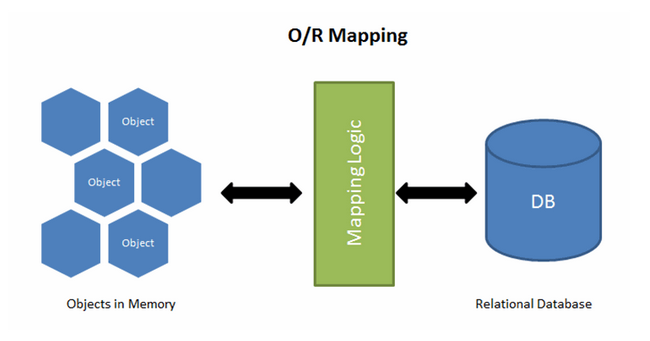
**ORM – Object Relational Mapping**Narzędzie mapowania obiektowo relacyjnego



**Entity Framework**

Generuje modele zgodne z tabelami baz danych  
Możemy mapować obiekt np. Student na odpowiadającą mu tabelę w bazie danych.  
Nazwy mogą być takie same, ale mogą być również różne

Możemy pisać logikę w C#, która będzie pod spodem zamieniana na kwerendy SQL.  
**Inne frameworki** ADO .NET, NHibernate, dopper

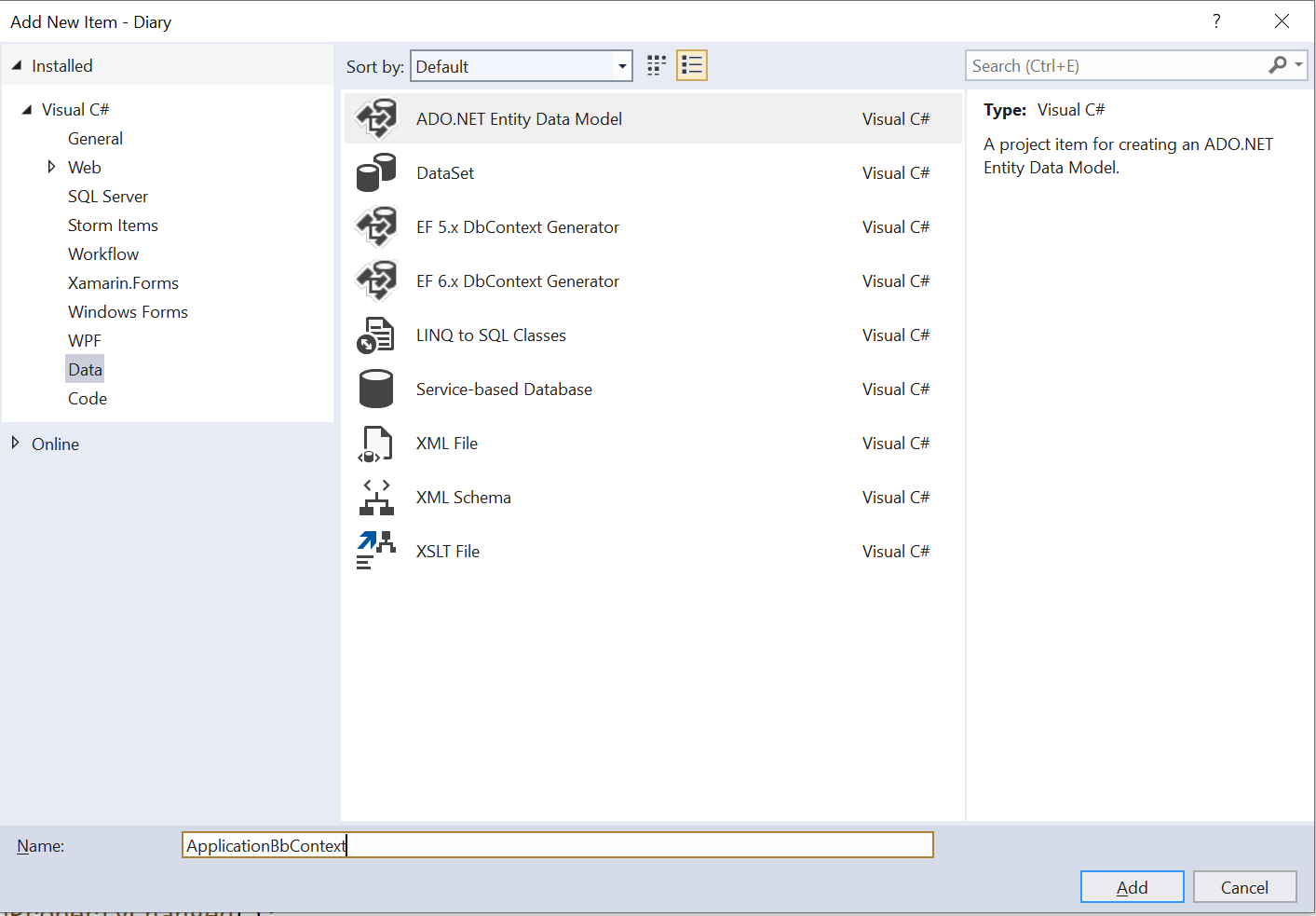
 **NHibernate** jest rozwiązaniem ORM dla platformy .NET. Dostarcza środowisko do mapowania obiektowo-relacyjnego dla tradycyjnego, relacyjnego modelu bazy danych. Jego podstawową funkcją jest mapowanie z klas platformy .NET do tabeli baz danych oraz od typów danych CLR do typów danych SQL.

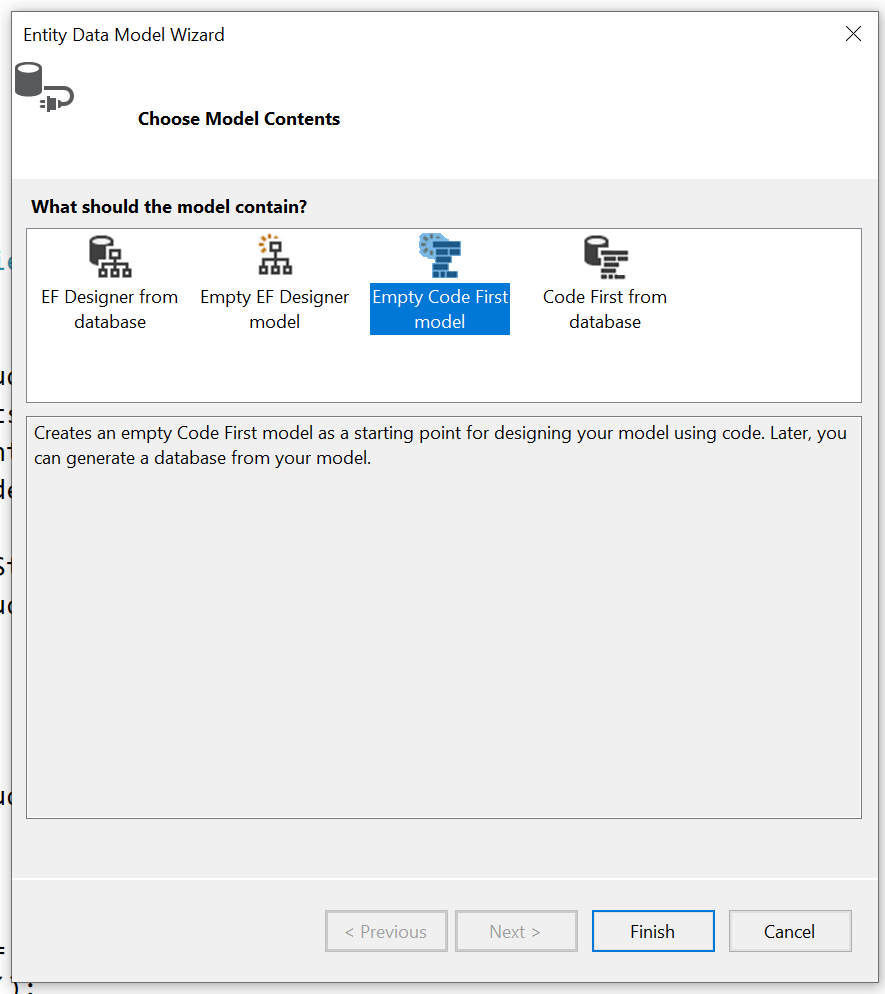


Dapper jest przykładem ORMa, którego funkcjonalność jest ograniczona do minimum. Robi on tylko i aż mapowanie pomiędzy zapytaniami i obiektami w kodzie. Jednak o to jak wyglądać będzie zapytanie musimy zatroszczyć się sami. Dapper wykonuje jedynie kod SQL, który sami napiszemy. Jego odpowiedzialnością jest jedynie zamiana zwróconych wartości na obiekty w kodzie C# albo wstawienie wartości z kodu C# do zapytania SQL. Dzięki temu mamy pełną kontrolę nad tym jak wyglądać będzie zapytanie. Efektem minimalnej funkcjonalności Dappera jest jego wydajność. Dapper do czasu odświeżenia Entity Frameworka wraz z nadejściem wersji Core był nie do doścignięcia przez standardowe ORMy jeżeli chodzi o czas wykonania zapytań. Nadal jest w czołówce.

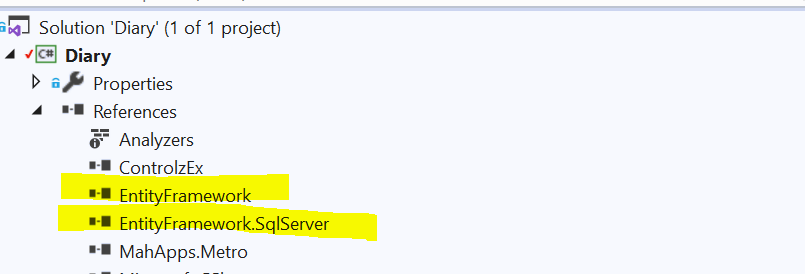
Code First – gdy nie mamy jeszcze bazy  
Code first from database – gdy mamy już bazę danych  
Database First – gdy mamy już bazę  
Model First – najgorsze podejście

**Instalacja**





Referencje powstałe po instalacji



Główna klasa

|  |
| --- |
| namespace Diary  {  public class ApplicationBbContext : DbContext  {  public ApplicationBbContext()  : base("name=ApplicationBbContext")  {  }  }  } |

W App.config wpisujemy connectionString

|  |
| --- |
| <connectionStrings>  <add name="ApplicationBbContext" connectionString="data source=(LocalDb)\MSSQLLocalDB;initial catalog=Diary.ApplicationBbContext;integrated security=True;MultipleActiveResultSets=True;App=EntityFramework" providerName="System.Data.SqlClient" />  </connectionStrings> |

|  |
| --- |
| <connectionStrings>  <add name="ApplicationBbContext"  connectionString="Driver={SQL Server Native Client 11.0};  Server=127.0.0.1;  Database=Diary;  Uid=user1;  Pwd=alamakota;" />  </connectionStrings> |

|  |  |
| --- | --- |
| W katalogu Models | tworzymy podkatalogi: |
| **Domains** | wszystkie obiekty domenowe na podstawie których będziemy tworzyć tabele w bazie danych |
| **Wrappers** | modele, które będziemy używać w aplikacji , właściwości i metody |
| **Configurations** | konfiguracje obiektów domenowych |
| **Converters** | konwertery obiekty domenowe na wrappery i odwrortnie |