Resumo

Resumo da Seção 2

Spring Boot Starter

O Spring Boot tem como base fornecer os recursos dos módulos do Spring Framework e demais projetos vinculados a ele, por meio de dependências nomeadas como starters.

Entre os diversos starters, que podem ser visualizados no capítulo 13.5 do manual de referência do Spring Boot 1.5.10.RELEASE, temos o principal que é o spring-boot-starter-parent. Esse starter tem como objetivo informar a versão do Spring Boot que será usada no projeto.

```
<parent>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-parent</artifactId>
   <version>1.5.10.RELEASE</version>
   <relativePath/> <!-- lookup parent from repository -->
</parent>
```

Todos os demais starters declarados no arquivo de gerenciamento de dependências, como o Maven, serão baseados nas versões referentes a do ...-starter-parent. Assim, não é necessário informar a tag de versão da dependência declarada como starter do módulo a ser incluído no projeto. Como exemplo, veja o módulo web, o qual fornece os recursos para uso do Spring MVC.

```
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
   </dependency>
```

Outro item importante que deve ser incluído no arquivo pom.xml do Maven é o plugin de build

spring-boot-maven-plugin. Ele tem como responsabilidade gerar um arquivo .jar executável como artefato do projeto.

O Spring Boot é executado por meio de uma classe principal, por isso, o artefato é um .jar executável.

```
  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>
```

A classe principal, responsável por executar, ou seja, inicializar o Spring Boot é similar a apresentada a seguir:

```
@SpringBootApplication
public class DemoMvcApplication {

  public static void main(String[] args) {
    SpringApplication.run(DemoMvcApplication.class, args);
  }
}
```

No código da classe DemoMvcApplication podemos ver o uso da anotação @SpringBootApplication, responsável por incluir no projeto alguns recursos de inicialização do Spring Framework.

Já no método main() a instrução SpringApplication.run() recebe como parâmetro a classe que contém a anotação @SpringBootApplication.

Páginas e arquivos estáticos

Quando trabalhamos com Thymeleaf junto ao Spring Boot, os arquivos de páginas (.html) e arquivos estáticos como CSS, JS ou imagens, devem ser adicionados no classpath do projeto, ou seja, diretório src/main/resources.

Nesse diretórios teremos dois sub-diretórios principais, o static, onde serão armazenados os arquivos estáticos e o template, onde serão adicionadas os htmls:

src/main/resources/static/css/style.css

src/main/resources/static/js/jQuery.js

src/main/resources/static/templates/index.html

WebJars

O projeto WebJars tem como objetivo fornecer arquivos .jar contendo bibliotecas de CSS, JS ou mesmo de imagens. Esses arquivo podem então ser adicionados no projeto como uma dependência no arquivo pom.xml.

Deste modo, ao invés de realizar o download dos fontes de recursos como o Bootstrap ou jQuery e incluí-los como recursos estáticos, incluímos no lugar a biblioteca do WebJar referente.

```
<dependency>
    <groupId>org.webjars</groupId>
    <artifactId>bootstrap</artifactId>
    <version>4.0.0</version>
</dependency>
```

Uma das vantagens das desse recurso é que se for necessário alterar a versão da biblioteca, basta alterá-la no arquivo pom.xml. Outra vantagen é que esta dependência estará disponível no repositório local do Maven, facilitando o uso dela em qualquer outro projeto que esteja sendo desenvolvido.

Para usar este recurso junto ao Spring Boot, é necessário incluir a dependência do webjars-locator. Essa dependencia já faz partes dos recursos fornecidos pelo Spring Boot e assim, não é

necessário adicionar uma versão ao declará-la. A inclusão dela no arquivo do Maven é para dizer ao Spring Boot que você vai fazer uso de seus recursos.

```
<dependency>
    <groupId>org.webjars</groupId>
    <artifactId>webjars-locator</artifactId>
</dependency>
```

E a finalidade principal da webjars-locator é relacionar as dependências de webjars (Bootstrap, jQuery, …) com as urls incluí-das nas páginas html.

```
<link href="/webjars/bootstrap/css/bootstrap.min.css" rel="stylesheet"/>
```

Desse modo, a página consegue localizar as bibliotecas estáticas entre as dependências do projeto e assim, quando a página abrir no navegador vai conseguir acessar tais recursos.

Resolvendo solicitações

Após a inclusão das páginas no projeto, será necessário preparar os controladores para que as solicitações sejam processadas e assim, as páginas possam ser acessadas.

Por exemplo, a página home.html tem no menu lateral o link Cadastrar para abrir a página cadastro.html referente a Departamentos. Este link é declarado da seguinte forma:

```
<a class="nav-link" href="/departamentos/cadastrar">Cadastrar</a>
```

Observe que a url possui dois caminhos que são:

/departamentos - tem como função acessar a classe DepartamentoController;

/cadastrar - tem como responsabilidade acessar o método mapeado com este caminho dentro do controller de departamentos.

Em DepartamentoController a anotação @RequestMapping, declarada sobre a assinatura da classe, tem como valor o caminho /departamentos. Dessa forma, a url do link Cadastrar vai

chegar até esse controller.

```
@Controller
@RequestMapping("/departamentos")
public class DepartamentoController {

   @GetMapping("/cadastrar")
   public String cadastrar() {
      return "/departamento/cadastro";
   }
}
```

Em seguida, o acesso chega ao método cadastrar(), por conta da anotação @GetMapping, que possui o caminho de acesso para /cadastrar.

Como resposta, o método cadastrar() retorna um objeto String com o valor /departamento/cadastro. Esse retorno vai abrir a página cadastro.html que está armazenada no diretório /templates/departamento. Entretanto, não se declara no retorno o /templates nem o .html, essas instruções estão implícitas para o Spring MVC.

Código Fonte

Caso tenha tido algum tipo de dificuldade para acompanhar a desenvolvimento do código fonte até o final desta seção, ele está disponível na área de arquivo para download.