Тема 11: Разработка на софтуер

**План-тезис**: Инструменти за разработка. Техники за продуктивно използване на интегрирана среда за разработка. Използване на външни библиотеки. Управление на пакети. Свързване на приложения с бази от данни. Създаване на приложения с няколко потребителски интерфейса.

**Инструменти за разработка**

Интегрирани среди за разработка(Integrated Development Environments(IDE)). Няма стандарти какво да притежава. Обикновено позволяват писане на код и компилиране и дебъгването му. Може да имат: модули за дизайн, свързани със системи за контрол на версиите, допълнителни модули и приставки.

Разлика между IDE и текстов редактор: IDE-тата имат интеграция с компилатор и дебъгер. Специализирани в един или няколко конкретни езика. Текстовите редактори поддържат редакция на код, но не и компилиране и дебъгване. Поддържат файлове от почти всички езици, имат модули за подчертаване и автоматично допълване.

**Използване на външни библиотеки**

Безплатни пакети, с отворен (open source) код. Лесно инсталиране и обновяване. Част от VS. Конфигуруеми източници на пакети. Лесно като добавяне на референция. Инсталатор с графичен дизайн. Налична конзола за управление на пакетите.

**Свързване на приложения с бази от данни**

Свързване към БД чрез C#

1. SqlClient:

Плюсове: Лесно управляван контрол на паралелни заявки. По голям шанс за работа с най - актуалната версия на информацията.

Минуси: Нуждае се от надеждна мрежова свързаност. Създава проблеми при скалирането.

1. ADO.NET:

ADO.NET e стандартна библиотека в .NET за достъп до данни и обработка на информация и XML. Поддържа свързани, несвързани и ORM модели за достъп до данни. Има добра интеграция с LINQ и позволява изпълнението на SQL. Позволява достъпване на информацията по ORM подхода. Доставчици на данни (Data providers) са различни библиотеки, които предоставят достъп до различни бази данни. Обикновено се дефинират 3 обекта: Connection (за свързване), Command(за изпълнение на SQL команда), DataReader(за извличане на данни).

Извличане на данни в свързан модел:

1. Отваряне на връзка (SqlConnection)
2. Изпълнение на команда (SqlCommand)
3. Обработка на множеството от резултати на заявката чрез четец (SqlDataReader)
4. Затваряне на четеца
5. Затваряне на връзката
6. ОRM(Object-Relational Mapping) Модел:

Съпоставя таблиците към класове и обекти. Обектите могат автоматично да бъдат съхранени в базата. Може да работи в свързан и несвързан режим. Извлича информацията от обектно-ориентирано API(LINQ заявки). Може да се създаде БД от обектен модел(code first) и обектен модел по базата(DB first).

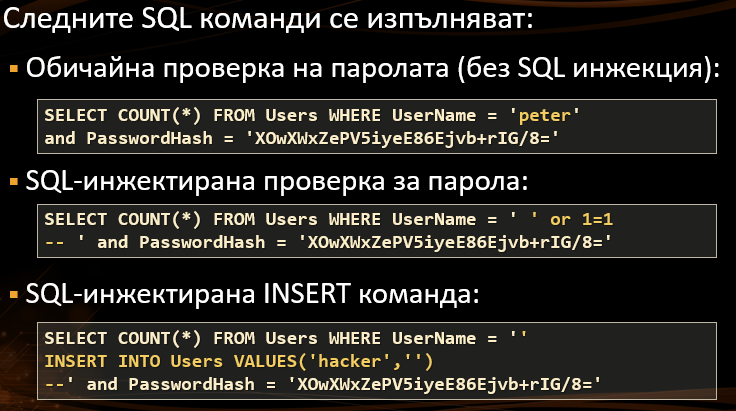
Плюсове: По малко код. Използва обекти с асоциации вместо таблици и SQL код. Интегриран механизъм за извличане на обекти.

Минуси: По-малко гъвкавост(SQL кодът се генерира автоматично).Не винаги ефикасен откъм производителност.

ADO.NET Еntity Framework Core:

ORM работна рамка с широко предназначение. Създава се модел, който съпоставя базата данни (entity data model). Отваря се обектен контекст (object context). Извличат се данни чрез LINQ / модифицира се таблицата в обектния контекст. Запазват се промените на обектния контекст в БД. Връзката се управлява автоматично

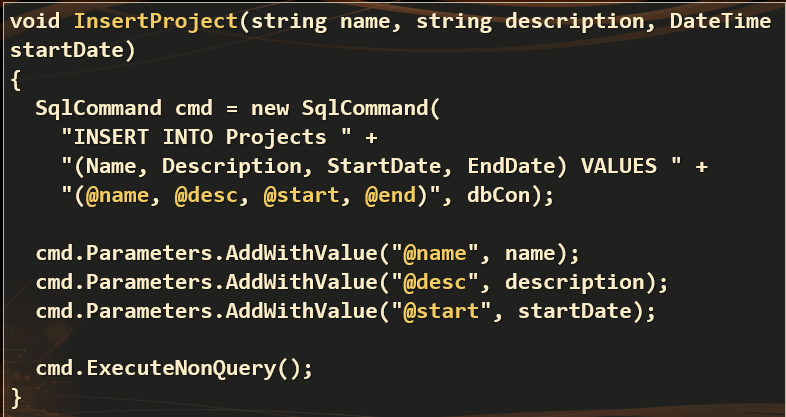
SQL Инжекция

****

Защита от SQL инжекции: Параметризирани заявки, отделяне на командата от нейните аргументи.

SqlParameter:

SQL заявките и съхранените процедури могат да имат входни и изходни параметри. Достъпват се чрез Parameters свойството на класа SqlCommand.



Класове в ORM

1. DbSet<T>:

Шаблонна колекция. Всеки DbSet съответства на таблица в БД. Имплементира ICollection<T>(foreach, LINQ).

Притежава функционалността на ICollection<T>(добавяне, премахване, проверка за съществуване, брой).

1. DbContext<T>:

Съставен от няколко DbSet<T>

Грижи се за попълването на DbSet-овете

Потребителя създава DbContext, който наследява DbContext.

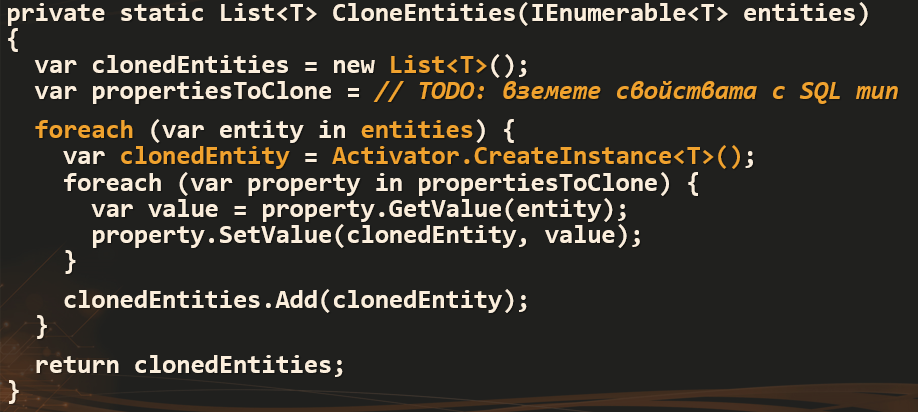
1. ChangeTracker<T>:

Контейнер за проследяване на промени.

Има 3 колекции: Всички данни, Добавени данни, Премахнати данни.

Проследява и променени данни(клониране при инициализация).

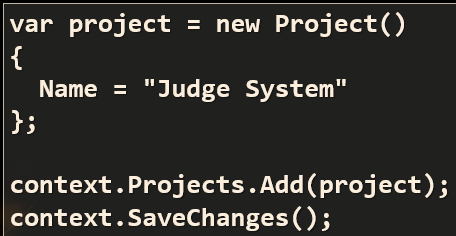
Клониране на данни



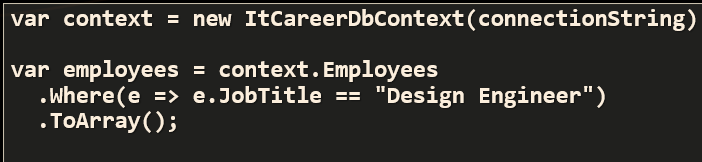
CRUD в ОRM

1. Инстанция на DbContext

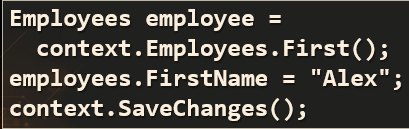
Create:



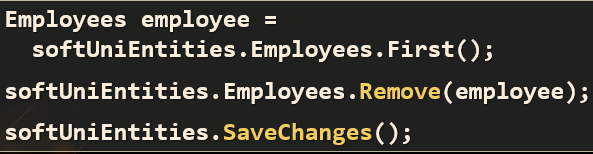
Read:



Update:



Delete:



**Създаване на приложения с няколко потребителски интерфейса**

ASP.NET

ASP.NET MVC е рамка за изграждане на уеб приложения.Базирана на MVC шаблона. Лесна и бърза за работа.Чудесен визуален интерфейс за нашето приложение

Winforms

Графичен потребителски интерфейс. Използва контролери за създаване на интерфейса.