04

Habilitando o cache

Transcrição

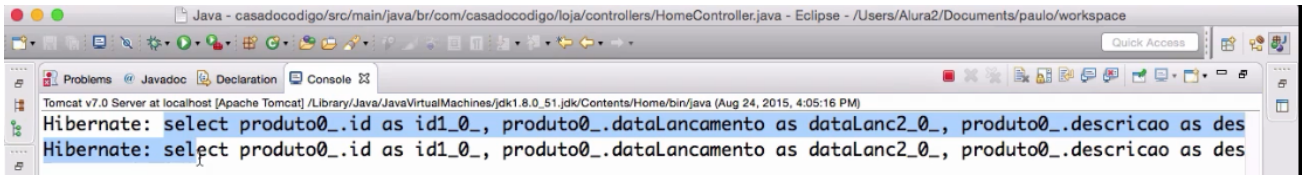
Em seguida, vamos pensar um pouco em performance. Você já pensou sobre a frequência em que os livros da Casa do Código são lançados? Os lançamentos não são feitos todos os dias, concorda? Em geral, passará um tempo antes de que a lista seja modificada (uma semana ou até mais).

Algo comum em aplicações deste tipo é o uso de **Cache** - um recurso que permite que guardemos informações no contexto da aplicação. Geralmente, os dados ficam armazenados no banco de dados - com acesso ao disco rígido. Com o uso de `cache`, quando as informações forem requisitadas, a aplicação já as terá e não precisará carregá-las novamente.

A subtituição por um acesso em memória, onde o **cache** é guardado, agiliza o processo. E a listagem dos livros parece ser um bom lugar para o uso de **cache**. Como a lista não muda com muita frequência, iremos usar este recurso, então. Faremos isto facilmente no **Spring**. Precisamos apenas marcar o método que tem os resultados a serem "cacheados" com a anotação `@Cacheable`. Assim faremos no método `index`, da classe `HomeController`.

```
@RequestMapping("/")
@Cacheable
public ModelAndView index(){
  List<Produto> produtos = produtoDao.listar();
  ModelAndView modelAndView = new ModelAndView("home");
  modelAndView.addObject("produtos", produtos);
  return modelAndView;
}
```

Agora precisamos testar se isso realmente vai funcionar. Reinicie o servidor e quando ele estiver apto novamente, limpe o console do **Eclipse** e abra a página inicial da aplicação no navegador. O código `SQL` da busca pelos livros no banco de dados deve ser impressa uma vez. E se atualizarmos a página? Com o cache habilitado, não deveria aparecer nada mais no console, porem vejá o que acontece.



É estranho, mas o código `SQL` aparece duas vezes. Isto não deveria acontece. Porém, em momento algum configuramos o cache da aplicação. Simplesmente usar a anotação não resolverá o problema, precisaremos habilitar o cache do **Spring**.

Esta configuração é feita na classe `AppWebConfiguration` e terá dois passos. O primeiro deles é marcar a classe com a anotação `@EnableCaching`. O segundo é fornecer um gerenciador de cache para que o **Spring** o use. Criaremos um novo método nesta mesma classe chamado `cacheManager`, que retornará um objeto do tipo `CacheManager`. Usaremos um gerenciador de cache que o **Spring** já fornece chamado `ConcurrentMapCacheManager`. Lembre-se de marcar este método com a anotação `@Bean`. Assim teremos:

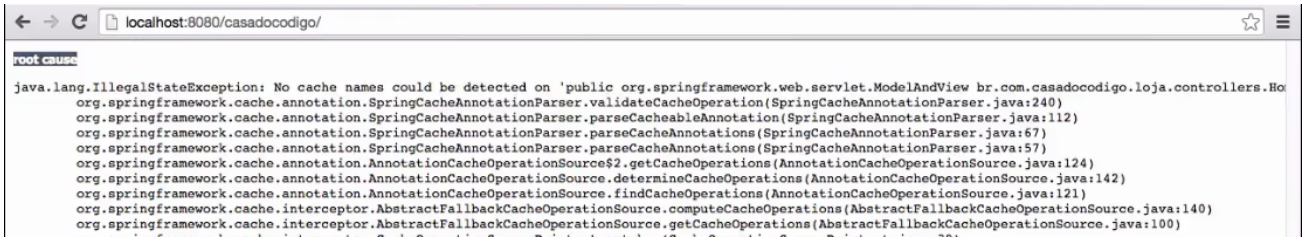
```
@EnableWebMvc
@ComponentScan(basePackageClasses={HomeController.class, ProdutoDAO.class, FileSaver.class, Carı
@EnableCaching
public class AppWebConfiguration {

  [...]

  @Bean
    public CacheManager cacheManager(){
        return new ConcurrentMapCacheManager();
    }

}
```

Mas ao reiniciar o servidor e tentar acessar a página inicial, teremos um erro. Veja:



O erro indica que nenhum nome de cache foi detectado na aplicação. Isso acontece porque o objeto da classe `ConcurrentMapCacheManager` usa `Map` para gerenciar o cache. Os `Map` s são estruturas de dados que armazenam valores referenciados a partir de chaves. Trata-se de um conjunto de chave e valor.

Para resolver o erro que ocorreu anteriormente, basta passarmos um parâmetro para a anotação `@Cacheable` indicando qual é a chave do cache daquele método. Usaremos `produtosHome` como a chave no método `index` da classe `HomeController` . Assim teremos:

```
@RequestMapping("/")
@Cacheable(value="produtosHome")
public ModelAndView index(){
    List<Produto> produtos = produtoDao.listar();
    ModelAndView modelAndView = new ModelAndView("home");
    modelAndView.addObject("produtos", produtos);
    return modelAndView;
}
```

Então, refaça o teste que fizemos anteriormente. Reinicie o servidor, limpe o console do **Eclipse** e abra a página inicial da aplicação, você verá uma linha de código `SQL` . Atualize a página, nada mais aparece no console. Experimente abrir outro navegador, o cache continua funcionando e nenhuma linha a mais de código `SQL` é impressa. Isto significa que não estamos mais acessando o banco de dados a todo momento. Estamos acessando o cache.