Entity Framework Tutorial
Second Edition

The ADO .NET Entity Framework from Microsoft is a new ADO .NET development framework that provides a level of abstraction for data access strategies and solves the impedance mismatch issues that exist between different data models.

This book explores Microsoft's Entity Framework and explains how it can be used to build enterprise-level applications. It will also teach you how you can work with RESTful Services and Google's Protocol Buffers with Entity Framework and WCF. You will explore how to use Entity Framework with the ASP.NET Web API and also how to consume the data exposed by Entity Framework from client applications of various types, such as ASP.NET MVC, WPF, and Silverlight. You will familiarize yourself with the new features and improvements introduced in Entity Framework, including enhanced POCO support, template-based code generation, tooling consolidation, and connection resiliency. By the end of the book, you will be able to successfully extend the new functionalities of Entity Framework into your project.

Who this book is written for
If you are a C# developer who wants to learn a new way of querying data and utilizing it in applications efficiently for data binding or other operations, then this book is for you. Basic knowledge of ADO .NET is assumed.

What you will learn from this book
- Explore the features of the ADO .NET Entity Framework
- Understand the concepts of entities and their relationships
- Create an Entity Data Model using the ADO .NET Entity Data Model Designer and the EdmGen tool
- Extend existing entity types to create your own entity types
- Write programs against the Entity Data Model to perform CRUD operations
- Discover WCF Data Services and learn how they can be used with Entity Framework
- Apply code-first, model-first, and database-first approaches


Joydip Kanjilal

A comprehensive guide to Entity Framework with insights into its latest features and optimizations for responsive data access in your projects.
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 2 'Getting Started'
- A synopsis of the book’s content
About the Author

Joydip Kanjilal has won a Microsoft Most Valuable Professional (MVP) award in ASP.NET. He is a speaker and author of several books and articles. He has over 18 years of industry experience in IT, with more than 12 years in Microsoft .NET and its related technologies. Joydip is currently working as a Principal Architect at SenecaGlobal IT Services Private Limited, Hyderabad. He has been selected as an MSDN Featured Developer of the Fortnight (MSDN) a number of times and has also been a Community Credit Winner at www.community-credit.com several times. He has authored the following books:

- *ASP.NET Web API: Build RESTful Web Applications and Services on the .NET Framework* by Packt Publishing
- *Visual Studio 2010 and .NET 4 Six-in-One* by Wrox Publishers
- *ASP.NET 4.0 Programming* by Mc-Graw Hill Publishing
- *Entity Framework Tutorial* by Packt Publishing
- *Pro Sync Framework* by APRESS
- *Sams Teach Yourself ASP.NET Ajax in 24 Hours* by Sams Publishing
- *ASP.NET Data Presentation Controls Essentials* by Packt Publishing


He has years of experience in designing and architecting solutions for various domains. His technical strengths include C, C++, VC++, Java, C#, Microsoft .NET, AJAX, WCF, web-based APIs, REST, SOA, design patterns, SQL Server, operating systems, and computer architecture.
He can be contacted at:

• http://aspadvice.com/blogs/joydip
• http://www.infoworld.com/blog/microsoft-coder
• Website: www.joydipkanjilal.com
• Twitter: https://twitter.com/joydipkanjilal
• Facebook: https://www.facebook.com/joydipkanjilal
• LinkedIn: http://in.linkedin.com/in/joydipkanjilal
Preface

The ADO.NET Entity Framework, the next generation of Microsoft’s data access technology, is an extended Object Relational Mapping (ORM) technology that makes it easy to tie together the data in your database with the objects in your applications. This is done by abstracting the object model of an application from its relational or logical model. It is an extended ORM in the sense that it provides many additional features that a traditional ORM does not.

This book is a clear and concise guide to the ADO.NET Entity Framework. Packed with plentiful code examples, this book helps you learn the ADO.NET Entity Framework and ADO.NET Data Services and build a better data access layer for your application.

The intent of writing this book is updating you to the latest trends and developments as far as Entity Framework is concerned.

What this book covers

Chapter 1, Introducing the ADO.NET Entity Framework, is an introduction to the basics of the ADO.NET Entity Framework (EF), its usefulness, features, and benefits.

Chapter 2, Getting Started, discusses how you can get started with EF, create an Entity Data Model (EDM), and write a program to query data.

Chapter 3, Entities, Relationships, and the Entity Data Model, gives a detailed explanation of entities, relationships, and each of the sections of the EDM.

Chapter 4, Working with Stored Procedures in the Entity Data Model, provides a sample application that illustrates how to perform CRUD operations against the EDM.

Chapter 5, Working with Entity Client and Entity SQL, discusses the Entity SQL query language and how to work with the Entity Client provider.
Chapter 6, Working with LINQ to Entities, includes a detailed discussion on LINQ to Entities with many code examples.

Chapter 7, Working with the Object Services Layer, provides a detailed discussion on the Object Services Layer and its helpful and useful features.

Chapter 8, Working with WCF Data Services, provides an introduction to WCF Data Services, the REST architectural paradigm, and how these can be used with the EDM to perform CRUD operations.

Appendix, Advanced Concepts, covers a few advanced concepts. These include: REST and REST-based service frameworks and OData. We would also explore the HTTP methods and the request and response codes. Lastly, we would take a look at the new features in Entity Framework 7.
In the previous chapter, we took a look at Entity Framework, including its architecture and its features. We also had a look at the new and enhanced features in Entity Framework 7. Note that Entity Framework 6.0 ships with Visual Studio 2013, but you can also install it via NuGet if needed. Also, Entity Framework 7 is yet to be released.

In this chapter, we will design our Security database, create an Entity Data Model (EDM) on top of it, and then use the EntityDataSource control to bind data exposed by the EDM to a GridView control.

The DataSource controls are those that are used to connect to a data source and then retrieve data from those data sources. If you use DataSource controls, the need of writing tedious code to perform Create, Read, Update, Delete (CRUD) operations on data-based controls is eliminated. The EntityDataSource control is a DataSource control that can connect to the data exposed by the EDM to perform CRUD operations.

In this chapter, we will cover the following points:

- Designing the UserAuthentication database
- Creating an EDM for the UserAuthentication database
- Introducing the EntityDataSource control
- Implementing our first application using Entity Framework 6
The latest version of Entity Framework is Entity Framework 7. As of this writing, Entity Framework 7 hasn't been released. It will be released as a "pre-release" at the same time ASP.NET 5 is released. You will be able to install the pre-release version of EF 7 using NuGet.

There are many changes coming up in Entity Framework 7—it is being rewritten from the ground up. The major goals of Entity Framework 7 include its support for new platforms and new data stores. So, support for additional providers will also be included in Entity Framework 7:

We will start this chapter with a discussion on our UserAuthentication database that we will be using throughout this book, followed by a discussion on how we can create an EDM using the Security database.
Designing the UserAuthentication database

Before we begin implementing a simple application that shows how data retrieved from the EDM can be consumed, let's take a quick look at the EDM again:

As you can see in this diagram, our application needs to interact with the CSDL layer. The SSDL layer will connect to the database (the Security database in our example), and the mapping layer will map these two layers so that they can communicate.

The Security database comprises of a list of the following tables:

- **Users**: This table contains the user details.
- **UsersAuthentication**: This table contains the user authentication details.
- **UserAuthenticationTypes**: This table contains user authentication type data. User authentication type can be Windows, Forms, Passport, and Anonymous.
Getting Started

- **UsersLoginHistory**: This table contains data related to user login history; that is, the user login history data of the user.
- **UserRoles**: This table contains the user role details.
- **Roles**: This table contains the role details.
- **Controls**: This table contains an entry per control (note that each control is an object).
- **ControlTypes**: This table contains the control type data.

Creating the EDM

Now that the **Security** database is ready, we will explore how we can create an EDM on top of the **Security** database.

Note that, before Entity Framework 7, there were two storage models—the EDMX file format based on XML schema or code. With Entity Framework 7, the EDMX file format will be dropped—we will have only the code-based format. Interestingly, this approach is also termed the "code-first only" approach.

You can create the Entity Data Model in one of two ways:

- Use the ADO.NET Entity Data Model Designer
- Use the command-line Entity Data Model Designer called `EdmGen.exe`

The first approach is preferred to the second. However, as we move through the chapters of the book, we will explore how we can follow the code-first approach to implement the model for our application that uses Entity Framework.

We will first take a look at how we can design an EDM using the ADO.NET Entity Data Model Designer.
Creating the Entity Data Model using the ADO.NET Entity Data Model Designer

To create an EDM using the ADO.NET Entity Data Model Designer, follow these simple steps:

1. Open Visual Studio.NET 2013 IDE, create a solution for a new web application project as follows, and save it with a name.
2. Switch to the Solution Explorer, and navigate to Add | New Item... to create a new Entity Data Model using Entity Data Model Wizard.
3. Next, select ADO.NET Entity Data Model from the list of templates displayed, as shown in the following screenshot:

   ![Creating a new ADO.NET Entity Data Model](image)

4. Name the Entity Data Model SecurityDB, and click on Add.
5. Select **Generate from database** from Entity Data Model Wizard, as shown in the following screenshot:

![Entity Data Model Wizard](image)

Generating the Entity Data Model from the database

- Note that you can also use the **Empty model** template to create the EDM yourself.
If you select the **Empty model** template and click on **Next**, the following screen appears:

![Empty Entity Data Model Wizard](image)

Empty Entity Data Model Wizard

As you can see from the previous screenshot, you can use this template to create the EDM yourself.

You can create the entity types and their relationships manually by dragging items from the toolbox.

We will not use this template in our discussion here, so let's get to the next step.

6. Click on **Next** in the **Entity Data Model Wizard** window shown earlier.

7. The modal dialog box will now appear and prompt you to choose your connection.
8. Click on **New Connection**. Now you will need to specify the connection properties and parameters for the database to connect to. In our example, the database is **Security**.

We will use a dot to specify the database server name. This implies that we will be using the database server of the localhost, which is the current system in use. You can also specify the server name here if your database resides on a different system. If your database resides on a different server, you need to specify the server name here.

9. After you specify the necessary user name, password, and server name, you can test your connection using the **Test Connection** button. When you do so, the message **Test connection succeeded** gets displayed in the message box, as shown in the following screenshot:
Note that the entity connection string is generated automatically. This connection string will be saved in the ConnectionStrings section of your application's web.config file. This is what it will look like:

```xml
<connectionStrings>
  <add name="SecurityDBEntities" connectionString="metadata=res://*/SecurityDB.csd|res://*/SecurityDB.ssdl|res://*/SecurityDB.msl;provider=System.Data.SqlClient;provider connection string="data source=.;initial catalog=SecurityDB;user id=sa;password=sa1@3;MultipleActiveResultSets=True;App=EntityFramework" providerName="System.Data.EntityClient" />
</connectionStrings>
```

10. Now, click on **Next** and specify the database objects you would like to have in your model from the **Choose Your Database Objects and Settings** window that is shown next:

Choosing the database objects to be used in the model
We will select all the tables of the Security database now. Refer to the following screenshot:

11. Lastly, click on Finish to generate the EDM for the Security database.

Your EDM has been generated and saved in a file named SecurityDB.edmx. We are done creating our first EDM using the ADO.NET Entity Data Model Designer tool.
When you open SecurityDB.edmx that we just created in the designer view, it will appear as shown in the following image:

The SecurityDB Entity Data Model

In the next section, we will learn to create an EDM using the EdmGen.exe command-line tool.

Creating Entity Data Model using the EdmGen tool

We will now take a look at how to create a data model using the EDM generation tool called EdmGen.

The EdmGen.exe command-line tool can be used to do one or more of the following:

- Generate the .cdsl, .msl, and .ssdl files as part of the EDM
- Generate object classes from a .csdl file
- Validate an EDM
The **EdmGen.exe** command-line tool generates the EDM as a set of three files: .csdl, .msl, and .ssdl. If you have used the ADO.NET Entity Data Model Designer to generate your EDM, the .edmx file generated will contain the CSDL, MSL, and the SSDL sections. You will have a single .edmx file that bundles all of these sections into it. On the other hand, if you use the **EdmGen.exe** tool to generate the EDM, you will find three separate files with .csdl, .msl, or .ssdl extensions.

Here is a list of the major options of the **EdmGen.exe** command-line tool:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/help</td>
<td>Use this option to display help on all the possible options of this tool. The short form is /?</td>
</tr>
<tr>
<td>/language:CSharp</td>
<td>Use this option to generate code using C# language</td>
</tr>
<tr>
<td>/language:VB</td>
<td>Use this option to generate code using VB language</td>
</tr>
<tr>
<td>/provider:&lt;string&gt;</td>
<td>Use this option to specify the name of the ADO.NET data provider that you would like to use</td>
</tr>
<tr>
<td>/connectionstring:&lt;connection string&gt;</td>
<td>Use this option to specify the connection string to be used to connect to the database</td>
</tr>
<tr>
<td>/namespace:&lt;string&gt;</td>
<td>Use this option to specify the name of the namespace</td>
</tr>
<tr>
<td>/mode:FullGeneration</td>
<td>Use this option to generate your CSDL, MSL, and SSDL objects from the database schema</td>
</tr>
<tr>
<td>/mode:EntityClassGeneration</td>
<td>Use this option to generate your entity classes from a given CSDL file</td>
</tr>
<tr>
<td>/mode:FromSsdlGeneration</td>
<td>Use this option to generate MSL, CSDL, and entity classes from a given SSDL file</td>
</tr>
<tr>
<td>/mode:ValidateArtifacts</td>
<td>Use this option to validate the CSDL, SSDL, and MSL files</td>
</tr>
<tr>
<td>/mode:ViewGeneration</td>
<td>Use this option to generate mapping views from the CSDL, SSDL, and MSL files</td>
</tr>
<tr>
<td>/entitycontainer:&lt;string&gt;</td>
<td>Use this option to specify the name of the Entity Container to be used in the conceptual model</td>
</tr>
<tr>
<td>/project:&lt;string&gt;</td>
<td>Use this option to specify the base name to be used for all the artifact files (.csdl, .msl, and .ssdl) to be generated. The short form of this option is /p</td>
</tr>
</tbody>
</table>
Note that you need to pass the connection string, and specify the mode and the project name of the artifact files (the .csdl, .msl, and .ssdl files) to be created. To create the EDM for our database, open a Visual Studio command window and type in the following:

```
edmgen /mode:fullgeneration /c:"Data Source=.;Initial Catalog=SecurityDB;User ID=sa;Password=sa1@3;" /p:SecurityDB
```

This will create a full Entity Data Model for our database. The output is shown in the following screenshot:

![Generating the Entity Data Model from the command line](image)

You can also validate the SecurityDB model that was just created, using the ValidateArtifacts option of the EdmGen command-line tool, as follows:

```
```

When you execute the preceding command, the output will be similar to what is shown in the previous screenshot.

As you can see in the preceding screenshot, there are no warnings or errors displayed. So, our EDM is perfect.

The section that follows discusses the DataSource controls included in ASP.NET and also the new EntityDataSource control, which was first introduced as part of the Visual Studio.NET 2008 SP1 release. Note that the EntityDataSource control is included as part of Visual Studio 2010 and onward.
The DataSource controls

The DataSource controls are those that can be bound to data from external data sources. These data sources may include databases, XML files, or even flat files. ASP.NET 2.0 introduced some DataSource controls with a powerful data binding technique, so the need for writing lengthy code for binding data-to-data controls has been eliminated.

In ASP.NET, the term data binding implies binding the controls to data retrieved from a data source and providing read or write connectivity between these controls and the data that they are bound to.

A DataSource control acts like a layer in between your data source and the data bound control. Data bound controls can be any control that actually interacts with the end user while consuming the data services provided by the DataSource control to which it is bound. It defines certain methods and properties that perform data-specific operations like insert, delete, update, and select over the data exposed by the DataSource control while at the same time abstracting the data source.

In the sections that follow, we will discuss these controls with special emphasis on the EntityDataSource control.

The ObjectDataSource control

The ObjectDataSource control works with in-memory collections. It defines properties like InsertMethod, DeleteMethod, UpdateMethod, and SelectMethod, which perform basic data storage and retrieval operations. Appropriate methods must be created and mapped to the properties that perform the required task. When one of these properties is used, the ObjectDataSource control actually creates an instance, invokes the appropriate method, and destroys as soon as it completes its execution phase. ObjectDataSource are usually used in the business layer in your application, which helps you to directly bind to the data bound controls at the presentation layer.
The SqlDataSource control

The SqlDataSource control allows you to perform standard data operations, like insert, update, delete, and select on the data persisting in your relational database. The SqlDataSource control is not meant only for the SQL Server database; it can work with any managed ADO.NET provider, which means that you can use the SqlDataSource control with different relational data sources. The SqlDataSource control defines properties like InsertCommand, DeleteCommand, UpdateCommand, and SelectCommand, for performing standard data operations, like insert, delete, update, and select, over the data. The command properties need appropriate queries to be set before using them. When updates are performed on a data control that connects to a SqlDataSource control, the SqlDataSource control creates update parameters for all columns, even though few columns are updated. The control also supports caching capabilities, which assist in improving the performance of the application. For further reading on this topic, please refer to this link: http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.sqldatasource(v=vs.110).aspx.

The SiteMapDataSource control

The SiteMapDataSource control allows you to bind the site map of your website. The site map can represent a hierarchical structure. The SiteMapDataSource control needs an appropriate root node to be specified in a given hierarchy. The SiteMapDataSource control contains properties that allow you to specify the node locations. Primarily, the SiteMapDataSource control is used for the data navigation purpose, which means that you cannot perform standard data operations, like inserts, updates, deletes, sorting, and paging of the data.

The XMLDataSource control

The XMLDataSource control is another kind of DataSource control. It basically represents the data, which is in the form of XML. You can access the XML data from the XMLDataSource control by connecting to a XML file or to XML data embedded as a string within the DataSource control. Caching in the XMLDataSource control is enabled by default for increasing the performance. You can perform standard data operations like insert, delete, update, and select over the XML data that is represented by the XMLDataSource control. However, operations like sorting and paging are not supported by the XMLDataSource control. The control also provides support for applying XML transformations through a XML style sheet.
The LinqDataSource control

The LinqDataSource control is a new control that has been introduced in ASP.NET 3.5. It extends the DataSource control and resides in the System.Web.UI.WebControls namespace. It provides a new approach for binding LINQ models to web controls in your ASP.NET applications. The LinqDataSource control provides properties and events, using which you can perform operations like selecting, filtering, grouping, and ordering against LinqDataSource. The LinqDataSource data control provides a flexible mechanism to build a data control with wizard-based workflow. It allows you to perform CRUD operations on the data over a LINQ model with minimal need to write SQL queries.

The EntityDataSource control

The EntityDataSource control is an example of a data control that was first included as part of the Visual Studio 2008 SP1 release, and can be used to bind data retrieved from an EDM to the data bound controls of ASP.NET.


Implementing our first application using the Entity Framework

In this section, we will learn how to use the EDM and the EntityDataSource control to implement our first program using the Entity Framework. We will use a GridView control to display bound data.
Let's first have the environment ready. I will run you through the steps to download and install Entity Framework 6 now.

In the **Solution Explorer**, right-click on the project and select **EntityFramework** from the list of NuGet packages, as shown in the following screenshot:
Click on **Install** to start downloading and installing Entity Framework 6:

Once Entity Framework 6 has been downloaded, click on **Accept** to start the installation. After Entity Framework 6 has been successfully installed and the necessary changes applied to your project, you are ready to start writing your first application that makes use of this framework.
Refer to the solution we created earlier using the Entity Data Model Designer. Now follow these steps:

1. Drag and drop an **EntityDataSource** control from the toolbox onto your `Default.aspx` web form:
2. Now click on the **Configure Data Source...** option to specify the data source. Refer to the following screenshot:

![Configuring the EntityDataSource Control](image)

3. Specify the **ConnectionString** and **DefaultContainerName** fields, and then click on **Next**:

![Configure Data Source - EntityDataSource1](image)
4. Specify the fields you want to retrieve from the database table, and click on Finish when done:

5. Now drag and drop a GridView control from the toolbox onto the Default.aspx web form.
6. Next, use the **Choose Data Source** option of the GridView control to associate its data source with the EntityDataSource control we created earlier. Refer to the following screenshot:

![Configuring the Data Source property of the GridView control](image)

This is how the markup code of the GridView control looks with its templates defined. Note how the DataSourceID of the GridView control has been associated with the EntityDataSource control we created earlier.

```xml
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" DataSourceID="EntityDataSource1">
    <Columns>
        <asp:BoundField DataField="UserAuthenticationID"
            HeaderText="UserAuthenticationID" ReadOnly="True"
            SortExpression="UserAuthenticationID" />
        <asp:BoundField
            DataField="UserAuthenticationTypeID"
            HeaderText="UserAuthenticationTypeID"
            ReadOnly="True"
            SortExpression="UserAuthenticationTypeID" />
        <asp:BoundField
            DataField="UserID"
            HeaderText="UserID"
            ReadOnly="True"
            SortExpression="UserID" />
        <asp:BoundField
            DataField="UserName"
            HeaderText="UserName"
            ReadOnly="True"
            SortExpression="UserName" />
    </Columns>
</asp:GridView>
```
Chapter 2

The markup code of the EntityDataSource control looks like the following:

```xml
<asp:EntityDataSource ID="EntityDataSource1" runat="server"
ConnectionString="name=SecurityDBEntities"
DefaultContainerName="SecurityDBEntities" EnableFlattening="False"
EntitySetName="UserAuthentications"
EntityTypeFilter="UserAuthentication"
Select="it.[UserAuthenticationID], it.[UserAuthenticationTypeID],
it.[UserID], it.[UserName]">
</asp:EntityDataSource>

This is what the complete markup code of the Default.aspx web page looks like:

```xml
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<asp:EntityDataSource ID="EntityDataSource1" runat="server"
ConnectionString="name=SecurityDBEntities"
DefaultContainerName="SecurityDBEntities" EnableFlattening="False"
EntitySetName="UserAuthentications"
EntityTypeFilter="UserAuthentication"
Select="it.[UserAuthenticationID], it.[UserAuthenticationTypeID],
it.[UserID], it.[UserName]">
</asp:EntityDataSource>
<asp:GridView ID="GridView1" runat="server"
AutoGenerateColumns="False"
DataSourceID="EntityDataSource1">
<Columns>
<asp:BoundField DataField="UserAuthenticationID"
HeaderText="UserAuthenticationID" ReadOnly="True"
SortExpression="UserAuthenticationID" />
<asp:BoundField
DataField="UserAuthenticationTypeID"
HeaderText="UserAuthenticationTypeID"
ReadOnly="True"
SortExpression="UserAuthenticationTypeID" />
<asp:BoundField DataField="UserID"
HeaderText="UserID"
ReadOnly="True"
SortExpression="UserID" />
</Columns>
</asp:GridView>
</form>
</body>
</html>
```
When you execute the application, your output should be similar to what is shown in the following image:

<table>
<thead>
<tr>
<th>UserAuthenticationID</th>
<th>UserAuthenticationTypeID</th>
<th>UserID</th>
<th>UserName</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Jaydp</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>Udal</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>3</td>
<td>Saunesh</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>4</td>
<td>Sanjeeb</td>
</tr>
</tbody>
</table>

Data exposed by the Entity Data Model is displayed in the GridView control.

**Summary**

In this chapter, we discussed how we can get started with Entity Framework. You learned how to create an EDM and use it along with the EntityDataSource control, to bind data to a GridView data control. In this chapter, we created the UserAuthentication database and an EDM that exposed this database. We also explored the EntityDataSource control and used it to bind data in our first application that leverages Entity Framework 6.

In the next chapter, we will continue to explore the EDM including each of its sections and will learn how they are related to each other.
Where to buy this book


Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.