The Learning House

427 S. 4th Street #300

Louisville KY 40202

Entity Framework Database First

.NET Cohort

Coding Bootcamp

SOFTWARE GUILD

Lesson Goals

- What's an ORM?
- Learn to install Entity Framework from NuGet
- Build a simple model using the wizard
- Load some data from the database

Impedance Mismatch

- The problem many OO developers face is that object-oriented relationships do not always cleanly map to database structures.
- Thus, there is often a lot of plumbing work to be done to translate table schema to our objects.
- This code is tedious and error prone.

ORM – Object Relational Mapper

- An ORM, like Entity Framework, is designed to automatically generate classes and plumbing to map code objects to database schema and manage the relational mapping between them.
- Entity Framework 5.0 can start with an existing database, an object model, or code first.
  - ADO.NET, the .NET data frameworks, runs under the covers

Strengths and Weaknesses of ORM

Strengths

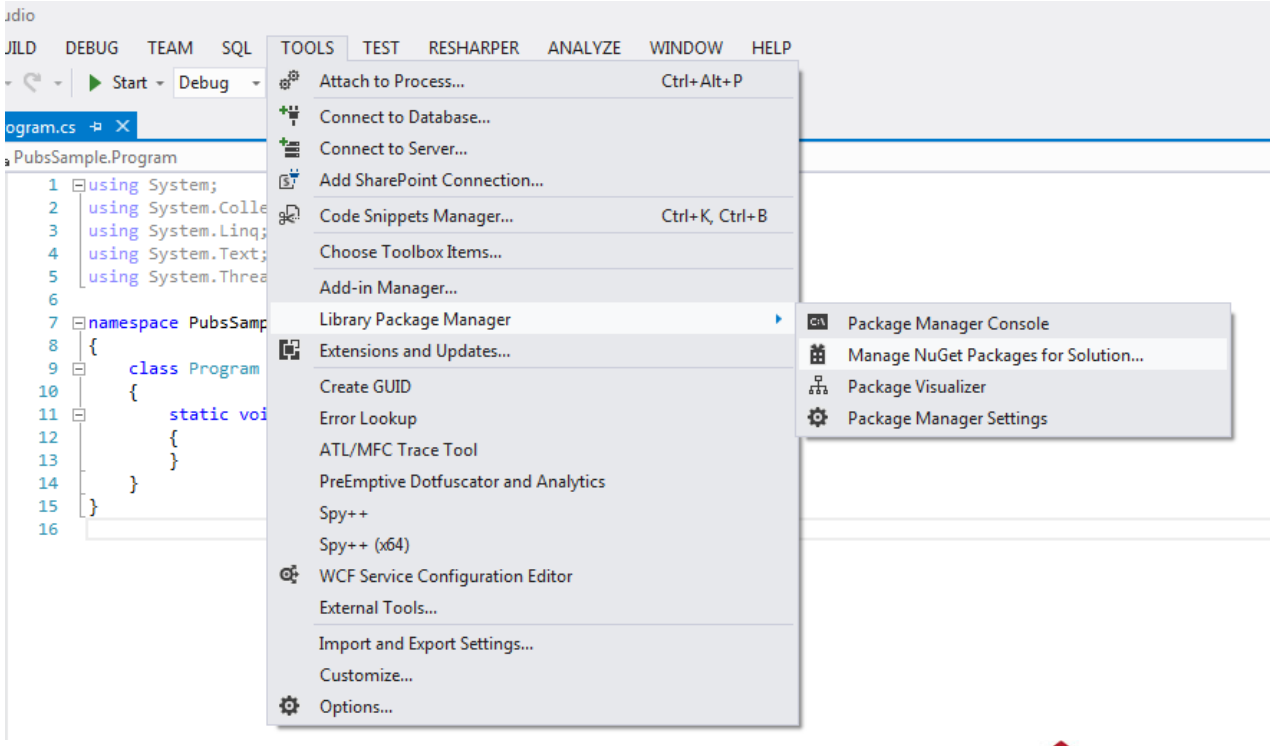
- Quickly able to generate data classes and get data into your application
- In simple cases, knowledge of SQL and database schema is not necessary to be effective
- Things like transaction management are much easier

Weaknesses

- Does not scale well, making it challenging for high-performance scenarios
- Getting down to the query level for granular control is difficult
- Database administrators tend to hate it because it makes performance optimization more difficult
  - Some queries and statements generate awful code.

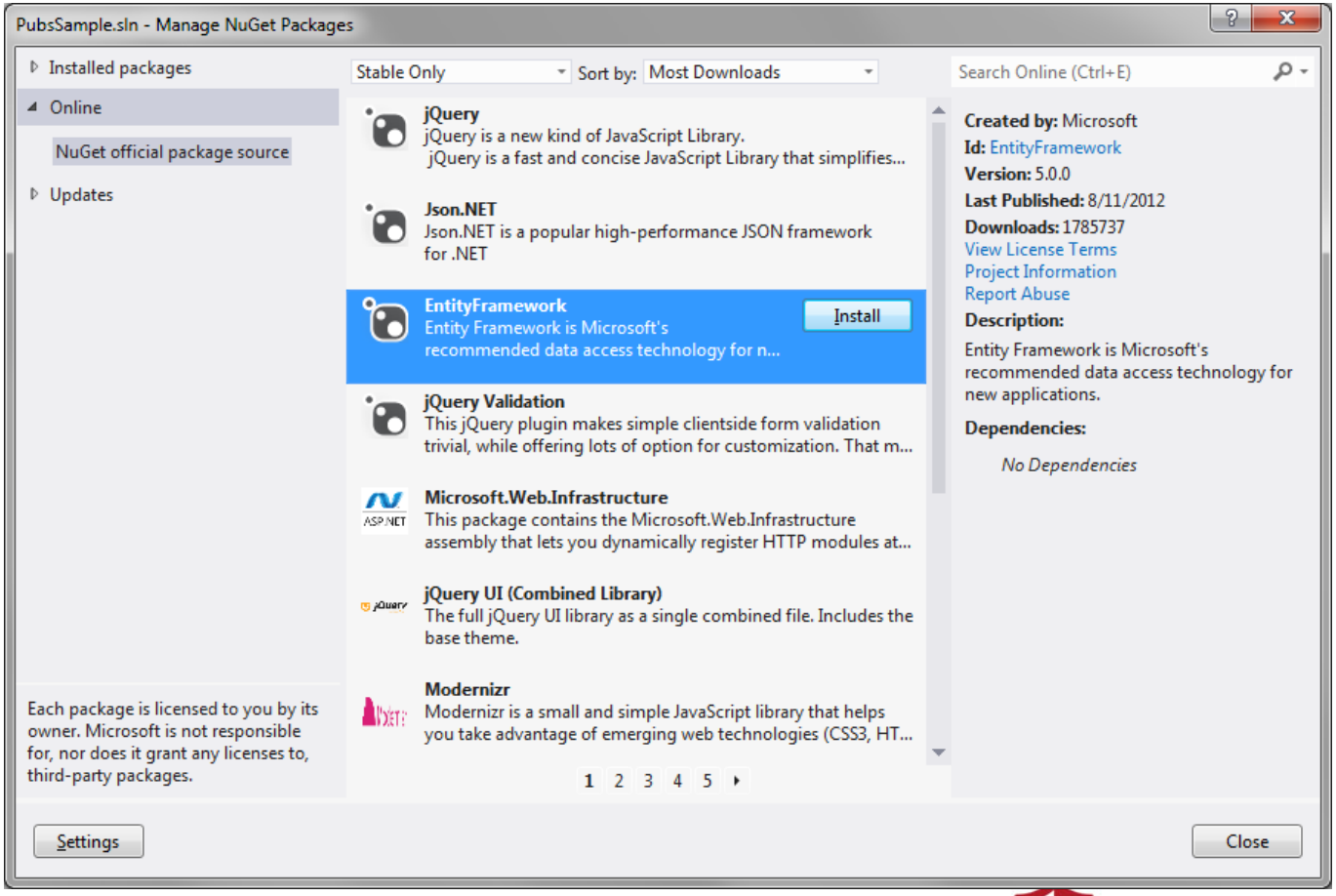
Adding Entity Framework

We can get the latest version of the Entity Framework from NuGet



SOFTWARE GUILD

NuGet has all sorts of useful goodies; you can search in the upper right and install a package by hitting the Install button.



SOFTWARE GUILD

Next Steps

1. Add a model to the project
2. Use the Entity Data Model Wizard
3. Edit the model
4. Write the client code

Every database-first Entity Framework app follows these same steps.

Add a model

- To add a model to the project:
  - Right-click the project and choose Add New Item
  - In the Add New Item dialog, click on the Data template section on the left and choose ADO.NET Entity Data Model
  - Give it a name (in our case, we're going to use Northwind's database, so we will call it Northwind.edmx)

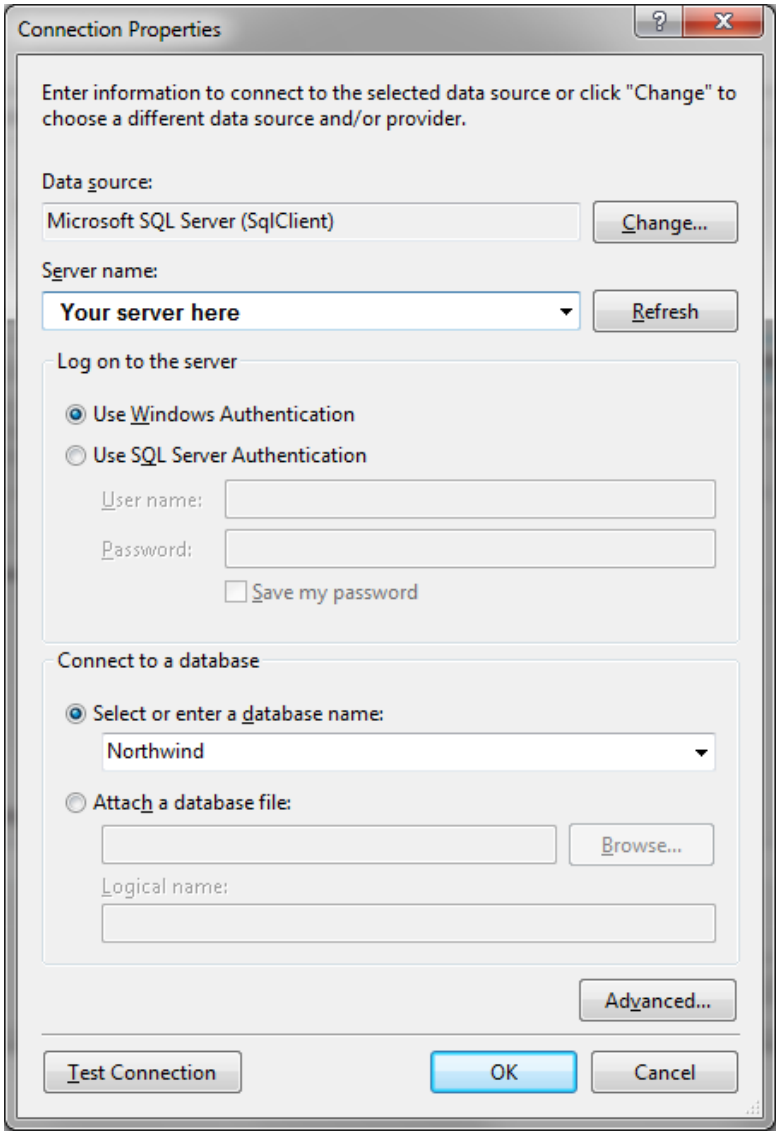
Entity Data Model Wizard

- Adding the model sends us into the wizard:
  - Choose Generate From Database
  - Set up a connection
  - Choose which database objects you want to bring into the model

Setting up a Connection

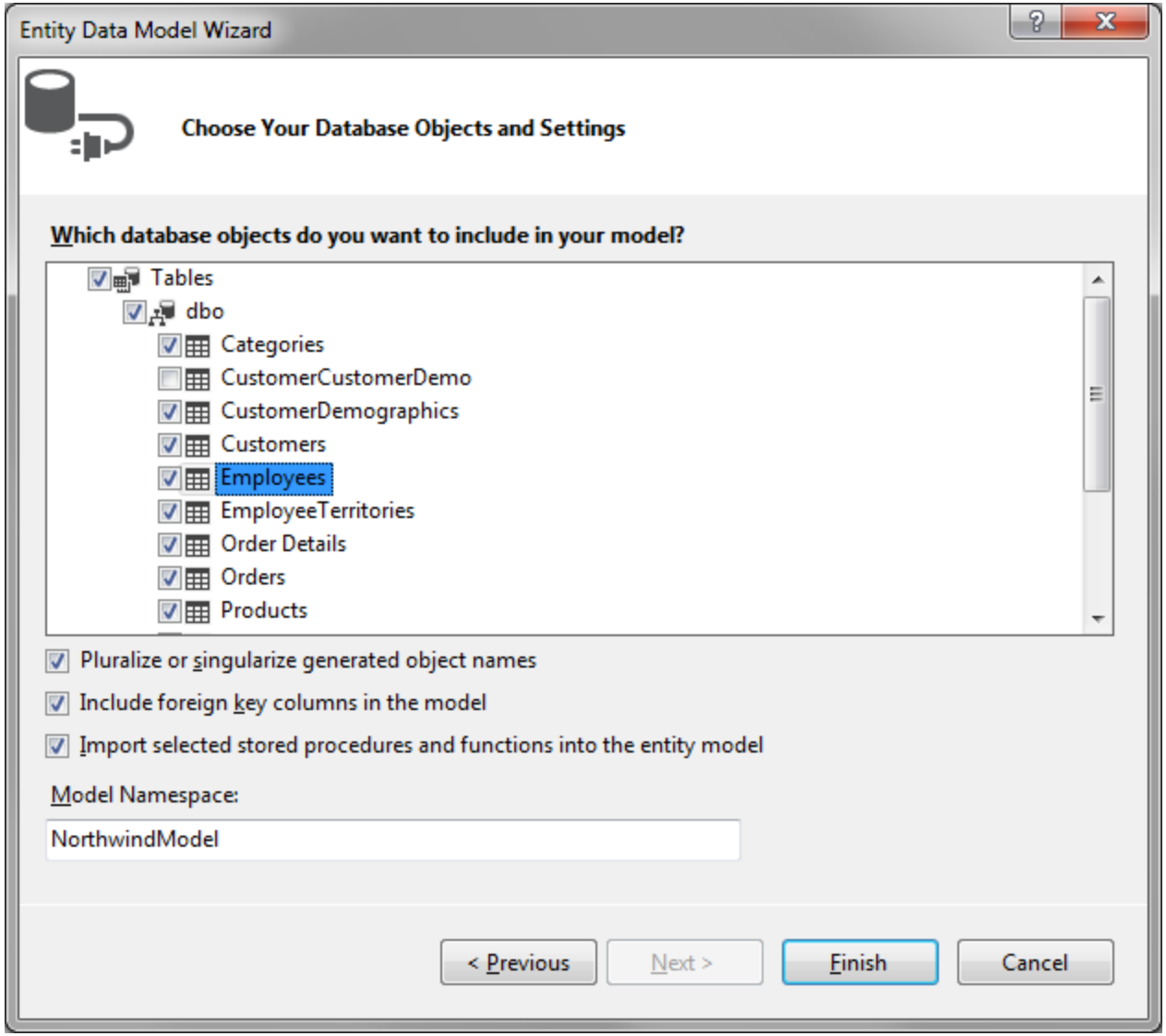
Generally you will use “localhost” as the server when developing locally.

Otherwise, ask your database administrator.

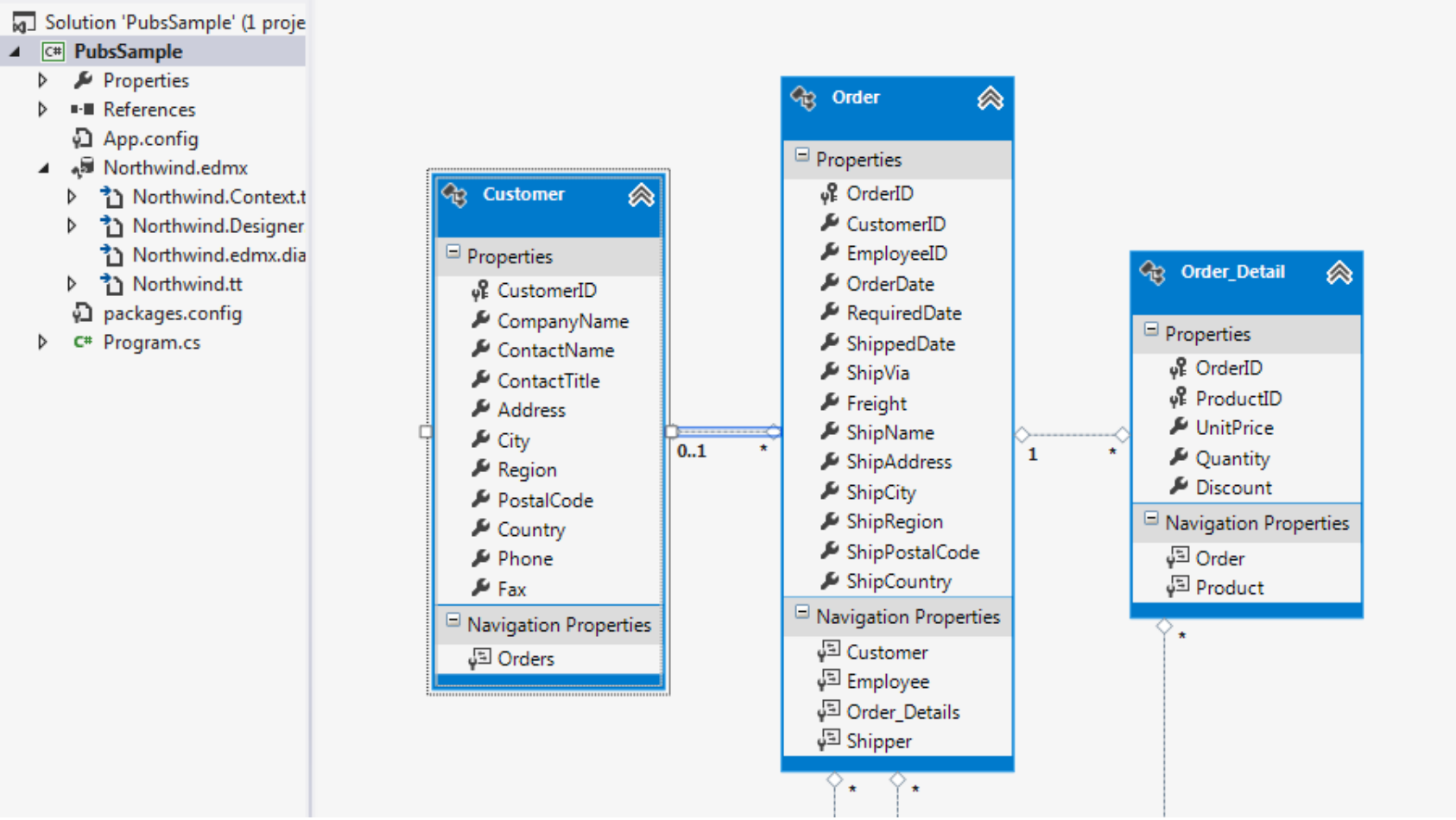


Choosing Objects

For now, we will simply get some of the tables.



Ta da!


Solution 'PubsSample' (1 proje
PubsSample
Properties
References
App.config
Northwind.edmx
Northwind.Context.t
Northwind.Designer
Northwind.edmx.dia
Northwind.tt
packages.config
Program.cs
Customer
Properties
CustomerID
CompanyName
ContactName
ContactTitle
Address
City
Region
PostalCode
Country
Phone
Fax
Navigation Properties
Orders
0..1
*
Order
Properties
OrderID
CustomerID
EmployeeID
OrderDate
RequiredDate
ShippedDate
ShipVia
Freight
ShipName
ShipAddress
ShipCity
ShipRegion
ShipPostalCode
ShipCountry
Navigation Properties
Customer
Employee
Order_Details
Shipper
1
*
Order_Detail
Properties
OrderID
ProductID
UnitPrice
Quantity
Discount
Navigation Properties
Order
Product

Edit the Model

- We can adjust names of model objects
- For example, I don’t like Order_Detail and Order_Details as a navigation property, let’s click on them in Order and Order_Detail and take out that underscore.

Write Some Code

- Now that we have a model and entity context in place, we can use the generated class (context) and LINQ to query the database.
- Let’s write a query to list out all the customer in the USA to the console…

Getting USA Customers

```
class Program
{
    static void Main(string[] args)
    {
        GetUSACustomers();
        Console.ReadLine();
    }

    private static void GetUSACustomers()
    {
        using (var context = new NorthwindEntities())
        {
            var usaCustomers = context.Customers.Where(c => c.Country == "USA");

            foreach (var customer in usaCustomers)
            {
                Console.WriteLine("{0,-35} {1} {2}", customer.CompanyName, customer.Phone, customer.Country);
            }
        }
    }
}
```

SOFTWARE GUILD

Changing Data

- We can manipulate the collections in our database context and then call SaveChanges() to apply all of the changes to the database.

```
private static void AddRegion()
{
    using (var context = new NorthwindEntities())
    {
        Region newRegion = new Region();
        newRegion.RegionDescription = "My New Region";

        context.Regions.Add(newRegion);
        context.SaveChanges();
    }
}
```

SOFTWARE GUILD

Conclusion

- With database-first development, we can be up and running in a few minutes.
- A downside on the security aspect is that the application needs permission to access tables directly. We will demonstrate binding to stored procedures later.