**Padrão MVC**

O padrão MVC (Model-View-Controller) separa um aplicativo em três grupos de componentes principais: Modelos, Exibições e componentes.

Usando esse padrão as solicitações de usuário são encaminhadas para um controlador, que é responsável por trabalhar com o Modelo para executar as ações do usuário e/ou recuperar os resultados de consultas. O controlador escolhe a Exibição a ser exibida para o usuário e fornece-a com os dados do Modelo solicitados.



Obs.: A exibição e o controlador dependem do modelo. No entanto, o modelo não depende da exibição nem do controlador. Esse é um dos principais benefícios da separação. Essa separação permite que o modelo seja criado e testado de forma independente da apresentação visual.

**Tutorial MVC – prático**

Num aplicativo MVC, a view apenas mostra a informação. A controller lidas com as intereções do usuário. Por exemplo, o controller lida com os segmentos URL e os valores de query-string, e passa esses valores para o Model.

O model deve usar esses valores para consultar o banco de dados.

**Adicionando uma controller**

Todo método **public** em uma controller é chamavel como HTTP endpoint.

Um HTTP endpoint:

* É um URL segmentável na aplicação web, como:
  + <https://localhost:5001/HelloWorld>
* Combina:
  + O protocolo usado: HTTP
  + O local na rede do web server, inluindo a porta TCP: localhost:5001.
  + A URI alvo: HelloWorld.

MVC invoca as classes da controller, e os métodos de ações junto delas dependendo da URL passada.

A lógica padrão das rotas URL usadas pelo MVC, usam um formato tipo esse para determinar qual código invocar:

/[Controller]/[ActionName]/[Parameters]

O Formato de rotas é setado no método “Configure” no arquivo Startup.cs.

**Criando uma View.**

Os templates View são criados usando Razor:

* Possuem uma extensão de arquivo .cshtml
* Prove uma maneira elegante de criar output HTML usando C#.

- Quando o nome da View não é especificado, a View padrão é retornada. A View padrão tem o mesmo nome da Action Method, Index, por exemplo.

**Mudando Views e Layout pages.**

Na View Layout.cshtml, temos o layout padrão dá página.

A View Layout nos permite:

* Especificar os containers layouts de um site em um só lugar.
* Aplicar o html container layout em varias paginas no site.

Dentro da Layout page possuímos o @RenderBody(): RenderBody é um placeholder onde todas as views especificas que criamos aparecem, amarradas na Layout page.

**Passando Data da Controller para a View**

Controller Actions são invocadas como resposta de requisições da URL. Uma classe da controller é onde o código escrito lida com as requisições do browser. A controller recupera dados da base de dados e decide qual tipo de resposta enviar de volta para o browser. Views Templates podem ser usadas pela controller para gerar e formatar uma resposta HTML para os browsers.

As controllers são responsáveis por prover os dados requeridos pela view template para renderizar uma resposta.

Views tenplates não devem:

- Conter regras de negócios

- Interagir diretamente com a base de dados

A view template deve trabalhar apenas com os datos fornecidos pela controler. Essa separação de conceitos ajuda a manter o código:

- Limpo

- Testável

- Fácil de Manter