Using Entity Frameworks to Simplify Data Access

John F. Gnazzo

Introduction

Microsoft Entity Framework (EF) is an object-relational mapper (ORM) based on ADO.NET, and is a set of technologies that support the development of data-oriented software applications. As a replacement to Linq-to-SQL, EF enables developers to work with data in the form of domain-specific objects and properties, such as students and student grades, without having to concern themselves with the underlying database tables and columns where this data is stored. Using EF, developers can work at a higher level of abstraction when they deal with data, and can create and maintain data-oriented applications with less code than in traditional applications.

EF provides a mechanism to simplify data access. EF also eliminates the need for most of the data-access code that developers usually need to write.

This Blog will demonstrate the ease of using EF to work with data by showing how to create an Entity Frameworks Data Model as well as providing concrete examples of using EF in a real-world application.

Create Entity Frameworks Data Model

An application generally utilizes a Data Model to describe the data that it will be using. The following example will use the Data Base First approach. This means that the data model will be defined from an existing database.

Given an existing project created from Microsoft Visual Studio 2013, the developer would first create an ADO.NET Entity Data model by electing to “Add New Item” to his project from Solution Explorer.

This would bring up the following Screen where he chooses to add an ADO.NET Entity Data Model. (See Figure 1 below)

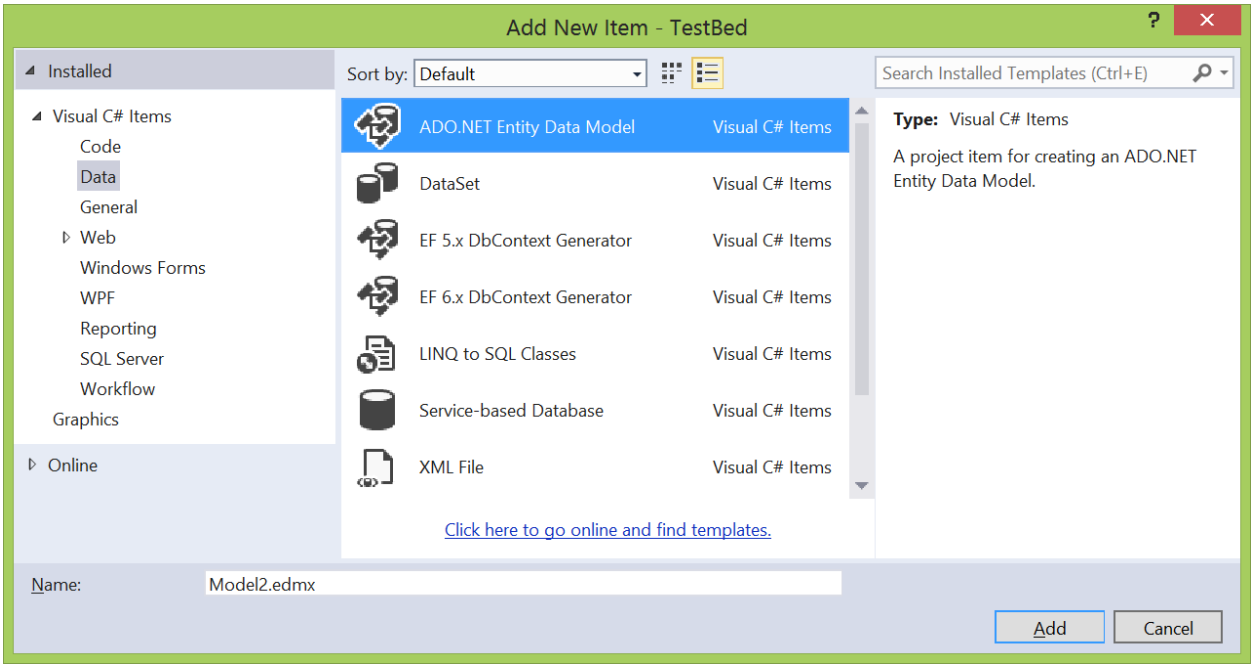

Figure 1: Add ADO.NET Entity Data Model

He would then go through steps to select a model from generating it through a Data Base. (See Figure 2 below)

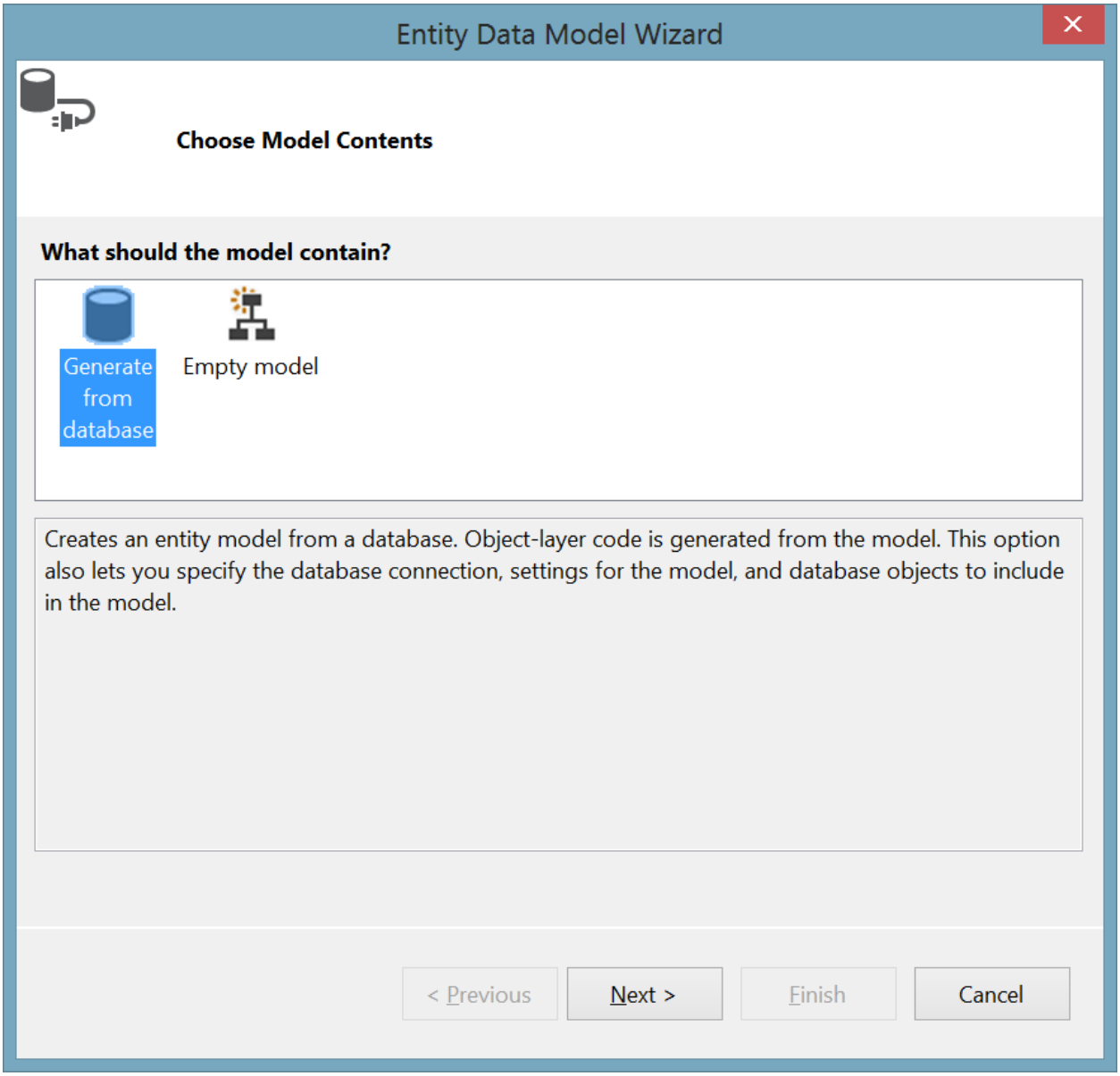

Figure 2: Select Generate from Database Option

And then select the Data Connection and name the Entity in this case MyEntities. (See Figure 3 Below)

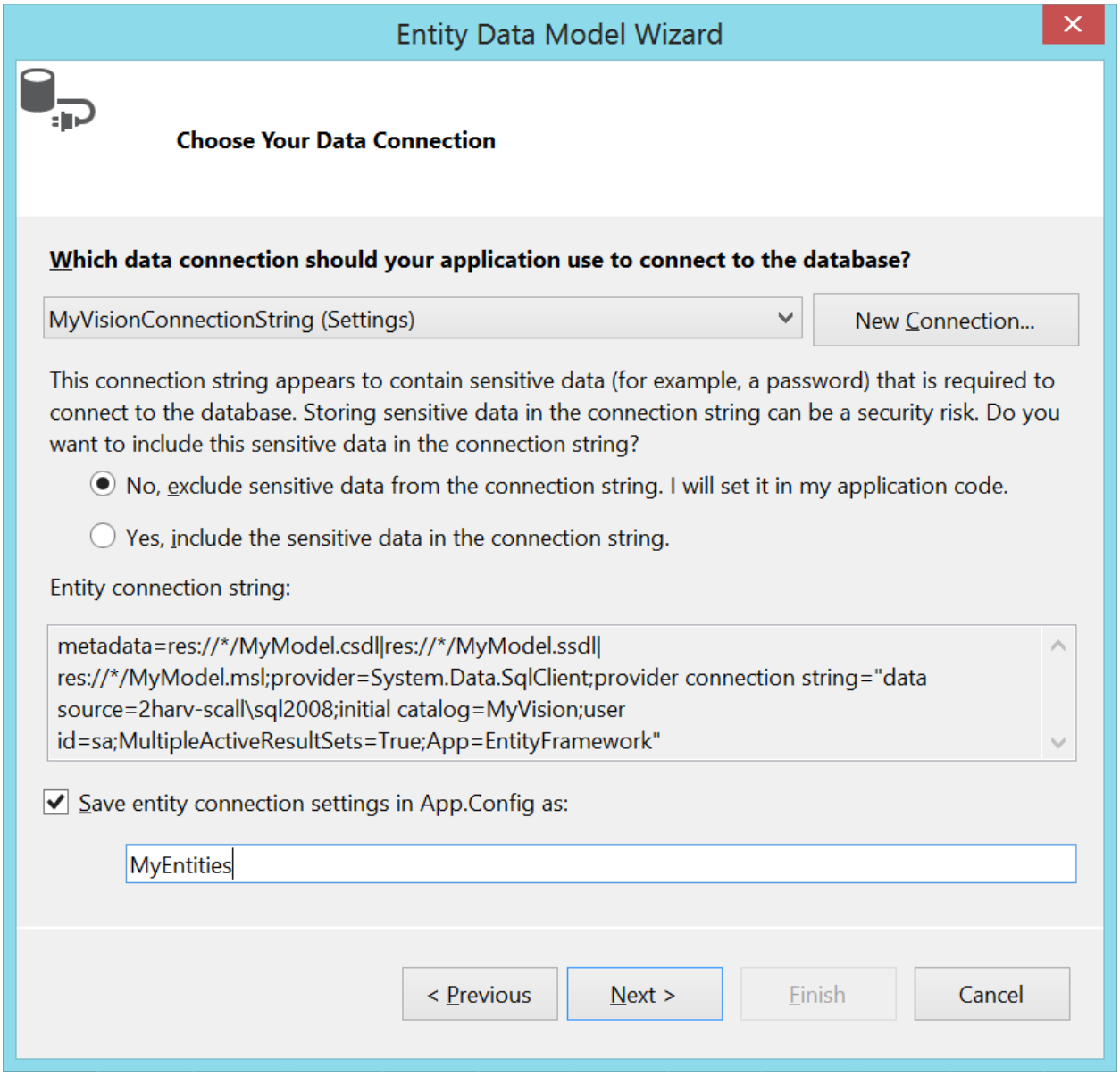

Figure 3: Create Database Connection

Then Select Entities to include in Model. In this case tables Bid, BidItem and BidItemMaterial. (See Figure Below)

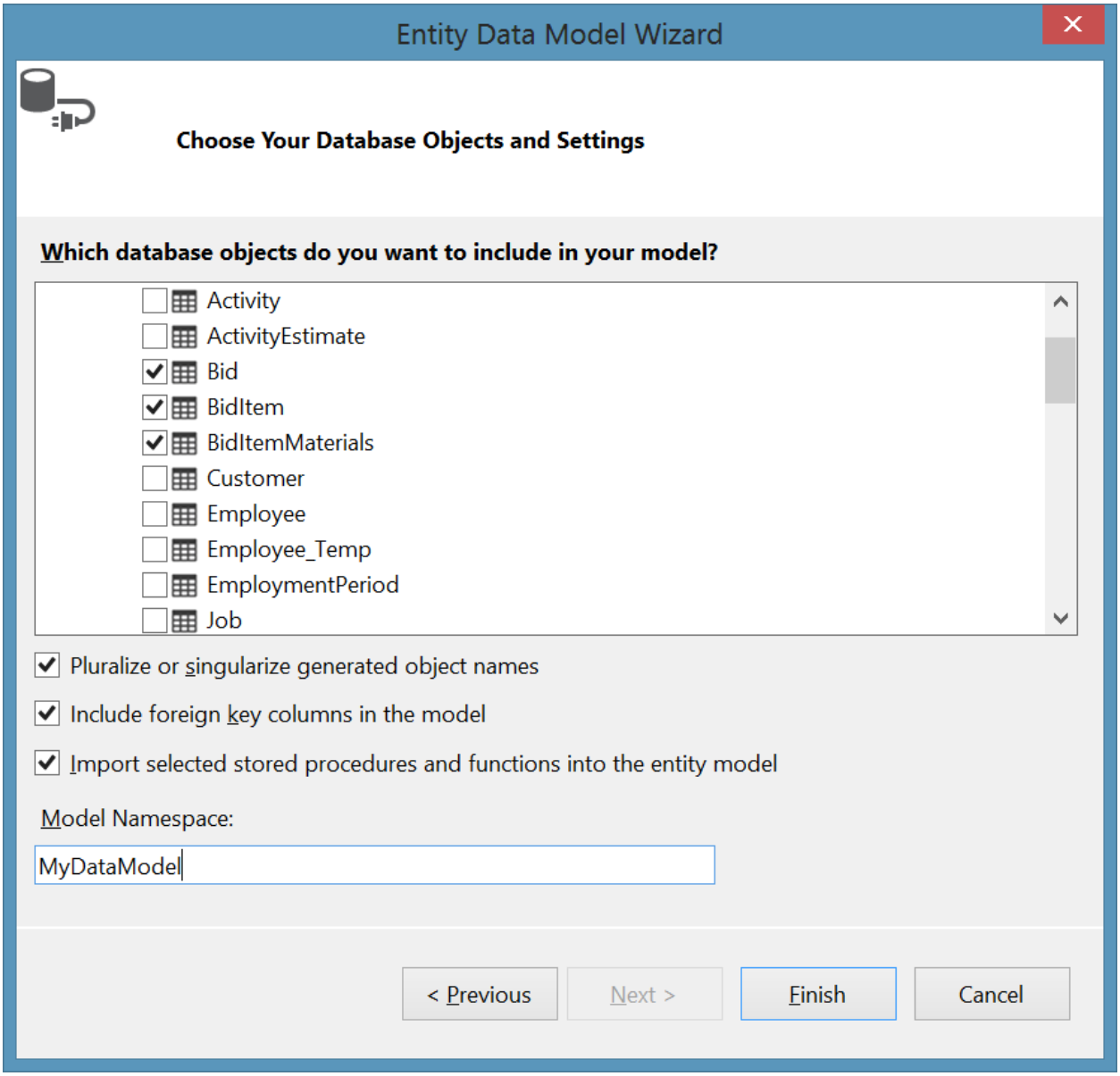

Figure 4: Select Entities to include in Model

The Data Model generated from the steps above includes a hierarchy of entities is shown below. In this figure, the model consists of a Bid (Bid Table). The Bid contains one (1) or more Bid Items (BidItem Table), and the Bid Item contains one or more Materials (BidItemMaterial Table).

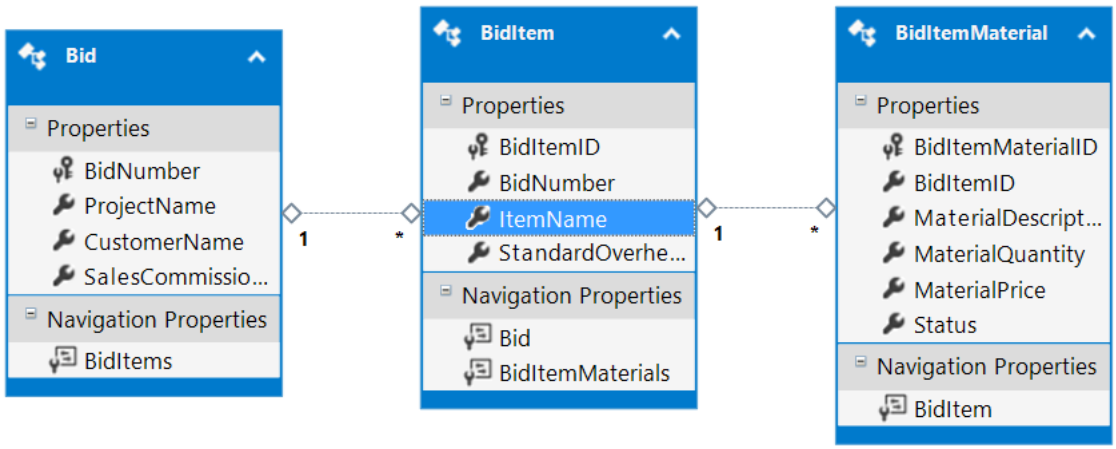

Figure 5: Data Model

Concrete Examples of using Entity Frameworks

The following shows several examples of using EF in C# code.

Create access to the database

```
MyEntities _context = new MyEntities();
```

Get all Bids

```
List<Bid> _bidList = _context.Bids.ToList();
```

Get a the bid having a bidNumber of 1

```
Bid MyBid = (Bid)context.Bids.FirstOrDefault(c => c.BidNumber == 1);
```

Update a form with data from a specific bid

```
bidBindingSource = new System.Windows.Forms.BindingSource( );

bidBindingSource.DataSource = _bidList.FirstOrDefault(c => c.BidNumber ==
_selectedBid.BidNumber).ToList();
```

A form having controls bound to the bidBindingSource would be updated accordingly.

Get a list of all materials having a specific BidItemId

```
List<BidItemMaterial> _materialList =
```

_context.BidItemMaterials.Where(c => c.BidItemID == _selectedItem.BidItemID).ToList();

Save Changes made to data bound to control

_context.SaveChanges();

Conclusion

From the examples above, one can see the power and ease of using Entity Frameworks for data base access.