Шаблони за дизайн и ADO.NET

Шаблон Unit of Work

- Шаблонът Unit of Work има две основни цели: поддръжка на данните и техните промени в паметта и изпращане на тези промени към базата данни, чрез една транзакция.
- За да се постигнат тези цели, са необходими следните :
  - Управление на списъците от обекти на бизнес логиката в паметта, които са били променяни (чрез вмъкване, актуализиране или изтриване) по време на транзакцията.
  - След като транзакцията е завършена, всички тези промени се изпращат като единствен **unit of work,** за да бъдат физически запазени в базата, чрез едно действие.

Какво е “Work” и “Unit”

- Базисна дефиниция на “Work” е: *изпълнението на някаква задача*.
- От перспективата на софтуерното приложение “Work” не е нищо повече от вмъкване, актуализиране и изтриване на данни.
- Например, ако имаме приложение, което управлява потребителски данни в база данни, то можем да добавяме, променяме или изтриваме потребителски запис в базата (което разглеждаме като един unit). С други думи:

1 customer CRUD = 1 unit of work

Акроним CRUD

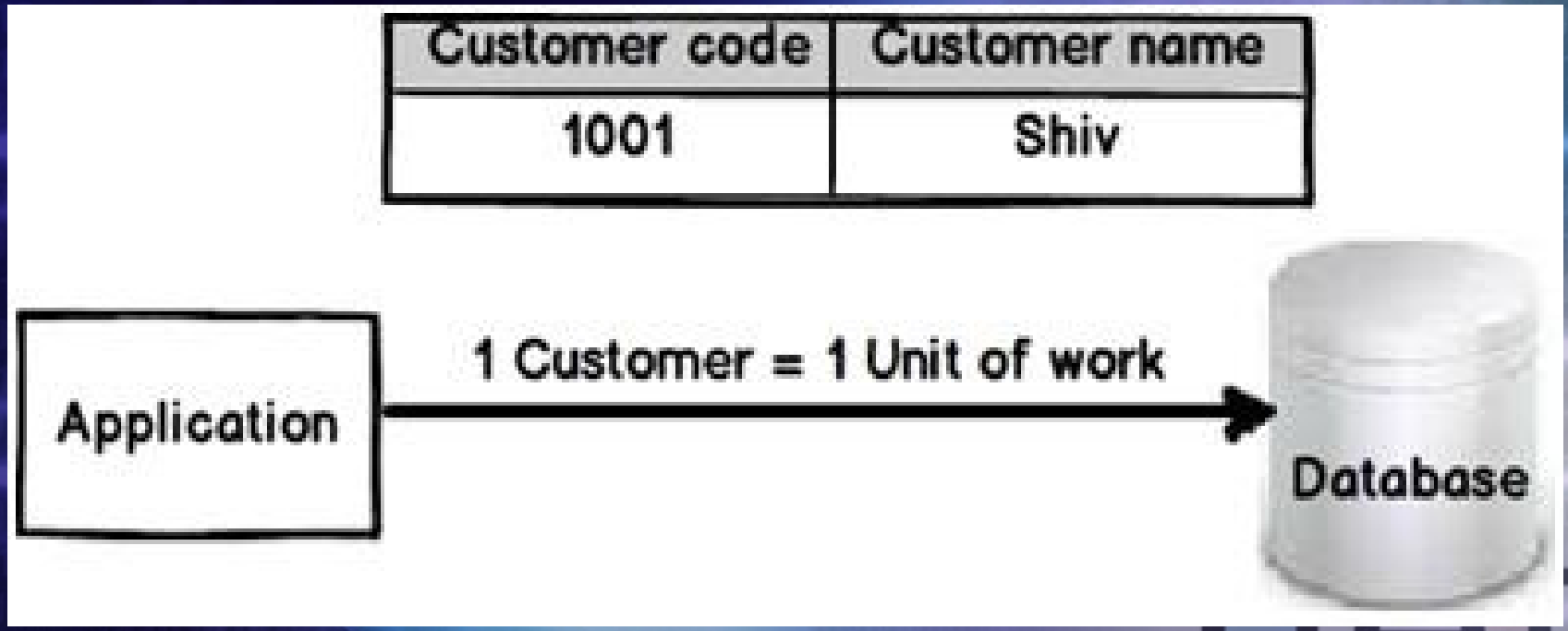
- Акронимът CRUD (Create, Read, Update, Delete) се отнася за всички основни функции, които могат да се имплементират в приложения, ползващи релационни бази данни;
- Всяка буква в този акроним може да бъде съпоставена към стандартна SQL заявка, HTTP метод или DDS (Data Distribution Service) операция:

Сравнение на CRUD команди

Операция	SQL	HTTP	DDS
Създаване	INSERT	PUT/POST	WRITE
Четене	SELECT	GET	READ/ TAKE
Промяна	UPDATE	PUT/POST/ PATCH	WRITE
Изтриване	DELETE	DELETE	DISPOSE

CRUD и Unit of Work

- CRUD операция към единичен потребителски запис



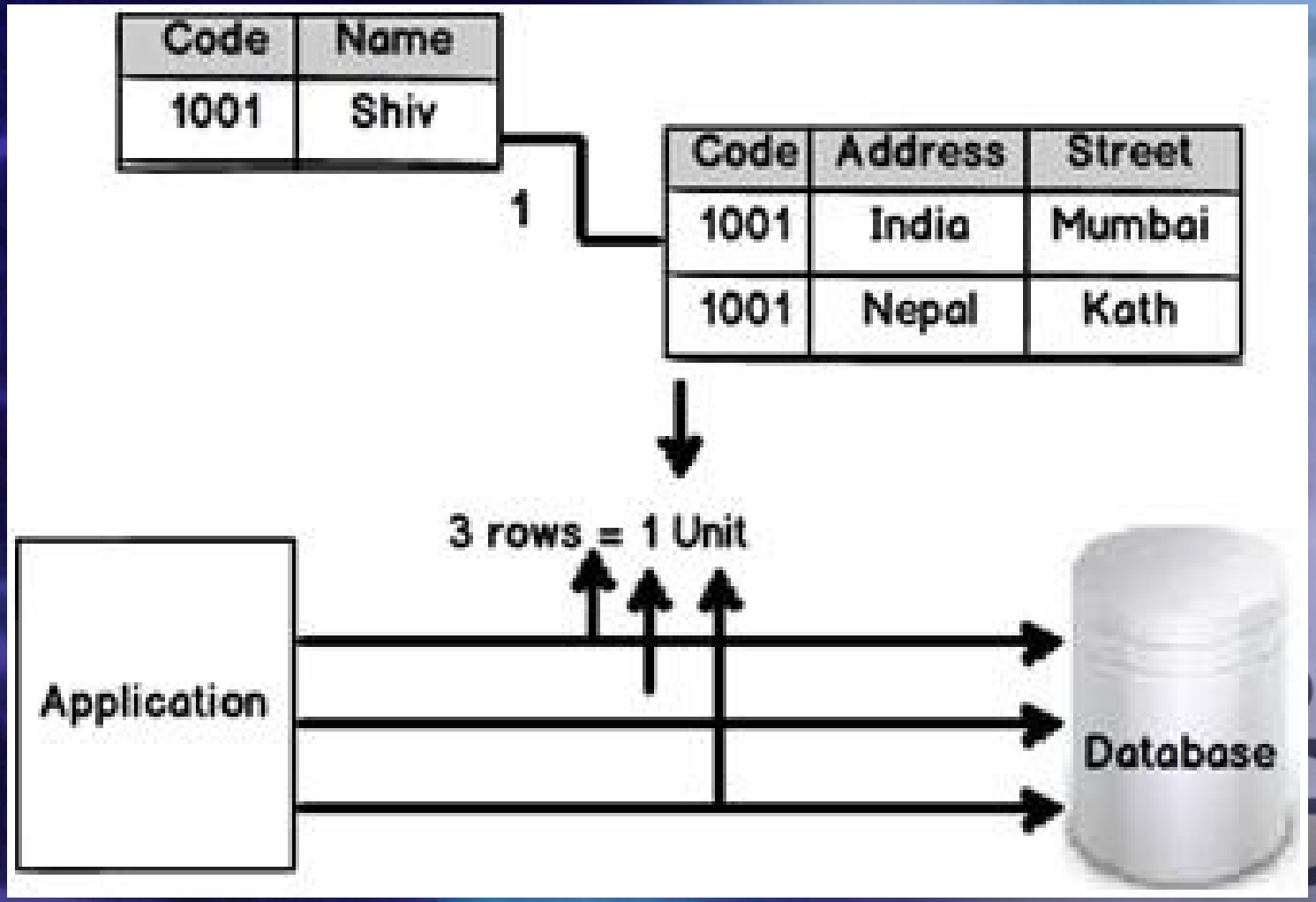
CRUD и Unit of Work

- Предното “уравнение” се променя значително в реални сценарии;
- Нека разгледаме сценарий, в който един потребител има няколко различни адреса;
- Тогава повече редове от базата данни ще са свързани с един unit of work. При 2 адреса получаваме:

3 Customer CRUD = 1 Logical unit of work

CRUD и Unit of Work

- 3 Customer CRUD = 1 Logical unit of work



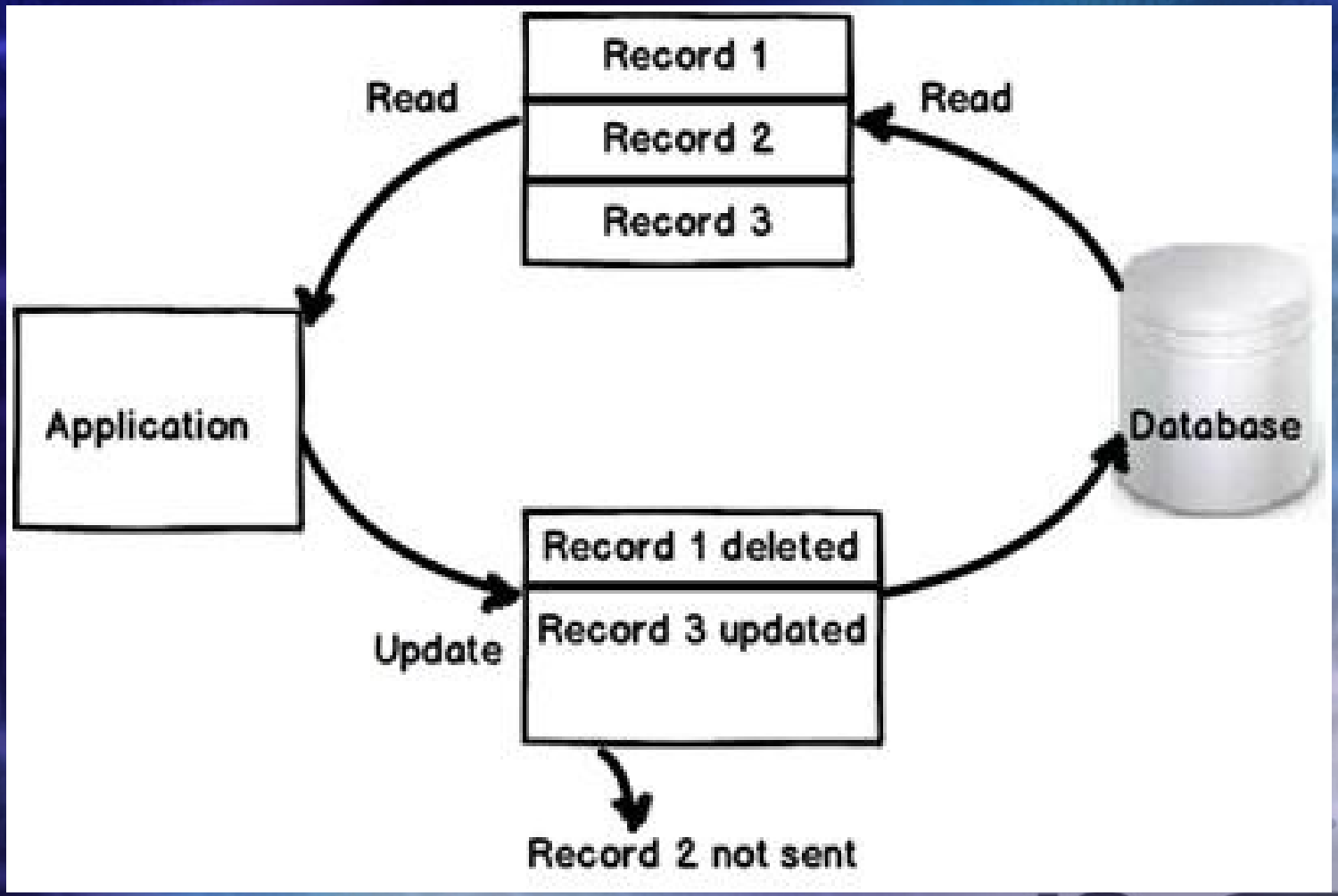
CRUD и Unit of Work

- С други думи, транзакциите за тези 3 записа трябва или всичките заедно да са успешни или заедно да са неуспешни;
- На практика получаваме, че е доста вероятно няколко CRUD операции да съответстват на един unit of work.

CRUD и Unit of Work

- Много разработчици може да заключат, че тези изисквания може да бъдат изпълнени чрез включване на всички CRUD операции в една транзакция. Но всъщност Unit of work представлява нещо много повече от обикновени транзакции към база данни. Всъщност, чрез него се изпращат единствено промените към базата данни;
- Нека например приложението изтегля три записа от базата данни, но модифицира само два. Така само променените записи се изпращат обратно към базата. Това оптимизира физическата натовареност на базата и увеличава производителността.

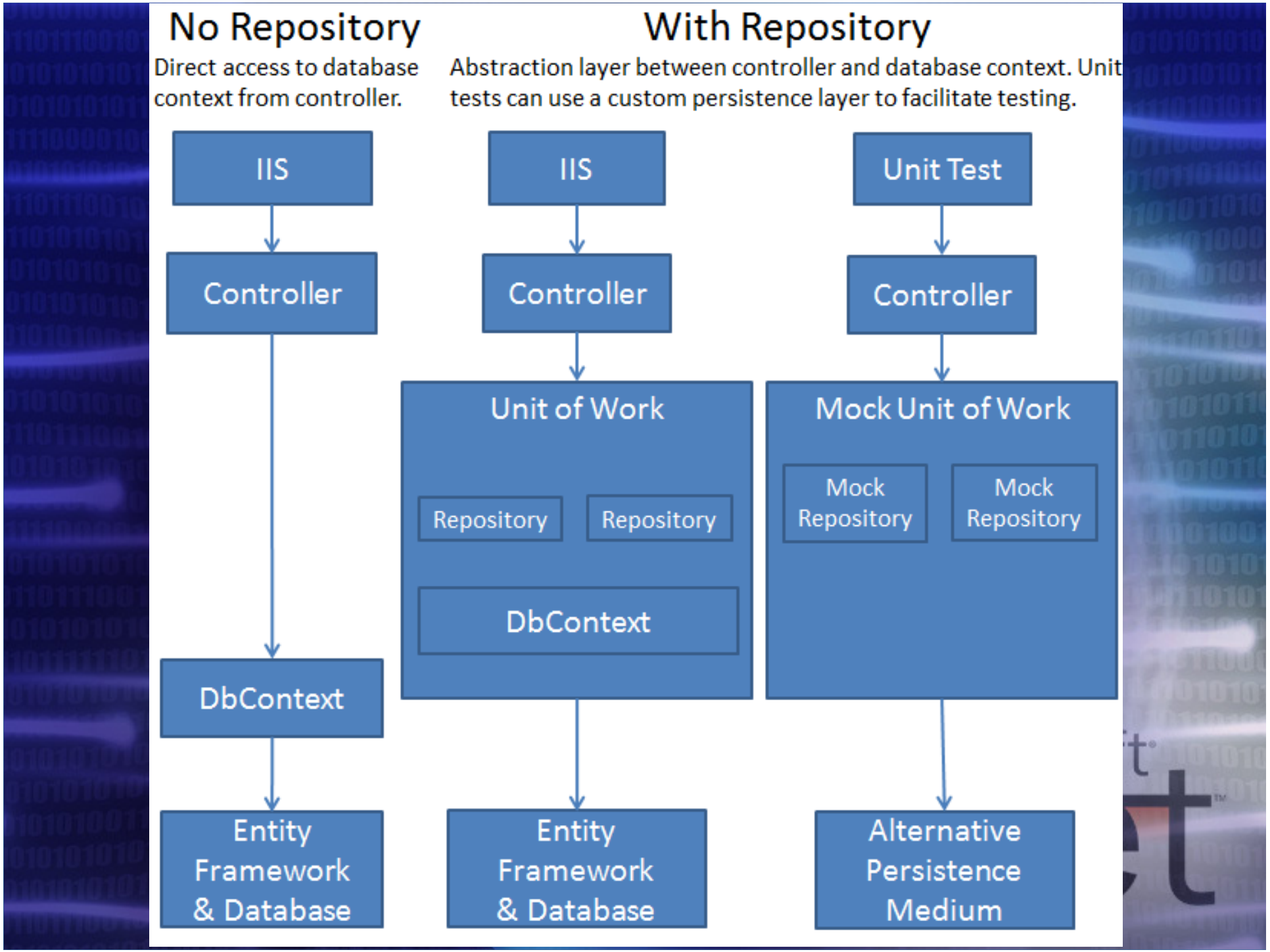
CRUD и Unit of Work



1 Unit of work = Modified records in a transaction

Repository и Unit of Work

- Двата шаблона служат за създаване на абстрактен слой между слоя за достъп до данни и слоя на бизнес логиката на дадено многослойно приложение.
- Прилагането на двата шаблона (съвместно или поотделно) помага за изолирането на приложението от промени в базата данни и може да улесни автоматизираното тестване или разработването чрез TDD (test-driven development) подход.

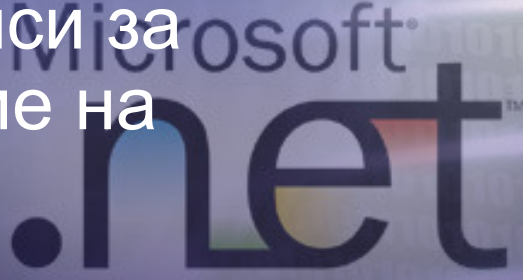

No Repository
Direct access to database context from controller.
IIS
Controller
DbContext
Entity Framework & Database
With Repository
Abstraction layer between controller and database context. Unit tests can use a custom persistence layer to facilitate testing.
IIS
Controller
Unit of Work
Repository
Repository
DbContext
Entity Framework & Database
Unit Test
Controller
Mock Unit of Work
Mock Repository
Mock Repository
Alternative Persistence Medium

Repository и Unit of Work

- На схемата, за всеки entity тип се имплементира repository клас;
- При необходимост, за дадено entity може да се създаде repository интерфейс и repository клас;
- Когато се инстанцира repository обект на вашия controller, той ще ползва този интерфейс, така че контролерът може да приеме референция към всеки обект, който имплементира repository интерфейса.
- Когато контролерът се стартира на уеб сървър, той получава repository обект, свързан с Entity Framework.
- Когато контролерът се стартира под unit test клас, той получава repository обект, свързан с данни, съхранени по начин, позволяващ лесно манипулиране във връзка с тестването (например колекция в паметта).

Repository и Unit of Work

- Могат да се използват няколко repositories и един unit of work клас за няколко entity типа в един контролер.
- Класът unit of work координира работата на няколкото repositories чрез създаване на единичен database context клас, споделен от всички тях.
- Ако е необходимо да се извърши автоматизирано тестване, може да се създадат и използват интерфейси за съответните класове, по подобие на примера от схемата.



Repository и Unit of Work

- Има много начини за прилагане на шаблоните Repository и Unit of work.
  - Може да се ползват repository класове с или без unit of work клас;
  - Може да се приложи едно repository за всички entity типове или по едно за всеки тип;
  - Ако се приложи repository за всеки тип, може да се използват отделни класове – общ базов клас и класове наследници или абстрактен базов клас и наследници;
  - Може да се включи бизнес логика в дадено repository или да има ограничение само за логика за достъп до данни;
  - Може да се създаде ниво на абстракция към съответния database context клас чрез използване на IDbSet интерфейси вместо DbSet типове за entity множествата.

Примерен Repository интерфейс

```
using System;
using System.Collections.Generic;

namespace StudentInterface {
    public interface IStudentRepository : IDisposable
    {
        IEnumerable<Student> GetStudents();
        Student GetStudentByID(int studentId);
        void InsertStudent(Student student);
        void DeleteStudent(int studentID);
        void UpdateStudent(Student student);
        void Save();
    }

}
```