FIAP GRADUAÇÃO

#04 – ASP.NET MVC - Razor e HTML Helpers

FIAP

# ENTERPRISE APPLICATION DEVELOPMENT

Prof. Me. Thiago T. I. Yamamoto

#04 – ASP.NET MVC - RAZOR E HTML HELPERS

thiagoyama thiagoyama@gmail.com

# #04 – RAZOR E HTML HELPERS

- Razor
- Bloco Razor
- HTML Helpers
- Criação de Helpers

CONTEÚDO PROGRAMÁTICO

FIAP

# RAZOR

# RAZOR

- View engine, para construção de páginas dinámicas:
- Lançada na versão MVC 3, em janeiro 2011;
- Sintaxe simplificada
- Fácil de aprender
- Possui extensão .cshtml

# BLOCO RAZOR

- Toda expressão Razor começa com o caracter @
  - Neste exemplo estamos declarando um bloco, onde podemos inserir código C#:

```
@{
    //Código C#
}
```

```
@{
    for (int i = 0; i < 10; i++)
    {
        <p>@i</p>
    }
}
```

# LOOPS E IF-ELSE

- Além do bloco, podemos inserir comandos C# de forma mais simples:

```
@if (true)
{
    <p>Olá Adminstrador!</p>
}
else
{
    <p>Olá Usuário</p>
}
```

```
@for (int i = 0; i < 10; i++)
{
    <p>@i</p>
}
```

Para acessar uma variável, podemos utilizar a sintaxe @NomeDaVariavel

FIAP

# HTML HELPERS

# HTML HELPERS

- Encapsula o código HTML
- É um modo de facilitar a construção de páginas:
  - ActionLink
  - BeginForm
  - Checkbox
  - TextBox
  - EditorFor
  - EditorModelFor
  - Etc...

# ACTION LINK

- É utilizado para gerar links;
- Pode ser utilizado de diversas maneiras:

```
@Html.ActionLink(“Cadastro Cliente","Cadastrar")
```

Texto do Link — Action

```
@Html.ActionLink(“Cadastro Cliente","Cadastrar“, “Cliente“)
```

Texto do Link — Action — Controller

```
@Html.ActionLink("Cadastro Cliente", "Cadastrar", new { id = 0})
```

Texto do Link — Action — Parâmetros

http://msdn.microsoft.com/en-us/library/system.web.mvc.html.linkextensions.actionlink(v=vs.100).aspx

# FORMULÁRIO

- Criação de Formulário;
- Por padrão, o formulário gerado pelo **BeginForm** utilizam requisições do tipo POST:

```
@{Html.BeginForm("Cadastrar","Cliente");}
                  Action      Controller

 @{Html.EndForm();}
```

- Podemos utilizar a palavra using, assim não precisamos utilizar o **EndForm**:

```
 @using(@Html.BeginForm("Cadastrar","Cliente")){
                        Action      Controller

 }
```

# FORMULÁRIO

- Se não passarmos parâmetros para o **BeginForm**, o formulário será submetido para o mesmo **Controller** e mesmo nome de **Action** da página:

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

}
```

Action = Cadastrar

Controller = Cliente

# TEXT E TEXTAREA

- Caixa de Texto:

```
@Html.TextBox("Nome")
```

Resultado

```
<input id="Nome" name="Nome" type="text" value="">
```

- TextArea:

```
@Html.TextArea(“Mensagem")
```

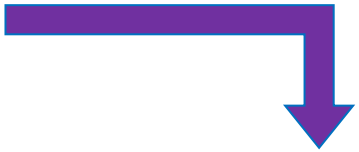
Resultado

```
<textarea cols="20" id="Mensagem" name="Mensagem" rows="2">
</textarea>
```

# CHECKBOX

```
@Html.CheckBox("Termo")
```

Resultado



```
<input id="Termo" name="Termo" type="checkbox" value="true">
<input name="Termo" type="hidden" value="false">
```

Obs. O HTML Helper gera o input hidden para enviar verdadeiro ou falso para o atributo “Termo”;

# RADIO E LABEL

- Radio Button:

```
@Html.RadioButton("sexo","Masculino")
@Html.RadioButton("sexo","Feminino")
```

Resultado

```
<input id="sexo" name="sexo" type="radio" value="Masculino">
<input id="sexo" name="sexo" type="radio" value="Feminino">
```

- Label:

```
@Html.Label(“Nome")
```

Resultado

```
<label for="Nome">Nome</label>
```

# PASSWORD E HIDDEN

- Password:

```
@Html.Password("Senha")
```

Resultado

```
<input id="Senha" name="Senha" type="password">
```

- Hidden:

```
@Html.Hidden("Id",1)
```

Resultado

```
<input id="Id" name="Id" value="1" type="hidden">
```

# STRONGLY TYPED VIEWS
# HTML HELPERS

# STRONGLY TYPED VIEWS

- Podemos utilizar o modelo de dados da página para criar os campos;
  - Vamos utilizar o seguinte modelo de exemplo:

```
public class Cliente
{
    public int Id { get; set; }
    public string Nome { get; set; }
    public DateTime DataNascimento { get; set; }
    public string Senha { get; set; }
    public bool IsAtivo { get; set; }
    public string Mensagem { get; set; }
    public string Sexo { get; set; }
}
```

Precisamos definir o model no inicio da página .cshtml

```
@model FIAPWebTemplate.Models.Cliente
```

# TEXTBOXFOR E TEXTAREAFOR

- Caixa de Texto:

```
@model FIAPWebTemplate.Models.Cliente

@Html.TextBoxFor(c => c.Nome)
```

Resultado

```
<input id="Nome" name="Nome" type="text" value="">
```

- TextArea:

```
@Html.TextAreaFor(c => c.Mensagem)
```

Resultado

```
<textarea cols="20" id="Mensagem" name="Mensagem" rows="2">
</textarea>
```

# CHECKBOXFOR E HIDDENFOR

- Checkbox:

```
@Html.CheckBoxFor(c => c.IsAtivo)
```

- Hidden:

```
@Html.HiddenFor(c => c.Id)
```

# LABELFOR E PASSWORDFOR

- LabelFor:

```
@Html.LabelFor(c => c.Nome)
```

- Password:

```
@Html.PasswordFor(c => c.Senha)
```

# DROPDOWNLISTFOR

- Cria uma lista de opções de acordo com o SelectList

Camada Controler

```
public ActionResult Index()
{
        ViewBag.Planos = new SelectList(lista, "PlanoId",
        "Descricao");
         return View();
}
```

Camada View

```
@Html.DropDownListFor(c => c.Plano.PlanoId,
ViewBag.Planos as SelectList)
```

Valores para a lista de seleção.

Atributo para receber o valor selecionado

# EDITORFOR

- Será escolhido o Helper mais indicado, dependendo do tipo da propriedade, ex: se a propriedade for boleano, será escolhido o checkbox, se nome o textbox.

```
@Html.EditorFor(c => c.Nome)
@Html.EditorFor(c=>c.DtNasc)
```

# EDITORFORMODEL

- Podemos construir um formulário completo com base nas propriedades do model.

```
@using (@Html.BeginForm())
{
    @Html.EditorForModel()
    <input type="submit" value="Enviar" />
}
```

# UIHINT

- Podemos utilizar alguns templates para exibir os campos.

  Basta anotar com **UIHint** a propriedade do model:

Model

```
[UIHint("Date")]
public DateTime DtNasc { get; set; }
```

# UIHINT

- Alguns Templates:

| Valor | Descrição |
|---|---|
| **Date** | Renderiza um componente para a escolha de data |
| **Time** | Renderiza um componente para a escolha de hora |
| **MultilineText** | Renderiza uma texarea |
| **String** | Renderiza um textbox |
| **Password** | Renderiza um Helper password |
| **Boolean** | Renderiza um checkbox ou DropDownList |
| **Decimal** | Renderiza um textbox com duas casas decimais |
| **Number** | Renderiza um texbox para números |

# DISPLAY

- Quando utilizamos **LabelFor**, o nome da propriedade será exibida na página.

```
@Html.LabelFor(c => c.Nome)
```

- Se quisermos alterar esse valor, podemos anotar a propriedade do Model com **Display:**

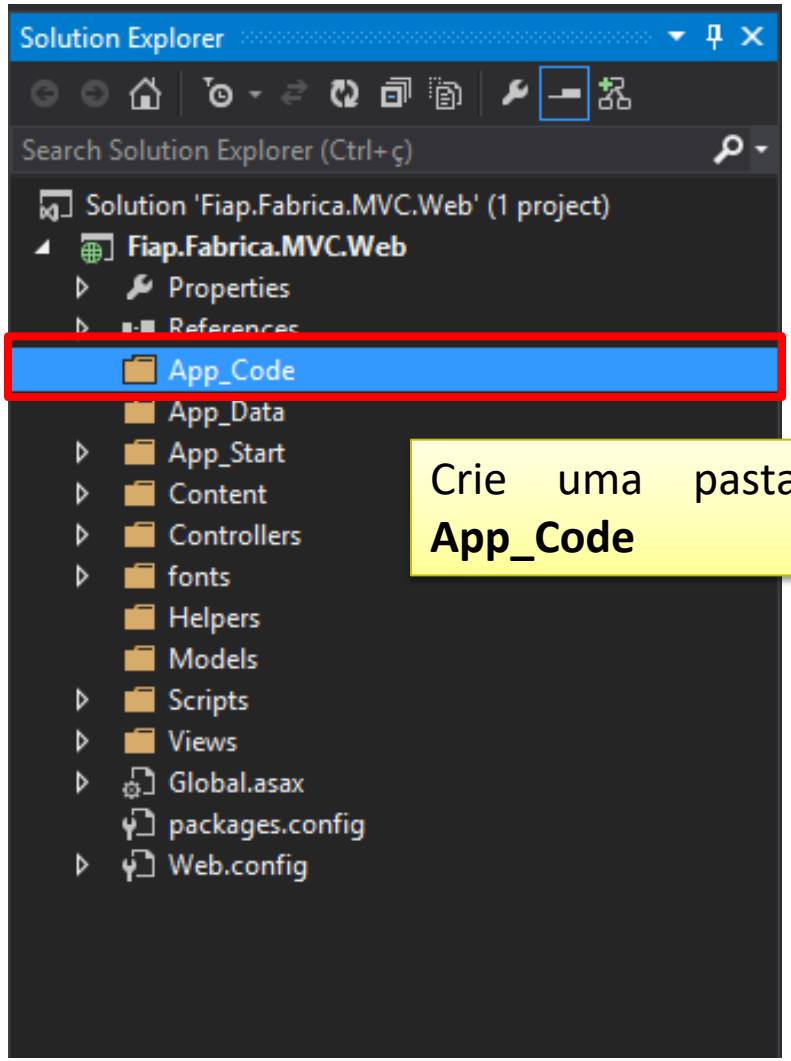
Model

```
[Display(Name = "Data Aniversário")]
public DateTime DtNasc { get; set; }
```

# CRIANDO HTML HELPERS

# CRIANDO HTML HELPERS

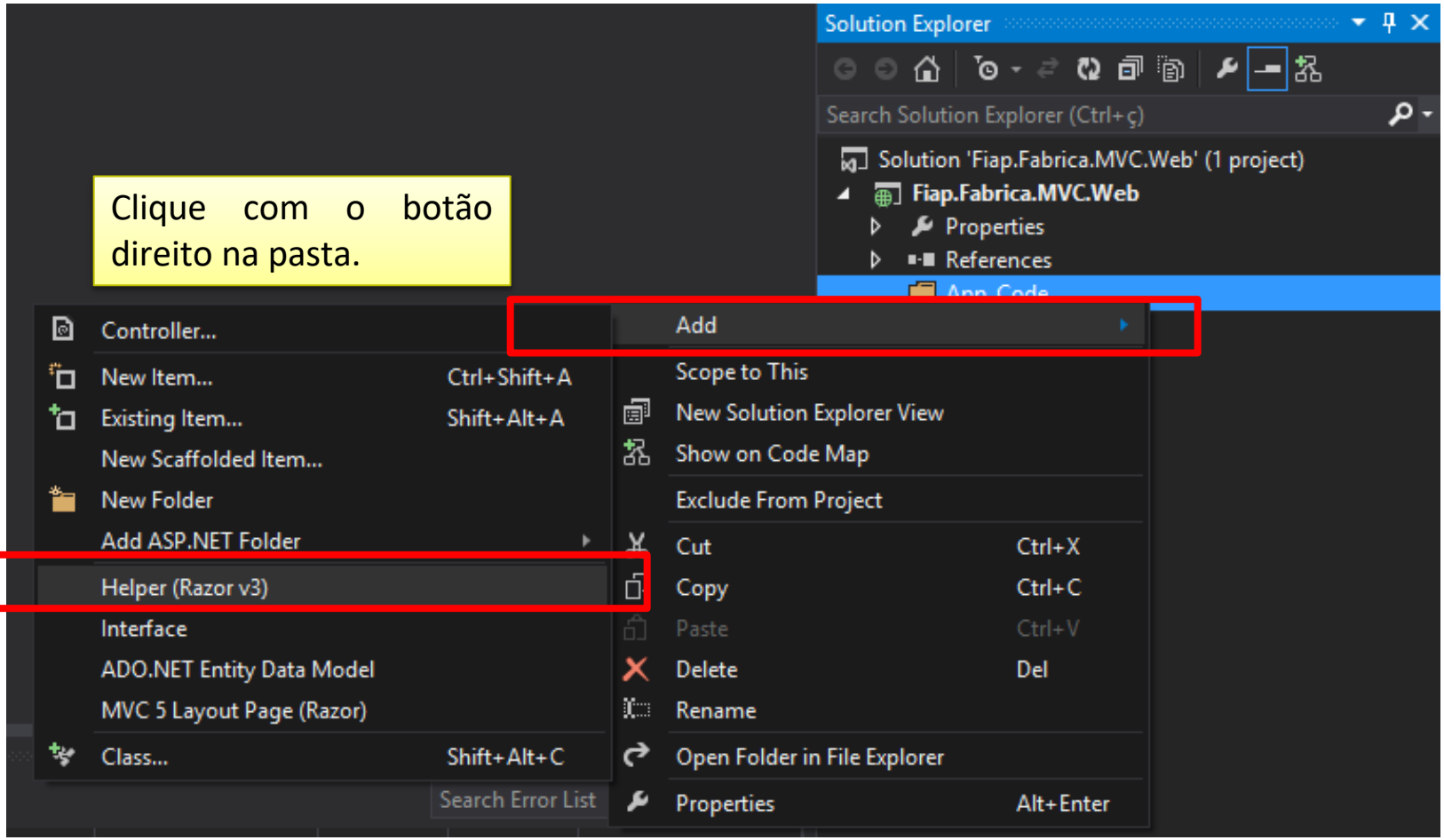
- Podemos criar os nossos próprios Helpers.




- **App_Code** é uma pasta reservada no ASP.NET para armazenar componentes como HTML Helpers.

# CRIANDO HTML HELPERS

- Adicione um arquivo de Helper (.cshtml).



# CRIANDO HTML HELPERS

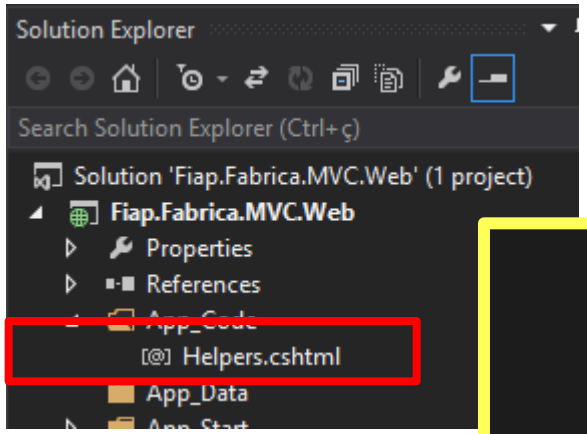
- Crie os Helpers:

```
@helper Submit(string name, String classe)
{
    <input type="submit" name="@name" class="@classe" />
}

@helper Message(string msg)
{
    <div class="alert alert-info">
        @msg
    </div>
}
```

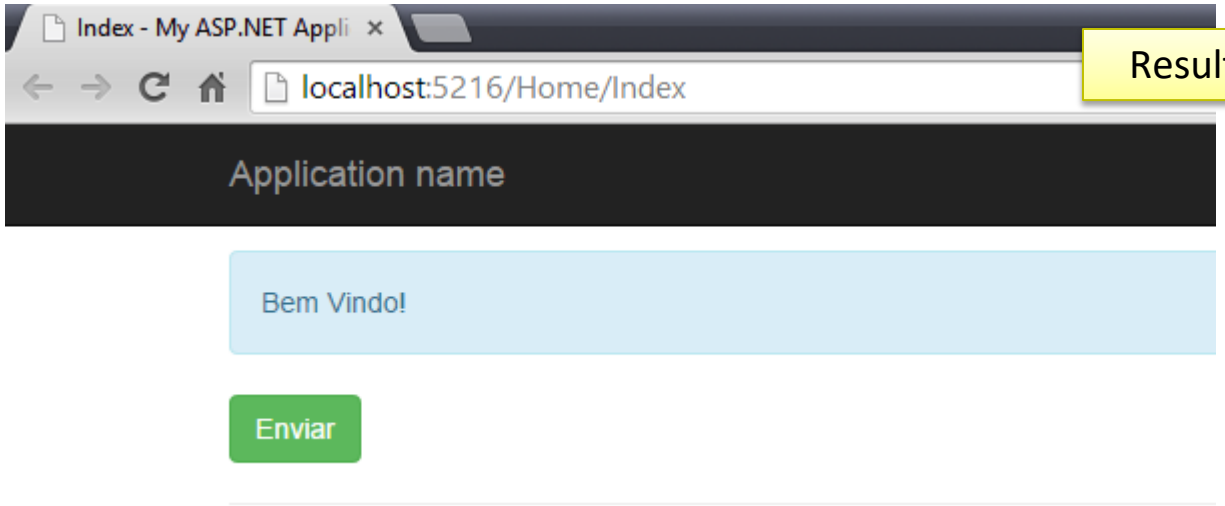
# CRIANDO HTML HELPERS

- Utilize o Helper: @NomeArquivo.Helper




Página que utiliza o Helper customizado

```
@Helpers.Message("Bem Vindo!")
@Helpers.Submit("Salvar","btn btn-success")
```




Resultado

FIAP



*"O sucesso normalmente vem para quem está ocupado demais para pensar nele"*