

StrCalc

Summary

ASP.NET Framework

포트폴리오를 위한
미니 프로젝트

기본적인 기능구현보다는...

- microsoft 권장의 **DB Connection**
- Entity Framework 를 사용한 **Code First** 방법
- **외부 라이브러리** 참조와 사용
- **IoC 컨테이너**를 활용
- **TRIGGER** 활용한 참조 테이블 레코드 자동생성

에 초점을 맞추고 제작하였습니다.

주의!! 작성자는 Java기반 spring 개발자로 본 포트폴리오의 목적은 ASP.NET환경에서도 충분히 동일 수준의 개발이 가능하다는 것을 알리기 위함입니다.

따라서 기능, 인터페이스 구현보다는 **ASP.NET Framework** 등의 microsoft 기반 개발환경을 어떻게 활용하는지에 대해 판단해주시면 감사하겠습니다.

개발 도구 및 환경

1. 환경



2. 개발도구



3. DB



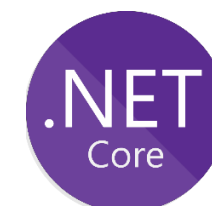
4. Server



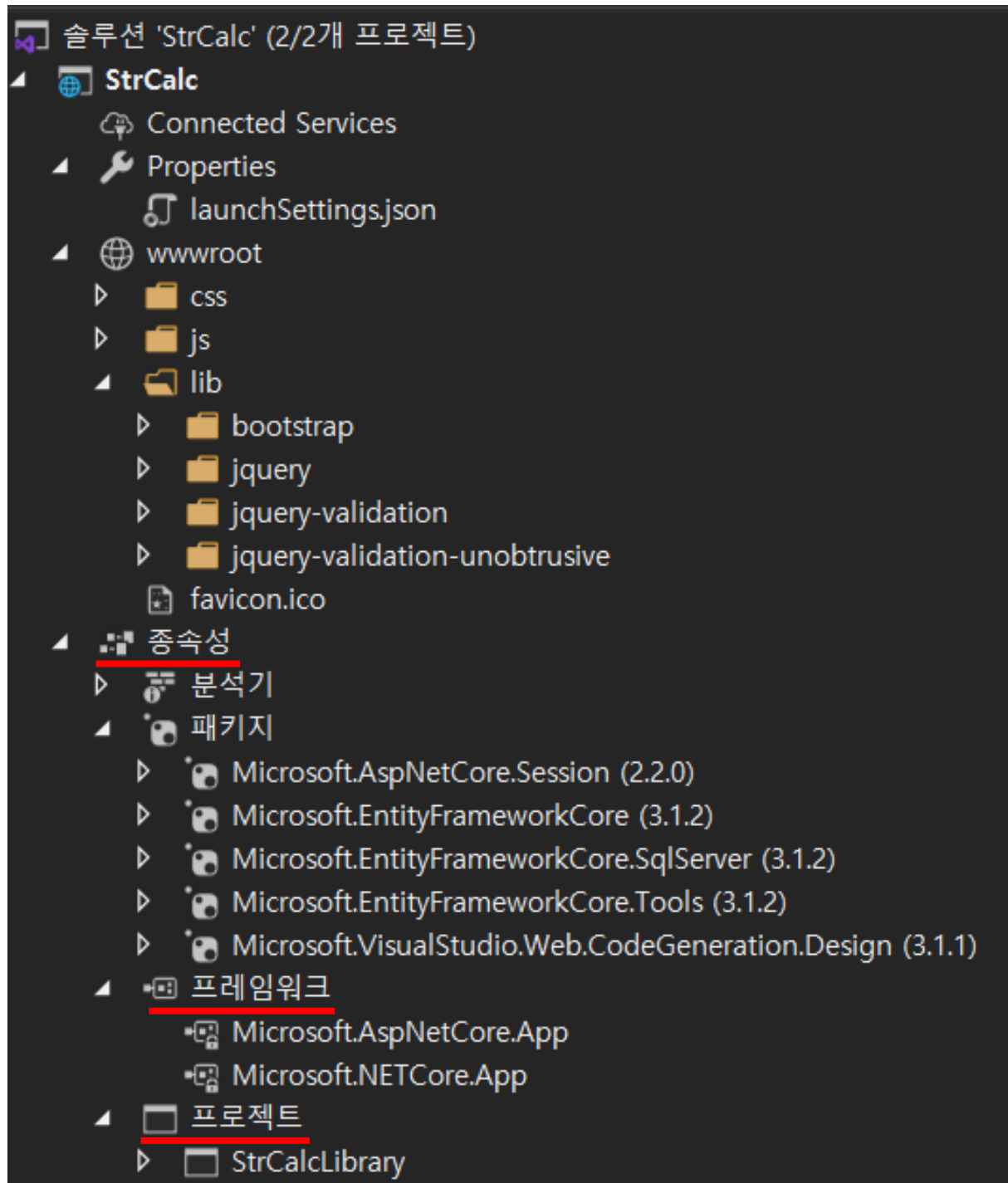
5. 언어



6. Framework

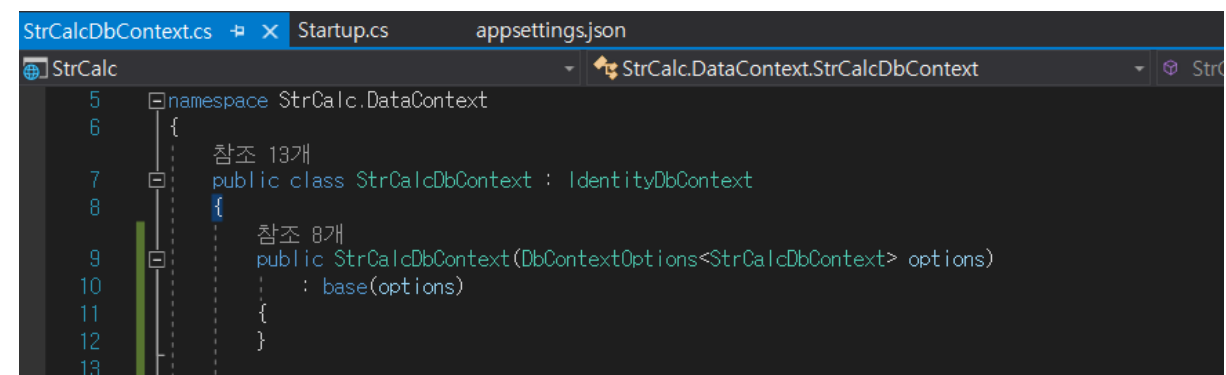
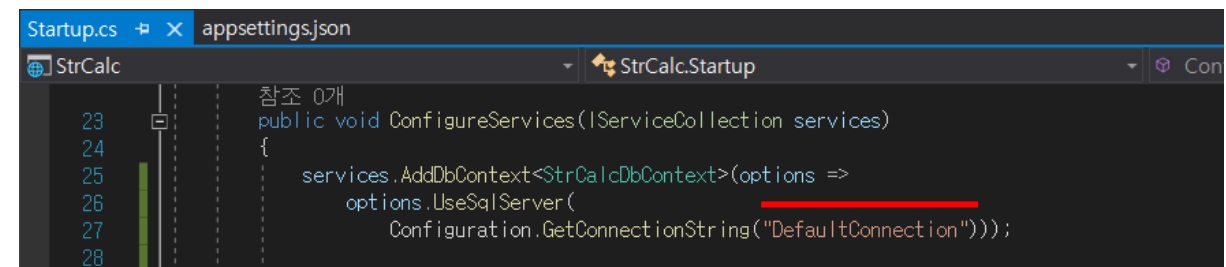
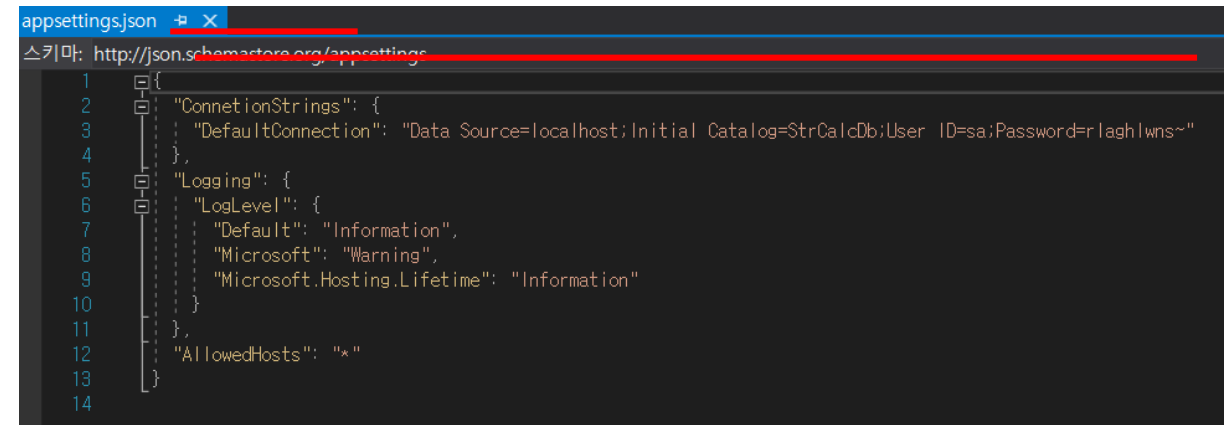


전체 구성



DB Connection 보안

- IoC 컨테이너 활용



IoC 컨테이너를 활용해

microsoft 권장의 **DB Connection 방식**을 사용하였습니다.

ConnetionString이 .json 파일에 저장되어 실제 코드에는 사용되지 않습니다.

Entity Framework - Code First

```

5 namespace StrCalc.DataContext
6 {
7     참조 13개
8     public class StrCalcDbContext : IdentityDbContext
9     {
10         참조 7개
11         public DbSet<Member> Members { get; set; }
12         참조 2개
13         public DbSet<Performance> MPfmc { get; set; }
14     }
15 }

```

```

PM> add-migration SCMig

```

```

2 namespace StrCalc.Migrations
3 {
4     참조 1개
5     public partial class SCMig : Migration
6     {
7         참조 0개
8         protected override void Up(MigrationBuilder migrationBuilder)
9         {
10             migrationBuilder.CreateTable(
11                 name: "Members",
12                 columns: table => new
13                 {
14                     No = table.Column<int>(nullable: false)
15                         .Annotation("SqlServer:Identity", "1, 1"),
16                     Id = table.Column<string>(nullable: false),
17                     Pw = table.Column<string>(nullable: false),
18                     NickName = table.Column<string>(nullable: false),
19                     Email = table.Column<string>(nullable: false)
20                 },
21             constraints: table =>
22             {
23                 table.PrimaryKey("PK_Members", x => x.No);
24             });
25         }
26     }
27 }

```

<Code First 방식의 테이블 생성 메서드>

개발자에게 **일관된 작업환경을 부여**할 수 있게 해주는 **Entity Framework**를 사용하여 일반 SQL문이 아닌 **C# 코드를 사용하는 DB 관리**가 가능하게 하였습니다.
(간단한 CRUD는 요약 밑에서 다루겠습니다.)

라이브러리 / IoC 컨테이너

```

솔루션 'StrCalc' (2/2개 프로젝트)
├── StrCalc
│   ├── StrCalcLibrary
│   ├── 종속성
│   ├── WilksPoint.cs
│   └── WilksRank.cs

```

```

종속성
├── 분석기
├── 패키지
├── 프레임워크
├── 프로젝트
└── StrCalcLibrary

```

```

8 using StrCalcLibrary;
9

```

```

23 참조 0개
24 public void ConfigureServices(IServiceCollection services)
25 {
26     services.AddTransient<WilksPoint>();
27     services.AddTransient<WilksRank>();
28 }

```

```

7 using StrCalcLibrary;
8
9 namespace StrCalc.Controllers
10 {
11     참조 1개
12     public class MemberController : Controller
13     {
14         private readonly WilksPoint _wp;
15         private readonly WilksRank _wr;
16
17         참조 0개
18         public MemberController(WilksPoint wp, WilksRank wr)
19         {
20             _wp = wp;
21             _wr = wr;
22         }
23     }
24 }

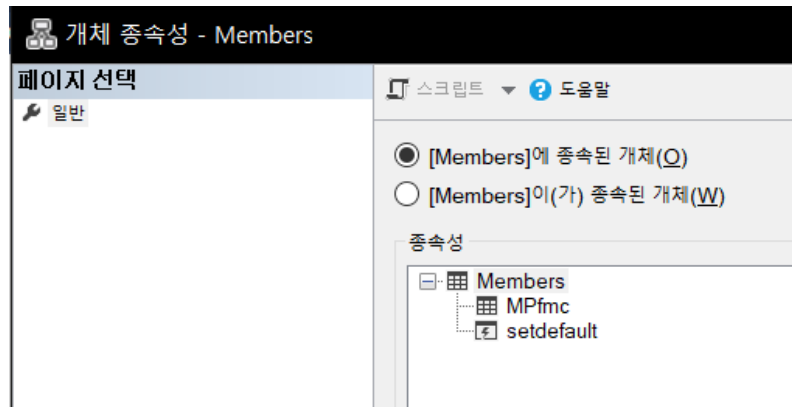
```

솔루션의 **메모리를 절약**하고 **모듈의 재사용**을 고려한 설계를 하였습니다.

Transient 방식으로 **메모리 누수를 방지** 하였습니다.

TRIGGER 활용

- 참조 테이블 레코드 자동생성



```
CREATE TRIGGER dbo.setdefault
ON dbo.members
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @num int;
    select @num = No FROM INSERTED;
    INSERT INTO dbo.MPfmt values (@num, 'Male', 0, 0, 0, 0, 0, 0, 0);
    SET NOCOUNT OFF;
END
GO
```

종속된 테이블의 참조 받는 레코드를 자동 생성하기 위해 **TRIGGER**를 작성하였습니다.

Server와 DB의

Connection 빈도를 최소화하여 병목현상을 방지할 수 있습니다.

Code First 방식

- CRUD 메서드

```
[HttpPost]
참조 0개
public IActionResult Regist(Member model)
{
    if (ModelState.IsValid)
    {
        using (var db = new StrCalcDbContext())
        {
            db.Members.Add(model);           // 메모리까지 올리기
            db.SaveChanges();               // commit
        }
        return RedirectToAction("Index", "Home");
    }
    return View();
}
```

```
using (var db = new StrCalcDbContext())
{
    var user = db.Members.FirstOrDefault(u => u.No.Equals(no));
    return View(user);
}
```

```
using (var db = new StrCalcDbContext())
{
    var no = HttpContext.Session.GetInt32("LOGIN_USER");
    var user = db.Members.FirstOrDefault(u => u.No.Equals(no));

    user.Pw = member.Pw;
    user.NickName = member.NickName;
    user.Email = member.Email;
    db.SaveChanges();
}
```

```
using (var db = new StrCalcDbContext())
{
    var user = db.Members.FirstOrDefault(u => u.No.Equals(no));
    db.Members.Remove(user);
    db.SaveChanges();
}
```

EF 라이브러리에 등록되어 있는 Add메서드로 insert sql을 주입하고 SaveChange로 commit 합니다.
Select와 Delete도 같습니다.
Update는 레코드의 칼럼에 직접적으로 접근하여 값을 변경 후 커밋합니다.