02

Formulário de cadastro de aluno

Transcrição

Com o cartão que indica que teremos um cadastro de alunos criado, o próximo passo é implementarmos um formulário que possibilite a inserção dos dados do aluno efetivamente. Porém, antes do formulário em si, precisaremos criar um novo *controller* para tratar das requisições do novo domínio da aplicação.

No pacote `br.com.alura.escolalura.controllers`, criaremos a classe `AlunoController` que contará com o método `cadastrar`, em que receberemos como argumento um modelo, este especificamente se tratando do Aluno, ainda não criado. Este método deverá retornar a página `cadastrar`, da pasta `aluno`. Assim, teremos como rota de mapeamento o endereço `/aluno/cadastrar`.

```
@Controller
public class AlunoController {

  @GetMapping("/aluno/cadastrar")
  public String cadastrar(Model model){
    return "aluno/cadastrar";
  }
}
```

A criação da pasta "aluno" em *resources* será utilizada para agrupar as páginas do domínio do aluno. Caso contrário todas as páginas ficarão misturadas no projeto.

O modelo Aluno é composto de vários atributos: primeiro temos seu nome, data de nascimento, curso, uma listagem de notas e uma de habilidades. Considerando isto, teremos a classe `Aluno` da seguinte forma:

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

public class Aluno {
  private ObjectId id;
  private String nome;
  private Date dataNascimento;
  private Curso curso;
  private List<Nota> notas;
  private List<Habilidade> habilidades;
}
```

Duas observações interessantes são: criamos um pacote diferente para agrupar todos os modelos (`br.com.alura.escolalura.models`), e o atributo `id` do tipo `ObjectId` (gerado automaticamente pelo MongoDB) é necessário na classe para algumas operações. Note também que Curso, Nota e Habilidade são outros três tipos a serem criados. Ainda no pacote de modelos, teremos:

a classe `Curso`;

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

public class Curso {
  private String nome;
}
```

a classe `Nota` ;

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

public class Nota {
  private Double valor;
}
```

a classe `Habilidade`

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

public class Habilidade {
  private String nome;
  private String nivel;
}
```

Como estamos utilizando o *SpringMVC*, precisaremos estar atentos a um detalhe: todos os atributos das classes necessitam dos *getters* e *setters*, pois o framework os utilizará quando necessário, para capturar ou inserir o nome do aluno em um formulário, por exemplo. Serão utilizados os respectivos métodos, porém, é preciso estar atento para não criar o conhecido *modelo anêmico*. Então, gerando todos os *getters* e *setters*, teremos:

a classe `Aluno` ;

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

import java.util.Date;
import java.util.List;

import org.bson.types.ObjectId;

public class Aluno {
  private ObjectId id;
  private String nome;
  private Date dataNascimento;
  private Curso curso;
  private List<Nota> notas;
  private List<Habilidade> habilidades;

  public ObjectId getId() {
    return id;
  }
  public void setId(ObjectId id) {
    this.id = id;
  }
  public String getNome() {
```

```
    return nome;
  }
  public void setNome(String nome) {
    this.nome = nome;
  }
  public Date getDataNascimento() {
    return dataNascimento;
  }
  public void setDataNascimento(Date dataNascimento) {
    this.dataNascimento = dataNascimento;
  }
  public Curso getCurso() {
    return curso;
  }
  public void setCurso(Curso curso) {
    this.curso = curso;
  }
  public List<Nota> getNotas() {
    return notas;
  }
  public void setNotas(List<Nota> notas) {
    this.notas = notas;
  }
  public List<Habilidade> getHabilidades() {
    return habilidades;
  }
  public void setHabilidades(List<Habilidade> habilidades) {
    this.habilidades = habilidades;
  }

}
```

a classe `Curso` ;

```
public class Curso {
  private String nome;

  public String getNome() {
    return nome;
  }

  public void setNome(String nome) {
    this.nome = nome;
  }
}
```

a classe `Nota` ;

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

public class Nota {
  private Double valor;

  public Double getValor() {
```

```
    return valor;
  }

  public void setValor(Double valor) {
    this.valor = valor;
  }
}
```

e a classe `Habilidade`

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

public class Habilidade {
  private String nome;
  public String getNome() {
    return nome;
  }
  public void setNome(String nome) {
    this.nome = nome;
  }
  public String getNivel() {
    return nivel;
  }
  public void setNivel(String nivel) {
    this.nivel = nivel;
  }
  private String nivel;
}
```

Com isso, precisaremos realizar mais dois passos para que nosso formulário de cadastro de alunos fique pronto. Primeiramente enviaremos o objeto `aluno` para ser utilizado no formulário, e o formulário em si. Utilizando o objeto `model` no método `cadastrar` da classe `AlunoController`, poderemos usar o método `addAttribute` para disponibilizar o objeto `aluno` à *view*. Assim, teremos:

```
@Controller
public class AlunoController {

  @GetMapping("/aluno/cadastrar")
  public String cadastrar(Model model){

    model.addAttribute("aluno", new Aluno());

    return "aluno/cadastrar";
  }
}
```

Para criarmos o formulário, temos o HTML da página `cadastrar.html` pronto para uso:

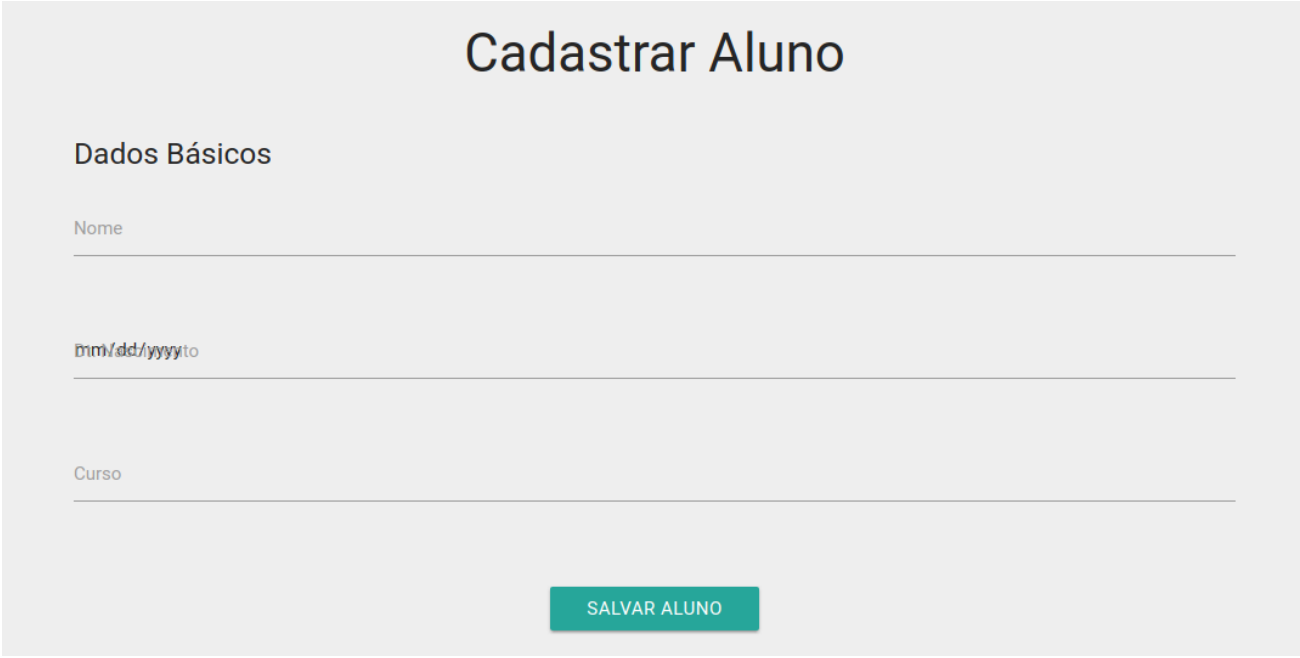
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8" />
```

```
  <link type="text/css" rel="stylesheet" href="../materialize/css/materialize.min.css" media="s
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet" />
  <title>EscolAlura</title>
  <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
  <script type="text/javascript" src="../js/inicializar.js"></script>
</head>
<body class="grey lighten-3">
  <div id="formularioEdicao" class="container">
    <h3 class="main-title center">Cadastrar Aluno</h3>
    <div class="row">
      <form class="col s12" action="#" th:action="@{/aluno/salvar}" th:object="${aluno}" method
        <div class="section">
          <h5>Dados Básicos</h5>
          <div class="row">
            <div class="input-field col s12">
              <input id="nome" type="text" th:field="*{nome}" />
              <label for="nome">Nome</label>
            </div>
          </div>
          <div class="row">
            <div class="input-field col s12">
              <input id="dataNascimento" type="date" class="datepicker" th:field="*{dataNascimer
              <label for="dataNascimento">Dt. Nascimento</label>
            </div>
          </div>
          <div class="row">
            <div class="input-field col s12">
              <input id="curso" type="text" class="validate" th:field="*{curso.nome}"  />
              <label for="curso">Curso</label>
            </div>
          </div>

        </div> <!-- Fim SECTION Dados Basicos -->

        <div class="row">
          <div class="input-field col s12 center">
            <button class="btn waves-effect waves-light" type="submit" name="action">Salvar Alu
          </div>
        </div>
      </form>
    </div>

  </div> <!-- Fim do formulario de edicao -->

  <script type="text/javascript" src="../materialize/js/materialize.min.js"></script>
</body>
</html>
```

Note que o formulário em si é apenas HTML utilizando-se as classes do Materialize, porém, com um pequeno incremento, o uso dos atributos do **Thymeleaf**, para relacionar os campos com o objeto (por exemplo, `th:field="*{contato.endereco}"`).

Lembre-se de criar o arquivo cadastrar.html na pasta "aluno" em **resources**.

Após estes passos, poderemos ver a página de cadastro dos alunos acessando `localhost:8080/aluno/cadastrar` :


Cadastrar Aluno
Dados Básicos
Nome
mm/dd/yyyy
Curso
SALVAR ALUNO