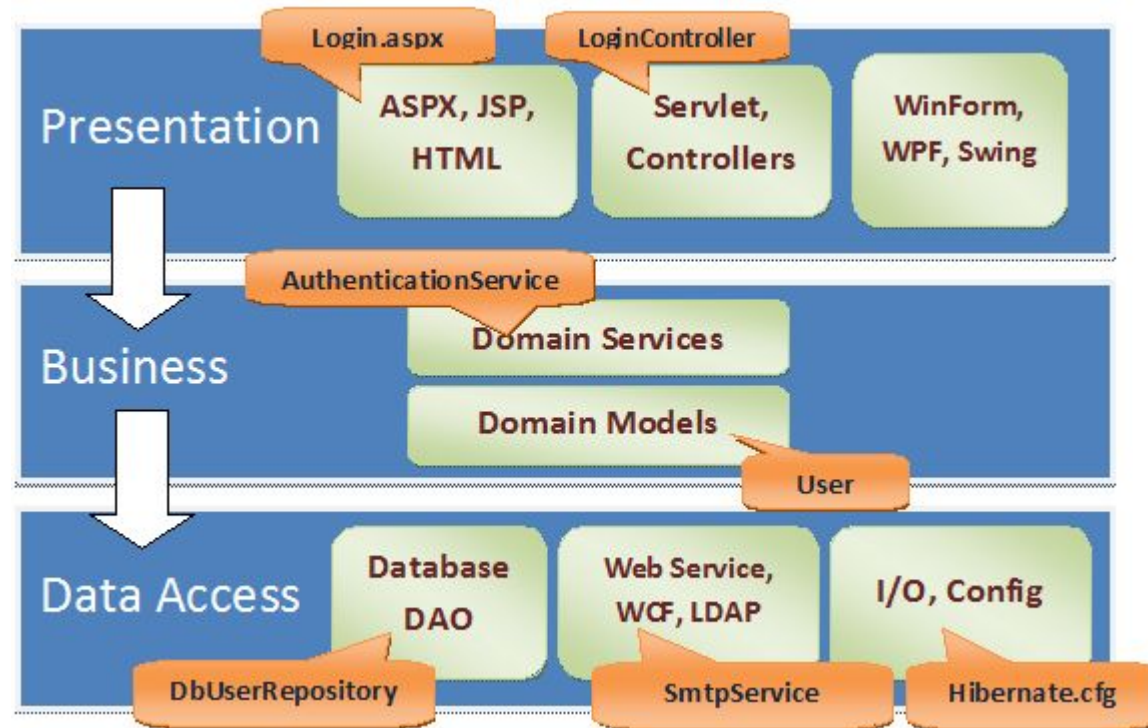


Repository and Unit of Work

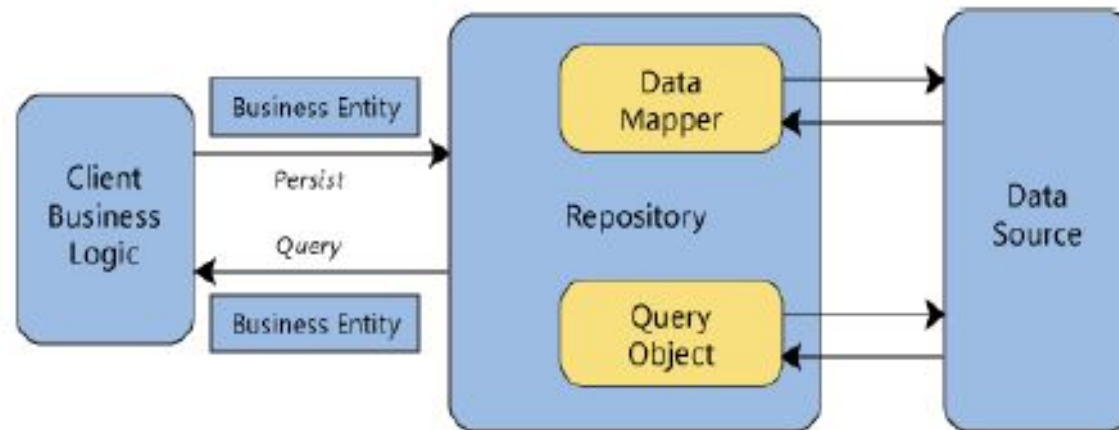
Design Patterns
with Entity Framework

- Патерн Repository
- Патерн Unit of Work



Repository Pattern

- Посередник між BLL та DAL(Data Source) рівнями
- Це рівень, який роздвляє Дані та Рівень Domain



Для чого потрібен Repository ?

- Рівень абстракції між Business Logic рівнем та Data Access рівнем.
- Ізолює програму від змін джерела даних.
- Полегшує автоматизоване юніт тестування, Test Driven Development.
- Джерело даних може бути змінено без будь-яких змін в бізнес логіці і з мінімальними змінами в Репозиторії.
- Легко створювати mock репозиторію.

EF DbContext implementer

```
public class SchoolContext : DbContext
{
    public DbSet<Course> Courses {get;set;}
    public DbSet<Department> Departments {get;set;}
    public DbSet<Enrollment> Enrollments {get;set;}
    public DbSet<Instructor> Instructors {get;set;}
    public DbSet<Student> Students {get;set;}

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        //...
    }
}
```

IGenericRepository<TEntity>

```
public interface IGenericRepository<TEntity> where TEntity : class
{
    IEnumerable<TEntity> Get( Expression<Func<TEntity, bool>> filter = null,
Func<IQueryable<TEntity>, IOrderedQueryable<TEntity>> orderBy
        = null, string includeProperties = "");
    TEntity GetByID(object id);
    void Insert(TEntity entity);
    void Delete(object id);
    void Delete(TEntity entityToDelete);
    void Update(TEntity entityToUpdate);
}
```


GenericRepository<TEntity> constructor

```
public class GenericRepository<TEntity> :  
IGenericRepository<TEntity> where TEntity : class  
{  
    internal SchoolContext context;  
    internal DbSet<TEntity> dbSet;  
  
    public GenericRepository(SchoolContext context)  
    {  
        this.context = context;  
        this.dbSet = context.Set<TEntity>();  
    }  
}
```

```

public virtual IEnumerable<TEntity> Get( Expression<Func<TEntity, bool>> filter = null,
                                       Func<IQueryable<TEntity>,
                                       IOrderedQueryable<TEntity>> orderBy = null,
                                       string includeProperties = "")
{
    IQueryable<TEntity> query = dbSet;
    if (filter != null)
    {
        query = query.Where(filter);
    }
    foreach (var includeProperty in includeProperties.Split
            (new char[] { ',' }, StringSplitOptions.RemoveEmptyEntries))
    {
        query = query.Include(includeProperty);
    }

    if (orderBy != null)
    {
        return orderBy(query).ToList();
    }
    else
    {
        return query.ToList();
    }
}

```

IEnumerable<TEntity> Get(...)

```
public virtual TEntity GetByID(object id)
{
    return dbSet.Find(id);
}
public virtual void Insert(TEntity entity)
{
    dbSet.Add(entity);
}
public virtual void Delete(object id)
{
    TEntity entityToDelete = dbSet.Find(id);
    Delete(entityToDelete);
}
public virtual void Delete(TEntity entityToDelete)
{
    if (context.Entry(entityToDelete).State == EntityState.Detached)
    {
        dbSet.Attach(entityToDelete);
    }
    dbSet.Remove(entityToDelete);
}
```

Імплементация CRUD методів

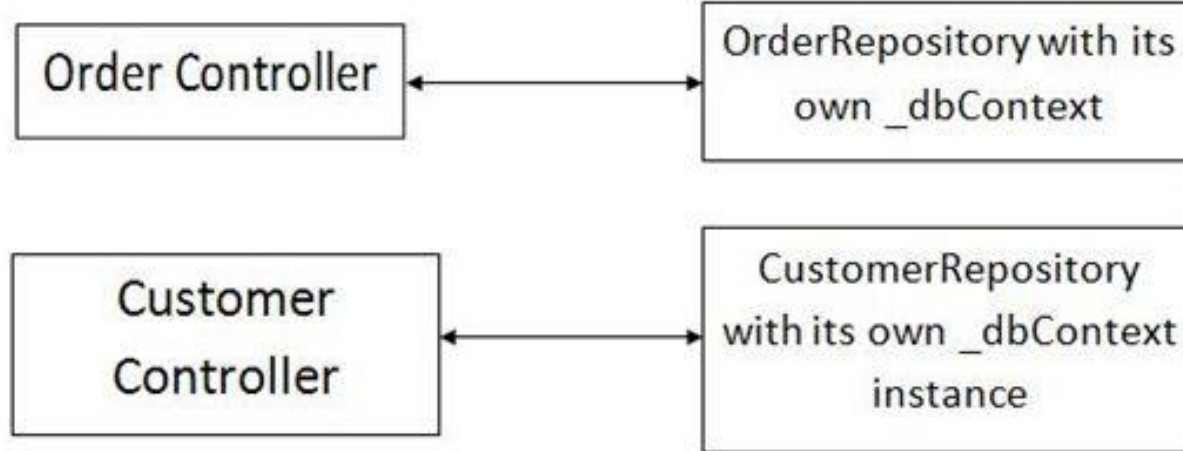
Імплементація CRUD методів

```
public virtual void Update(TEntity entityToUpdate)
{
    dbSet.Attach(entityToUpdate);
    context.Entry(entityToUpdate).State = EntityState.Modified;
}
```

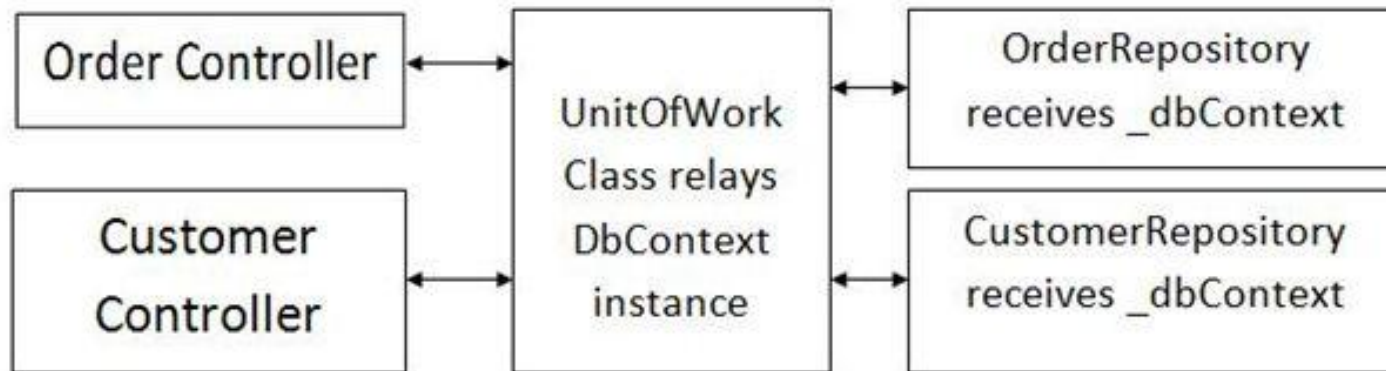
Unit of Work (UoW)

- Патерни об'єктно-реляційної поведінки:
 - Unit of Work
 - Identity Map
 - Lazy Load
- UnitOfWork покликаний відслідковувати всі зміни даних, які ми здійснюємо з доменною моделлю в рамках бізнес-транзакції. Після того, як бізнес-транзакція закривається, всі зміни потрапляють в БД у вигляді єдиної транзакції.

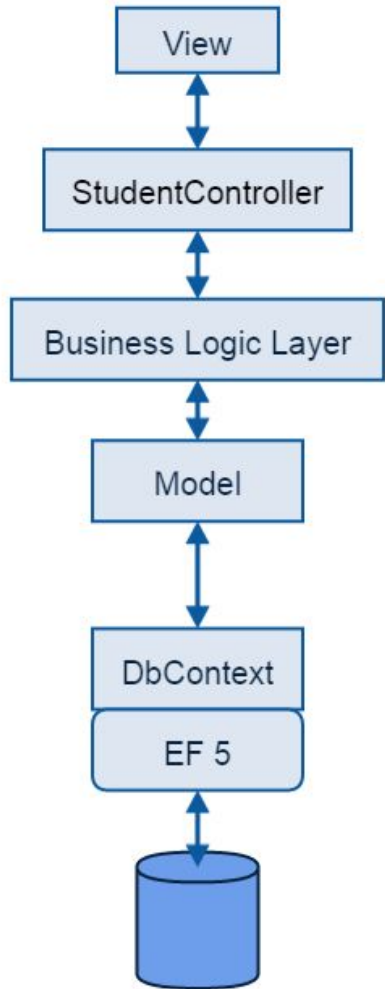
Without UnitOfWork



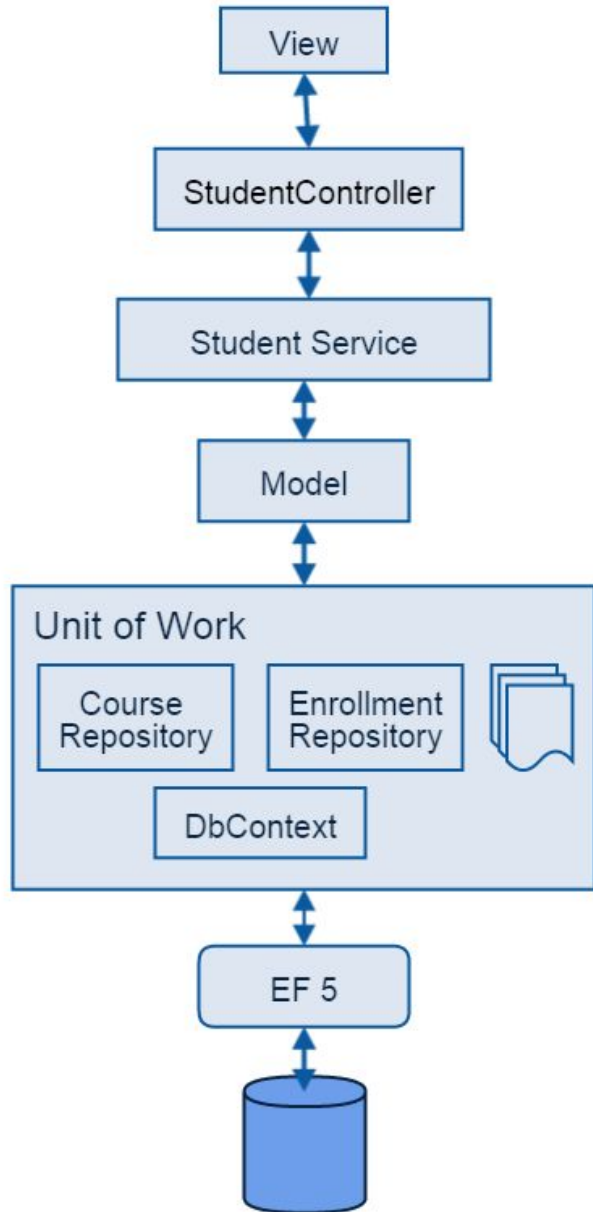
With UnitOfWork



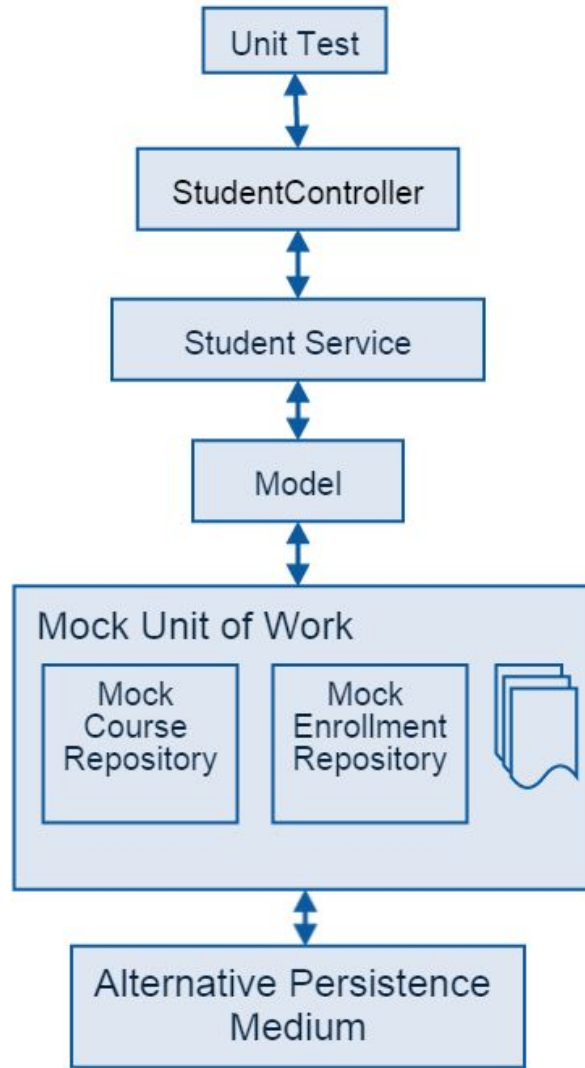
Without Repository and UoW



With Repository and UoW



Testing and Mocking



Search Solution Explorer (Ctrl+;)

- ▶ Properties
- ▶ References
- ▶ App_Data
- ▶ App_Start
- ▶ Content
- ▶ Controllers
- ▶ DAL
 - ▶ Interfaces
 - ▶ IGenericRepository.cs
 - ▶ IUnitOfWork.cs
 - ▶ GenericRepository.cs
 - ▶ SchoolContext.cs
 - ▶ UnitOfWork.cs
- ▶ Filters
- ▶ Images
- ▶ Migrations
- ▶ Models
- ▶ Scripts
- ▶ Services
 - ▶ Interfaces
 - ▶ IExaminationService.cs
 - ▶ IRegistrationService.cs
 - ▶ ExaminationService.cs
 - ▶ RegistrationService.cs
- ▶ ViewModels
- ▶ Views
- ▶ favicon.ico
- ▶ Global.asax
- ▶ packages.config
- ▶ Web.config

able project structure

Object Explorer
Test Explorer
Solution Explorer
Team Explorer
Server Explorer
Properties

IUnitOfWork interface

```
public interface IUnitOfWork
{
    GenericRepository<Department> DepartmentRepository {get;}
    GenericRepository<Course> CourseRepository {get;}
    GenericRepository<Person> PersonRepository{get;}
    GenericRepository<Student> StudentRepository{get;}
    GenericRepository<Instructor> InstructorRepository{get;}
}
```

UnitofWork.cs

```
public class UnitOfWork : IUnitOfWork, IDisposable
{
    private SchoolContext context = new SchoolContext();
    private GenericRepository<Department> departmentRepository;
    private GenericRepository<Course> courseRepository;
    private GenericRepository<Person> personRepository;
    private GenericRepository<Student> studentRepository;
    private GenericRepository<Instructor> instructorRepository;

    //Generic repository instantiation for every entity-set in
domain
    // using a single shared DbContext object within a Uow
wrapper
    //...
}
```

```
public GenericRepository<Department> DepartmentRepository
{
    get
    {
        if (this.departmentRepository == null)
        {
            this.departmentRepository = new GenericRepository<Department>(context);
        }
        return departmentRepository;
    }
}

public GenericRepository<Course> CourseRepository
{
    get
    {
        if (this.courseRepository == null)
        {
            this.courseRepository = new GenericRepository<Course>(context);
        }
        return courseRepository;
    }
}
```

Persist DbContext changes and clean up resources

```
public void Save()
{
    context.SaveChanges();
}
```

```
private bool disposed = false;

protected virtual void Dispose(bool disposing)
{
    if (!this.disposed)
    {
        if (disposing)
        {
            context.Dispose();
        }
    }
    this.disposed = true;
}

public void Dispose()
{
    Dispose(true);
    GC.SuppressFinalize(this);
}
```

How does the architectural wiring come live in the controller ?

```
public class CourseController
{
    private UnitOfWork unitOfWork = new UnitOfWork();

    public ActionResult Index()
    {
        var courses = unitOfWork.CourseRepository.Get(includeProperties: "Department");
        return View(courses.ToList());
    }

    public ActionResult Details(int id)
    {
        Course course = unitOfWork.CourseRepository.GetByID(id);
        return View(course);
    }
    //...
}
}
```

Sample Edit, Get(...) calls

```
public ActionResult Edit( [Bind(Include = "CourseID,Title,Credits,DepartmentID")]
Course course)
{
    try
    {
        if (ModelState.IsValid)
        {
            unitOfWork.CourseRepository.Update(course);
            unitOfWork.Save();
            return RedirectToAction("Index");
        }
    }
    catch (DataException dex)
    {
        ModelState.AddModelError("", "Unable to save changes. Try again, and if the
problem persists, see your system administrator.");
    }
    PopulateDepartmentsDropDownList(course.DepartmentID);
    return View(course);
}
```

```
private void PopulateDepartmentsDropDownList(object selectedDepartment = null)
{
    var departmentsQuery = unitOfWork.DepartmentRepository.Get( orderBy: q =>
q.OrderBy(d => d.Name));
    ViewBag.DepartmentID = new SelectList(departmentsQuery, "DepartmentID",
"Name", selectedDepartment);
}
```

References

- 1) <http://www.asp.net/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application>
- 2) <http://www.asp.net/mvc/overview/getting-started/getting-started-with-ef-using-mvc/advanced-entity-framework-scenarios-for-an-mvc-web-application#repo>

?