Aplicação MVC + Entity + MySQL

Criar a Classe Cliente

Adicionar a classe Cliente na pasta Models

```
    public class Cliente
    {
        [Key]
        public int ClienteID { get; set; }

        [Required(ErrorMessage = "Preencha o nome")]
        [DisplayName("Nome")]
        [StringLength(255, MinimumLength = 3, ErrorMessage = "O nome deve ter entre 3 e 255
caracteres.")]
        public string Nome { get; set; }

        [Required(ErrorMessage = "Preencha o email")]
        [DisplayName("Email")]
        [StringLength(255, MinimumLength = 3, ErrorMessage = "O email deve ter entre 3 e 255
caracteres.")]
        public string Email { get; set; }

        [Required(ErrorMessage = "Preencha a senha")]
        [DisplayName("Senha")]
        [StringLength(50, MinimumLength = 3, ErrorMessage = "A senha deve ter entre 3 e 50
caracteres.")]
        [DataType(DataType.Password)]
        public string Senha { get; set; }

        [DisplayName("Foto")]
        public string Foto { get; set; }

        [DisplayName("Data de Cadastro")]
        [DataType(DataType.DateTime, ErrorMessage = "Formato de data inválido")]
        [ScaffoldColumn(false)]
        [DisplayFormat(DataFormatString = "{0:dd/MM/yyyy HH:mm:ss tt}")]
        public DateTime DataCadastro { get; set; }

        [Required(ErrorMessage = "Selecione a Cidade")]
        [DisplayName("Cidade")]
        public int CidadeID { get; set; }

        public virtual Cidade Cidade { get; set; }

    }
```

Alterar a classe Cidade

Acrescentar o relacionamento

```
public virtual ICollection<Cliente> Clientes { get; set; }
```

Alterar a classe Context

```
        public DbSet<Estado> Estados { get; set; }
        public DbSet<Cidade> Cidades { get; set; }
        public DbSet<Cliente> Clientes { get; set; }
```

Adicionar Migration
Add-Migration cliente

Atualizar o Banco de Dados
Update-Database

Verifique no banco de dados se as tabelas Estado, Cidade e Cliente estão corretas.

Gerar as Telas de Cidade

Comentar os blocos system.data e entityFramework no web.config.

Adicionar um Novo Controller na pasta Controllers.

- MVC 5 Controller with views, using EntityFramework
  - Model Class: Cliente
  - Data Context class: Context
  - Controller name: ClientesController
  - Deixar marcado: Generate views; Reference script libraries; Use a layout page.

Descomentar os blocos system.data e entityFramework no web.config.

Altere o Layout principal

Na pasta Views/Shared/_Layout.cshtml

Adicione o item no menu

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

Execute a aplicação.
Acesse a lista de Clientes.
Acesse o Cadastro de Cliente (já deve ter gerado com a Lista de Cidades).
- Preencha o e-mail com qualquer texto que não seja um e-mail válido. Vai permitir Salvar.
- A senha já está com máscara.

Alterar o tipo de dados do atributo Email da classe Cliente

```
[Required(ErrorMessage = "Preencha o email")]
[DisplayName("Email")]
[DataType(DataType.EmailAddress)]
[StringLength(255, MinimumLength = 3, ErrorMessage = "O email deve ter entre 3 e 255
caracteres.")]
public string Email { get; set; }
```

Não precisa de nova Migration pois essa alteração é apenas em Tela.

Execute a aplicação.
Acesse a lista de Clientes.
Acesse o Cadastro de Cliente.
- Preencha o e-mail com qualquer texto que não seja um e-mail válido. **Não** vai permitir Salvar.
Porém, a mensagem de erro deve estar em inglês.

Alterar o tipo de dados do atributo Email da classe

```
[Required(ErrorMessage = "Preencha o email")]
[DisplayName("Email")]
[DataType(DataType.EmailAddress)]
[EmailAddress(ErrorMessage = "E-mail inválido")]
[StringLength(255, MinimumLength = 3, ErrorMessage = "O email deve ter entre 3 e 255
caracteres.")]
public string Email { get; set; }
```

Não precisa de nova Migration pois essa alteração é apenas em Tela.

Execute a aplicação.
Acesse a lista de Clientes.
Acesse o Cadastro de Cliente.
- Preencha o e-mail com qualquer texto que não seja um e-mail válido. **Não** vai permitir Salvar.
Mensagem vai estar OK.

Salvar a data de cadastro

Se observarmos a data de cadastro no banco de dados, ela está com uma informação errada.
Precisamos alterar algumas linhas no ClientesController.
Na Action Create, adicione a linha abaixo.

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create([Bind(Include =
"ClienteID,Nome,Email,Senha,Foto,DataCadastro,CidadeID")] Cliente cliente)
        {
            if (ModelState.IsValid)
            {
                cliente.DataCadastro = DateTime.Now;

                db.Clientes.Add(cliente);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.CidadeID = new SelectList(db.Cidades, "CidadeID", "Nome",
cliente.CidadeID);
            return View(cliente);
        }
```

Execute a aplicação.
Acesse o Cadastro de Cliente.
Adicione um cliente e verifique no banco de dados a data de cadastro.

Upload da Foto no Cadastro

Criar pasta para Upload

- Criar Pasta Files na raiz do Projeto.
- Dentro desta pasta, criar Pasta Cliente.

Adicionar variáveis no web.config

- Dentro do <appSettings>, adicionar mais 2 variáveis
- `<add key="PathFiles" value="C:\...\Documents\Visual Studio 2013\Projects\WebSite1\WebSite1\Files"/>`
- `<add key="PathWeb" value="http://localhost:9742"/>`

Alterar a tela de Cadastro de Cliente

Na tela Views/Clientes/Create.cshtml, alterar a declaração do formulário.
Substituir.

```
@using (Html.BeginForm())
```

Por.

```
@using (Html.BeginForm("Create", "Clientes", null, FormMethod.Post, new { enctype = "multipart/form-data" }))
```

Alterar a div da Foto

Substituir

```
        <div class="form-group">
            @Html.LabelFor(model => model.Foto, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Foto, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Foto, "", new { @class = "text-danger" })
            </div>
        </div>
```

Por

```
         <div class="form-group">
             @Html.LabelFor(model => model.Foto, htmlAttributes: new { @class = "control-label col-md-2" })
             <div class="col-md-10">
                 <input type="file" id="Foto" name="foto" />
                 @{
                     string str = ViewBag.FotoMensagem;
                     if (str != "")
                     {
                         <span class="text-danger contenterror">
                             @ViewBag.FotoMensagem
                         </span>
                     }
                 }
             </div>
         </div>
```

Alterar a Action Create

No arquivo ClientesController, alterar a Action Create
Adicionar ao final da assinatura do método, o parâmetro que vai receber a imagem.

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create([Bind(Include =
"ClienteID,Nome,Email,Senha,Foto,DataCadastro,CidadeID")] Cliente cliente,
HttpPostedFileBase foto)
```

Alterar a Action para salvar a imagem

```
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include =
"ClienteID,Nome,Email,Senha,Foto,DataCadastro,CidadeID")] Cliente cliente,
HttpPostedFileBase foto)
        {
            ViewBag.FotoMensagem = "";
            try
            {
                if (ModelState.IsValid)
                {
                    string fileName = "";
                    string contentType = "";
                    string path = "";

                    if (foto != null && foto.ContentLength > 0)
                    {
                        fileName = System.IO.Path.GetFileName(foto.FileName);
                        contentType = foto.ContentType;
                        path =
System.Configuration.ConfigurationManager.AppSettings["PathFiles"] + "\\Cliente\\" +
fileName;
                        foto.SaveAs(path);
                        cliente.Foto = fileName;
                    }

                    cliente.DataCadastro = DateTime.Now;
                    db.Clientes.Add(cliente);
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
            }
            catch (Exception ex)
            {
                ViewBag.FotoMensagem = "Não foi possível salvar a foto";
            }
            ViewBag.CidadeID = new SelectList(db.Cidades, "CidadeID", "Nome",
cliente.CidadeID);
            return View(cliente);

        }
```

Execute a aplicação.
Acesse o Cadastro de Cliente.
Adicione um cliente e verifique no banco de dados e na pasta Files/Cliente se a foto foi salva.

Alterar a Lista de Clientes para exibir a Foto

Na página Views/Clientes/Index.cshtml, alterar a coluna <td> referente a Foto, que está dentro do foreach.

Alterar de

```
            <td>
                @Html.DisplayFor(modelItem => item.Foto)
            </td>
```

Para

```
<td>
@{
       string foto = item.Foto;
       if (foto != string.Empty)
       {
          if (foto != null)
          {
              string file = foto;
              string path =
System.Configuration.ConfigurationManager.AppSettings["PathWeb"] + "/Files/Cliente/" + file;
              <img src='@Url.Content(path)' style="height:50px;" title="Foto" /><br />
              <a href="@path" target="_blank">Visualizar</a>
           }
         }
  }
</td>
```

Execute a aplicação.
Acesse a Lista de Clientes.

Alterar a tela de Edição de Cliente

Na tela Views/Clientes/Edit.cshtml, alterar a declaração do formulário.
Substituir.

```
@using (Html.BeginForm())
```

Por:

```
@using (Html.BeginForm("Edit", "Clientes", null, FormMethod.Post, new { enctype =
"multipart/form-data" }))
```

Alterar a página Views/Clientes/Edit.cshtml.
Substituir a div que mostra a Foto.

```
<div class="form-group">
@Html.LabelFor(model => model.Foto, htmlAttributes: new { @class = "control-label col-md-2" })
   <div class="col-md-10">
     @Html.EditorFor(model => model.Foto, new { htmlAttributes = new { @class = "form-control" } })
     @Html.ValidationMessageFor(model => model.Foto, "", new { @class = "text-danger" })
   </div>
</div>
```

Alterar para

```
<div class="form-group">
@Html.LabelFor(model => model.Foto, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
    @{
    string foto = Model.Foto;
    if (foto != string.Empty)
    {
        if (foto != null)
        {
            string file = foto;
            string path =
System.Configuration.ConfigurationManager.AppSettings["PathWeb"] + "/Files/Cliente/" + file;
            <img src='@Url.Content(path)' style="height:auto;" title="Foto" /><br />
            <a href="@path" target="_blank">Visualizar</a>
            @Html.ActionLink("Excluir", "DeleteFile", "Clientes", new { id = Model.ClienteID, arquivo =
"Foto" }, null)
        }
        else
        {
            <div class="col-md-10">
                <input type="file" id="Foto" name="foto" />
            </div>
        }
    }
    else
    {
        <div class="col-md-10">
            <input type="file" id="Foto" name="foto" />
        </div>
    }
    string str = ViewBag.FotoMensagem;
    if (str != "")
    {
        <span class="text-danger contenterror">
            @ViewBag.FotoMensagem
        </span>
    }
   }
</div>
</div>
```

No arquivo ClientesController, alterar a Action Edit
Adicionar ao final da assinatura do método, o parâmetro que vai receber a imagem.

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit([Bind(Include =
"ClienteID,Nome,Email,Senha,Foto,DataCadastro,CidadeID")] Cliente cliente,
HttpPostedFileBase foto)
```

Alterar a Action para salvar a imagem

```
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include =
"ClienteID,Nome,Email,Senha,Foto,DataCadastro,CidadeID")] Cliente cliente, HttpPostedFileBase
foto)
        {
            ViewBag.FotoMensagem = "";
            try
            {
                string fileName = "";
                string contentType = "";
                string path = "";

                Cliente clienteBD = db.Clientes.Find(cliente.ClienteID);
                if (foto != null && foto.ContentLength > 0)
                {
                    fileName = System.IO.Path.GetFileName(foto.FileName);
                    contentType = foto.ContentType;
                    path = System.Configuration.ConfigurationManager.AppSettings["PathFiles"] +
"\\Cliente\\" + fileName;
                    foto.SaveAs(path);
                    cliente.Foto = fileName;
                }
                else
                {
                    if (foto == null)
                    {
                        if (clienteBD.Foto != null)
                        {
                            if (clienteBD.Foto.Length > 0)
                            {
                                //usa valores que ja estao no BD
                                cliente.Foto = clienteBD.Foto;
                            }
                        }
                    }
                }
                ((IObjectContextAdapter)db).ObjectContext.Detach(clienteBD);
                db.Entry(cliente).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            catch (Exception ex)
            {
                ViewBag.FotoMensagem = "Não foi possível salvar a foto";
            }

            ViewBag.CidadeID = new SelectList(db.Cidades, "CidadeID", "Nome", cliente.CidadeID);
            return View(cliente);
        }
```

Criar uma Action para excluir o arquivo

```
        public ActionResult DeleteFile(int? id, string arquivo)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Cliente cliente = db.Clientes.Find(id);
            try
            {
                if (ModelState.IsValid)
                {
                    string file = "";
                    switch (arquivo)
                    {
                        case "Foto":
                            file =
System.Configuration.ConfigurationManager.AppSettings["PathFiles"] + "\\Cliente\\" + cliente.Foto;
                            break;
                    }
                    if (System.IO.File.Exists(file))
                    {
                        System.IO.File.Delete(file);
                        Cliente clienteBD = db.Clientes.Find(cliente.ClienteID);
                        switch (arquivo)
                        {
                            case "Foto":
                                cliente.Foto = string.Empty;
                                break;
                        }
                        ((IObjectContextAdapter)db).ObjectContext.Detach(clienteBD);
                        db.Entry(cliente).State = EntityState.Modified;
                        db.SaveChanges();
                         ViewBag.CidadeID = new SelectList(db.Cidades, "CidadeID", "Nome",
cliente.CidadeID);
                        return View("Edit", cliente);
                    }
                    else
                    {
                        ViewBag.FotoMensagem = "Não foi possível excluir a foto";
                    }
                }
                else
                {
                    ViewBag.FotoMensagem = "Não foi possível excluir a foto";
                }
            }
            catch (Exception ex)
            {
                ViewBag.FotoMensagem = "Não foi possível excluir a foto";
            }
            ViewBag.CidadeID = new SelectList(db.Cidades, "CidadeID", "Nome", cliente.CidadeID);
            return View("Edit", cliente);
        }
```

Execute a aplicação.
Acesse a Edição de Clientes.
Editar e Excluir uma imagem do cliente.

Alterar a Lista de Clientes para exibir o Estado

No ClientesControllers, alterar a Action Index

```
        public ActionResult Index()
        {
            var clientes = db.Clientes.Include(c => c.Cidade).Include(e => e.Cidade.Estado);
            return View(clientes.ToList());
        }
```

Na página Views/Clientes/Index.cshtml, alterar a primeira coluna <td> dentro do foreach

```
            <td>
                @Html.DisplayFor(modelItem => item.Cidade.Nome)
(  @Html.DisplayFor(modelItem => item.Cidade.Estado.Descricao)    )
            </td>
```

Execute a aplicação.
Acesse a Lista de Clientes.

Alterar a Lista de Clientes para exibir os clientes em ordem alfabética

No ClientesControllers, alterar a Action Index

```
        public ActionResult Index()
        {
            var clientes = db.Clientes.Include(c => c.Cidade).Include(e =>
e.Cidade.Estado).OrderBy(o => o.Nome);
            return View(clientes.ToList());
        }
```

Execute a aplicação.
Acesse a Lista de Clientes.