NET2 - 10

Databas och C# - ORM

Förra gången

- Databasdesign - Parkeringshuset
  - Normalisering
  - Primary/Foreign key
  - JOIN-exempel
- ER-diagram
  - Lucidchart
- Övning – Bygg bokhandel

Idag

- Kort redovisning av Bokhandeln
- Ansluta till en databas med C#
  - ADO
- Object-relational mapping (ORM)
  - Vad är det?
  - Fördelar och nackdelar
  - Olika ramverk
    - Dapper
    - Entity Framework
  - Code-along med Dapper: Parkeringshuset2
  - Övningsuppgift
    - Fortsätta med parkeringshuset

ADO - ActiveX Data Objects

- ADO är ett programmeringsgränssnitt för att komma åt data i en databas
  - Äldre teknik som fortfarande används
  - Microsoft
- Det vanliga sättet att komma åt en databas inifrån en ASP-sida är att:
  - Skapa en ADO-anslutning till en databas
  - Öppna databasanslutningen
  - Extrahera de data du behöver från postmängden
  - Stäng postmängden
  - Stäng anslutningen

ADO - Ansluta till databas

- Connection string:
  - `var connString = "data source=.\\SQLEXPRESS; initial catalog = Parking10; persist security info = True; Integrated Security = True;";`
- SQL
  - `var sql = "SELECT * FROM Cars";`
- Ansluta till databasen
  - `var connection = new SqlConnection(connString)`
  - `connection.Open();`
- Göra något med databasen
  - `var command = new SqlCommand(sql, connection)`
  - `var reader = command.ExecuteReader()`
- Läsa svaret från databasen
  - `reader.Read()`

Connection string

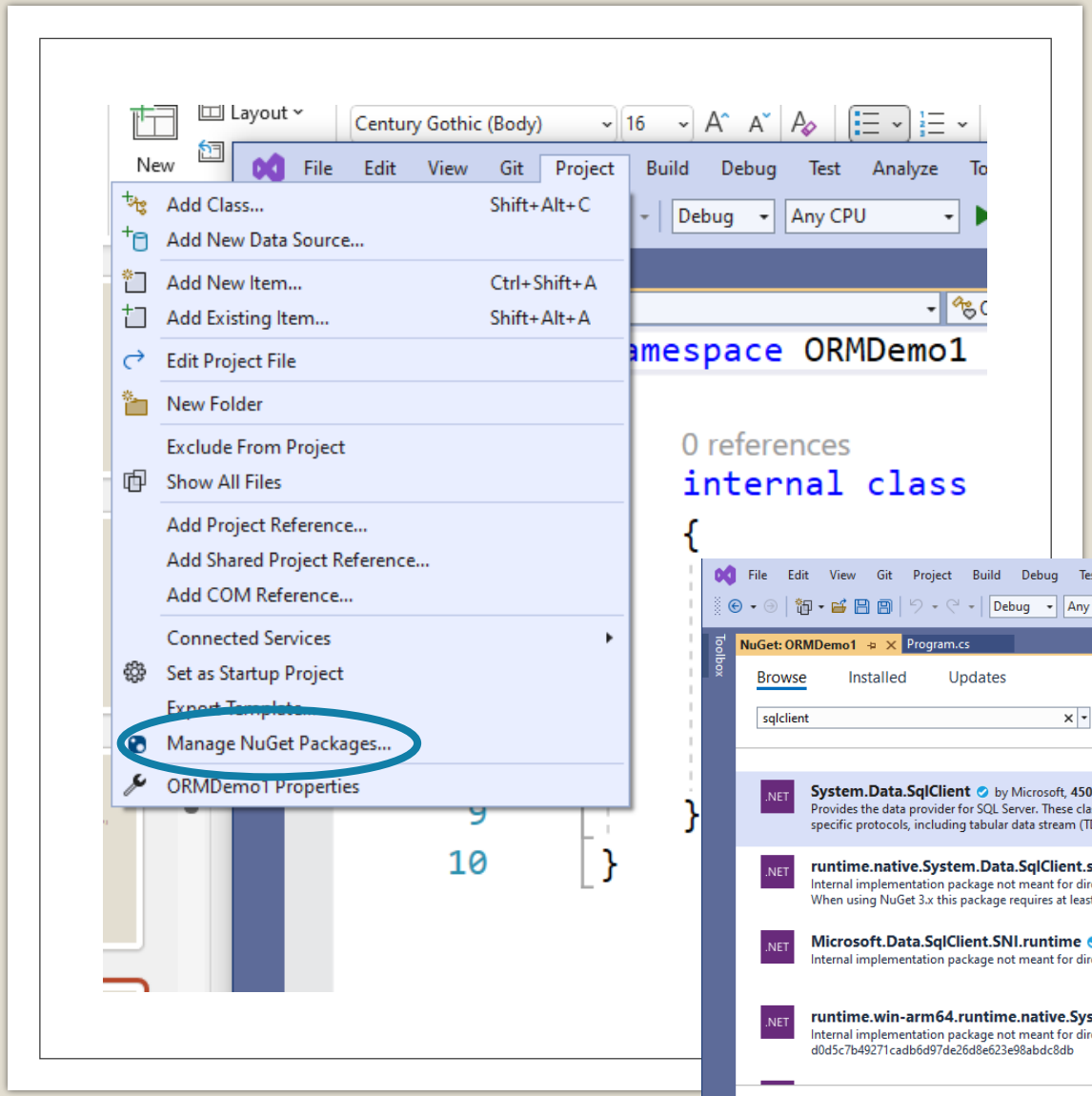
- `data source=.\\SQLEXPRESS; initial catalog = <Databasnamnet>; persist security info = True; Integrated Security = True;`

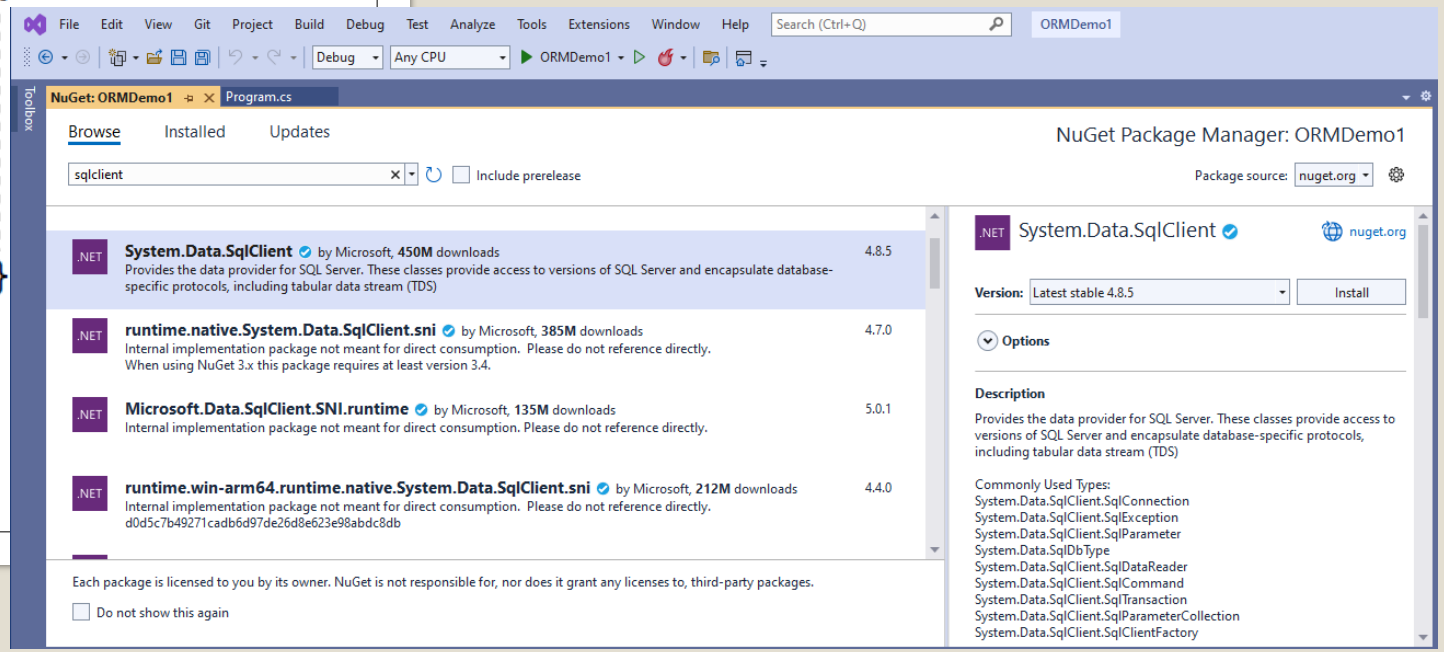
- Data source = Databasservern
- Initial catalog = Databasens namn
- persist security info = Komma ihåg lösenordet
- Integrated Security = Windows-inloggning

ADO – läsa från databasen

Paket

- Nuget Packages
  - Färdiga kod-paket skrivna av både Microsoft, och andra utvecklare





Lite C#

- Databasfält kan vara NULL, vilket C# inte gillar.
- För att markera att ett värde FÅR vara NULL, så anger vi det med ett ?-tecken vid datatypen.
  - Exempel: `public int? ParkingSlotsId { get; set; }`

- Korrekt hantering av instanser av t ex databaskopplingen

```
using (var connection = new SqlConnection(connString))
{
    // Kod som använder connection
}
```

Demo – Klassisk ADO

- Dokument: ORMDemo1.zip

ADO

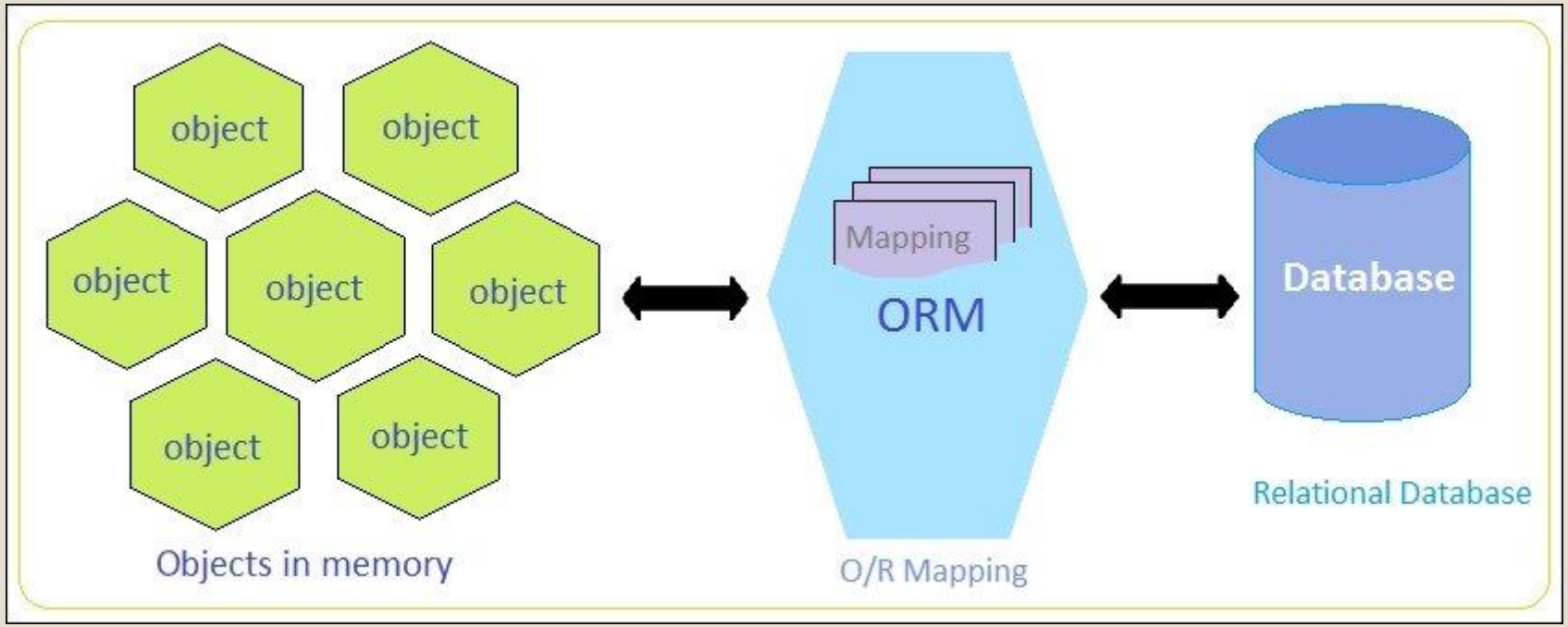
- Gammalt
- Krångligt
- Mer och mer ovanligt
- Koppla databasen med object sker manuellt

- Lösningen
  - Object relational mapping

Vad är ORM?

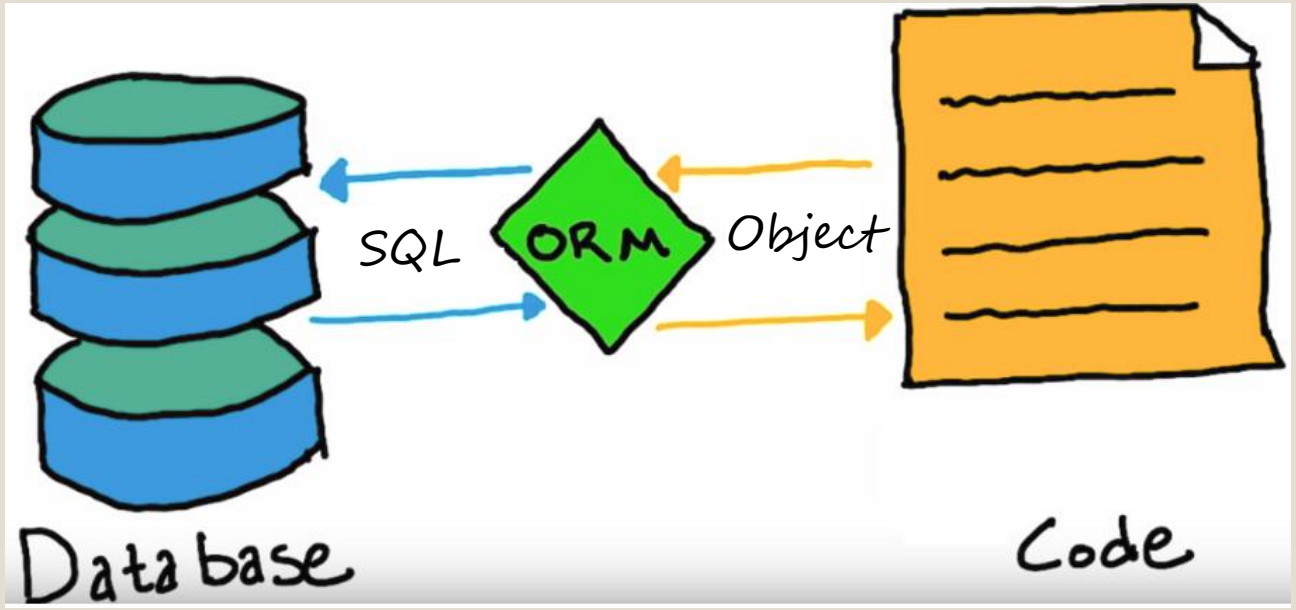
- **Object-relational mapping** (ORM) är ett verktyg för att koppla tabeller i en databas till klasser och objekt i sin kod.
- På det sättet blir det lättare att manipulera databasen när man programmerar.
- Syftet är att man lätt ska få tillgång och hantera databasen när man programmerar en applikation
- Istället för att skriva ren SQL-frågor för att hämta ut data från databasen, använder man sig istället av sitt vanliga programmeringsspråk för att fråga databasen.
- Oftast krävs det att man mappar ut tabeller i databasen till klasser och objekt i koden.
- Dessutom gör man det möjligt i koden att öppna upp en anslutning till databasen så att man kan manipulera data

ORM


object
object
object
object
object
object
object
object
Objects in memory
Mapping
ORM
O/R Mapping
Database
Relational Database

Enkelt uttryckt

- ORM är ett sätt att koppla ihop vår vanliga C#-kod med en databas.



Fördelar och nackdelar

- Fördelar
  - Minskar utvecklingstiden - eliminerar behovet av repetitiv SQL-kod.
  - Minskar utvecklingskostnaderna.
  - Övervinner leverantörsspecifika SQL-skillnader - ORM vet hur man skriver leverantörsspecifik SQL så att du inte behöver
  - Du kan koda i välbekant C#
- Nackdelar:
  - Förlust av utvecklarproduktivitet medan de lär sig att programmera med ORM.
  - Utvecklare förlorar förståelsen för vad koden faktiskt gör - utvecklaren har mer kontroll med SQL.
  - ORM har en tendens att vara långsam.
  - ORM klarar SQL-frågor för komplexa frågor sämre

Olika ORM-ramverk

- Dapper:
  - Ett micro-ORM som fokuserar på prestanda, kan kombineras ihop med andra ORM.
  - Anses vara den snabbaste ORM:et på marknaden för .NET.
  - Den är skapad av Stack Overflow och den är ett öppet källkodsprojekt.
  - Dapper saknar många funktioner som andra ORM har.
  - Det viktigaste med Dapper är prestandan, den är väldig snabb på att hämta ut data från databasen.
  - Eftersom den saknar funktioner som andra ORM har så rekommenderas det att använda Dapper i en applikation där man främst endast hämtar ut data från en databas.

Olika ORM-ramverk

- Entity Framework:
  - Entity Framework (EF) är Microsofts ORM för .NET som släpptes för första gången 2008.
  - Tidigare hade EF bara stöd för Microsofts egna databaser som Microsoft SQL Server. Numera finns det paket som kan installeras som ger stöd för många andra typer av databaser.
  - Väldigt mycket händer under huven med Entity Framework. Den skapar konfigurationer, mappar ut tabeller till objekt med mera genom att autogenerera kod med hjälp av några knapptryckningar. Detta gör att en utvecklare sparar mer tid och gör det även lättare för en utvecklare att använda sig av ORM:et då en inte behöver lika stor kunskap om hur allting fungerar.
  - EF får dock en del kritik från utvecklare just på grund av att EF skapar kod automatiskt. Man får mindre kontroll på sin egen kod då man måste förlita sig på kod som skapas av en maskin. Det blir till exempel svårare att debugga koden om själv inte har skrivit den eller kan förstå sig på den
  - EF pratar vi mer om snart…

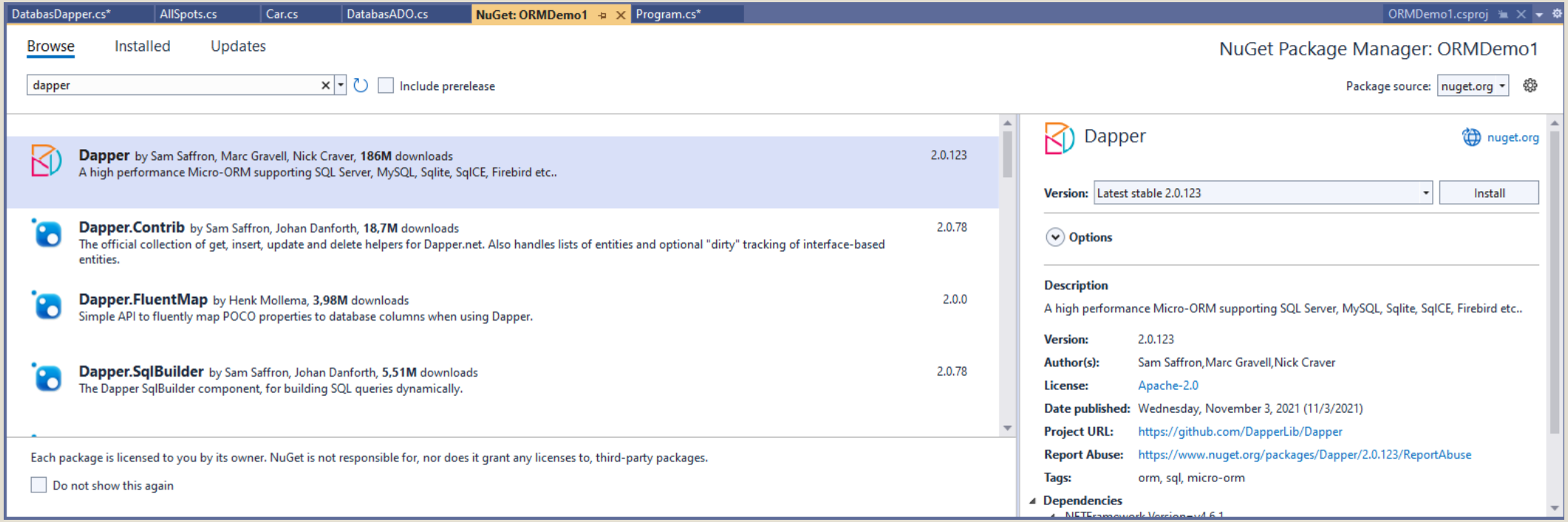
Dapper exempel

```
var sql = "select * from products";
var products = new List<Product>();
using (var connection = new SqlConnection(connString))
{
                connection.Open();
                using (var command = new SqlCommand(sql, connection))
                {
                                using (var reader = command.ExecuteReader())
                                {
                                                var product = new Product
                                                {
                                                                ProductId = reader.GetInt32(reader.GetOrdinal("ProductId")),
                                                                ProductName = reader.GetString(reader.GetOrdinal("ProductName")),
                                                                SupplierId = reader.GetInt32(reader.GetOrdinal("SupplierId")),
                                                                CategoryId = reader.GetInt32(reader.GetOrdinal("CategoryId")),
                                                                QuantityPerUnit = reader.GetString(reader.GetOrdinal("QuantityPerUnit")),
                                                                UnitPrice = reader.GetDecimal(reader.GetOrdinal("UnitPrice")),
                                                                UnitsInStock = reader.GetInt16(reader.GetOrdinal("UnitsInStock")),
                                                };
                                                products.Add(product);
                                }
                }
}
```

Med Dapper ersätts det grårmarkerade, med koden nedan

```
products = connection.Query<Product>(sql);
```

Paket - Dapper



Lägg till detta där du vill använda Dapper: `using Dapper;`

Demo – ORM: Dapper

- Dokument: ORMDemo1.zip

Övningsuppgift - Grupp – Fortsätta Parkeringshuset

- I Dokumenter ORMDemo1.zip(från dagens demo) finns en del lösningar fixade och I tidigare dokumentet DeluxeParkeringSQL.sql finns en del av den SQL ni behöver
- Din uppgift är att bygga vidare på detta, I en console-app
  - Skapa ett helt nytt VS-document, och använd dagens demo enbart som inspiration.
  - Funktioner I applikationen:
    - Kunna se lista på städer, och lägga till städer
    - Kunna se lista på parkeringshus och lägga till parkeringshus och parkeringsplatser (samt ange om platsen har eluttag)
    - Kunna växla mellan olika städer och parkeringshus för att se vilka platser som är lediga
    - Kunna välja en bil och ett parkeringshus + plats för att parkera bilen
    - Kunna välja en bil för att "köra iväg" bilen
    - Kunna lägga till ny bil, via inmatning
    - Kunna se följande vyer, I console-appen:
      - `Antal Elplatser per parkeringshus`
      - `Antal Elplatser per stad`
      - `Se Parkerade bilar`
      - `Lediga platser`

Länkar

- Welcome To Learn Dapper | Learn Dapper
- Executing Non Query Commands With Dapper | Learn Dapper
- The advantages and disadvantages of using ORM - Stack Overflow
- What is an ORM and Why You Should Use it | by Mario Hoyos | Bits and Pieces (bitsrc.io)

- using statement - C# Reference | Microsoft Learn