PROGRAMAÇÃO VISUAL

ASP.NET Core MVC – Entity Framework

ENTITY FRAMEWORK

Entity Framework (EF) é uma framework ORM standard, parte da plataforma .NET

- Fornece uma infraestrutura em tempo de execução para a gestão de dados baseados em SQL efetuada através de objetos .NET

O esquema relacional é mapeado para um modelo de objetos (classes e associações)

- O Visual Studio possui ferramentas integradas que criam os mapeamentos EF de dados SQL
  - Data mappings são compostos por classes C# e XML
- É fornecida uma API standard de manipulação de dados

CARACTERÍSTICAS DA ENTITY FRAMEWORK

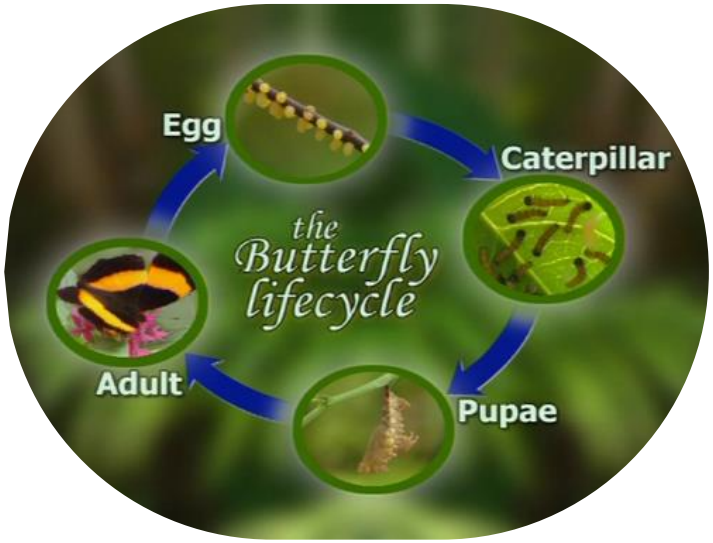
Características standard da Entity Framework (EF):

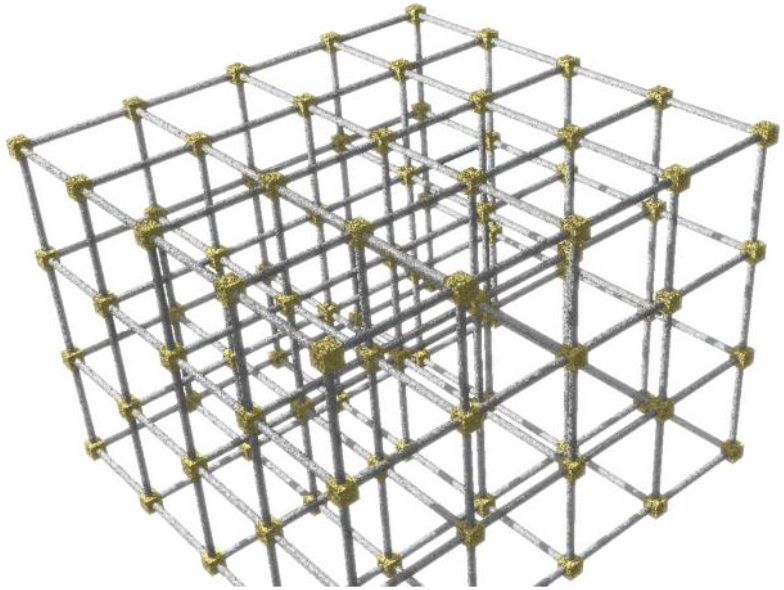
- Mapeamento de tabelas, views, stored procedures e funções para objetos .NET
- Disponibiliza queries baseadas em LINQ
  - Executadas como comandos SQL SELECT no servidor de base de dados
- Operações CRUD – Create/Read/Update/Delete
- Criação de queries compiladas – para a execução da mesma query parametrizada várias vezes
- Criação ou remoção de esquemas de base de dados

ENTITY FRAMEWORK LIFECYCLE

Quando se inicia a aplicação

- A EF traduz em comandos SQL as queries ao modelo de objetos
- Envia-as para a base de dados para a sua execução posterior





ENTITY FRAMEWORK LIFECYCLE (2)

Quando a base de dados retorna os resultados

- A Entity Framework traduz os registos obtidos da base de dados novamente para objectos NET

O servidor de base de dados é praticamente transparente ficando Escondido por detrás da API

- O LINQ é executado sobre `IQueryable<T>`
  - Em tempo de compilação é criada uma *query expression tree*
  - Em tempo de execução as instruções SQL são criadas e executadas.

COMPONENTES EF

A classe `ObjectContext`

- `ObjectContext` guarda a ligação à base de dados e as classes da EF
- Fornece o acesso aos dados feito através de LINQ
- Implementa o seguimento (tracking) das identidades, das alterações, e a API para as operações de CRUD

Entity classes

- Cada tabela da base de dados é normalmente mapeada numa única classe de identidade (classe C#)

COMPONENTES EF (2)

Associações

- Uma associação é uma relação entre duas classes identidade do tipo chave primária / chave secundária
- Permite a navegação entre entidades. `Student.Courses`

Controlo de Concurrência

- A Entity Framework utiliza um controle de concorrência otimístico (sem locking por omissão)
- Fornece a deteção automática de conflitos de concorrência e meios para a sua resolução.

LINQ

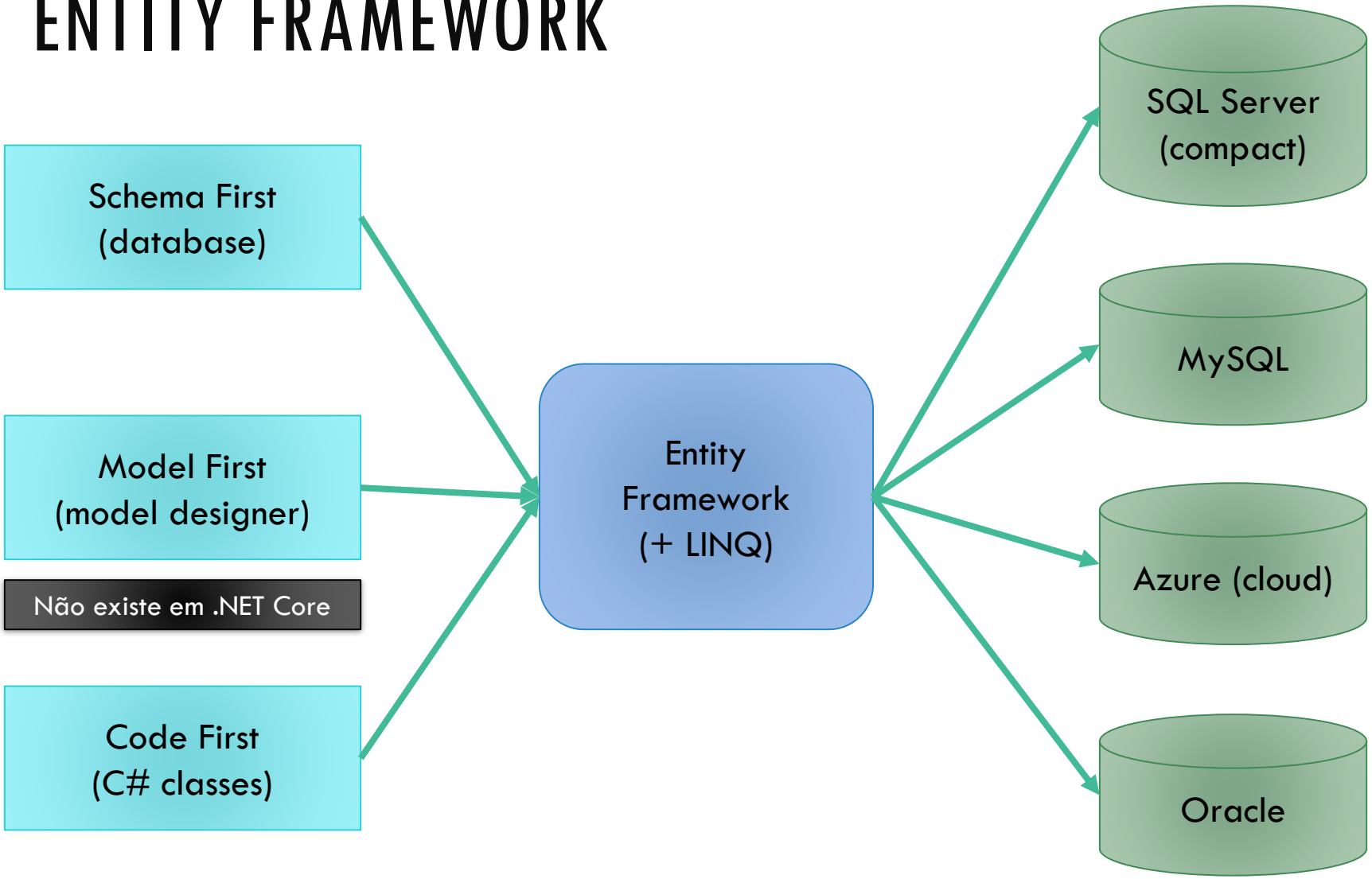
Sintaxe integrada

```
var query = from r in _db.Restaurants
            where r.Country == "USA"
            orderby r.Name
            select r;
```

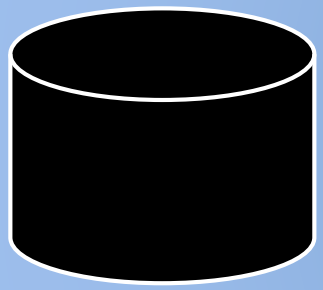
Sintaxe usando métodos de extensão

```
var query = _db.Restaurants
              .Where(r => r.Country == "USA")
              .OrderBy(r => r.Name)
              .Skip(10)
              .Take(10);
```

ENTITY FRAMEWORK



MODELOS ENTITY FRAMEWORK

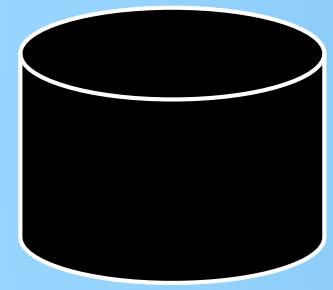


Model First

Create .edmx model in designer
Generate DB from .edmx
Classes auto-generate from .edmx

Code First

Define classes & mapping in code
Database auto-created at runtime



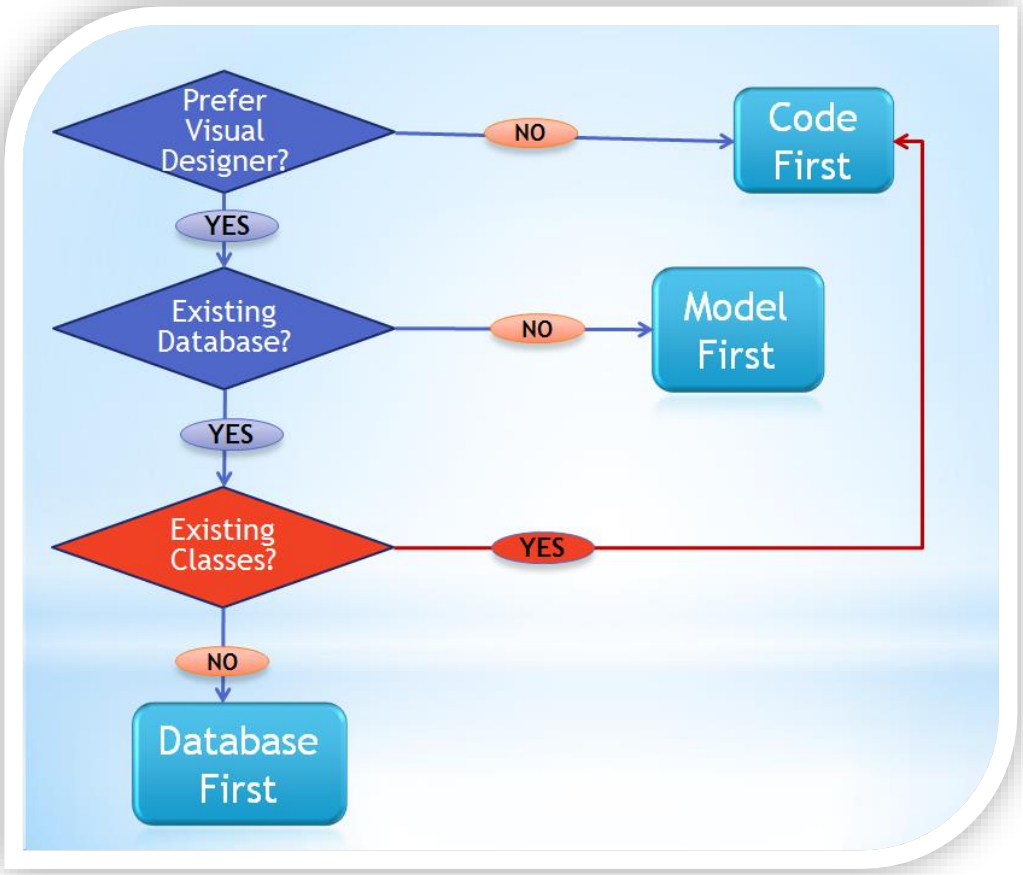
Database First

Reverse engineer .edmx model
Classes auto-generate from .edmx

Code First

Define classes & mapping in code

ABORDAGENS ENTITY FRAMEWORK



ABORDAGEM CODE FIRST

Convenções em vez de configurações

- Database naming
- Chave primária
- Relationships (propriedade de navegação)

Anotações de dados

- Dizem à EF como mapear o modelo de objetos para o modelo de dados relacional da base de dados
- As anotações são colocadas diretamente sobre as propriedades da classe do modelo
  - System.ComponentModel.DataAnnotations

TIPOS DE ANOTAÇÃO COMUNS

`Key` – Define a chave primária

`Column` – Define o nome do campo da BD

`Table` – Define o nome da tabela para uma classe

`Required` – Define um campo obrigatório da BD

`NotMapped` – Propriedade não mapeada para a BD

`MinLength()` – Tamanho mínimo para a propriedade

`MaximumLength()` – Tamanho máximo para a propriedade

`Range()` – Define uma gama de valores válida.

MIGRAÇÃO DA BASE DE DADOS

Package Manager Console

- Enable-Migrations (-ContextTypeName)
  - Cria uma pasta de migração no projeto
  - Cria uma tabela de sistema __MigrationHistory na BD
- Update-Database (-Verbose)
  - Aplica as alterações de migração na BD
  - –script – cria o *script* de SQL das alterações

CUSTOM MAPPING

Quando a base de dados não segue as convenções …

```
public class OdeToFoodDB  : DbContext
{
    ...

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Restaurant>()
                    .Property(r => r.ID).HasColumnName("restaurant_id");

        modelBuilder.Entity<Restaurant>()
                    .HasMany(restaurant => restaurant.Reviews)
                    .WithRequired(review => review.Restaurant);

        modelBuilder.Entity<Review>()
                    .ToTable("tbl_Reviews");

        base.OnModelCreating(modelBuilder);
    }
}
```

A problem has been detected and windows has been shut down to prevent damage
to your computer.

DRIVER_IRQL_NOT_LESS_OR_EQUAL

If this is the first time you've seen this Stop error screen,
restart your computer, If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical information:

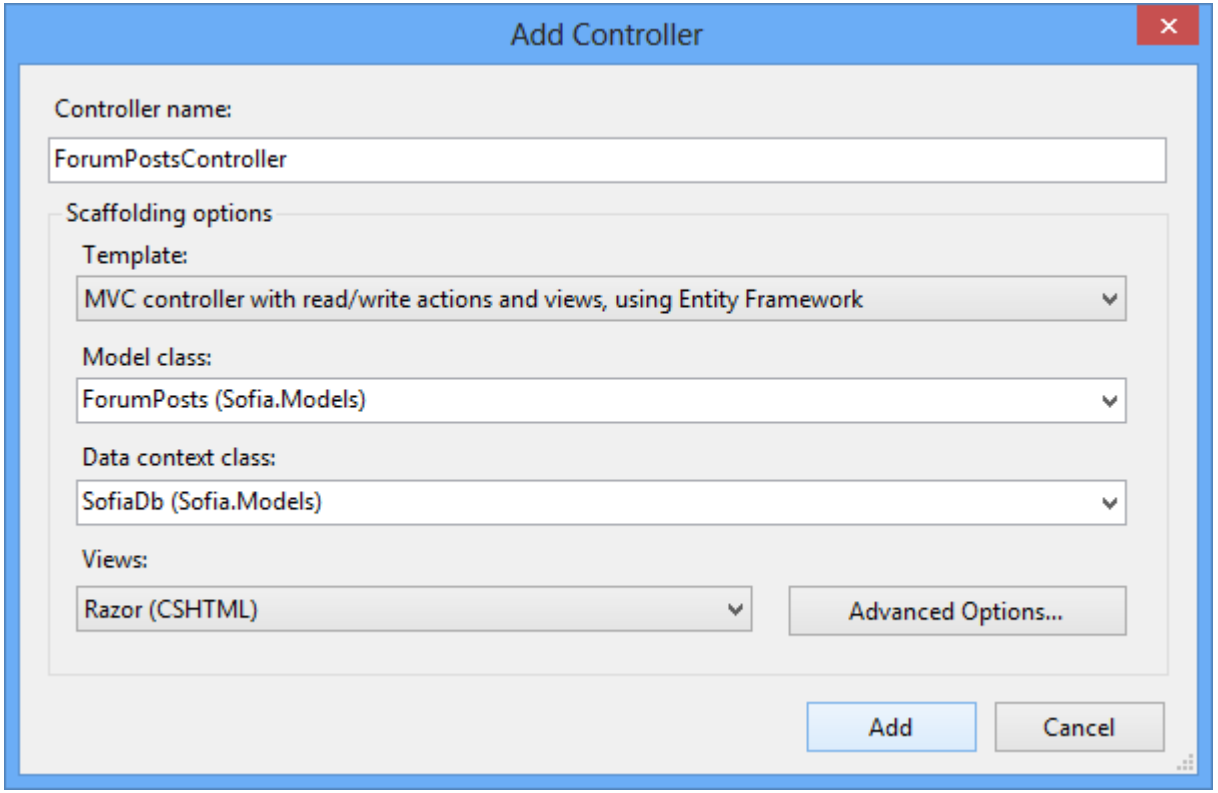
*** STOP: 0x000000D1 (0xOOOOOOOC,0x00000002,0x00000000,0xF86B5A89)

***   gv3.sys - Address F86B5A89 base at F86B5000, DateStamp 3dd991eb

Beginning dump of physical memory
Physical memory dump complete.
Contact your system administrator or technical support group for further
assistance.

SCAFFOLDING – MODELOS CONTROLLER E VIEW

- TODO: Editor templates e view templates



MVC – VALIDAÇÃO COM ANOTAÇÕES

Atributos definidos em
System.ComponentModel.DataAnnotations

Cobrem os principais padrões de validação

- **Required**
- **StringLength**
- **Regex**
- **Range**

```
public class LogOnModel
{
    [Required]
    public string UserName { get; set; }

    [Required]
    public string Password { get; set; }

    public bool RememberMe { get; set; }
}
```

MVC – ATRIBUTOS DE VALIDAÇÃO DOS DADOS

Atributo	Descrição
`Compare`	Verifica se duas propriedades do modelo são idênticas .
`CreditCard`	A propriedade deve ter o formato usado pelos cartões de crédito.
`EmailAddress`	A propriedade deve ter o formato de um endereço de email
`Range`	O valor da propriedade deve estar dentro de um determinado intervalo.
`RegularExpression`	A propriedade deve estar de acordo com uma determinda expressão regular.
`Required`	O valor da propriedade deve ser fornecido sempre.
`StringLength`	A propriedade de texto não pode ultrapassar o valor fornecido
`Url`	A propriedade deve ter o formato de um URL.

MVC – VALIDAÇÃO DEFINIDA PELO UTILIZADOR

Atributos definidos pelo utilizador

Derivam de **`ValidationAttribute`**

Redefinem o método **`IsValid`**

```
[AttributeUsage(AttributeTargets.Property)]
public sealed class MinLengthAttribute : ValidationAttribute
{

    // …

    public override bool IsValid(object value)
    {
        string valueAsString = value as string;
        return (valueAsString != null &&
                valueAsString.Length >= _minCharacters);
    }
}
```

MVC – MODELO DE VALIDAÇÃO DO CONTROLADOR

`ModelState.IsValid` – Indica se a validação teve sucesso

`ModelState.AddModelError` – Erro definido pelo utilizador

```
[HttpPost]
public ActionResult Edit(ForumPosts forumPost)
{
    if (ModelState.IsValid)
    {
        if (forumPost.Author != "Nikolay.IT")
        {
            ModelState.AddModelError("Author", "Wroooooooong!");
        }
        db.Entry(forumPost).State = EntityState.Modified;
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    return View(forumPost);
}
```

MVC – MODELO DE VALIDAÇÃO DA VISTA

`@Html.ValidationSummary` – mostra erros

`@Html.ValidationMessageFor(…)` – Mostra mensagem de validação para a propriedade referida

É preferível usar os **Tag Helpers** da validação

```
@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <div class="editor-label">
        @Html.LabelFor(model => model.Title)
    </div>
    <div class="editor-field">
        @Html.EditorFor(model => model.Title)
        @Html.ValidationMessageFor(model => model.Title)
    </div>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Text box with integrated client-side validation

jQuery validation library required for unobtrusive JavaScript validation

MVC – MODELO DE VALIDAÇÃO DA VISTA

Exemplo com **Tag Helpers**

```
<form action="/Movies/Create" method="post">
    <div class="form-horizontal">
        <h4>Movie</h4>
        <div class="text-danger"></div>
        <div class="form-group">
            <label class="col-md-2 control-label" for="ReleaseDate">ReleaseDate</label>
            <div class="col-md-10">
                <input class="form-control" type="datetime"
                data-val="true" data-val-required="The ReleaseDate field is required."
                id="ReleaseDate" name="ReleaseDate" value="" />
                <span class="text-danger field-validation-valid"
                data-valmsg-for="ReleaseDate" data-valmsg-replace="true"></span>
            </div>
        </div>
    </div>
</form>
```

MVC – ANOTAÇÕES DE DISPLAY / EDIT

Atributo	Descrição
DataType	Dá pistas ao *View engine* para a formatação dos dados. Ex: email, datas, moeda, etc.
DisplayFormat	Especifica explicitamente o formato das datas
Display(Name=…)	Texto a aparecer nas vista como Label para a propriedade anotada.

REFERÊNCIAS

- Telerik Software Academy
  - academy.telerik.com