



C# Developer.

Linq2DB, Dapper



Проверить, идет ли запись

Меня хорошо видно && слышно?



Ставим "+", если все хорошо
"-", если есть проблемы

Тема вебинара

Linq2DB, Dapper



Елена Сычева

Team Lead Full-Stack Developer

Об опыте:

Более 15 лет опыта работы разработчиком (C#, Angular, .Net, React, NodeJs)

Телефон / эл. почта / соц. сети:

<https://t.me/lentsych>



Правила вебинара



Активно
участвуем



Задаем вопрос
в чат или голосом



Вопросы вижу в чате,
могу ответить не сразу

Условные обозначения



Индивидуально



Время, необходимое
на активность



Пишем в чат



Говорим голосом




Документ



Ответьте себе или
задайте вопрос

Маршрут вебинара



ADO.Net

Что такое ORM

Зачем мне это

Dapper

Примеры

Рефлексия

Цели вебинара

К концу занятия вы сможете

1. Работать с базой с помощью Ado.Net
2. Объяснить что такое ORM, как выбрать подходящий
3. Писать запросы на Dapper

Смысл

Зачем вам это уметь

1. Иметь возможность выбрать технологию для работы с БД
2. Уметь читать/писать данные в БД



ADO.NET

Что это?

ADO.NET

ADO.NET по-прежнему остается актуальной и широко используемой технологией доступа к данным в приложениях .NET.

Он предоставляет набор классов для автономного доступа к источникам данных, таким как базы данных и XML-файлы.

ADO.NET является частью .NET Framework и .NET Core/.NET 5+ (теперь называемой просто .NET) и остается основополагающей технологией доступа к данным. Хотя появились новые технологии и платформы доступа к данным, такие как Entity Framework (EF) и Dapper, ADO.NET далеко не устарел.

ADO.NET

Производительность: ADO.NET предлагает детальный контроль над взаимодействием с базой данных, который может быть более эффективным для определенных сценариев с высокой производительностью по сравнению с инструментами ORM (объектно-реляционное сопоставление), такими как Entity Framework.

Гибкость: он предоставляет возможность выполнять необработанные SQL-запросы, хранимые процедуры и обрабатывать сложные транзакции, предлагая больший контроль над операциями с базой данных.

ADO.NET

Легкость: для приложений, не требующих накладных расходов на ORM, ADO.NET может быть более простым и понятным выбором.

Совместимость: ADO.NET совместим с широким спектром поставщиков данных, что делает его универсальным для доступа к различным типам баз данных.

Зрелая и стабильная. Будучи зрелой технологией, ADO.NET имеет хорошо зарекомендовавший себя и стабильный API с обширной документацией и поддержкой сообщества.

ADO.NET

| | | |
|----|----------------|---|
| 1. | SqlConnection | Подключение к БД с использованием Connection String |
| 2. | SqlCommand | Команда (обертка над запросом) для отправки на сервер Позволяет также подставлять параметры в запрос |
| 3. | SqlTransaction | Транзакция для выполнения нескольких запросов как единого целого |
| 4. | SqlDataReader | Удобное средство для чтения множества строк из БД |
| 5. | DataSet | Редко используемое (узкоспециализированное) средство для чтения таблицы БД и оперирования с ней оффлайн |

ADO.NET

ExecuteScalar — это метод в ADO.NET, который используется для выполнения запроса SQL и возврата первого столбца первой строки в наборе результатов, возвращаемом запросом. Любые другие столбцы и строки игнорируются. Этот метод обычно используется, когда вы хотите получить одно значение из базы данных, например число, сумму или значение определенного столбца из одной записи.

```
public static async Task<int> GetUserCountAsync(string connectionString)
{
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        await connection.OpenAsync();
        string query = "SELECT COUNT(*) FROM Users";

        using (SqlCommand command = new SqlCommand(query, connection))
        {
            // ExecuteScalar returns the first column of the first row in the result set
            object result = await command.ExecuteScalarAsync();
            if (result != null)
            {
                return Convert.ToInt32(result);
            }
            return 0;
        }
    }
}
```

ADO.NET async

SqlConnection.OpenAsync()
SqlCommand.ExecuteNonQueryAsync()
SqlCommand.ExecuteReaderAsync()
SqlCommand.ExecuteScalarAsync()
SqlDataReader.ReadAsync()

Преимущества и недостатки ADO.NET

- + Использование чистого SQL
- + Полный контроль за отправляемыми запросами
- Конвертация “С#-объект <-> SQL-представление” целиком на разработчике
- IDE не поможет с переименованиями
- Компилятор не поможет с контролем типов

Когда использовать ADO.NET

Сложные транзакции, требующие детального контроля над соединением, командами и транзакциями.

Ситуации, когда разработчику необходимо использовать расширенные функции поставщика базы данных.

Обширных пользовательских манипуляций с данными и тонкой настройки производительности на уровне взаимодействия с базой данных.

Минимальные внешние зависимости, поскольку ADO.NET является частью .NET Framework.

Пример

ORM.

Что это?

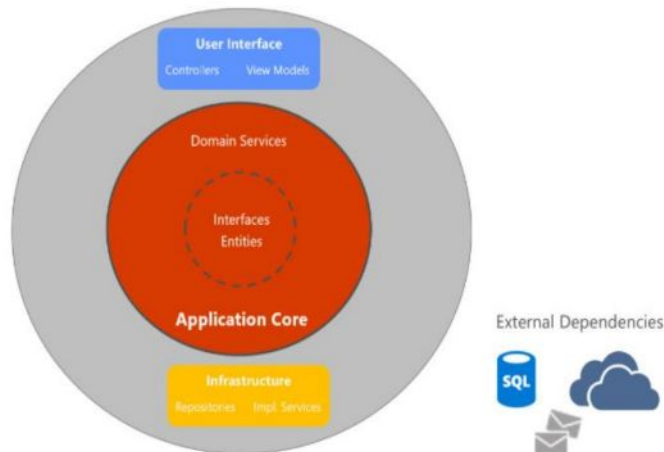
ORM

ORM (Object-Relational Mapping): объектно-реляционное отображение — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

ORM объединяет по своей сути различные парадигмы объектно-ориентированного программирования, где сущности представлены в виде классов и объектов, и реляционных баз данных, где данные хранятся в таблицах со строками и столбцами

Какие модели маппит ORM

Clean Architecture Layers (Onion view)

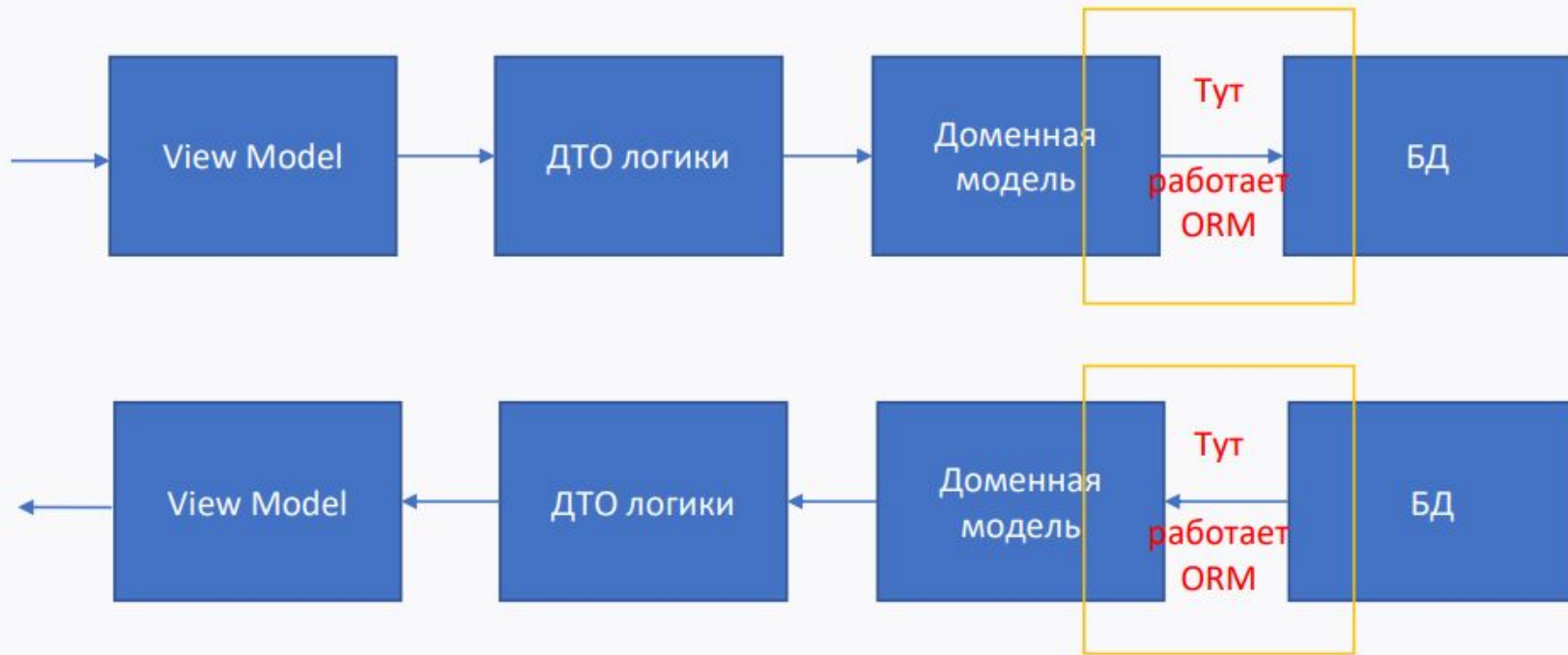


Виды моделей

1. Доменные модели (Entities), находятся в ядре
2. Модели логики (DTO), находятся в Domain Services
3. Модели для работы с клиентом (View Models), находятся в API или слое пользовательского интерфейса

К БД мапим Entities

Какие модели маппит ORM



Преимущества и недостатки ORM

Преимущества

- **Простота:** ORM предоставляет простой интерфейс для работы с базой данных, который может быть понятным любому программисту. ORM скрывает сложности SQL-запросов, позволяя работать с данными на более высоком уровне абстракции.
- **Переносимость:** ORM может работать с различными СУДБ, что делает его более переносимым, чем SQL. Это позволяет разработчикам легко переносить свое приложение на другую СУБД без изменения кода.
- **Сопровождаемость:** ORM может значительно упростить сопровождение приложения, так как изменения в структуре базы данных могут быть внесены непосредственно в код ORM, а не в каждый SQL-запрос.
- **Безопасность:** ORM может предотвратить SQL-инъекции, поскольку ORM автоматически экранирует данные, которые передаются в базу данных.

Преимущества и недостатки ORM

Недостатки

- **Сложность:** ORM может быть сложным для понимания, особенно для новых разработчиков. ORM требует определенных знаний и опыта, чтобы использовать его эффективно.
- **Производительность:** ORM может быть менее эффективным, чем работа с SQL напрямую. ORM должен обрабатывать запросы и преобразовывать их в SQL, что может замедлить производительность.
- **Ограничения:** ORM может иметь ограничения в отношении того, какие запросы могут быть выполнены. В случае, когда нужно выполнить сложный запрос или использовать специфичные функции базы данных, может потребоваться написание SQL-запроса напрямую.

Dapper

Что такое Dapper

Легкий микро-ORM, разработанный Stack Overflow, известный своей простотой и производительностью. Позволяет разработчикам выполнять необработанные SQL-запросы и сопоставлять результаты запросов с объектами с минимальными издержками.

Практика

Преимущества Dapper

- быстрый
- позволяет использовать хранимые процедуры
- видно какой именно SQL запрос вы исполняете

Сценарии использования Dapper

- приложения, критичные к производительности, которым требуется детальный контроль над взаимодействием с базой данных.
- когда требуются сложные запросы или оптимизация базы данных

LINQ2DB

LINQ2DB

LINQ to DB — это быстрая библиотека доступа к базе данных LINQ, предлагающая простой, легкий, быстрый и типобезопасный уровень между POCO-объектами и базой данных.

Архитектурно это на один шаг выше микро-ORM, таких как Dapper, вы работаете с выражениями LINQ. Запросы проверяются компилятором C#. Однако он не такой сложный, как Entity Framework. Отслеживание изменений не осуществляется, поэтому вам придется управлять этим самостоятельно, но есть и положительная сторона: вы получаете больше контроля и более быстрый доступ к своим данным. Другими словами, LINQ to DB — это типобезопасный SQL.

Преимущества

1. Проверка во время компиляции
2. Оптимизированное выполнение запросов
3. Поддержка нескольких поставщиков баз данных: LinqToDB поддерживает различных поставщиков баз данных, включая SQL Server, MySQL, PostgreSQL, SQLite, Oracle и другие.
4. Поддержка транзакций: LinqToDB обеспечивает встроенную поддержку транзакций, позволяя разработчикам выполнять атомарные операции с базой данных и обеспечивать согласованность данных.
5. Богатый набор функций
6. Расширенные параметры сопоставления

Когда использовать

1. Нужна высокая производительность
2. Сложные запросы
3. Независимость базы данных
- 4. запросы LINQ**
5. Microservices Architecture

Тестирование

Список материалов для изучения

1. <https://www.learndapper.com/>
2. <https://ling2db.github.io/>
3. <https://learn.microsoft.com/enus/dotnet/api/system.data.sqlclient.sqlconnection.connectionstring?view=dotnet-plat-ext7.0&viewFallbackFrom=net-6.0>
4. <https://martinfowler.com/eaCatalog/repository.html>
5. <https://learn.microsoft.com/en-us/ef/core/modeling/inheritance>
6. <https://www.entityframeworktutorial.net/efcore/configure-one-to-one-relationship-using-fluent-api-in-ef-core.aspx>
7. <https://www.entityframeworktutorial.net/efcore/configure-one-to-many-relationship-using-fluent-api-in-ef-core.aspx>
8. <https://www.entityframeworktutorial.net/efcore/configure-many-to-many-relationship-in-ef-core.aspx>
9. Мартин Фаулер. Архитектура корпоративных программных приложений

Вопросы?



Ставим "+",
если вопросы есть



Ставим "-",
если вопросов нет

Рефлексия

Цели вебинара

Проверка достижения целей

-
1. Назвать и объяснить основные принципы ФП
 2. Писать функциональный код на C#
-

Рефлексия



С какими впечатлениями уходите с вебинара?



Как будете применять на практике то, что узнали на вебинаре?