02

Criando rotas na aplicação

Customização de rotas

Suponha que a empresa em que você trabalha queira modificar o sistema de controle de estoque para o sistema que você está desenvolvendo nesse curso, porém como existem diversos clientes que dependem das URLs do sistema atual, devemos fazer com que as URLs do novo sistema sejam iguais às antigas.

Vamos fazer com que a listagem de produtos seja servida pela URL `/produtos` e que a visualização por `/produtos/{produtoId}`. A customização de uma url no Asp.Net MVC 5 é feita utilizando-se uma anotação definida no framework chamada `RouteAttribute`, essa anotação é utilizada sobre actions dos controllers e recebe como argumento a nova url do método.

Para fazermos com que o `Index` do `ProdutoController` possa ser acessado através de `/produtos`, utilizamos o seguinte código:

```
[Route("produtos")]
public ActionResult Index()
{
    // implementação
}
```

Podemos fazer o mesmo para a lógica de visualização, porém nela precisamos capturar qual é o `id` que será enviado para a lógica do controller. Para capturarmos uma variável da url, precisamos colocar o nome da variável que queremos capturar dentro de chaves, `{id}` para capturar uma variável chamada `id`.

```
[Route("produtos/{id}")]
public ActionResult Visualiza(int id)
{
    // implementação
}
```

Agora podemos acessar a visualização do produto de id 1 através da seguinte url: `/produtos/1`.

Além de utilizar o `RouteAttribute`, precisamos também habilitar a customização de rotas dentro do arquivo `App_Start/RouteConfig.cs`. Dentro desse arquivo podemos encontrar o método `RegisterRoutes`.

Para habilitarmos a customização de rotas, precisamos utilizar o método `MapMvcAttributeRoutes` do objeto `RouteCollection` que é recebido como argumento no método `RegisterRoutes`.

```
public static void RegisterRoutes(RouteCollection routes)
{
    routes.MapMvcAttributeRoutes();

    // Código que já estava no método
}
```

Como utilizamos o `ActionLink` para construir os links do capítulo anterior, os endereços dos links serão automaticamente atualizados para utilizar as urls que acabamos de configurar.

Mas no `ActionLink` precisamos passar qual é o nome da action e do controller para onde queremos enviar o usuário, mas o que aconteceria se por algum motivo a equipe decidisse mudar o nome do método ou do controller? Nesse caso teríamos que modificar todos os `ActionLink` s de todo projeto!

Para resolvermos esse problema, podemos utilizar uma configuração adicional do `RouteAttribute` , o `Name` . O parâmetro `Name` define qual é o nome da rota e esse nome pode ser utilizado posteriormente para montarmos links nas views.

```
[Route("produtos", Name="ListaProdutos")]
public ActionResult Index()
{
    // implementação
}

[Route("produtos/{id}", Name="VisualizaProduto")]
public ActionResult Visualiza(int id)
{
    // implementação
}
```

E agora, na view, podemos utilizar o método `RouteLink` do `HtmlHelper` passando o texto do link e o nome da rota para montar o link. Para criarmos o link para a página de visualização, utilizamos o seguinte código:

```
@Html.RouteLink("Voltar para a lista", "ListaProdutos")
```

No link de visualização, precisamos também passar quais são as informações que devem ser enviadas para o servidor. Essas informações são enviadas, novamente, através de objetos anônimos do C#.

```
@Html.RouteLink(produto.Nome, "VisualizaProduto", new {id = produto.Id})
```