04

Mudando locale dinamicamente

Transcrição

Apesar de termos traduzido partes da aplicação para os idiomas requeridos, de certa forma, deixamos o usuário preso a uma única opção, que depende das configurações de seu navegador. O comum nas aplicações é vermos a possibilidade de alterar o idioma por ação de algum menu.

Adicionaremos no menu do cabeçalho as duas opções de idioma, inglês e português. Vejamos como se encontra o menu atualmente:

```
<nav id="main-nav">
    <ul class="clearfix">
        <li>
            <a href="${s:mvcUrl('CCC#itens').build()}" rel="nofollow">
                <s:message code="menu.carrinho" arguments="${carrinhoCompras.quantidade}" />
            </a>
        </li>
        <li>
            <a href="/pages/sobre-a-casa-do-codigo" rel="nofollow">
                <fmt:message key="menu.sobre"/>
            </a>
        </li>
    </ul>
</nav>
```

O que podemos fazer é criar mais duas opções de menu, duplicado as tags `li` e em vez de um link para uma página, passaremos apenas um parâmetro do tipo `GET` usando como chave o `locale` e como valor o sufixo do idioma. Considerando isso, teremos:

```
<li>
    <a href="?locale=pt" rel="nofollow">
        <fmt:message key="menu.pt"/>
    </a>
</li>

<li>
    <a href="?locale=en_US" rel="nofollow">
        <fmt:message key="menu.en"/>
    </a>
</li>
```

E o menu completo será:

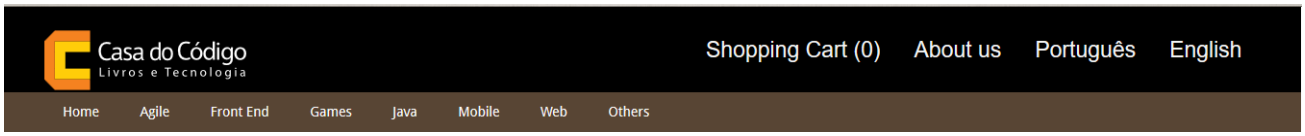
```
<nav id="main-nav">
    <ul class="clearfix">
        <li>
            <a href="${s:mvcUrl('CCC#itens').build()}" rel="nofollow">
                <s:message code="menu.carrinho" arguments="${carrinhoCompras.quantidade}" />
```

```
            </a>
        </li>
        <li>
            <a href="/pages/sobre-a-casa-do-codigo" rel="nofollow">
                <fmt:message key="menu.sobre"/>
            </a>
        </li>

        <li>
            <a href="?locale=pt" rel="nofollow">
                <fmt:message key="menu.pt"/>
            </a>
        </li>

        <li>
            <a href="?locale=en_US" rel="nofollow">
                <fmt:message key="menu.en"/>
            </a>
        </li>
    </ul>
</nav>
```

Veja que já utilizamos o `fmt:message` para usar as mensagens do `messages.properties`. Desta forma teremos que por nos arquivos de idiomas as chaves com os valores:

```
menu.pt = Português
menu.en = English
```

E o menu já é apresentado com as opções de idiomas.



Agora será que funciona? Ao clicarmos vemos que nada muda, ou seja, ainda não funciona. Claro, o *Spring* não sabe se deve mudar o idioma baseado nesta simples mudança na URL. Precisamos configurá-lo para que isso ocorra.

O que precisamos fazer é que alguém verifique na requisição a mudança do `locale` e também armazenar essa mudança em algum lugar, se não o usuário terá que mudar de idioma toda vez que mudar de página. O responsável por interpretar a mudança será um `interceptor` e armazenaremos o valor do `locale` do usuário nos **cookies** do navegador.

Na classe `AppWebConfiguration`, usaremos o método `addInterceptors` que recebe um objeto chamado `registry` do tipo `InterceptorRegistry` e através deste objeto, usaremos o método `addInterceptor` para adicionar um novo interceptador, este que verificará a mudança de `locale` do usuário, um objeto do tipo `LocaleChangeInterceptor`. Veja o código:

```
public void addInterceptors(InterceptorRegistry registry) {
    registry.addInterceptor(new LocaleChangeInterceptor());
}
```

O segundo passo é fornecer para o *Spring* um resolvedor de `locale` que, além de armazenar a configuração de `locale` do usuário, possa também carregar as páginas no idioma correto. Para isso, criaremos o método `localeResolver` que

retorna um objeto do mesmo tipo e dentro deste método apenas retornaremos um objeto da classe `CookieLocaleResolver`.

```
@Bean
public LocaleResolver localeResolver(){
    return new CookieLocaleResolver();
}
```

Como este método retornará um objeto a ser usado pelo *Spring*, precisaremos anotá-lo com `@Bean`. E é isso! Já funciona. Podemos mudar de idioma à vontade e navegar pelas páginas no idioma escolhido.

Nesta aula, aprendemos a lidar com a problemática da internacionalização, vimos que o processo é bastante simples e que poucas configurações são necessárias. Vimos também o uso das tags `fmt:message` e `s:message` e como podemos organizar os textos das páginas em arquivos com extensão `.properties`. Agora chegou o momento de praticar um pouco mais com os exercícios.