Fundamentos de Desenvolvimento Web/BD

Aula 05: Forms e Validação

Professor: Rinaldo Ferreira Junior

E-mail: rinaldo.fjunior@prof.infnet.edu.br

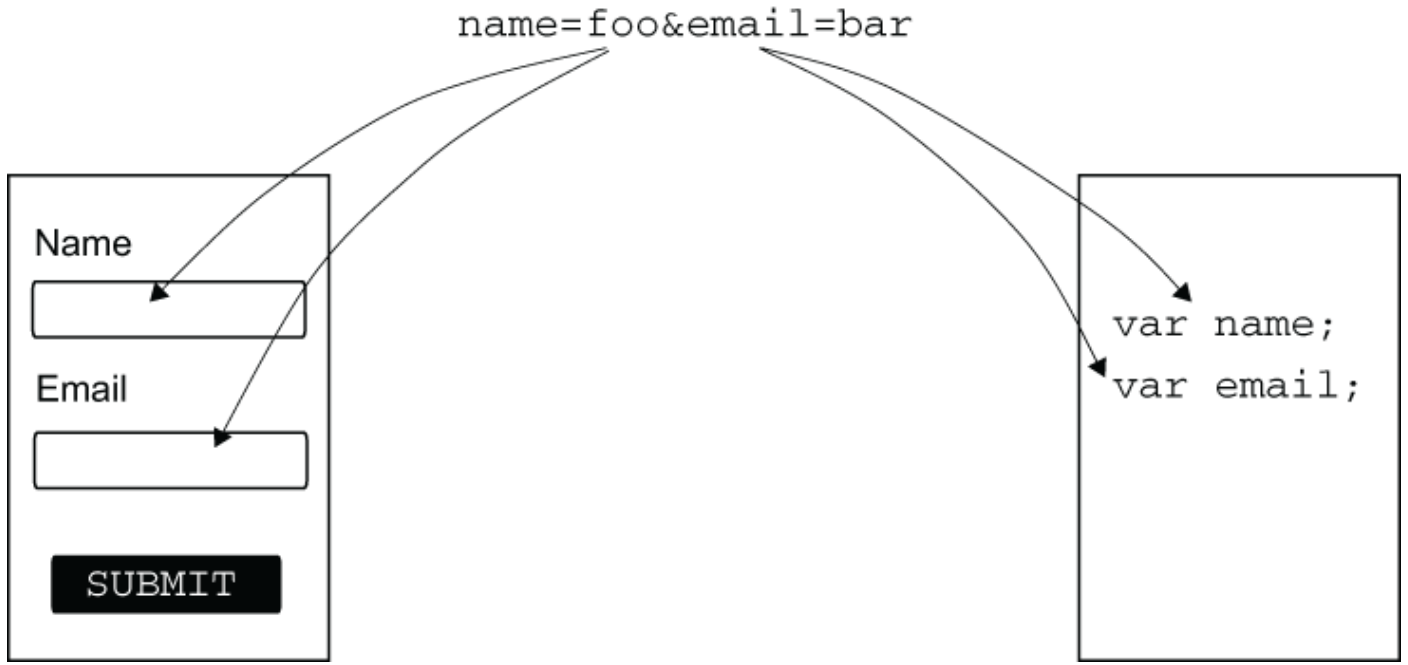
- **Professor:** Rinaldo Ferreira Junior
- **Graduação:** Pós-graduado em Arquitetura de Softwares
- **Atuação:** .Net | C# | SQL | NoSQL | Engenheiro de Software
- **E-mail:** rinaldo.fjunior@prof.infnet.edu.br
- **Linkedin:** https://www.linkedin.com/in/rinaldo-ferreira-junior-787326a

Aula 05: Roteiro

- Forms
- Validação
- HTML Helpers

Aula 05: Forms

- Forms é um element do HTML que agrupa uma série de outros elementos, com o objetivo de coletar entrada de dados do usuário..
- O elemento Forms é denotado pela tag <forms></forms>.
- Deve haver um controle na tela que submeta o form para processamento no servidor. Isso se dá por meio de um request.
  - Se o request for via GET, os dados são adiconados à URL
  - Se o request for via POST, os dados são adiconados ao body



Aula 05: O Padrão post-redirect-get

Instituto Infnet

- Normalmente, após submeter dados através de um form, o usuário é redirecionado para outra tela. Isso é o padrão post-redirect-get.
- Tão logo o processamento do form termine, uma página de confirmação é exibida ao usuário. Esse é o GET do parão.
- Esse padrão é amplamente usado e recomendado, pois evita que o usuário faça múltiplos refreshs na página, submetendo os mesmo dados várias vezes e causando reprocessamento.
  - Se os dados forem armazenado em um BD por exemplo, isso pode causar duplicidade
- Outra vantagem do uso desse padrão, é que ele ajuda a tornar o site mais amigável e intuitivo. Sem o redirect, o usuário pode ser confundido e achar que o form não foi enviado.

Aula 05: Lendo Dados por Request

Instituto Infnet

- No ASP.Net Core, dados do form enviados por POST são armazenados na coleção Request.Forms.
- No ASP.Net Core, dados do form enviados por GET são armazenados na coleção Request.Query.
- Cada item da coleção Forms pode ser acessado através de uma string, representando o nome do campo, por exemplo, Request.Form[“email"].

```
public void OnPost()
{
    if(!string.IsNullOrEmpty(Request.Form["cidade"]))
    {
        Message = $"Campo Cidade lido no request: {Request.Form["cidade"]}";
    }
}
```

```
public void OnGet()
{
    if (!string.IsNullOrEmpty(Request.Query["cidade"]))
    {
        Message = $"Campo Cidade lido no GET {Request.Query["cidade"]}";
    }
}
```

Aula 05: Percorrendo a Coleção

- Como se tratam de coleções, tanto Forms quanto Query podem ser percorridos, para se obter todos os valores.

```
<ul>
    @if (Request.HasFormContentType)
    {
        foreach (var item in Request.Form)
        {
        <li>@item.Key: @Request.Form[item.Key]</li>
        }
    }
</ul>
```

Aula 05: Validação Client Side

Instituto Infnet

- Através de Model Binding, é possível gerar validação a nível de cliente, automaticamente. O Razor gera os scripts necessários para a validação e exibição de erros.
- A construção da validação se dá por Tag Helpers e atributos de validação, adicionados ao modelo da página.

```
public class ValidacaoModel : PageModel
{
    [BindProperty]
    [Required(ErrorMessage = "Logradouro é obrigatório")]
    2 references
    public string Logradouro { get; set; } = string.Empty;

    [BindProperty]
    [StringLength(20, ErrorMessage = "Bairro deve conter no máximo, 20 caracteres")]
    2 references
    public string Bairro { get; set; } = string.Empty;

    [BindProperty]
    [Required(ErrorMessage = "Cidade é obrigatório")]
    2 references
    public string Cidade { get; set; } = string.Empty;

    [BindProperty]
    1 reference
    public string Mensagem { get; set; } = string.Empty;

    0 references
    public void OnGet()
    {
    }
}
```

Atributos de validação são recursos do namespace System.ComponentModel.DataAnnotations;

Aula 05: Validação Client Side

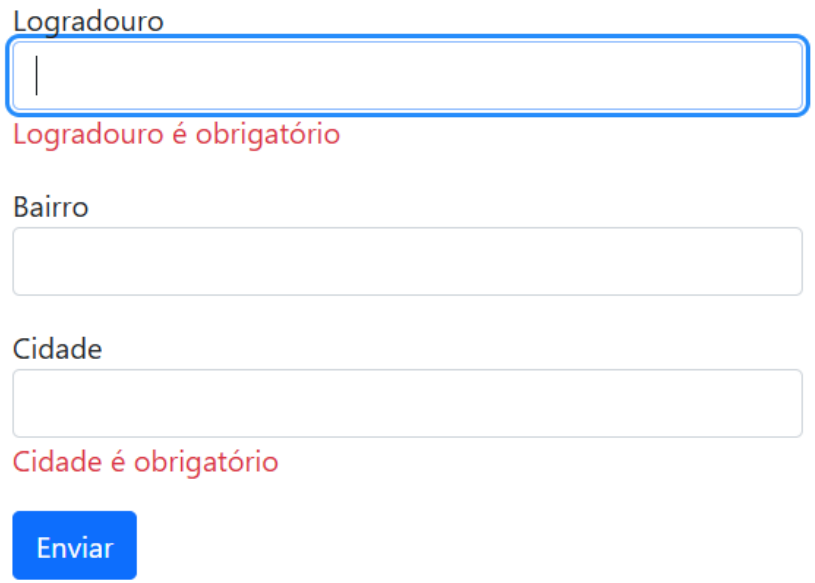
- No lado do front, as Tag Helpers associam os controles ao modelo e vinculam os atributos de validação. O arquivo _ValidationScriptsPartial possui as chamadas para o Jquery.

```
@section scripts {
    <partial name="_ValidationScriptsPartial" />
}
<div class="col-4">
    <form method="post">
        <div class="mb-3">
            <label for="name">Logradouro</label>
            <input asp-for="Logradouro" class="form-control" type="text" maxlength="10" />
            <span asp-validation-for="Logradouro" class="text-danger"></span>
        </div>
        <div class="mb-3">
            <label for="name">Bairro</label>
            <input asp-for="Bairro" class="form-control" type="text" />
            <span asp-validation-for="Bairro" class="text-danger"></span>
        </div>
        <div class="mb-3">
            <label for="name">Cidade</label>
            <input asp-for="Cidade" class="form-control" type="text" />
            <span asp-validation-for="Cidade" class="text-danger"></span>
        </div>
        <button class="btn btn-primary">Submit</button>
    </form>
</div>
```

Aula 05: Validação Client Side

Instituto Infnet

- As validações configuradas server side, são projetadas para os controles durante a renderização. Ao submeter a página, os scripts de validação são executados.




```
<form method="post">
    <div class="mb-3">
        <label for="name">Logradouro</label>
        <input class="form-control" type="text" maxlength="10" data-val="true" data-val-required="Logradouro &#xE9; obrigat&#xF3;rio" id="Logradouro" name="Logradouro" value="" />
        <span class="text-danger field-validation-valid" data-valmsg-for="Logradouro" data-valmsg-replace="true"></span>
    </div>
    <div class="mb-3">
        <label for="name">Bairro</label>
        <input class="form-control" type="text" data-val="true" data-val-length="Bairro deve conter no m&#xE1;ximo, 20 caracteres" data-val-length-max="20" id="Bairro" maxlength="20" name="Bairro" value="" />
        <span class="text-danger field-validation-valid" data-valmsg-for="Bairro" data-valmsg-replace="true"></span>
    </div>
    <div class="mb-3">
        <label for="name">Cidade</label>
        <input class="form-control" type="text" data-val="true" data-val-required="Cidade &#xE9; obrigat&#xF3;rio" id="Cidade" name="Cidade" value="" />
        <span class="text-danger field-validation-valid" data-valmsg-for="Cidade" data-valmsg-replace="true"></span>
    </div>
    <button class="btn btn-primary">Enviar</button>
```

Aula 05: Validação Client Side com Sumário

- Além de exibir erros por controles individuais, a validação pode centralizer todos os erros em um sumário, através da Tag Helper asp-validation-summary.

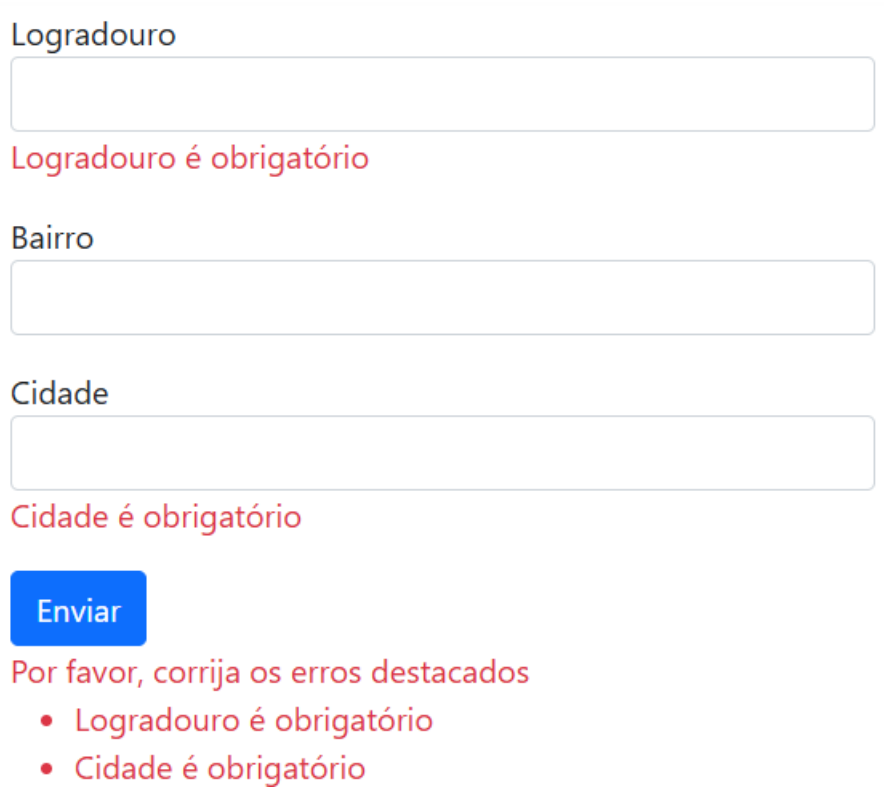
```
@section scripts {
    <partial name="_ValidationScriptsPartial" />
}
<div class="col-4">
    <form method="post">
        <div class="mb-3">
            <label for="name">Logradouro</label>
            <input asp-for="Logradouro" class="form-control" type="text" maxlength="10" />
            <span asp-validation-for="Logradouro" class="text-danger"></span>
        </div>
        <div class="mb-3">
            <label for="name">Bairro</label>
            <input asp-for="Bairro" class="form-control" type="text" />
            <span asp-validation-for="Bairro" class="text-danger"></span>
        </div>
        <div class="mb-3">
            <label for="name">Cidade</label>
            <input asp-for="Cidade" class="form-control" type="text" />
            <span asp-validation-for="Cidade" class="text-danger"></span>
        </div>
        <button class="btn btn-primary">Submit</button>
    </form>
    <div asp-validation-summary="All" class="text-danger">Foram encontrados erros</div>
```

Aula 05: Validação Client Side com Sumário

Instituto Infnet

- A exibição do sumário é controlada por CSS. Para evitar que o sumário apareça mesmo sem validação, altere a classe CSS responsável pela validação.

```css
.validation-summary-valid {
    display: none;
}
```



Aula 05: Validação Server Side

- Validação no cliente deve ser entendido como uma mera conveniência, por dar feedback imediato ao usuário. Não deve ser o único meio de validação da aplicação, pois pode facilmente ser contornada.
- Já a validação server side é executada sempre e, não pode ser desabilitada. Todas as validações configuradas para os controles, são reexecutadas no servidor. Se houver violação, o servidor devolve os mesmos erros que são retornados pela validação no nível do cliente.
- A página mantém um objeto ModelStateDictionary, um dicionário que mantém informações relacionadas aos erros encontrados pelo processo de validação.
- Para testar a validação no servidor, comente ou retire da página, a chamada para os scripts de validação

```
@section scripts {
    <partial name="_ValidationScriptsPartial" />
}
```

Aula 05: Validação com HTML Helpers

- HTML Helpers também podem ser usadas para validação. O HTML gerado é compatível com os browsers mais modernos.

```
<div class="col-4">
    <form method="post">
        <div class="mb-3">
            @Html.LabelFor(m ⇒ m.Logradouro, new { @class = "col-md-2 control-label" })
            @Html.TextBoxFor(m ⇒ m.Logradouro, new { @class = "form-control" })
            @Html.ValidationMessageFor(m ⇒ m.Logradouro, "*")
        </div>
        <div class="mb-3">
            @Html.LabelFor(m ⇒ m.Bairro, new { @class = "col-md-2 control-label" })
            @Html.TextBoxFor(m ⇒ m.Bairro, new { @class = "form-control" })
            @Html.ValidationMessageFor(m ⇒ m.Bairro, "*")
        </div>
        <div class="mb-3">
            @Html.LabelFor(m ⇒ m.Cidade, new { @class = "col-md-2 control-label" })
            @Html.TextBoxFor(m ⇒ m.Cidade, new { @class = "form-control" })
            @Html.ValidationMessageFor(m ⇒ m.Cidade, "*")
        </div>
        <button class="btn btn-primary">Enviar</button>
    </form>
    @Html.ValidationSummary(false, "Por favor, corrija os erros encontrados")
</div>
```