Goiânia 29 de novembro de 2020

**Introdução de teoria do MVC.**

O que framework: E o conjunto de classe que facilita a execução.

O que MVC: E uma arquitetura de organização de classes as principais são Model, View, Controller.

Model: Responsabilidade de conversa com as classes de persistência e manipulação de banco de dados.

View: Responsabilidade pela Visão do sistema recebe os dados que serão apresentados nas telas.

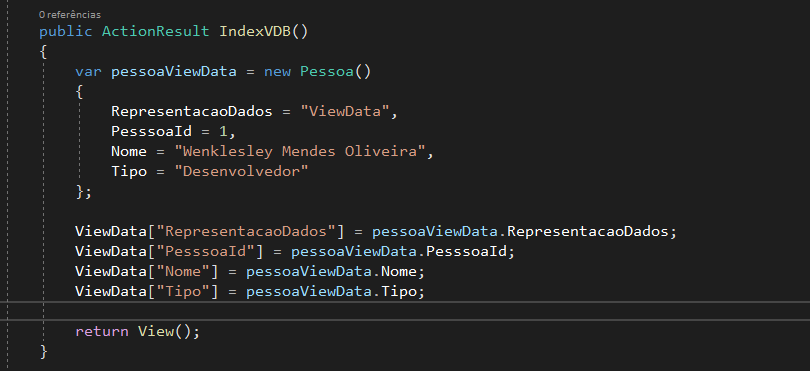
Controller: Responsabilidade de intermediar a comunicação com as Views e Models.

**Detalhando um projeto MVC**

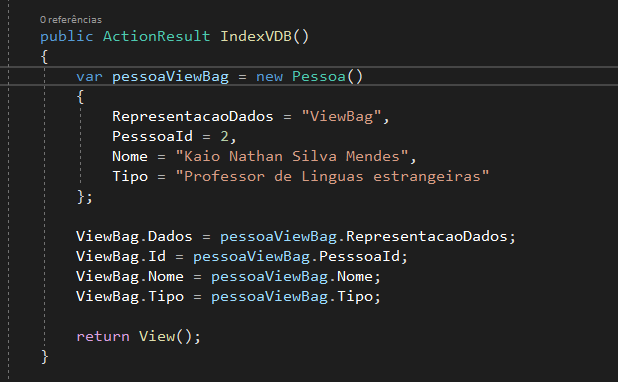
Falar um pouco sobre arquitetura do template criada no MVC

1. App\_Data: E pasta criada pelo template
   1. Sua responsabilidade e em contém dados do banco.
2. App\_Start: E pasta criada pelo template
   1. Sua responsabilidade e unir 3 tipos de arquivos de Configurações que já vem como padrão na criação do template que são **BundleConfig**, **BundleConfig.cs** e **App\_RouteConfig.cs** vamos detalha melhor responsabilidade de cada um.
      1. **BundleConfig.cs:** A responsabilidade deste arquivo esta em organiza e gera único arquivo de scripts ou folha de estilo para evitar de ter varias requisições do browser no sistema mais detalhes na documentação do link abaixo.
         1. <http://go.microsoft.com/fwlink/?LinkId=254725>
      2. **App\_FilterConfig.cs:** A responsabilidade deste arquivo e centralizar as mensagens de erros e alertas não sendo obrigatório mas para melhorar organização e simplesmente e arquitetura do MVC.
      3. **App\_RouteConfig.cs:** A responsabilidade deste arquivo está em carregar as rotas (Endereço Web).
3. Content: E pasta criada pelo template
   1. A responsabilidade desta pastar está em conter arquivos CSS (folha de estilo) e de imagens do sistema.
4. **Controllers:** E pasta criada pelo template que simboliza o **C** do nome da arquitetura **MVC** sendo umas das pastas principais que defini há arquitetura.
   1. Sua responsabilidade está em conter arquivos dos controladores do sistema como padrão será criado pelo template um **HomeController.cs.**
   2. Quais forma de instanciar uma View dentro dos controladores vamos citar 3 tipos mais importantes aqui que são **ViewData**, **ViewBag**, **View**.
      * 1. **ViewData:** Nesta forma e passada os dados da **Models** para **Views** em Arreios.

**Figura de um exemplo do código.**

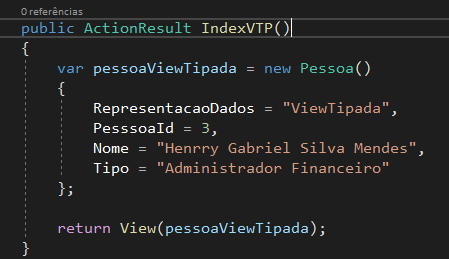


* + 1. **ViewBag:** Nesta forma e passada os dados da Models para Views em forma de paramentos

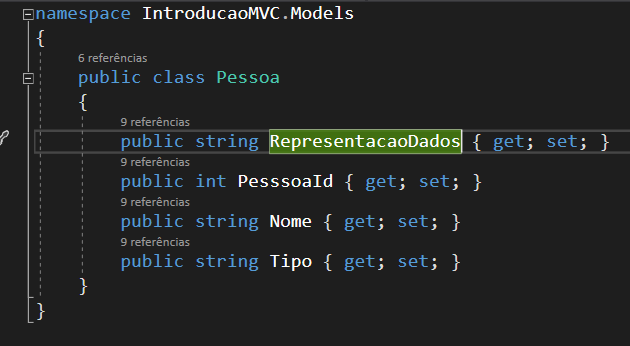
**Figura de um exemplo do código.**

* + 1. **View Tipada:** muito utilizado na para os get set representado no exemplo abaixo em exemplo de instancia.

**Figura de um exemplo do código.**

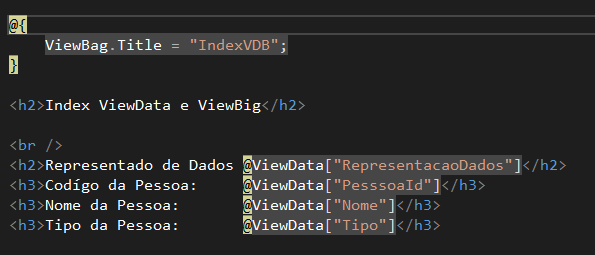


1. Fonts:
   1. Aqui e armazenados somente as fontes especiais coisa assim.
2. **Models:** E pasta criada pelo template que simboliza o **M** do nome da arquitetura **MVC** sendo umas das pastas principais que defini há arquitetura.
   1. Sua responsabilidade está em conter arquivos de modelos do banco de dados simplificando ainda mais onde vai ficar os get set da arquitetura.
   2. Os códigos são representados pela sintaxe abaixo no exemplo de classe pessoa.

**Figura de um exemplo do código.** 

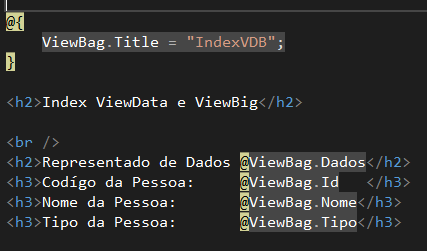
1. Script:
   1. Aqui fica os Java Scripts e outros scripts.
2. **Views:** E pasta criada pelo template que simboliza o **V** do nome da arquitetura **MVC** sendo umas das pastas principais que defini há arquitetura.
   1. Sua responsabilidade está em conter arquivos de da visão dos sistemas simplificando são arquivos com extensão cshtml que aceita código ser codificado os seguintes código C#, html, CSS e Script.
   2. Existe 3 formas de receber dados dos controladores que são **ViewData**, **ViewBag**, **ViewTipada** veremos um exemplo de cada na forma instanciada no Views.
      1. **ViewData:** Esta forma receber dados sendo passado pelo arreio.

**Figura de um exemplo do código.**

****

* + 1. **ViewBag:** Esta forma receber dados sendo passado por parâmetros.

**Figura de um exemplo do código.**

****

* + 1. **ViewTipada:** Esta forma receber dados sendo passado por parâmetros.
  1. Para entender como funciona o \_layaut.cshtml usaremos o seguinte código @Html.ActionLink("Inicial","Index","Home")
  2. Como exemplo mostrado acima nos demostra.
     + 1. Primeiro Parâmetro Nome da página.
       2. Segundo parâmetro Nome da Action (Ação) que vai ser executar.
       3. Terceiro parâmetro Nome do controlador que vai ser executado.

1. Global.
   1. asax: Onde inicializa nossa aplicação.
2. Web.config:
   1. E responsável por carregar todos as configurações que será utilizado na aplicação.