Fundamentos de Desenvolvimento Web/BD

Aula 03: Page Model

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 03: Roteiro

Instituto Infnet

- Page Model

Aula 03: PageModel

- PageModel é uma classe gerada automaticamente quando uma Razor Page é adicionada.
- A palavra Model aparece como sufixo no nome da Page, por exemplo, IndexModel.
- As classes são derivadas do tipo PageModel, que possui diversos métodos e propriedades para trabalhar com requisições HTTP.
- Fornece uma separação entre a camada visual e a camada de processamento da página.
  - Reduz a complexidade da camada de UI, deixando-a mais simples de manter.
  - Facilita a adoção de testes unitários.
  - Permite que equipes possam trabalhar UI e processamento separadamente.
- A classe PageModel é acessada pela camada de UI através da diretiva @model.

```
@page
@model IndexModel
@{
}
```

Aula 03: Processamento em um PageModel

Instituto Infnet

- Código é processado em um PageModel através de Handler Methods, que são análogos às Actions de um Controller MVC.
- Esses métodos são executados automaticamente pelos Requests e retornam um objeto PageResult.
- Os métodos são chamados de acordo com a operação do Request. O Razor verifica o nome da ação com o respectivo nome do método, assim, uma operação de GET executará o método OnGet e assim por diante.

```
public class IntroModel : PageModel
{

    public string Mensagem { get; set; } = string.Empty;

    public void OnGet()
    {
        Mensagem = "Get acionado!";
    }

    public void OnPost()
    {
        Mensagem = "Post acionado!";
    }
}
```

```
@page
@model SampleWeb.Pages.Methods.IndexModel
@{
}

<h3>@Model.Mensagem</h3>
<form method="post">
    <button class="btn btn-default">
        Disparando um post
    </button>
</form>
<p>
    <a href="" class="btn btn-default">Disparando um Get</a>
</p>
```

Aula 03: Processamento com Métodos Nomeados

Instituto Infnet

- Razor possui um recurso chamado Named Handler Methods, que permite a criação de métodos diferentes para um mesmo verbo HTTP.
- Atende à Pages mais complexos, onde há a necessidades de múltiplas operações.

```
public void OnGetUpperMessage()
{
    Mensagem = "Get acionado em maiúsculas!".ToUpper();
}

public void OnPostDelete()
{
    Mensagem = "Delete acionado!";
}

public void OnPostEdit(int id)
{
    Mensagem = "Edit acionado!";
}

public void OnPostView(int id)
{
    Mensagem = "View acionado!";
}
```

```
<div class="row">
    <div class="col-lg-1">
        <form asp-page-handler="edit" method="post">
            <button class="btn btn-default">Editar</button>
        </form>
    </div>
    <div class="col-lg-1">
        <form asp-page-handler="delete" method="post">
            <button class="btn btn-default">Deletar</button>
        </form>
    </div>
    <div class="col-lg-1">
        <form asp-page-handler="view" method="post">
            <button class="btn btn-default">Exibir</button>
        </form>
    </div>
    <div class="col-lg-2">
        <a asp-page-handler="upperMessage" method="get">Em Maiúsculas</a>
    </div>
</div>
```

Aula 03: Parâmetros em Method Handlers

- Razor também permite que Handler Methods possuam parâmetros.

```
public void OnGetCase(bool allUpper)
{
    if (allUpper)
    {
        Mensagem = $"Parâmetro allUpper como {allUpper}".ToUpper();
    }
    else
    {
        Mensagem = $"Parâmetro allUpper como {allUpper}";
    }
}
```

```
<div class="col-lg-2">
    <a asp-page-handler="case" asp-route-allUpper="false" method="get">Case com Parâmetro</a>
</div>
```

Aula 03: Processamento com Objetos Complexos

Instituto Infnet

- O atributo [BindProperty] indica que o Model deve vincular uma propriedade com o parâmetro de mesmo nome que vier de um Request.
- Dessa forma, uma propriedade de Page, que receba um objeto complexo, pode ser associada a um parâmetro de mesmo nome

```
public class BindingSampleModel : PageModel
{
    [BindProperty]

    public Login? Login { get; set; }

    public void OnGet()
    {
    }

    public void OnPost()
    {
        Login!.Nome = Login.Nome.ToUpper();
    }
}
```

```
public class Login
{

    public string Nome { get; set; } = string.Empty;

    public string Email { get; set; } = string.Empty;

    public string Password { get; set; } = string.Empty;
}
```

```
<form class="form-horizontal" method="post">
    <div class="form-group">
        <label for="Nome" class="col-sm-2 control-label">Nome</label>
        <div class="col-sm-10">
            <input type="text" class="form-control" name="Login.Nome">
        </div>
    </div>
    <div class="form-group">
        <label for="Email" class="col-sm-2 control-label">Email</label>
        <div class="col-sm-10">
            <input type="text" class="form-control" name="Login.Email">
        </div>
    </div>
    <div class="form-group">
        <label for="Senha" class="col-sm-2 control-label">Senha</label>
        <div class="col-sm-10">
            <input type="text" class="form-control" name="Login.Password">
        </div>
    </div>
    <div class="form-group">
        <div class="col-sm-offset-2 col-sm-10">
            <button type="submit" class="btn btn-default">Criar</button>
        </div>
    </div>
</form>
```