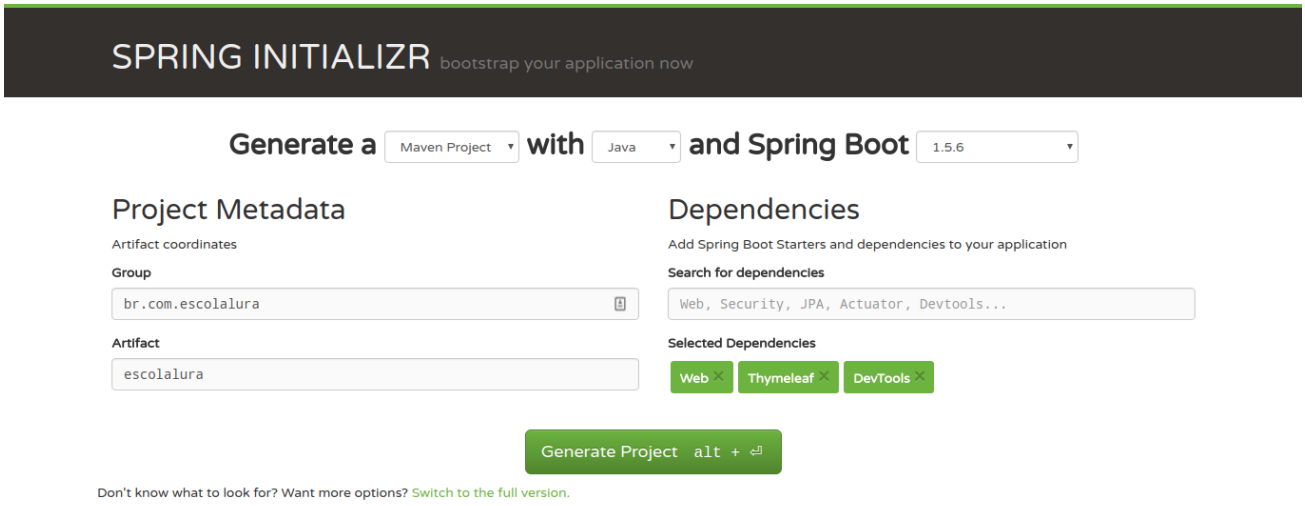
01

Criação do projeto web

Transcrição

Agora que já revisamos o básico sobre o MongoDB, precisaremos utilizá-lo para resolver um problema real do cliente. Desenvolveremos uma aplicação para gerenciamento de alunos de uma escola, e um dos principais problemas está ligado à sua localização. Por não ser bem localizada, é interessante que haja alguma funcionalidade que permita os alunos se agruparem por localização, compartilhando-se carona. Essa é uma das motivações de se utilizar MongoDB neste projeto.

As ferramentas que utilizaremos serão o próprio MongoDB e o framework SpringBoot, que facilitará bastante a criação dessa aplicação. Para termos nosso projeto base com algumas das configurações pré-realizadas, utilizaremos o site do *Spring Initializr* (http://start.spring.io/), em que podemos criar o projeto com as seguinte definições: utilizando Maven, SpringBoot 1.5.3, tendo `br.com.alura.escolalura` como *group*, `escolalura` como *artifact* e, como dependências, termos os *starters Thymeleft*, *Dev tools* e *Web*.



Depois disto, basta clicarmos em *Generate Project*, baixando-o, descompactando-o e importando o projeto como um Maven. A etapa de *import* do projeto pode demorar um pouco porque o Maven precisa baixar todas as dependências do projeto.

Por último, precisaremos configurar o *driver* de conexão do MongoDB que utilizamos na aula anterior como dependência. Adicionaremos este *driver* no `pom.xml`:

```
<dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver</artifactId>
    <version>3.5.0</version>
</dependency>
```

Criaremos um *controller* para testarmos se tudo está funcionando corretamente, começando pela classe `AdminController`, em que criaremos um método chamado `index`, que retorna uma `String "index"`. Anotaremos a classe com `@Controller`, e o método com `@GetMapping`, informando que trata-se de um método que responde a uma requisição `GET` na raiz do projeto, `/`.

```
package br.com.alura.escolalura.controllers;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class AdminController {

    @GetMapping("/")
    public String index(){
        return "index";
    }

}
```

Note que criamos um novo pacote no projeto para agrupar todos os *controllers*. Como o método `index` retorna uma `string` que indica o nome do arquivo template que será carregado, precisaremos criar este template. Neste HTML adicionaremos apenas uma mensagem de boas vindas. Criaremos o `index.html` na pasta `resources/templates` :

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8"/>
    <title>Insert title here</title>
</head>
<body>
    <h1>Bem-vindo ao Sistema Escolalura</h1>
</body>
</html>
```

Lembre-se que o Thymeleaf é minucioso na validação do HTML, e todas as tags precisam estar sempre fechadas. Notou o fechamento da tag meta? Fique atento! Esses detalhes podem gerar erros.

Agora precisaremos de fato executar a aplicação. A classe `EscolaluraApplication` , criada automaticamente pelo gerador do projeto, possui o método `main` , em que utilizaremos as opções "botão direito -> Run as > Java Application".

O SpringBoot não precisa de um servidor para ser executado. O mesmo já vem preparado com um contêiner pré-configurado para executar o projeto sem maiores problemas.

Esta é a página que devemos visualizar ao acessarmos o projeto no navegador pelo endereço `localhost:8080` . Isto indica que tudo está funcionando como esperado. A requisição foi recebida, mapeada pelo *controller*, e o template foi carregado corretamente.

A aplicação já funciona, mas para que a deixemos um pouco mais elegante visualmente, utilizaremos um framework conhecido, chamado *Materialize* (http://materializecss.com/). Podemos baixá-lo no site, descompactá-lo e copiar sua pasta para a `static` na pasta `resources` do projeto.

> Não estudaremos o Materialize especificamente neste curso, apenas o utilizaremos para deixar as páginas mais atrativas.

Depois de copiarmos a pasta `materialize` para a pasta `static` em nosso `index.html` precisaremos configurar as tags de importação de estilos e scripts deste framework. Na documentação do próprio Materialize é possível ver esses imports:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8"/>
    <title>Insert title here</title>
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet" />
    <link type="text/css" rel="stylesheet" href="materialize/css/materialize.min.css"  media="s
</head>
<body>
    <h1>Bem-vindo ao Sistema Escolalura</h1>

    <script type="text/javascript" src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
    <script type="text/javascript" src="materialize/js/materialize.min.js"></script>
</body>
</html>
```

Adicionaremos o *Material Icons Font* e o `materialize.min.css` como dependências de estilo, e o *jQuery* e `materialize.min.js` como dependências de scripts em nossa página. E agora? Como utilizar este framework? Por meio de suas classes! Para deixar a página com o fundo um pouco mais cinza, por exemplo, podemos usar as classes `gray lighten-3` . Para centralizarmos o `h1` da página, `main-title center` .

Para cada funcionalidade do nosso sistema, criaremos um cartão na página e, desta forma, deixaremos o primeiro cartão pronto. Note a estrutura do HTML:

```
<div class="row center">
    <div class="col s12 m4 l4 waves-effect waves-light">
        <a href="/aluno/cadastrar">
            <div class="card-panel hoverable z-depth-1 center gray lighten-4">
                <i class="large material-icons">perm_identity</i>
                <div class="truncate">Cadastrar Alunos</div>
            </div>
        </a>
    </div>
</div>
```

Estamos criando uma linha centralizada em que um cartão com ícone e link informa que se trada do cadastro de alunos. Nosso HTML final, com o página mais cinza e o título centralizado com o cartão, fica da seguinte forma:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8"/>
    <title>Insert title here</title>
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet" />
    <link type="text/css" rel="stylesheet" href="materialize/css/materialize.min.css"  media="s
</head>
<body class="grey lighten-3">
    <h1 class="main-title center">Bem-vindo ao Sistema Escolalura</h1>

    <div class="row center">
        <div class="col s12 m4 l4 waves-effect waves-light">
            <a href="/aluno/cadastrar">
                <div class="card-panel hoverable z-depth-1 center grey lighten-4">
                    <i class="large material-icons">perm_identity</i>
                    <div class="truncate">Cadastrar Alunos</div>
                </div>
            </a>
        </div>
    </div>

    <script type="text/javascript" src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
    <script type="text/javascript" src="materialize/js/materialize.min.js"></script>
</body>
</html>
```

Se atualizarmos o navegador, veremos: