

## PROGRAMAÇÃO VISUAL TESP/TPSI

### Introdução à Programação para a Web com .NET MVC ViewBag / ViewData / Layouts

Prof. José Brás

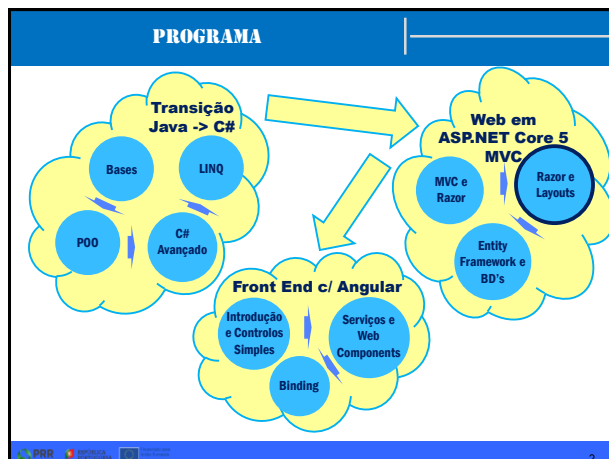
F262 – jose.bras.pereira@estsetubal.iips.pt

DEI / EST Setúbal / IPS

2025/26



1



2

## 5.2 WEB SITE C/ LAYOUTS

### 5. .NET CORE WEB

- 5.2.0 – Criação do projeto MVC c/ Layouts
- 5.2.1 – Controladores (Controller), ações e resultados
  - 5.2.1.1 – Ações dos controladores
  - 5.2.1.2 – Resultados das ações dos controladores
- 5.2.2 – Passagem de Dados p/ Vistas (View)
  - 5.2.2.1 Model
  - 5.2.2.2 ViewBag
  - 5.2.2.3 ViewData
- 5.2.3 Criação e Utilização de novos Layouts
  - 5.2.3.1 Utilização de Layouts
  - 5.2.3.2 HTML Helpers
  - 5.2.3.3 Secção footer do \_Layout preenchida na View
  - 5.2.3.4 Definição do Layout na \_ViewStart
- 5.2.4 Sintaxe Razor (Vistas)



3

3

## 5.2.0 CRIAÇÃO DO PROJECTO

### 5. .NET CORE WEB 5.2 WEB SITE C/ LAYOUTS

Create a new project

1 - C# / All Platforms / Web

3 - Nome e PASTA! -> NEXT

2 - ASP.NET Core WebApp (Model-View-Controller) -> Next

4 - Framework: .NET 8.0 ou 9.0  
Authentication Type: none  
Configure for HTTPS: No  
Do not use top-level statements: yes  
-> Create

4

4

## 5.2.1 - CONTROLADORES

### 5.2 - WEB SITE C/ LAYOUTS

WebSiteComLayouts

- Connected Services
- Dependencies
- Properties
- wwwroot
- Controllers
  - FotoController.cs
  - HomeController.cs
  - MusicaDoDiaController.cs
- Models
- ErrorViewModel.cs
- Foto.cs
- Views
  - Foto
    - Index.cshtml
  - Home
    - Contactos.cshtml
    - ContSemScaff.cshtml
    - Index.cshtml
    - Privacy.cshtml

Estão sempre na pasta com o nome **Controllers**

Por convenção têm o sufixo **Controller** como em **FotoController**

- O Controller é o elemento central do padrão MVC
- Os Routers instanciam um controlador (um objeto da classe) por cada pedido
  - Cada pedido é mapeado para uma ação específica
- Todos os controladores herdam da classe **Controller**
  - A classe Controller fornece o acesso ao Request (a informação do pedido)



5

5

## 5.2.1.1 - AÇÕES

### 5.2 - WEB SITE C/ LAYOUTS

- As ações são métodos dos controladores e representam o destino final dos pedidos
  - São métodos públicos
  - Não são de classe (não são static)
  - Por omissão são HTTPGet (pelo que podemos omitir o "Decoration Attribute" (HttpGet))
  - Não impõem restrições ao tipo de retorno mas ...

```
public class FotoController : Controller
{
    public IActionResult Index()
    {
        Foto f = new Foto() {
            Codigo = "ABC000001",
            Titulo = "Deserto"
        };
        return View(f);
    }
}
```

retornam normalmente um **IActionResult (\*)**

(\*) Uma variável do tipo interface **IActionResult**

6

6

## 5.2.1.2 – RESULTADOS DAS AÇÕES

## ■ Os resultados das ações

- Respondem a um pedido do browser
- Normalmente retornam uma variável do tipo Interface IActionResult
  - Ou Seja de um tipo que implemente os métodos requeridos pela interface IActionResult

## ■ Existem vários tipo de retorno:

Nome	Comportamento	Produto
<b>ContentResult</b>	Por omissão retorna uma string mas pode retornar outros tipos	Conteúdo
<b>EmptyResult</b>	Sem resposta. Usado para comandos como create, update e delete	
<b>FileContentResult</b> <b>PhysicalFileResult</b> <b>FileStreamResult</b> <b>VirtualFileResult</b>	Retorna o conteúdo de um ficheiro. Subclasse de FileResult	File
<b>ChallengeResult</b>	Authentication credential not valid or not present. Returns an HTTP 401 status code	

7

## 5.2.1.2 – RESULTADOS DAS AÇÕES

Nome	Comportamento	Produto
<b>UnauthorizedResult</b>	Returns an HTTP 401 status code and nothing else	
<b>RedirectResult</b> <b>RedirectToActionResult</b> <b>RedirectToRouteResult</b> <b>LocalRedirectResult</b>	Redirects the client to a new URL. Derived from RedirectToActionResult	Redirect / RedirectPermanent
<b>ForbiddenResult</b>	To refuse a request to a particular resource. Returns an HTTP 403 status	
<b>JsonResult</b>	Returns data in JSON format	Jason
<b>ViewResult</b> <b>PartialViewResult</b>	Response is the responsibility of a view engine-To render a view or part of it.	View / PartialView
<b>ViewComponentResult</b>	Returns html content for a view	
<b>SignInResult</b> <b>SignOutResult</b>	Sign in or sign out the user based on provided mechanism	

... Existem mais tipos de retorno

8

## 5.2.2 - PASSAGEM DE DADOS P/ VISTAS

5. .NET CORE WEB

## □ É possível passar informação para as vistas através de:

## ■ ViewBag

- Action: ViewBag.Message = "Text of message";
- View: @ViewBag.Message

Veremos em seguida

## ■ ViewData

- Action: ViewData["messageIdentifier"] = "message text";
- View: @ViewData["messageIdentifier"]

Vimos na A13 no ficheiro \_Layouts, iremos aprofundar

## ■ Model (strongly-typed)

- Action: return view(model);
- View: @model ModelDataType

Vimos na A13 e aqui no slide 6, iremos aprofundar

9

## 5.2.2.1 – PASSAGEM DE DADOS C/MODEL

5. .NET CORE WEB

5.1 – WEB SITE SIMPLES

```
namespace TP11.Models
{
    public class Foto
    {
        public String Codigo { get; set; }
        public String Titulo { get; set; }
    }
}
```

1. BDR sobre a pasta Models -> Add -> New Item -> Class -> Foto

2. Adicionar 2 propriedades Codigo, Titulo

```
public IActionResult PorTitulo(string titulo)
{
    Foto f = new Foto
    {
        Codigo = "ABC000001",
        Titulo = titulo
    };
    return View(f);
}
```

3. Adicionar ao HomeController uma ação PorTitulo que aceite um titulo e retorne uma view com um objeto da classe foto

10

## 5.2.2.1 – PASSAGEM DE DADOS C/MODEL

5. .NET CORE WEB

5.2 – WEB SITE C/ LAYOUTS

1. BDR sobre a pasta Home -> Add -> View

2. No diálogo Add New Scaffolded Item -> RazorView

3. No diálogo Add Razor View: View name: PorTitulo Template: Details Model Class: Foto (TP11.Models.Foto) layout page: \_Layout -> Add

4. Run & Try

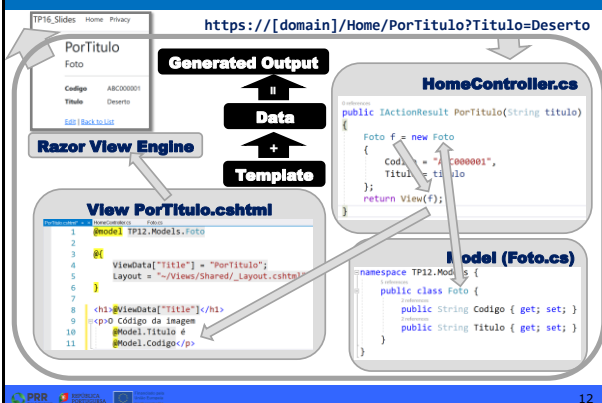
Copy/paste url to browser navigation box -> https://[domain]/ Home/ PorTitulo?Titulo=Deserto

11

## 5.2.2.1 – PASSAGEM DE DADOS C/MODEL

5. .NET CORE WEB

5.1 SIMPLE WEB SITE



12

### 5.2.2.2-PASSAGEM DE DADOS C/ VIEWBAG

#### 5. .NET CORE WEB

##### 5.2 WEB SITE C/ LAYOUTS

```
public IActionResult Index()
{
    ViewBag.Welcome = "Bem Vindo";
    Foto f = new Foto() { Titulo = "Deserto" };
    ViewBag.MyFoto = f;
    return View();
}
```

**1. Reescreva a action Index no ficheiro HomeController.cs**

**2. Reescreva o código do ficheiro Index.cshtml da view Home**

**3. Run & Try**

**ViewBag** – é uma propriedade de tipo dinâmico da classe Controller

Serve para passar dados do controlador para a view

ViewBag.MyFoto

```
@{
    ViewData["Title"] = "Home Page";
}
```

```
<div class="text-center">
  <h1 class="display-4">@ViewBag.Welcome</h1>
  <h1 class="display-4">@ViewBag.MyFoto.Titulo</h1>
</div>
```

TP16\_Slides

Bem Vindo  
Deserto

© 2022 - TP16\_Slides - Privacy

13

### 5.2.2.3-PASSAGEM DE DADOS C/ VIEWDATA

#### 5. .NET CORE WEB

##### 5.2 WEB SITE C/ LAYOUTS

```
public IActionResult Index()
{
    List<String> lista = new List<String>() {
        "aqui vai encontrar", "as melhores fotos" };
    ViewData["ObjetoLista"] = lista;
    ViewData["Text01"] = " Bem Vindo ao FotoShopping";
    return View();
}
```

**1. Reescreva a action Index no ficheiro HomeController.cs**

**2. Reescreva o código do ficheiro Index.cshtml da view Home**

**3. Run & Try**

**ViewData** – é um Dicionário de pares chave-valor em que cada chave é uma String

Tal como a ViewBag serve para passar dados do controlador para a view

ViewData["Text01"]  
ViewData["ObjetoLista"]

```
@{
    ViewData["Title"] = "Home Page";
    List<String> lista = ViewData["ObjetoLista"] as List<String>;
}
```

```
<div class="text-center">
  <h1 class="display-4">@ViewData["Text01"]</h1>
  <p>@lista[0]</p>
  <p>@lista[1]</p>
</div>
```

TP12\_StandAuto

Bem Vindo  
ao  
FotoShopping

aqui vai encontrar  
as melhores fotos

© 2019 - LARUI StandAuto - Privacy

14

### 5.2.3 CRIAÇÃO E USO DE LAYOUTS

#### 5. .NET CORE WEB

##### 5.2 WEB SITE C/ LAYOUTS

**1. BDR na pasta Shared -> New Item**

**2. Selecionar C# -> Razor Layout**

**3. Nome: \_MyLayout (com \_ no início) -> Add**

```
<!DOCTYPE html>
<html>
<head>
<met>
<tit>
  @ViewData["Title"] - TP1
</tit>
</head>
<body>
<table style="width:100%;">
<tr>
<td>Links</td>
<td style="text-align:">
</td>
</tr>
<tr>
<td>
<p>@Html.ActionLink("Home", "Index")</p>
<p>@Html.ActionLink("Privacy", "Privacy")</p>
</td>
<td>@RenderBody()</td>
</tr>
</table>
</body>
</html>
```

As views são injetadas aqui por baixo

**4. Substituir o html original pelo deste slide**

15

### 5.2.3.1 UTILIZAÇÃO DE LAYOUTS

#### 5. .NET CORE WEB

##### 5.2 WEB SITE C/ LAYOUTS

Cada vista pode especificar a sua página de layout

```
@{
    Layout = "~/Views/Shared/_MyLayout.cshtml";
}
```

```
@{
    Layout = null;
}
```

16

### 5.2.3.1 UTILIZAÇÃO DE LAYOUTS

#### 5. .NET CORE WEB

##### 5.2 WEB SITE C/ LAYOUTS

**1. Abrir o ficheiro Index da view Home**

**2. Inserir o path para o \_MyLayout.cshtml logo abaixo da ViewData["Title"]**

```
@{
    ViewData["Title"] = "Home Page";
    Layout = "~/Views/Shared/_MyLayout.cshtml";
}
```

```
<div class="text-center">
  <h1>@ViewBag.Welcome</h1>
  <h3>@ViewData["Text01"]</h3>
  <h4>@ViewData["Text02"]</h4>
</div>
```

As views são injetadas aqui por baixo

**3. Repetir para o ficheiro Privacy.cshtml**

**4. Run & Try**

Bem Vindo  
Ao Fotoshopping  
aqui vai encontrar as melhores fotos

17

### 5.2.3.2 HTML HELPERS

#### 5. .NET CORE WEB

##### 5.1 - WEB SITE SIMPLES

A classe **HtmlHelper** gera elementos de html usando objectos da classe model.

Liga (bind) o modelo a elementos de html para apresentar os valores das propriedades do modelo no elemento html

A vantagem dos html Helpers sobre as tags html é que os métodos dos html helper são desenhados para facilitar o binding entre a view e o modelo de dados

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width" />
<title>@ViewBag.Title</title>
</head>
<body>
<table style="width:100%;">
<tr>
<td>Links</td>
<td>As views entram aqui em baixo</td>
</tr>
<tr>
<td>
<p>@Html.ActionLink("Home", "Index")</p>
<p>@Html.ActionLink("Privacy", "Privacy")</p>
</td>
<td>@RenderBody()</td>
</tr>
</table>
</body>
</html>
```

**Html helper ActionLink**

18

5.2.3.2 HTML HELPERS		5. .NET CORE WEB
		5.1 - WEB SITE SIMPLES
HtmlHelper	Strongly Typed HtmlHelpers	Html Control
Html.ActionLink		Anchor link
Html.TextBox	Html.TextBoxFor	Textbox
Html.TextArea	Html.TextAreaFor	TextArea
Html.CheckBox	Html.CheckBoxFor	Checkbox
Html.RadioButton	Html.RadioButtonFor	Radio button
Html.DropDownList	Html.DropDownListFor	Dropdown, combobox
Html.ListBox	Html.ListBoxFor	multi-select list box
Html.Hidden	Html.HiddenFor	Hidden field
Password	Html.PasswordFor	Password textbox
Html.Display	Html.DisplayFor	Html text
Html.Label	Html.LabelFor	Label
Html.Editor	Html.EditorFor	Generates Html controls based on data type of specified model property e.g. textbox for string property, numeric field for int, double or other numeric type.

19

5.2.3.3 SECÇÃO FOOTER

5. .NET CORE WEB

5.2 WEB SITE C/ LAYOUTS

```

<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>@ViewData["Title"] - TP12</title>
<link rel="stylesheet" href="~/css/site.css" />
</head>
<body>
<table style="width:100%;>
<tr>
<td>Links</td>
<td style="text-align:center">As views são injetadas aqui por baixo</td>
</tr>
<tr>
<td>
<p>@Html.ActionLink<br>@Html.ActionLink</p>
<td>@RenderBody()</td>
</tr>
</table>
<footer>
@RenderSection("Footer", required: false)
</footer>
</body>
</html>

```

1. Inserir na **\_MyLayout** uma **secção footer** com **@RenderSection** denominada **Footer**

2. Inserir uma **secção denominada Footer** na **Razor View Index.cshtml**

```

@section Footer
{
<span>Texto para o Footer<br>do _MyLayout</span>
}

```

20

5.2.3.1 UTILIZAÇÃO DE LAYOUTS

5. .NET CORE WEB

5.2 WEB SITE C/ LAYOUTS

As vistas não necessitam obrigatoriamente de especificar o layout usado se recorrerem ao layout por omissão especificado no ficheiro **\_ViewStart.cshtml**

\* Basta inserir o layout por omissão no ficheiro **~/Views/\_ViewStart.cshtml\***

```

@{
Layout = "~/Views/Shared/_MyLayout.cshtml";
}

```

21

5.2.3.4 UTILIZAÇÃO DO \_VIEWSTART

5. .NET CORE WEB

5.2 WEB SITE C/ LAYOUTS

```

Layout = "~/Views/Shared/_MyLayout.cshtml";
}

```

2. Inserir o **path** para o **\_MyLayout.cshtml** no bloco de **c#**

```

@{
ViewData["Title"] = "Home Page";
//Layout = "~/Views/Shared/_MyLayout.cshtml";
}

```

3. **Remover os path** para o ficheiro **\_MyLayout.cshtml** dos cabeçalhos de **c#** nos ficheiros **Index.cshtml** e **Privacy.cshtml**

1. Abrir o ficheiro **\_ViewStart** da pasta **Views**

4. **Run & Try**

As views são injetadas aqui por baixo

Bem Vindo  
Ao Fotoshopping  
aqui vai encontrar as melhores fotos  
Texto para o Footer do \_MyLayout

22

5.2.4 SINTAXE RAZOR

5. .NET CORE WEB

5.2 WEB SITE C/ LAYOUTS

- Razor view engine**
  - Fornece e executa o código HTML das vistas
  - Disponibiliza vários html tag helpers para a geração do HTML;
- A linguagem Razor** é uma sintaxe para templates de etiquetas (Template markup syntax)
  - Não é uma nova linguagem;
  - Editável com qualquer editor de texto
  - Tem um bom intellisense (built in no VisualStudio)
  - Facilmente testável por unidades (unit-test)

**NEW RAZOR**

Na sintaxe Razor os blocos de código começam com o caracter **@** e não necessitam que o bloco tenha um fecho explícito.

```

@DateTime.Now

```

23

5.2.5 LINGUAGEM RAZOR

5. .NET CORE WEB

5.2 WEB SITE C/ LAYOUTS

1. Abra a view **PorTitulo.cs.html**.

2. Declare e inicie a variável **someVar** com o caracter **@**

```

@model A15_NomeApelido.Models.Foto
@{
ViewData["Title"] = "PorTitulo";
Layout = "~/Views/Shared/_Layout.cshtml";
char someVar = '@';
}

```

3. Insira o código no final.

4. **Run & Try**

```

@* Para valores (HTML Encoded) *@
<p>
use @@ para valores (HTML encoded)<br />
Current time is: @DateTime.Now <br />
Not HTML encoded value: @Html.Raw(someVar)
</p>

```

24

### 5.2.5 LINGUAGEM RAZOR

#### 5. .NET CORE WEB

##### 5.2 WEB SITE C/ LAYOUTS

3. Logo após o código do slide anterior, insira o código deste bloco

4. Run & Try

```
@* @ {...} Para blocos de Código *@
@{
    var fotoName = "North Sea";
    if (Model != null) {
        fotoName = Model.Titulo;
    }
    else if (ViewBag.FotoName != null) {
        fotoName = ViewBag.FotoName;
    }
    <p> A fotografia "@fotoName" foi adicionada à sua lista de compras</p>
}
```

PRR

25

25

### 5.2.5 LINGUAGEM RAZOR

#### 5. .NET CORE WEB

##### 5.2 WEB SITE C/ LAYOUTS

```
public IActionResult ListarFotos()
{
    List<Foto> listaDeFotos = new List<Foto>();
    Foto f1 = new Foto{ Titulo = "North Sea", Codigo = "NS111"};
    Foto f2 = new Foto{ Titulo = "Turtle", Codigo = "TU222"};
    Foto f3 = new Foto { Titulo = "Sun Shining", Codigo = "SUSH333"};
    listaDeFotos.Add(f1);
    listaDeFotos.Add(f2);
    listaDeFotos.Add(f3);
    return View(listaDeFotos);
}
```

1. No controlador Home crie uma ação ListarFotos com este código

2. Adicione o ActionLink ao MyLayout.cshtml

3. Crie uma Razor View com View Name: ListarFotos Template: Details Model Class: Foto layout page: MyLayout

<p>@Html.ActionLink("Fotos", "ListarFotos")</p>

PRR

26

26

### 5.2.5 LINGUAGEM RAZOR

#### 5. .NET CORE WEB

##### 5.2 WEB SITE C/ LAYOUTS

```
@model IEnumerable<TP12.Models.Foto>
@{
    ViewData["Title"] = "Listar Fotos";
    Layout = "~/Views/Shared/_MyLayout.cshtml";
}
<div class="lista-de-fotos">
    @{
        if (Model.Count() == 0)
        {
            <p> Desculpe, não foram encontradas fotografias</p>
        }
        else
        {
            @: Lista de fotografias disponíveis:
            foreach (var foto in Model)
            {
                <br /><b>@foto.Codigo -- @foto.Titulo</b>
            }
        }
    }
</div>
```

1. Substitua o código da View ListarFotos pelo código ao lado

As linhas de markup HTML podem ser inseridas em qualquer parte

Usamos @: para renderização de texto simples

PRR

27

27

### 5.2.5 LINGUAGEM RAZOR

#### 5. .NET CORE WEB

##### 5.2 WEB SITE C/ LAYOUTS

```
@* Comentários em Razor *@
@{
    // Comentários em c#
    /* Comentários multilinha
       que acordamos não usar em C# */
}
@* E como é com os "@" dos e-mails? *@
<p>
    O sinal que separa os nomes do e-mail dos domínios é: @@<br/>
    E isto mostra como o razor pode ser espertalhão: <br/>
    spam_them@gmail.com
</p>
@* Expressão de código explícita *@
<p>
    Preços actuais (0-10): @Model.Preco / 10.0
    Preços actuais (0-1): @(Model.Preco / 10.0M)
    spam_them@Model.Preco
    spam_them@(Model.Preco)
</p>
```

1. Adicione o código ao final do ficheiro PorTitulo.cshtml

2. Run & Try

TP12

PorTitulo

O Código da Imagem Deserto é ABC000001

use @ para valores (HTML, encoded)

Current time is: 05/12/2021 00:44:54

Not HTML, encoded value: >

A fotografia "Deserto" foi adicionada à sua lista de compras

O sinal que separa os nomes do e-mail dos domínios é: @

E isto mostra como o razor pode ser espertalhão:

spam\_them@gmail.com

Preços actuais (0-10): 0 / 10.0

Preços actuais (0-1): 0

spam\_them@Model.Preco

spam\_them@

PRR

28

28