02

Atualizando nosso aluno

Transcrição

Comentamos anteriormente sobre a necessidade de mapearmos os alunos, mas em nossa aplicação não há uma forma de endereçamento, de contato com eles. Vamos criá-lo agora como uma nova classe denominada `Contato`, no pacote `br.com.alura.escolalura.models`.

Esta classe em particular terá três atributos: o endereço, sendo uma *String*, uma lista de coordenadas, do tipo *Double*, visto que coordenadas são números de ponto flutuante. E, por último, uma *String* que indica o tipo de ponto da coordenada.

Os dois últimos atributos precisam estar escritos em inglês porque serão utilizados internamente pelos algoritmos de cálculo do MongoDB, portanto chamados `coordinates` e `type`. O atributo `type` será inicializado como `Point`, estratégia utilizada pelo Mongo. Dessa forma teremos a classe `Contato`:

```
public class Contato {
  private String endereco;
  private List<Double> coordinates;
  private String type = "Point";

  public String getEndereco() {
    return endereco;
  }
  public void setEndereco(String endereco) {
    this.endereco = endereco;
  }
  public List<Double> getCoordinates() {
    return coordinates;
  }
  public void setCoordinates(List<Double> coordinates) {
    this.coordinates = coordinates;
  }
  public String getType() {
    return type;
  }
  public void setType(String type) {
    this.type = type;
  }

}
```

Além disso criaremos dois construtores, um padrão a ser utilizado pelo *Spring* e o segundo com parâmetros de endereço e coordenadas, para criarmos o objeto `contato` informando estes dados.

```
public Contato() { }

public Contato(String endereco, List<Double> coordinates) {
  this.endereco = endereco;
```

```
    this.coordinates = coordinates;
  }
```

Precisaremos também criar um atributo na classe `Aluno` do tipo `Contato` juntamente com seus *getters and setters*:

```
public class Aluno {
  // código omitido
  private Contato contato;

  public Contato getContato() {
    return contato;
  }
  public void setContato(Contato contato) {
    this.contato = contato;
  }
  // código omitido
}
```

Certo, agora é necessário possibilitarmos o preenchimento dos campos de endereço, latitude, longitude, as coordenadas. Como o aluno saberá a sua latitude e longitude? Existem sites que, a partir de um determinado endereço, conseguem informar estes valores. Um deles é o latlong.net (http://www.latlong.net/).

O problema é que não é comum pedir essas informações ao usuário, passando a ele a responsabilidade de saber disso. Seria interessante que, dado um endereço, pudéssemos saber suas coordenadas sem que o aluno precisasse se preocupar com isso.

Existe a biblioteca Java Client For Google Maps Services (https://mvnrepository.com/artifact/com.google.maps/google-maps-services), que resolve muito bem este problema. Podemos utilizá-la em nosso projeto, recebendo um endereço e depois preenchendo os dados de coordenada do aluno. Em `pom.xml` adicionaremos esta nova dependência:

```
<dependency>
  <groupId>com.google.maps</groupId>
  <artifactId>google-maps-services</artifactId>
  <version>0.1.20</version>
</dependency>
```

Criaremos um novo campo no formulário de cadastro de alunos em `resources/template/aluno/cadastrar.html` :

```
<div class="row">
  <div class="input-field col s12">
    <input id="endereco" type="text" class="validate" th:field="*{contato.endereco}"  />
    <label for="endereco">Endereço</label>
  </div>
</div>
```

Com isso já temos o necessário para obtermos o endereço do aluno em nossa aplicação e a partir dela capturarmos suas coordenadas.