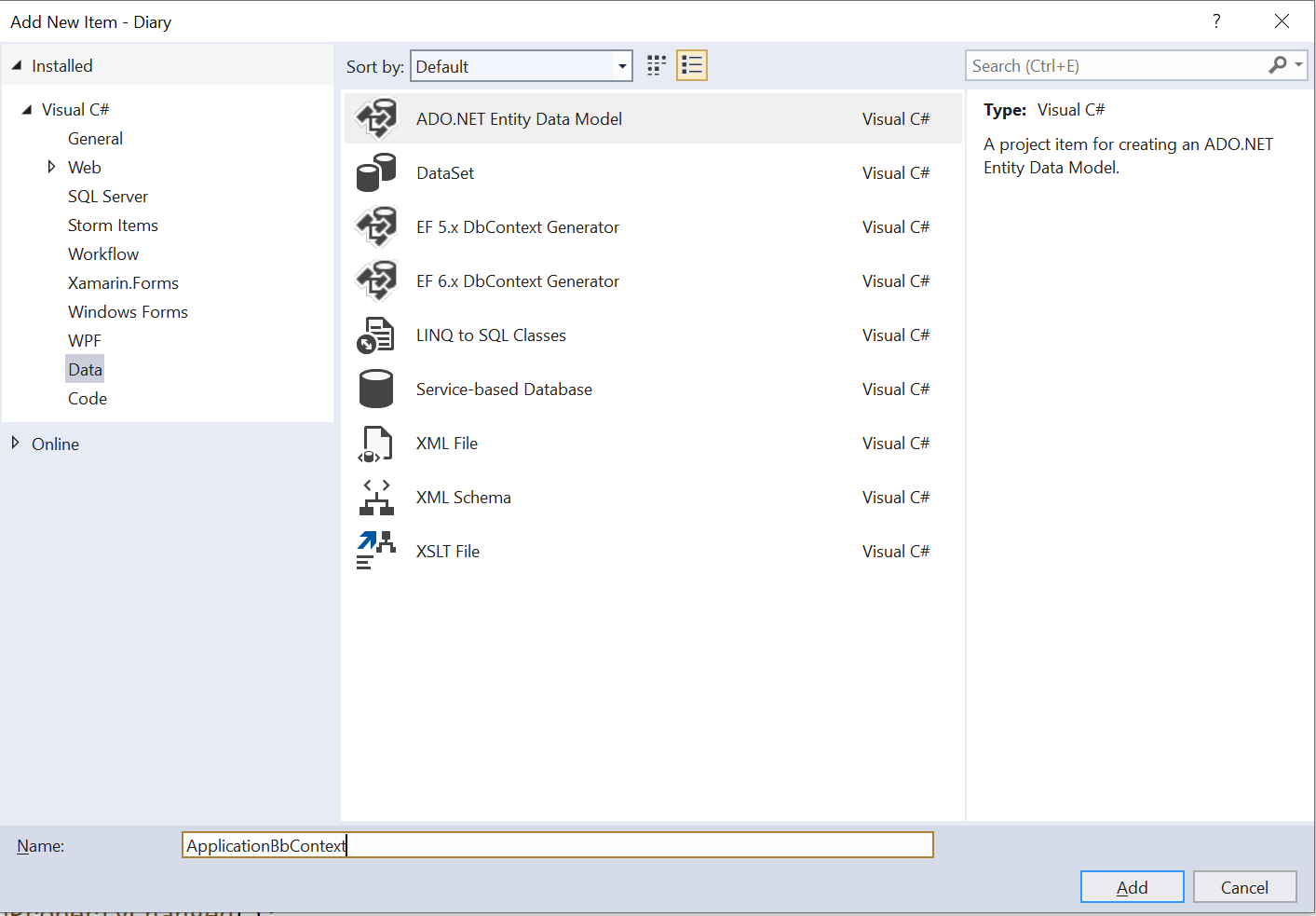
Entity Framework

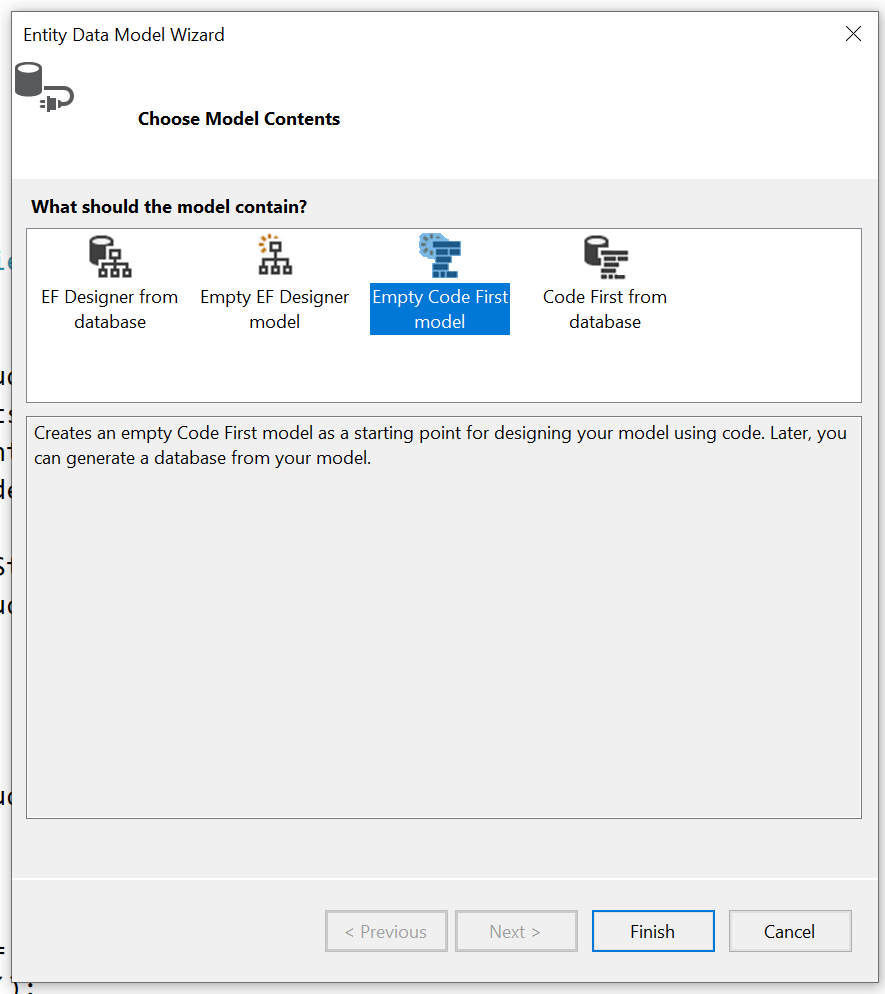
ORM – narzędzie mapowania obiektowo relacyjnego  
Generuje modele zgodne z tabelami baz danych  
Możemy mapować obiekt np. Student na odpowiadającą mu tabelę w bazie danych.  
Nazwy mogą być takie same, ale mogą być również różne

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

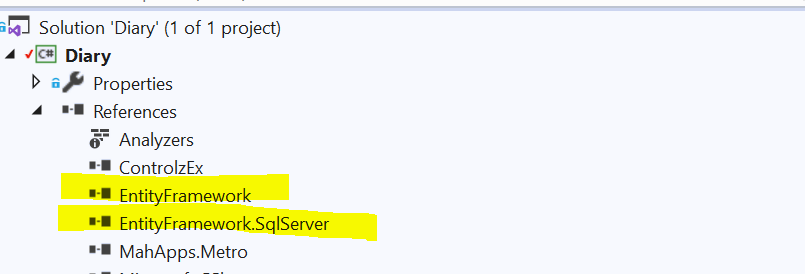
Code First – gdy nie mamy jeszcze bazy  
Code first from database – gdy mamy już bazę danych  
Database First – gdy mamy już bazę  
Model First – najgorsze podejście

**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 modele domenowe na podstawie których będziemy tworzyć tabele w bazie danych

Wrappers modele, które nbędziemu używać w aplikacji , właściwości i metody

Configurations – konfiguracje obiektów domenowych

Converters – konwertery obiekty domenowe na wrappery i odwrortnie