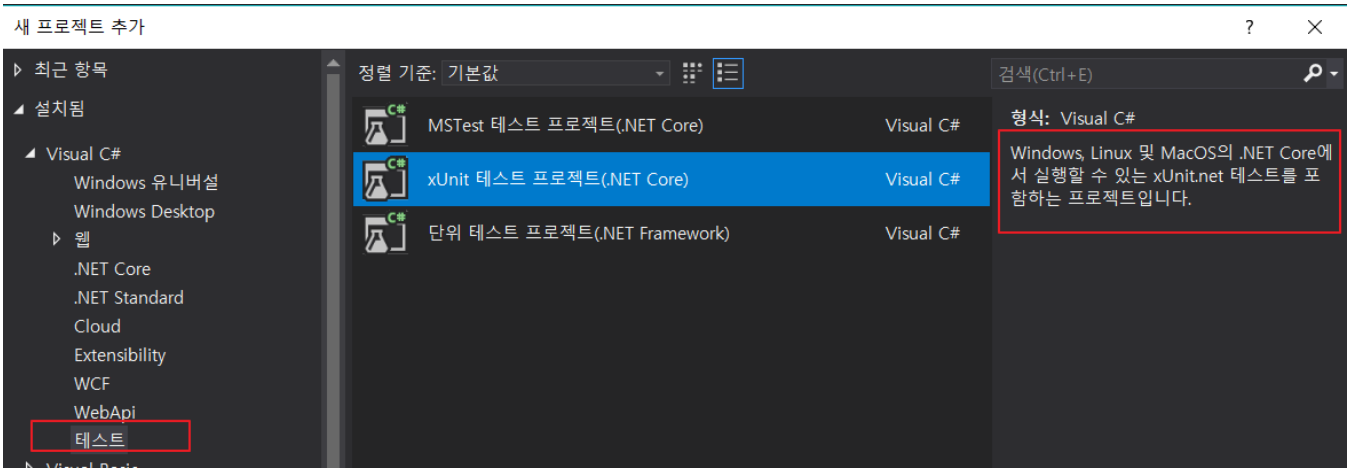
00. UNITTEST FOR ORM

( )

ORM .

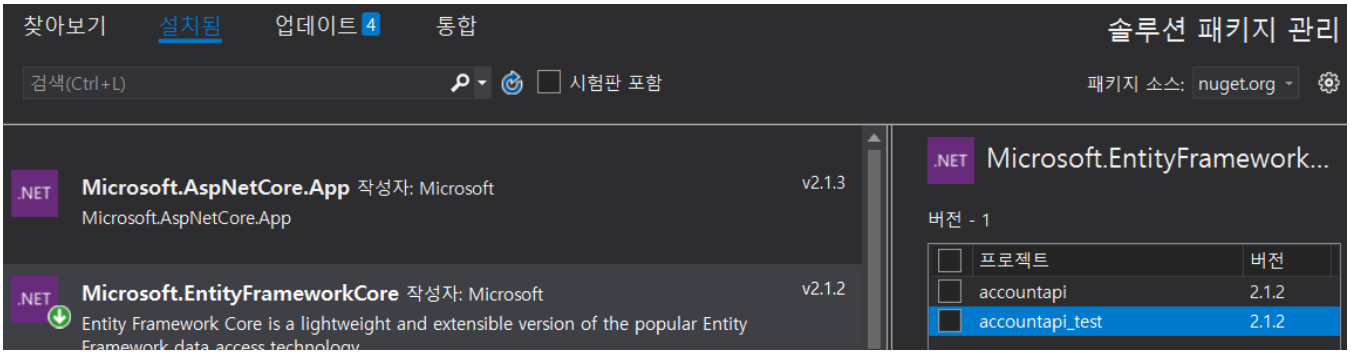
API .

xUnit



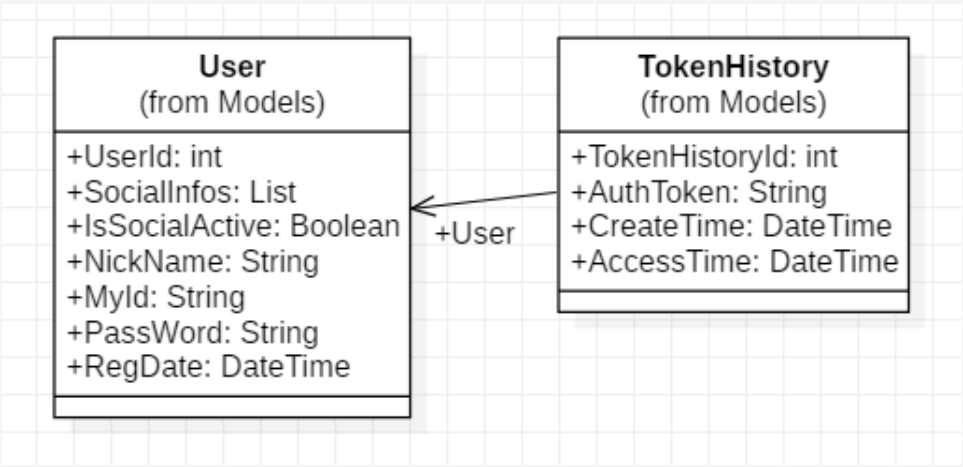
core xUnit .

-
-



Nuget ,

.




```
    [Table("user")]
    public class User
    {
        [Key]
        [DatabaseGenerated
(DatabaseGeneratedOption.Identity)]
        public int UserId { get; set; }

        public List<SocialInfo>
SocialInfos { get; set; }

        public Boolean IsSocialActive {
get; set; }

        [StringLength(50,MinimumLength =
3)]
        [Required]
        public String NickName { get;
set; }

        [StringLength(50, MinimumLength =
3)]
        [Required]
        public String MyId { get; set; }

        [StringLength(50, MinimumLength =
3)]
        [Required]
        public String PassWord { get;
set; }

        public DateTime RegDate { get;
set; }


    }
```

- Table :
- Key : Primary Key
- GeneratedOption : (DB)
- Required : Not null
- StringLength :

DB ,

.

```
    [Table("tokenhistory")]
    public class TokenHistory
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.
Identity)]
        public int TokenHistoryId { get; set; }

        [ForeignKey("UserForeignKey")]
        public User User { get; set; }

        public String AuthToken { get; set; }

        public DateTime CreateTime { get; set; }

        public DateTime AccessTime { get; set; }

    }
```

(FoeignKey) (UserId)

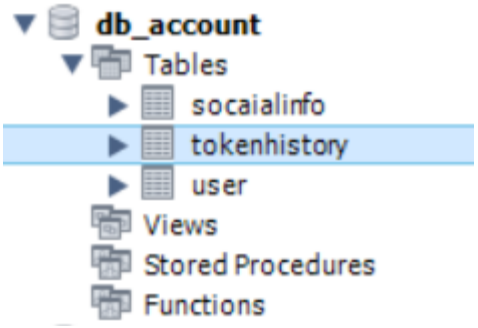
.

SQL	ORM
Select u.NickName * from TokenHistory t join User u u.userid=t.userid where u.Userid=1;	TokenHistory token=_context.TokenHistory.First(t => t.UserId == 1); token.User.NickName;

> *Join SQL .*
>
> *ORM SQL , SQL .*
>
> *Null ? inner join SQL*
>
> *Entity .*

```
    public class AccountContent : DbContext
    {
        public DbSet<User> Users { get; set; }
        public DbSet<SocialInfo> SocialInfos { get; set; }
        public DbSet<TokenHistory> TokenHistories { get;
set; }

        public AccountContent(
DbContextOptions<AccountContent> options )
        : base(options)
        {

        }

        protected override void OnModelCreating
(ModelBuilder modelBuilder)
        {
            //( Fluent API     )
            modelBuilder.Entity<TokenHistory>()
                .HasIndex(p => new { p.AuthToken })
                .IsUnique(true);
        }

    }
```

- HasIndex .



C# Object . 1 VS N

. .

,

. 1 .

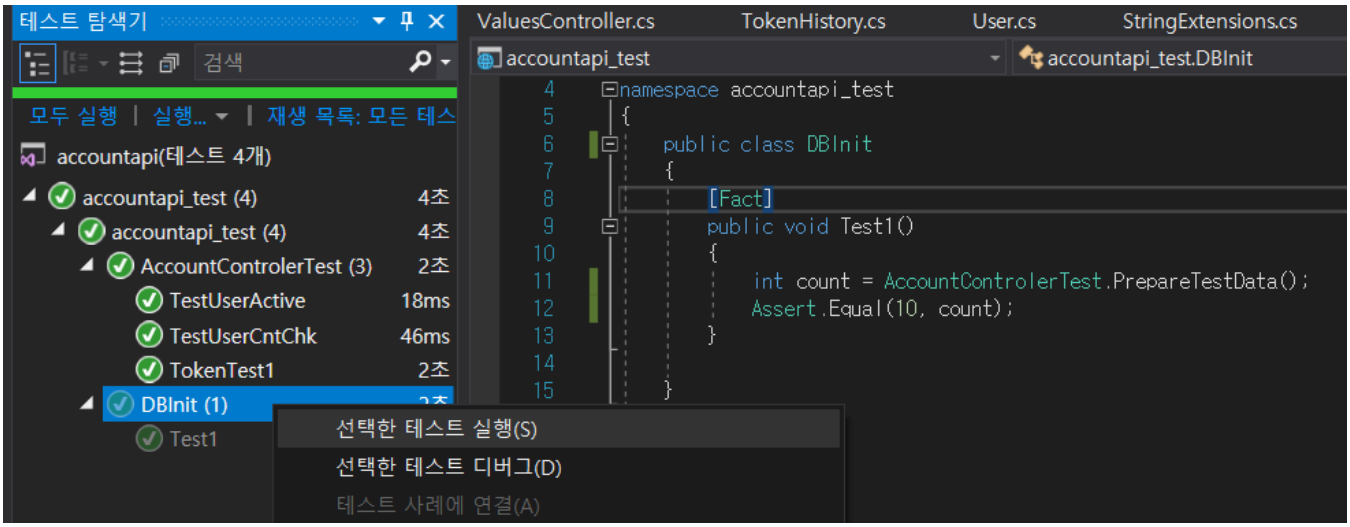
ORM , .

```
public class AccountService
{
    private readonly AccountContent _context;

    public AccountService(AccountContent context)
    {
        _context = context;
    }

    public User GetUserByid(int id)
    {
        return _context.Users.First(p => p.UserId == id);
    }

    public String GetAccessToken(string userid,string userpw)
    {
        User accessUser = _context.Users.First(p => p.MyId == userid && p.PassWord == userpw);
        if (accessUser == null)
        {
            throw new Exception("401");
        }
        else
        {
            string token = Convert.ToBase64String(Guid.NewGuid().ToByteArray());
            _context.TokenHistories.Add(new TokenHistory()
            {
                User = accessUser,
                AuthToken = token,
                CreateTime = DateTime.Now,
                AccessTime = DateTime.Now
            });

            _context.SaveChanges();
            return token;
        }
    }

    public User GetMyInfo(string accessToken)
    {
        User myinfo= _context.TokenHistories.First(p => p.AuthToken == accessToken).User;
        if (myinfo == null) throw new Exception("401");
        myinfo.PassWord = "******";
        return myinfo;
    }

}
```

, AccessToken .. AccessToken

API ..., SQL DB .

TestCode



- DBInit : Entity DB / .
- AccountControlerTest : DB . (DB)

AccountControlerTest

```
    public class AccountControlerTest
    {
        public static readonly string ConnectionString = "server=localhost;database=db_account;user=psmon;
password=db1234";

        public AccountControlerTest()
        {
            InitContext();
        }

        private AccountContent _context;

        internal static int PrepareTestData()
        {
            var builder = new DbContextOptionsBuilder<AccountContent>()
                .UseMySql(AccountControlerTest.ConnectionString);
            var context = new AccountContent(builder.Options);

            context.Database.EnsureDeleted();
            context.Database.EnsureCreated();
            var users = Enumerable.Range(1, 10)
                .Select(i => new User { MyId = "TestID" + i, PassWord = "TEST123" + i, NickName = "Mynick" + i,
RegDate = DateTime.Now, IsSocialActive = false });

            context.Users.AddRange(users);
            context.SaveChanges();
            return context.Users.Count(t => t.IsSocialActive == false);
        }

        protected void InitContext()
        {
            var builder = new DbContextOptionsBuilder<AccountContent>()
                .UseMySql(AccountControlerTest.ConnectionString);

            var context = new AccountContent(builder.Options);
            _context = context;
        }

        [Fact]
        public void TestUserCntChk()
        {
            var count = _context.Users.Count(t => t.IsSocialActive==false);
            Assert.Equal(10, count);
        }

        [Fact]
        public void TokenTest1()
        {
            var service = new AccountService(_context);
            var accessToken = service.GetAccessToken("TestID1", "TEST1231");
            User myInfo = service.GetMyInfo(accessToken);
            Assert.Equal("Mynick1", myInfo.NickName);
            //TestID1 TEST1231
        }

        [Fact]
        public void TestUserActive()
        {
            bool expected = false;
            var controller = new AccountControler(_context);
            User result = controller.GetUserByid(1);
            Assert.Equal(expected, result.IsSocialActive);
        }
    }
```

, .

, Entity .

C# Object Entity Entity .

GIT : http://git.webnori.com/projects/OPPJ/repos/accountapi/browse/accountapi_test