06

Usando Guava

Transcrição

Tudo está certo, mas devemos seguir sempre algumas recomendações. A documentação do **Spring**, apesar de fornecer o gerenciador de cache que estamos usando, alerta para que este seja usado apenas em ambiente de desenvolvimento. A sugestão é porque o Spring não permite algumas configurações - como por exemplo, o limite da quantidade de elementos que podem ser guardados no cache ou até mesmo um tempo de vida. Estas configurações são bem interessantes no mundo real e faremos uso delas, com o objetivo de melhorar o gerenciamento de cache pela nossa aplicação.

A documentação recomenda o uso do **Guava**, um ***framework*** de cache fornecido pelo **Google**. Para usá-lo, precisaremos alterar algumas de nossas configurações. Primeiro, precisaremos declarar o Guava como uma das dependências do projeto no `pom.xml`

```
<dependency>
    <groupId>com.google.guava</groupId>
    <artifactId>guava</artifactId>
    <version>18.0</version>
</dependency>

<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context-support</artifactId>
    <version>4.1.0.RELEASE</version>
</dependency>
```

Uma das dependências é o próprio Guava. A outra será uma biblioteca que faz a integração do mesmo com o **Spring**. O próximo passo para configurar o Guava em nosso projeto é instanciá-lo com as configurações do cache no mesmo lugar, onde estávamos criando o `ConcurrentMapCacheManager`, no método `cacheManager` da classe `AppWebConfiguration`.

Para podermos adicionar as configurações de tamanho e tempo limite do cache, precisaremos primeiro criar um objeto do tipo `CacheBuilder` e ao chamar o método `newBuilder` da classe, podemos chamar logo em seguida os métodos `maximumSize` e `expireAfterAccess`, passando valores de `100` para o tamanho máximo de elementos que serão guardados no cache e `5` e `TimeUnit.MINUTES` para o segundo método, respectivamente, indicando que o cache será expirado a cada cinco minutos. O código ficará assim:

```
CacheBuilder<Object, Object> builder = CacheBuilder.newBuilder().maximumSize(100).expireAfterAc
```

Após isto, criaremos um objeto do tipo `GuavaCacheManager` e usar o método `setCacheBuilder` para passar o `builder` de cache que criamos anteriormente. Os dois passos feitos agora ficam assim:

```
CacheBuilder<Object, Object> builder = CacheBuilder.newBuilder().maximumSize(100).expireAfterAc
GuavaCacheManager manager = new GuavaCacheManager();
manager.setCacheBuilder(builder);
```

Por último, retornamos o `manager` para que seja utilizado pelo Spring para gerenciar o cache da aplicação. O método completo deve estar parecido com este:

```
@Bean
public CacheManager cacheManager(){
  CacheBuilder<Object, Object> builder = CacheBuilder.newBuilder().maximumSize(100).expireAfter
  GuavaCacheManager manager = new GuavaCacheManager();
  manager.setCacheBuilder(builder);
  return manager;
}
```

Experimente fazer todos os testes novamente. Reinicie o servidor, limpe o console, abra a página inicial e veja se aparece algo no console. Atualize a página, abra a mesma em outro navegador, adicione um novo livro, diminua o tempo de vida do cache para verificar se será gerado um novo depois do tempo limite. Explore outras unidades de tempo da classe `TimeUnit` para testes.