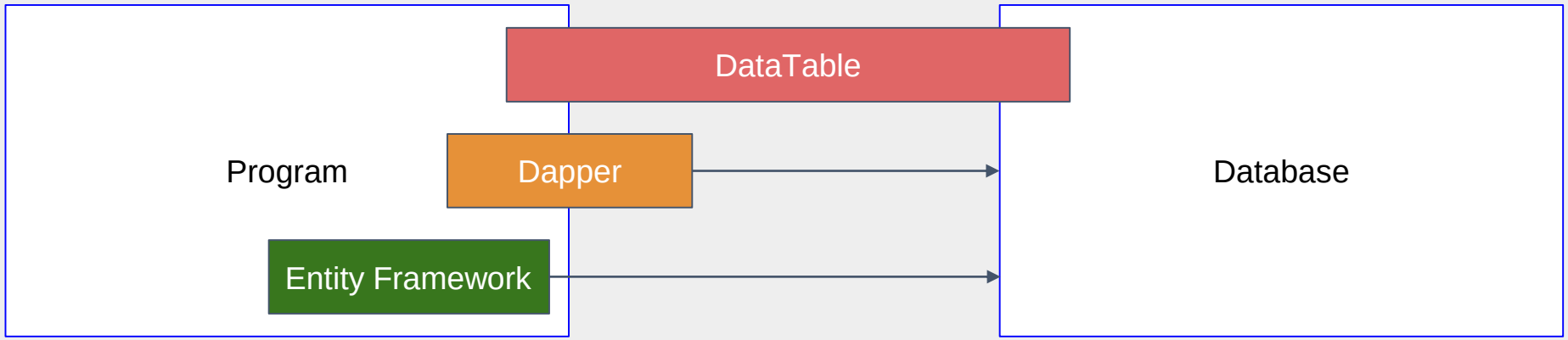
1. ORM 이란
   - Object Relational Mapping
   - Database를 영속성 객체로 저장
   - 완전한 ORM 이라고 한다면 EntityFramework 와 같이 코드로 DB를 제어 하게됨

2. 그렇다면 Micro ORM 은 무엇인가?
   - SqlMapper 라고도 불림
   - Database 핸들링은 직접하되, class instance 에 바인딩을 도와줌.

1. Dapper 소개

Q. 아무리 봐도 모르겠다, 그러니까 대체 Dapper 가 뭔데?
A. SELECT 쿼리를 DataTable 말고 class instance 로 받는거

Q. DataTable 보다 좋아? Entity Framework 보다 좋아?
A. 바라보는 지향점이 다르다고 볼수 있음.



2. DataTable과의 차이점

	DataTable	Dapper
Connection	ADO.NET Provider	
Query 작성	Sql Text 또는 StoredProcedure	
호출 객체	SqlCommand (new instance)	SqlMapper (static method)
Parameter Binding	cmd.Parameters.AddWithValue("param1", "변수1");	new { param1 = “변수1” } 또는 DynamicParameters
Select에 대한 반환값	DataTable -> DataRowCollection -> object	IEnumerable<T>
Execute에 대한 반환값	int	
비고	ExecuteReader, ExecuteScalar 같은 기능도 동일하게 사용 가능함.	

2. DataTable과의 차이점

1. DataTable 은 Column 및 Row 를 자동으로 생성하고 내부 값(Cell)은 object 형태로 반환합니다.

   즉 DataTable 에 사용된 Query를 완전 이해하고 있어야 사용이 가능합니다.
   예) dt.Rows[“Col1”] 이 실제로 존재하는지, 무슨 타입인지는 쿼리를 통해 정해짐

2. DataTable과의 차이점

2. Dapper 는 쿼리를 통해서 가져온 데이터를 Generic 형태의 Instance에 바인딩합니다. 바인딩시 자료형이 다르다면 Exception 이 발생할 수 있음
   Model 작성시점에는 쿼리를 알아야하나 사용하는 시점에는 클래스 프로퍼티이기때문에 인텔리센스의 도움을 받을 수 있음
   파라미터 바인딩이 매우 편리함

3. 예제코드 (DataTable)

```
            string selectQuery = @"
SELECT
    A.COL1
    ,A.COL2
FROM
    dbo.TABLE1 A
WHERE
    A.COL1 = @col1
";

            string insertQuery = @"
INSERT INTO dbo.TABLE1 (
    COL1
    ,COL2
)
VALUES (
    @COL1
    ,@COL2
)
";
```

```
#region DataTable 예제
var dt = new DataTable();

using (var conn = new SqlConnection("connectionString"))
{
    conn.Open();

    //SELECT 예제
    using (var cmd = new SqlCommand())
    {
        cmd.Connection = conn;
        cmd.CommandText = selectQuery;
        cmd.Parameters.AddWithValue("col1", "col1에 바인딩할 값");

        dt.Load(cmd.ExecuteReader());
    }

    foreach (DataRow row in dt.Rows)
    {
        Console.WriteLine(row["COL1"] as string);
        Console.WriteLine(row["COL2"] as long?);
    }

    //INSERT 예제
    using (var cmd = new SqlCommand())
    {
        cmd.Connection = conn;
        cmd.CommandText = insertQuery;
        cmd.Parameters.AddWithValue("col1", "col1에 바인딩할 값");
        cmd.Parameters.AddWithValue("col2", 1234567);

        int r = cmd.ExecuteNonQuery();
    }
}
#endregion
```

3. 예제코드 (Dapper)

```
public class DataModel
{
    참조 2개
    public string COL1 { get; set; }
    참조 2개
    public long? COL2 { get; set; }
}
```

```
            string selectQuery = @"
SELECT
    A.COL1
    ,A.COL2
FROM
    dbo.TABLE1 A
WHERE
    A.COL1 = @col1
";

            string insertQuery = @"
INSERT INTO dbo.TABLE1 (
    COL1
    ,COL2
)
VALUES (
    @COL1
    ,@COL2
)
";
```

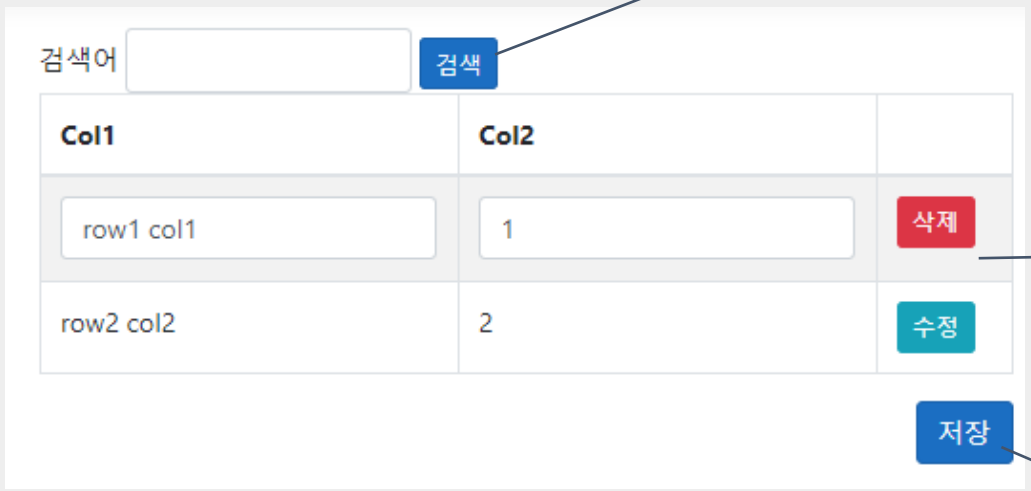
```
#region Dapper 사용 예제
using (var conn = new SqlConnection("connectionString"))
{
    conn.Open();

    //SELECT 예제
    //case 1. 직접 호출
    var dataModelList = SqlMapper.Query<DataModel>(conn, selectQuery, new { col1 = "col1에 바인딩할 값" }).ToList();
    foreach (var item in dataModelList)
    {
        Console.WriteLine(item.COL1);
        Console.WriteLine(item.COL2);
    }

    //INSERT 예제
    //case 1. 익명 타입
    SqlMapper.Execute(conn, insertQuery, new {
        col1 = "col1에 입력할 값",
        col2 = 1234567
    });
}
#endregion
```

3. 예제코드 (Dapper)

```
    public class DataModel
    {
        참조 2개
        public string COL1 { get; set; }
        참조 2개
        public long? COL2 { get; set; }

        참조 1개
        public static List<DataModel> GetList(IDbConnection conn, string col1)
        {
            string selectQuery = @"
SELECT
    A.COL1
    ,A.COL2
FROM
    dbo.TABLE1 A
WHERE
    A.COL1 = @col1
";
            return SqlMapper.Query<DataModel>(conn, selectQuery, new { col1 = col1 }).ToList();
        }

        참조 1개
        public int Insert(IDbConnection conn)
        {
            string insertQuery = @"
INSERT INTO dbo.TABLE1 (
    COL1
    ,COL2
)
VALUES (
    @COL1
    ,@COL2
)
";
            return SqlMapper.Execute(conn, insertQuery, this);
        }
    }
}
```

```
#region Dapper 사용 예제

using (var conn = new SqlConnection("connectionString"))
{
    conn.Open();

    //SELECT 예제
    //case 2. 정적메소드로 만들기
    var dataModelList2 = DataModel.GetList(conn, "col1에 바인딩할 값");
    foreach (var item in dataModelList2)
    {
        Console.WriteLine(item.COL1);
        Console.WriteLine(item.COL2);
    }

    //INSERT 예제
    //case 2. Instance Property 바인딩
    var model = new DataModel();
    model.COL1 = "col1에 입력할 값";
    model.COL2 = 1234567;

    var r = SqlMapper.Execute(conn, insertQuery, model);

    //case 3. 모델 내부에서 만드는 경우
    var r2 = model.Insert(conn);
}
#endregion
```

4. 실제 사용 예시 (ASP.NET MVC Core + vue.js)



Contoller를 통해 Json 으로 반환
return Json(DataModel.GetList())

버튼을 통해 ItemState 의 상태값을 변경 하고,
데이터를 수정함 (object binding)

변경된 object 를 Controller Method의 Parameter받아
List<DataModel>형태로 변환처리

받은 List의 State 를 확인하여 Insert, Update, Delete 작업

4. 실제 사용 예시 (ASP.NET MVC Core + vue.js)

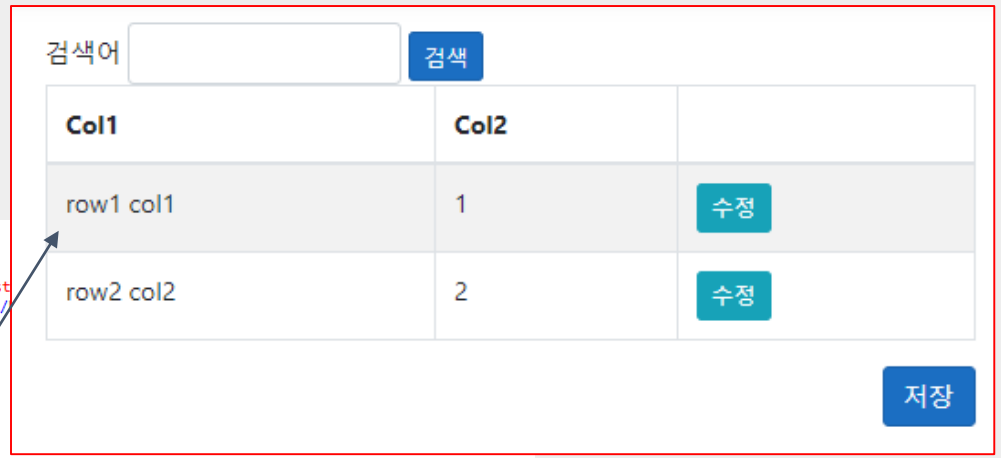
```
//web에서만 쓸거라 INotifyPropertyChanged를 상속받지는 않겠습니다
참조 4개
public abstract class ModelBase
{
    참조 5개
    public enum ItemStateEnum
    {
        None = 0,
        Modified = 1,
        Added = 2,
        Deleted = 3
    }

    참조 1개
    public ItemStateEnum ItemState { get; set; } = ItemStateEnum.None;
}
```

```
[Route("/SampleData")]
[HttpGet]
참조 0개
public JsonResult GetSampleData(string search)
{
    //core가 아닌 asp.net mvc 에서는 , 뒤에 JsonRequestBehavior.AllowGet 를 추가해줘야합니다.
    return Json(DataModel.GetList(search));
}
```

```
public class DataModel : ModelBase
{
    참조 0개
    public string Col1 { get; set; }
    참조 0개
    public long Col2 { get; set; }

    참조 1개
    public static List<DataModel> GetList(IDbConnection conn, IDbTransaction tran, string search)
    {
        string sql = @"
SELECT
    ,Col1
    ,Col2
FROM
    Table1
WHERE
    Col1 LIKE @search + '%'";

        return SqlMapper.Query<DataModel>(conn, sql, new { search = search }).ToList();
    }

    참조 1개
    public int Insert(IDbConnection conn, IDbTransaction tran = null)
    {
        return SqlMapper.Execute(conn, "insert query", this, tran);
    }

    참조 1개
    public int Update(IDbConnection conn, IDbTransaction tran = null)
    {
        return SqlMapper.Execute(conn, "update query", this, tran);
    }
    참조 1개
    public int Delete(IDbConnection conn, IDbTransaction tran = null)
    {
        return SqlMapper.Execute(conn, "delete query", this, tran);
    }
}
```

```
[Route("/SampleData")]
[HttpPost]
참조 0개
public IActionResult SaveSampleData(List<DataModel> input)
{
    if (ModelState.IsValid == false)
        return BadRequest(new { msg = "잘못된 요청입니다" });

    using (var conn = new SqlConnection("connectionstring"))
    {
        using (var tran = conn.BeginTransaction())
        {
            try
            {
                foreach (var item in input)
                {
                    switch (item.ItemState)
                    {
                        case ModelBase.ItemStateEnum.Deleted:
                            item.Delete(conn, tran);
                            break;
                        case ModelBase.ItemStateEnum.Added:
                            item.Insert(conn, tran);
                            break;
                        case ModelBase.ItemStateEnum.Modified:
                            item.Update(conn, tran);
                            break;
                    }
                }

                tran.Commit();
            }
            catch (Exception ex)
            {
                tran.Rollback();
                return BadRequest(new { msg = ex.Message });
            }
        }
    }

    return Ok();
}
```

4. 실제 사용 예시 (ASP.NET MVC Core + vue.js)




```
<div id="app1">
    <div>
        <span>검색어</span>
        <input v-model="searchText" type="text" class="form-control" st
        <button v-on:click="Load" class="btn btn-sm btn-primary">검색</
    </div>

    <table class="table table-bordered table-hover table-striped">
        <thead>
            <tr>
                <th>Col1</th>
                <th>Col2</th>
                <th></th>
            </tr>
        </thead>
        <tbody>
            <tr v-for="(item, index) in data" v-if="item.itemState < 3">
                <td>
                    <div v-if="item.itemState == 0">
                        {{ item.col1 }}
                    </div>
                    <div v-else>
                        <input v-model="item.col1" type="text" class="form-control" />
                    </div>
                </td>
                <td>
                    <div v-if="item.itemState == 0">
                        {{ item.col2 }}
                    </div>
                    <div v-else>
                        <input v-model="item.col2" type="text" class="form-control" />
                    </div>
                </td>
                <td>
                    <button v-if="item.itemState == 0" v-on:click="item.itemState = 1" class="btn btn-sm btn-info">수정</button>
                    <button v-if="item.itemState == 1" v-on:click="item.itemState = 3" class="btn btn-sm btn-danger">삭제</button>
                </td>
            </tr>
        </tbody>
    </table>

    <div class="text-right">
        <button v-on:click="Save" class="btn btn-primary">저장</button>
    </div>
```

```
var app1 = new Vue({
    el: "#app1",
    data: {
        searchText: "",
        data: []
    },
    methods: {
        Load: function () {
            var $this = this;

            $.ajax({
                type: "GET",
                url: "/SampleData",
                dataType: "json",
                data: {
                    search: $this.searchText
                },
                success: function (data, status, xhr) {
                    $this.data = data;
                },
                error: function (xhr, ajaxOptions, thrownError) {
                    if (xhr.status == 400) {
                        var data = JSON.parse(xhr.responseText);
                        alert(data.msg);
                    }
                    else {
                        alert(xhr.responseText)
                    }
                }
            });
        },
        Save: function () {
            var $this = this;

            $.ajax({
                type: "POST",
                url: "/SampleData",
                dataType: "json",
                data: {
                    input: $this.data
                },
                success: function (data, status, xhr) {
                    $this.Load();
                },
```

5. 주의사항

1. Winform DataGridView 또는 WPF DataGrid 바인딩시 List 가 아닌 BindingList (Winform) 또는 ObservableCollection (WPF)으로 바인딩 해야하며, Model에 INotifyPropertyChanged를 상속받아 적용해야합니다.

2. DataRowState와 같이 변화된 값에 대한 관리도 직접 추가해야합니다.
   (ModelBase 등 부모클래스 작성을 권장합니다)

3. Property 가 있으나 SELECT 쿼리에 없는 경우, 기본값으로 설정됩니다.
   즉, 쿼리와 모델 작성에 주의하지 않으면 의도하지않은 null 값을 만날수 있습니다.

6. 결론

1. DataTable의 object Casting 문제, EF 의 Db 핸들링에 대한 부담감이 있다면 Dapper 도입을 적극 추천

2. “한방 쿼리” 에 대한 쿼리 부담을 줄이고, 객체지향 관점으로 생각할 수 있음

3. 아주 단순하게 DB Table = Model로 만 생각한다 하더라도 생성된 모델을 반복적으로 사용하게될 수록 편의도가 증가

4. WPF MVVM, ASP.NET MVC 처럼 모델을 작성하여 관리하는 경우 도입을 적극적으로 권장