Herança

Criar uma classe **Transporte.cs** dentro da pasta Models.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace WebSite1.Models
{
    public abstract class Transporte
    {
        [Key]
        public int TransporteID { get; set; }

        [Required(ErrorMessage = "Preencha o nome do transporte")]
        [DisplayName("Nome")]
        public string Nome { get; set; }

    }
}
```

Criar uma classe **Automovel.cs** dentro da pasta **Models**, que herda **Transporte**.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace WebSite1.Models
{
    public class Automovel:Transporte
    {
        [Required(ErrorMessage = "Preencha o chassi do veículo")]
        [DisplayName("Chassi")]
        public string Chassi { get; set; }

        [Required(ErrorMessage = "Preencha o renavam do veículo")]
        [DisplayName("Renavam")]
        public string Renavam { get; set; }
    }
}
```

Criar uma classe **Aviao.cs** dentro da pasta **Models**, que herda **Transportes**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace WebSite1.Models
{
    public class Aviao:Transporte
    {


    }
}
```

OBS: Quando existem pelo menos 2 classes que herdam uma outra, no banco de dados é criado um discriminator para diferenciar as classes filhas. A técnica usada acima é a Sigle Table Inheritance ou Table Per Hierarchy (TPH). Existe ainda a técnica Table Per Type (TPT) – uma tabela por tipo (classe) - e Table Per Concrete Class (TPC) – cria tabelas das classes concretas e acrescenta os campos da classe mãe nas classes filhas.

Adicionar o acesso no Context

```
        public DbSet<Transporte> Transportes { get; set; }
```

Adicionar uma Migration

```
PM> Add-Migration transporte
```

Atualizar o banco de dados

```
PM> Update-Database
```

Observe o banco de dados e veja como foi criada a tabela transporte.

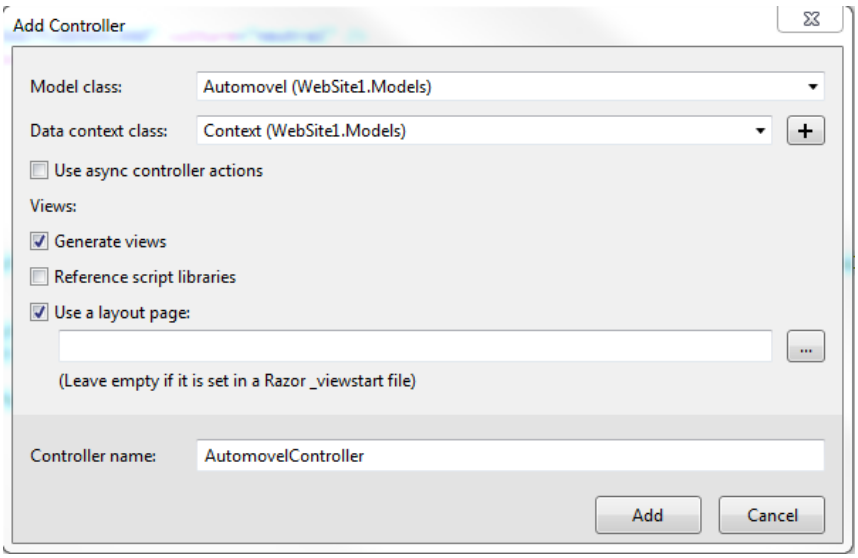
Scaffolding das telas de Automóvel

Comentar o bloco system.data e entityframework no web.config

```
<!--<system.data>
    <DbProviderFactories>
      <remove invariant="MySql.Data.MySqlClient"/>
      <add name="MySQL Data Provider" invariant="MySql.Data.MySqlClient" description=".Net
Framework Data Provider for MySQL" type="MySql.Data.MySqlClient.MySqlClientFactory,
MySql.Data, Version=6.8.7.0, Culture=neutral, PublicKeyToken=c5687fc88969c44d"/>
    </DbProviderFactories>
  </system.data>
  <entityFramework codeConfigurationType="MySql.Data.Entity.MySqlEFConfiguration,
MySql.Data.Entity.EF6">
    <defaultConnectionFactory type="System.Data.Entity.Infrastructure.SqlConnectionFactory,
EntityFramework"/>
    <providers>
      <provider invariantName="MySql.Data.MySqlClient"
type="MySql.Data.MySqlClient.MySqlProviderServices, MySql.Data.Entity.EF6"/>
    </providers>
  </entityFramework>-->
```

Criar o Controller

Botão direito na pasta Controllers, Add Controller
Selecionar MVC5 Controller with views, using EntityFramework



Descomentar o bloco system.data e entityframework no web.config

Ajustar o AutomovelController

Index

```
return View(db.Transportes.OfType<Automovel>().ToList());
```

Details

```
Automovel automovel = (Automovel)db.Transportes.Find(id);
```

Edit

```
Automovel automovel = (Automovel)db.Transportes.Find(id);
```

Delete

```
Automovel automovel = (Automovel)db.Transportes.Find(id);
```

DeleteConfirmed

```
Automovel automovel = (Automovel)db.Transportes.Find(id);
```

Adicionar um item no Menu

Acrescentar a linha abaixo no arquivo _Layout.cshtml

```
<li>@Html.ActionLink("Automóveis", "Index", "Automovel")</li>
```

Executar a aplicação

Acessar a URL http://localhost:porta/Publico/Logar