08

Como visto na aula

Para essa aula, a gente precisa criar uma aplicação que vai ser usada para atender nosso cliente. Para criar essa aplicação foi escolhido o site : http://start.spring.io/ (http://start.spring.io/) que cria uma aplicação baseada no Spring Boot.

Depois de criada a aplicação, importe ela para sua IDE. Nesse curso estamos usando o Eclipse, mas se sinta livre para escolher a melhor na sua opinião.

Adicione o driver do MongoDB no seu pom.xml

```
<dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver</artifactId>
    <version>3.4.2</version>
</dependency>
```

Para começar o projeto, crie uma Controller que vai receber as requisições iniciais:

```
@Controller
public class AdminController {

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

}
```

E depois crie um HTML que vai responder para essa rota

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

</body>
</html>
```

Vamos usar nesse curso também um framework css que é o Materialize. Faça o download nesse site: http://materializecss.com/ (http://materializecss.com/)

Rode a aplicação e veja se está tudo funcionando até agora. Para rodar a aplicação é só rodar o método main na classe EscolaluraRoteiroApplication

Vamos avançando o conteúdo e criando as controller e páginas necessárias, conforme a gente viu na aula.

A parte mais importante aula é a criação do codec

```
public class AlunoCodec implements CollectibleCodec<Aluno> {

    private Codec<Document> codec;

    public AlunoCodec(Codec<Document> codec) {
        this.codec = codec;
    }

    @Override
    public void encode(BsonWriter writer, Aluno aluno, EncoderContext encoderContext) {
        Document document = new Document();

        ObjectId id = aluno.getId();
        String nome = aluno.getNome();
        Date dataNascimento = aluno.getDataNascimento();
        Curso curso = aluno.getCurso();
        List<Habilidade> habilidades = aluno.getHabilidades();
        List<Nota> notas = aluno.getNotas();
        Contato contato = aluno.getContato();

        document.put("_id", id);
        document.put("nome", nome);
        document.put("dataNascimento", dataNascimento);
        document.put("curso", new Document().append("nome", curso.getNome()));

        if (habilidades != null) {
            List<Document> habilidadesDocument = new ArrayList<Document>();
            for (Habilidade habilidade : habilidades) {
                habilidadesDocument.add(
                        new Document().append("nome", habilidade.getNome()).append("nivel", hab
                }
                document.put("habilidades", habilidadesDocument);
            }

            if (notas != null) {
                List<Document> notasDocument = new ArrayList<Document>();
                for (Nota nota : notas) {
                    notasDocument.add(new Document().append("valor", nota.getValor()));
                }
                document.put("notas", notasDocument);

            }

            List<Double> coordinates = new ArrayList<Double>();
            for (Double location : contato.getCoordinates()) {
                coordinates.add(location);
            }

            document.put("contato", new Document()
                    .append("endereco", contato.getEndereco())
                    .append("coordinates", coordinates)
                    .append("type", contato.getType()));
```

```
        codec.encode(writer, document, encoderContext);
    }

    @Override
    public Class<Aluno> getEncoderClass() {
        return Aluno.class;
    }

    @SuppressWarnings("unchecked")
    @Override
    public Aluno decode(BsonReader reader, DecoderContext decoderContext) {
        Document document = codec.decode(reader, decoderContext);

        Aluno aluno = new Aluno();

        aluno.setId(document.getObjectId("_id"));
        aluno.setNome(document.getString("nome"));
        aluno.setDataNascimento(document.getDate("dataNascimento"));
        Document curso = (Document) document.get("curso");
        if (curso != null) {
            String nomeCurso = curso.getString("nome");
            aluno.setCurso(new Curso(nomeCurso));
        }

        List<Document> notasDocument = (List<Document>) document.get("notas");
        if (notasDocument != null) {
            List<Nota> notas = new ArrayList<Nota>();
            for (Document documentNota : notasDocument) {
                notas.add(new Nota(documentNota.getDouble("valor")));
            }
            aluno.setNotas(notas);
        }
        List<Document> habilidadesDocument = (List<Document>) document.get("habilidades");
        if (habilidadesDocument != null) {
            List<Habilidade> habilidades = new ArrayList<Habilidade>();
            for (Document documentHabilidade : habilidadesDocument) {
                habilidades.add(
                        new Habilidade(documentHabilidade.getString("nome"), documentHabilidade
            }
            aluno.setHabilidades(habilidades);
        }

        Document contato = (Document) document.get("contato");
        if (contato != null) {
            String endereco = contato.getString("endereco");
            List<Double> coordinates =  (List<Double>) contato.get ("coordinates");
            aluno.setContato(new Contato(endereco, coordinates));

        }

        return aluno;
    }

    @Override
    public Aluno generateIdIfAbsentFromDocument(Aluno aluno) {
        return documentHasId(aluno) ? aluno : aluno.gerarNovoId();
    }
```

```
    @Override
    public boolean documentHasId(Aluno aluno) {
        return aluno.getId() != null;
    }

    @Override
    public BsonValue getDocumentId(Aluno aluno) {
        if (!documentHasId(aluno)) {
            throw new IllegalStateException("O aluno não tem um _id");
        }

        return new BsonString(aluno.getId().toHexString());
    }

}
```

Com ele somos capazes de fazer o parse de um objeto Java para um documento no MongoDB.