# Usando o MongoDB a partir do Spring

Certified Developer
The Ultimate Tech Degree

DigitalHouse >

# Dependências

Com o SpringBoot Initializer no IntelliJ IDE, podemos criar um projeto Spring com as seguintes dependências no pom.xml:

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

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

O **spring-boot-starter-data-mongodb** tem tudo o que você precisa para criar um projeto SpringBoot com o MongoDB.

**spring-boot-starter-data-mongodb**
- mongodb-driver
  - mongodb-driver:bson
  - mongodb-driver:driver-core
- spring-data-mongodb
  - spring-data-commons

# Propriedades da conexão

Em resources, criamos um arquivo com o nome application.properties onde adicionaremos a configuração.

```
#mongodb
spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
spring.data.mongodb.database=mongoexample
```

# Classes e estruturas do projeto

A classe **Book** mapeia a Collection books.

```
@Document(collection = “books”)
public class Book{

@Id
private String id;
private String author;
@Field(name = “book”)
private String bookTitle;
}
```

# Classes e estruturas do projeto

O **BookRepository** extende de MongoRepository.

```
@repository
public interface BookRepository extends
MongoRepository<Book,String> {
}
```

# Classes e estruturas do projeto

O **BookService** com o método findAllBooks.

```
@Service
public class BookService{

  private final BookRepository bookRepository;

  public BookService(BookRepository  bookRepository){
    this.bookRepository = bookRepository;
  }

  public List<Book> findAllBooks();
    return bookRepository.findAll();
}
```

# Classes e estruturas do projeto

O **Controller**.

```
@RestController
@RequestMapping(value = “/mongoexample”)
public class BookController{

  private final BookService bookService;

  public BookController(BookService bookService){
    this.bookService = bookService;
  }

  @GetMapping(value = “/books”)
  public List<Book> getAllBooks();
    return bookService.findAllBooks();
}
```

# Queries especiais

No caso de querer fazer uma query que a partir de seu parâmetro realize o filtro por ele, como findBooksByAuthor(), basta informar o parâmetro com o mesmo nome do campo no DB, neste caso “author”, da seguinte forma:

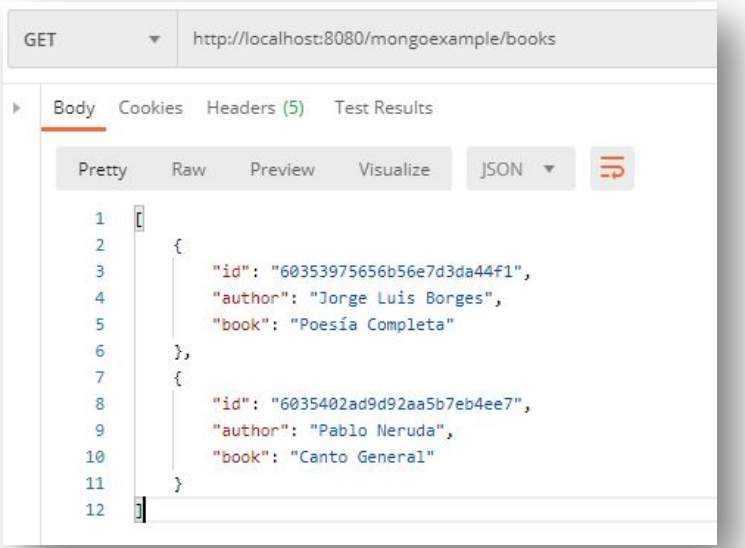
```
@Repository
public interface BookRepository extends MongoRepository
<Book,String>{
  List<Book> findBooksByAuthor(String author);
}
```

# Testando a aplicação

Execute a aplicação com mvn spring-boot:run. Em seguida, no navegador ou no Postman faça um GET para a url:

http://localhost:8080/mongoexample/books

Será realizado um request que produzirá uma query em nosso banco de dados do MongoDB local e nos retornará um response com todos os documentos da collection Books.



DigitalHouse>