Build an ASP.NET Core Service and App with .NET (Core) 5.0 Two-Day Hands-On Lab

Lab 2

This lab walks you through creating the Models and ViewModels. Prior to starting this lab, you must have completed Lab 1.

Part 1: Enable Nullability

To turn on C# 8 nullability, update the project files for `AutoLot.Models` and `AutoLot.Dal` to the following:

```
<PropertyGroup>
  <TargetFramework>net5.0</TargetFramework>
  <Nullable>enable</Nullable>
</PropertyGroup>
```

Part 2: Creating the Entities in AutoLot.Models

The entities represent the data that is persisted in SQL Server and can be shaped to be more application specific. Begin by deleting the autogenerated `Class1.cs`.

Step 1: Create the Base Entity

- Create a new folder in the `AutoLot.Models` project named `Entities`. Create a subfolder named `Base` under the `Entities` folder.
- Add a new class to the `Base` folder named `BaseEntity.cs`
- Add the following using statements to the class:

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
```

- Update the code for the class to the following:

```
namespace AutoLot.Models.Entities.Base
{
  public abstract class BaseEntity
  {
    [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public int Id { get; set; }
    [Timestamp] public byte[]? TimeStamp { get; set; }
  }
}
```

Step 2: Create the Car Entity

NOTE: The project will not compile until the remaining entities have been added.

- Add a new class to the `Entities` folder named `Car.cs`
- Add the following using statements to the class:

```
using System.Collections.Generic;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text.Json.Serialization;
using AutoLot.Models.Entities.Base;
```

- Update the code for the class to the following:

```
namespace AutoLot.Models.Entities
{
  [Table("Inventory", Schema = "Dbo")]
  public partial class Car : BaseEntity
  {
    [Required]
    [DisplayName("Make")]
    public int MakeId { get; set; }
    [ForeignKey(nameof(MakeId))]
    [InverseProperty(nameof(Make.Cars))]
    public Make? MakeNavigation { get; set; }
    [StringLength(50), Required]
    public string Color { get; set; } = "Gold";
    [StringLength(50), Required]
    [DisplayName("Pet Name")]
    public string PetName { get; set; } = "My Precious";
    [JsonIgnore]
    [InverseProperty(nameof(Order.CarNavigation))]
    public IEnumerable<Order> Orders { get; set; } = new List<Order>();
    [NotMapped]
    public string MakeColor => $"{MakeNavigation?.Name} ({Color})";
    public override string ToString()
    {
        // Since the PetName column could be empty, supply
        // the default name of **No Name**.
        return $"{PetName ?? "**No Name**"} is a {Color} {MakeNavigation?.Name} with ID {Id}.";
    }
  }
}
```

Step 3: Create the Person Owned Class

Owned classes can be reused between other entities.

- Add a folder named `Owned` under the `Entities` folder, and add a new class named `Person.cs`.
- Add the following using statements to the class:

```
using System.ComponentModel.DataAnnotations;
using Microsoft.EntityFrameworkCore;
```

- Update the code for the class to the following:

```
namespace AutoLot.Models.Entities.Owned
{
  [Owned]
  public class Person
  {
    [Required, StringLength(50)] public string FirstName { get; set; } = "New";
    [Required, StringLength(50)] public string LastName { get; set; } = "Customer";
  }
}
```

Step 4: Create the Customer Entity

- Add a new class to the `Entities` folder named `Customer.cs`
- Update the using statements to the following:

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text.Json.Serialization;
using AutoLot.Models.Entities.Base;
using AutoLot.Models.Entities.Owned;
```

- Update the code for the class to the following:

```
namespace AutoLot.Models.Entities
{
  [Table("Customers", Schema = "Dbo")]
  public partial class Customer : BaseEntity
  {
    public Person PersonalInformation { get; set; } = new Person();
    [JsonIgnore]
    [InverseProperty(nameof(CreditRisk.CustomerNavigation))]
    public IEnumerable<CreditRisk> CreditRisks { get; set; } = new List<CreditRisk>();
    [JsonIgnore]
    [InverseProperty(nameof(Order.CustomerNavigation))]
    public IEnumerable<Order> Orders { get; set; } = new List<Order>();
    [NotMapped]
    public string FullName
      => $"{PersonalInformation?.FirstName} {PersonalInformation?.LastName}";
  }
}
```

Step 5: Create the CreditRisk Entity

The CreditRisk entity also uses the Person owned class.

- Add a new class to the `Entities` folder named `CreditRisk.cs`
- Update the using statements to include the following:

```
using System.ComponentModel.DataAnnotations.Schema;
using AutoLot.Models.Entities.Base;
using AutoLot.Models.Entities.Owned;
```

- Update the code for the class to the following:

```
namespace AutoLot.Models.Entities
{
  [Table("CreditRisks", Schema = "Dbo")]
  public partial class CreditRisk : BaseEntity
  {
    public Person PersonalInformation { get; set; } = new Person();
    public int CustomerId { get; set; }

    [ForeignKey(nameof(CustomerId))]
    [InverseProperty(nameof(Customer.CreditRisks))]
    public virtual Customer? CustomerNavigation { get; set; }
  }
}
```

Step 6: Create the Make Entity

- Add a new class to the `Entities` folder named `Make.cs`
- Add the following using statements to the class:

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text.Json.Serialization;
using AutoLot.Models.Entities.Base;
```

- Update the code for the class to the following:

```
namespace AutoLot.Models.Entities
{
  [Table("Makes", Schema = "dbo")]
  public partial class Make : BaseEntity
  {
    [StringLength(50), Required] public string Name { get; set; } = "Ford";
    [JsonIgnore]
    [InverseProperty(nameof(Car.MakeNavigation))]
    public IEnumerable<Car> Cars { get; set; } = new List<Car>();
  }
}
```

Step 7: Create the Order Entity

- Add a new class to the `Entities` folder named `Order.cs`
- Update the using statements to match the following:

```
using System.ComponentModel.DataAnnotations.Schema;
using AutoLot.Models.Entities.Base;
```

- Update the code for the class to the following:

```
namespace AutoLot.Models.Entities
{
  [Table("Orders", Schema = "Dbo")]
  public partial class Order : BaseEntity
  {
    public int CustomerId { get; set; }
    public int CarId { get; set; }
    [ForeignKey(nameof(CarId))]
    [InverseProperty(nameof(Car.Orders))]
    public virtual Car? CarNavigation { get; set; }
    [ForeignKey(nameof(CustomerId))]
    [InverseProperty(nameof(Customer.Orders))]
    public virtual Customer? CustomerNavigation { get; set; }
  }
}
```

Step 8: Create the Logging Entity

SeriLog has an option to write log entries to a database table.

- Add a new class to the `Entities` folder named `SeriLogentry.cs`
- Add the following using statements to the class:

```
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Xml.Linq;
```

- Update the code for the class to the following:

```
namespace AutoLot.Models.Entities
{
  [Table("SeriLogs", Schema = "Logging")]
  public class SeriLogEntry
  {
    [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public int Id { get; set; }
    public string? Message { get; set; }
    public string? MessageTemplate { get; set; }
    [MaxLength(128)]
    public string? Level { get; set; }
    [DataType(DataType.DateTime)]
    public DateTime? TimeStamp { get; set; }
    public string? Exception { get; set; }
    public string? Properties { get; set; }
    public string? LogEvent { get; set; }
    public string? SourceContext { get; set; }
    public string? RequestPath { get; set; }
    public string? ActionName { get; set; }
    public string? ApplicationName { get; set; }
    public string? MachineName { get; set; }
    public string? FilePath { get; set; }
    public string? MemberName { get; set; }
    public int? LineNumber { get; set; }
    [NotMapped]
    public XElement? PropertiesXml => (Properties != null)? XElement.Parse(Properties):null;
  }
}
```

NOTE: The project compiles at this point.

Part 3: Create the ViewModels in AutoLot.Models

There are two ViewModels used by the applications.

- Create a new folder in the named `ViewModels` under the `Entities` folder.

Step 1: Create the CustomerOrderViewModel

- Add a new class to the `ViewModels` folder named `CustomerOrderViewModel.cs`
- Add the following using statements to the class:

```
using System.ComponentModel.DataAnnotations.Schema;
using Microsoft.EntityFrameworkCore;
```

- Update the code for the class to the following:

```
namespace AutoLot.Models.ViewModels
{
  [Keyless]
  public class CustomerOrderViewModel
  {
    public string? FirstName { get; set; }
    public string? LastName { get; set; }
    public string? Color { get; set; }
    public string? PetName { get; set; }
    public string? Make { get; set; }
    [NotMapped]
    public string FullDetail
      => $"{FirstName} {LastName} ordered a {Color} {Make} named {PetName}";
    public override string ToString() => FullDetail;
  }
}
```

Step 2: Create the DealerInfo Class

The `DealerInfo` is a non-persisted class.

- Add a new class to the `ViewModels` folder named `DealerInfo.cs`
- Update the code for the class to the following:

```
namespace AutoLot.Models.ViewModels
{
    public class DealerInfo
    {
        public string? DealerName { get; set; }
        public string? City { get; set; }
        public string? State { get; set; }
    }
}
```

Summary

In this lab, you created the Models (Entities) and the ViewModels for the applications.

Next steps

In the next part of this tutorial series, you will create the DbContext, DbContext Factory, and run your first migration.