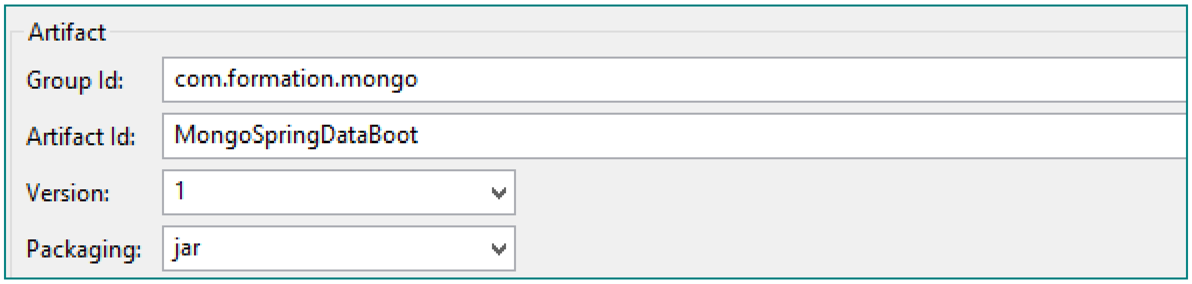
SpringData + SpringBoot + MongoDb

Création du projet maven dans Eclipse



Artifact

Group Id:	com.formation.mongo
Artifact Id:	MongoSpringDataBoot
Version:	1
Packaging:	jar

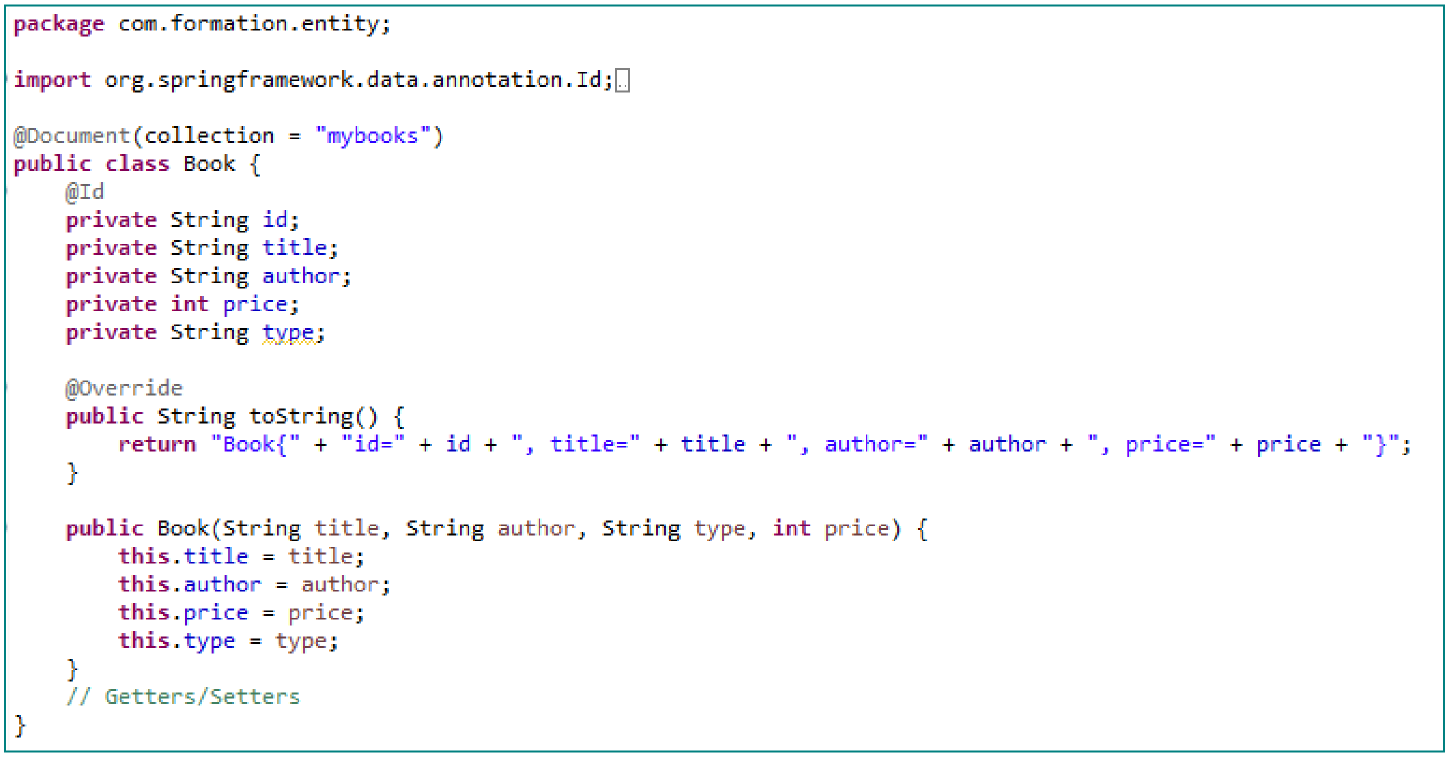
Configuration du fichier pom.xml

```
        <properties>
                <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
                <maven.compiler.source>1.8</maven.compiler.source>
                <maven.compiler.target>1.8</maven.compiler.target>
                <start-class>com.formation.app.Application</start-class>
        </properties>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-parent</artifactId>
                <version>1.3.2.RELEASE</version>
        </parent>

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

        <build>
                <plugins>
                        <plugin>
                                <groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-plugin</artifactId>
                        </plugin>
                </plugins>
        </build>
```

Dependances : web : springdatamongo

Création des entités

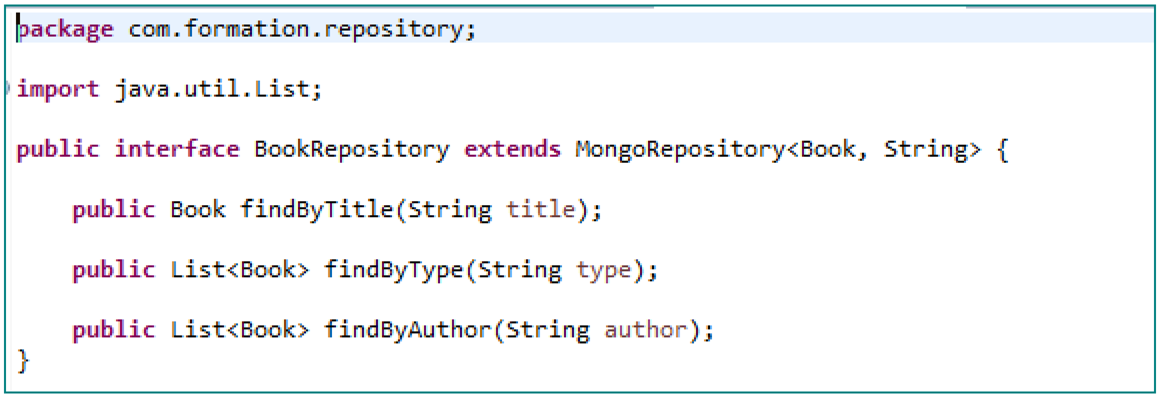
Book.java



```java
package com.formation.entity;

import org.springframework.data.annotation.Id;

@Document(collection = "mybooks")
public class Book {
    @Id
    private String id;
    private String title;
    private String author;
    private int price;
    private String type;

    @Override
    public String toString() {
        return "Book{" + "id=" + id + ", title=" + title + ", author=" + author + ", price=" + price + "}";
    }

    public Book(String title, String author, String type, int price) {
        this.title = title;
        this.author = author;
        this.price = price;
        this.type = type;
    }
    // Getters/Setters
}
```

Création des DAOs

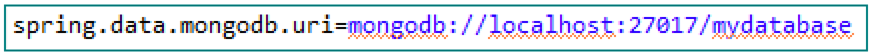
BookRepository.java



```java
package com.formation.repository;

import java.util.List;

public interface BookRepository extends MongoRepository<Book, String> {

    public Book findByTitle(String title);

    public List<Book> findByType(String type);

    public List<Book> findByAuthor(String author);
}
```

Configuration pour MongoDB

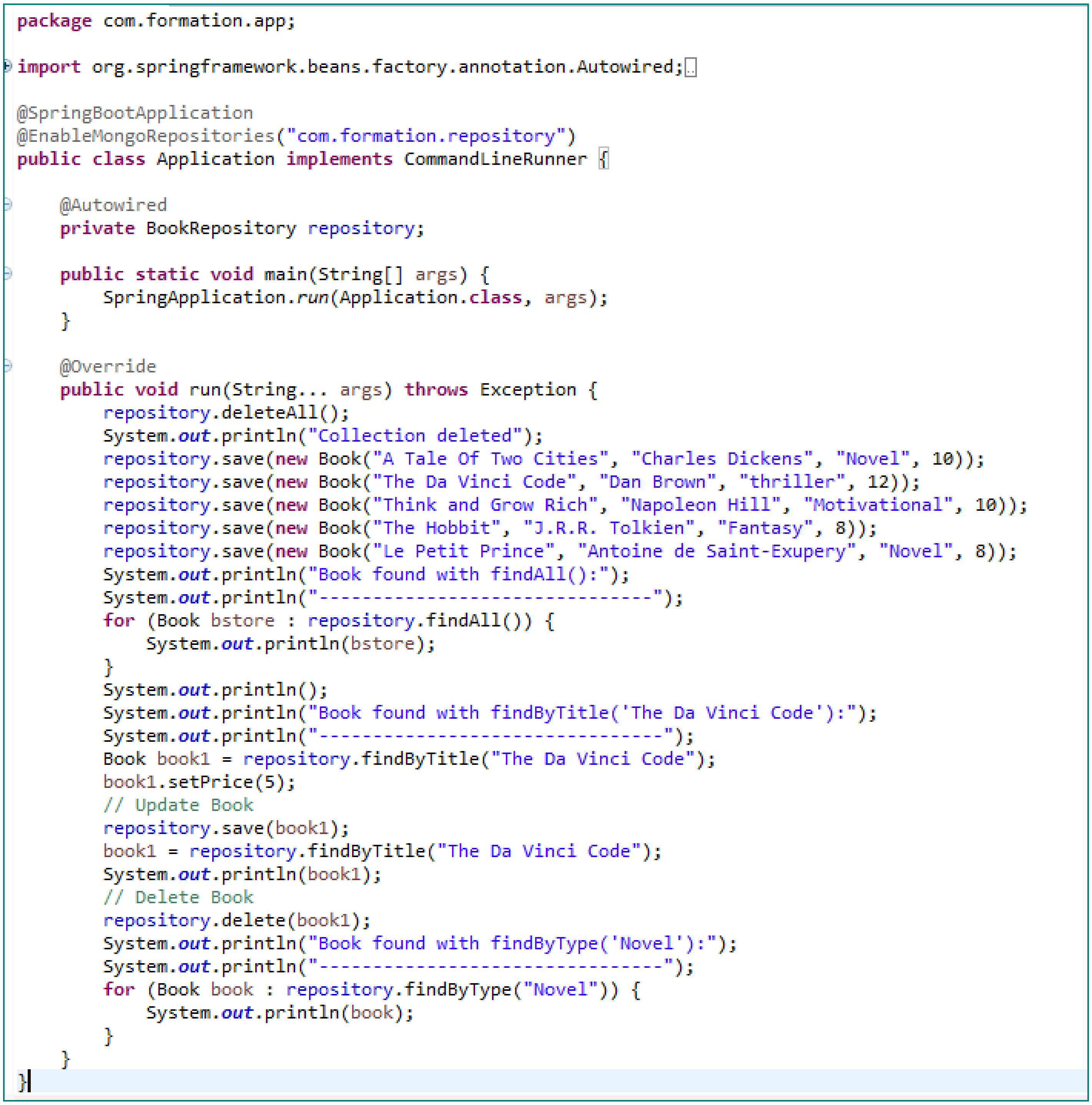
Créez un fichier application.properties dans le répertoire src/main/resources, ajouter le code suivant :



```
spring.data.mongodb.uri=mongodb://localhost:27017/mydatabase
```

Test sans eclipse

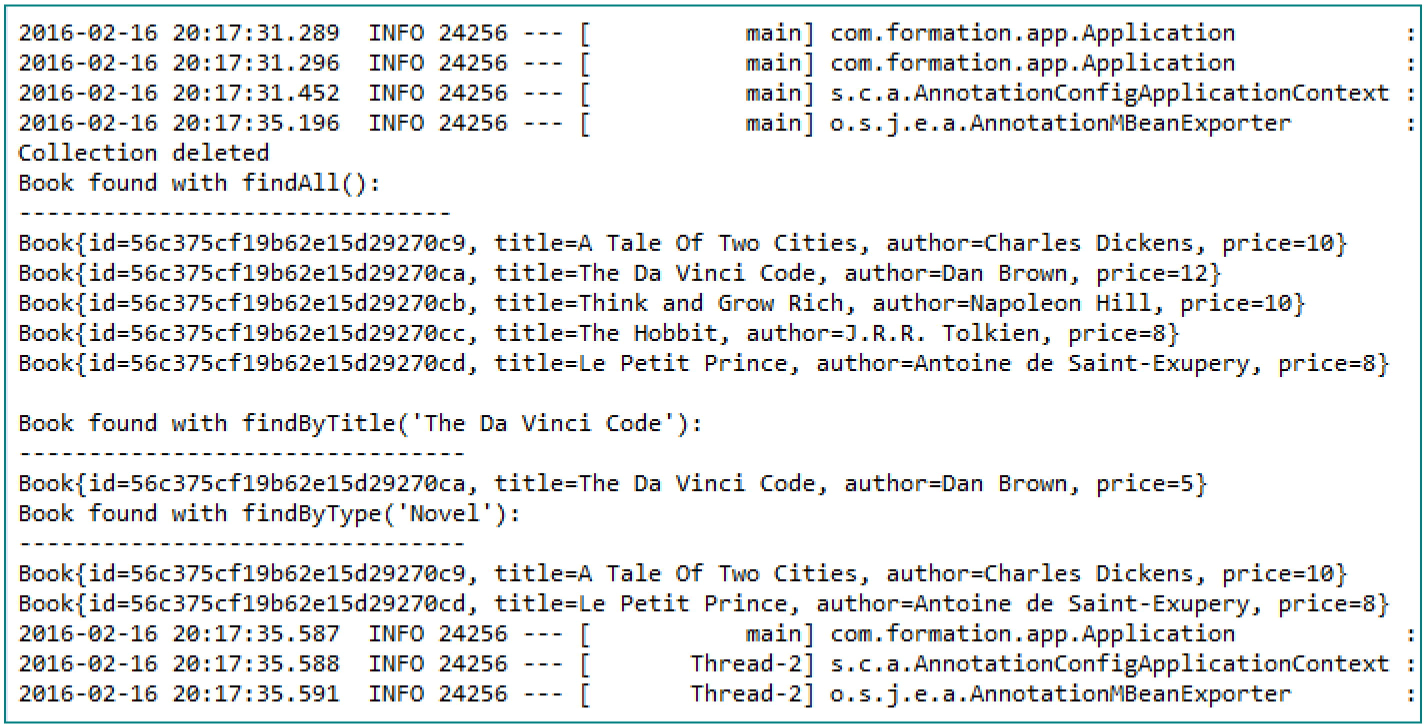
Ecrire une classe pour tester les différentes fonctions. Vérifier à chaque fois avec un client mongo en ligne de commande.



```
package com.formation.app;

import org.springframework.beans.factory.annotation.Autowired;

@SpringBootApplication
@EnableMongoRepositories("com.formation.repository")
public class Application implements CommandLineRunner {

    @Autowired
    private BookRepository repository;

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        repository.deleteAll();
        System.out.println("Collection deleted");
        repository.save(new Book("A Tale Of Two Cities", "Charles Dickens", "Novel", 10));
        repository.save(new Book("The Da Vinci Code", "Dan Brown", "thriller", 12));
        repository.save(new Book("Think and Grow Rich", "Napoleon Hill", "Motivational", 10));
        repository.save(new Book("The Hobbit", "J.R.R. Tolkien", "Fantasy", 8));
        repository.save(new Book("Le Petit Prince", "Antoine de Saint-Exupery", "Novel", 8));
        System.out.println("Book found with findAll():");
        System.out.println("-------------------------------");
        for (Book bstore : repository.findAll()) {
            System.out.println(bstore);
        }
        System.out.println();
        System.out.println("Book found with findByTitle('The Da Vinci Code'):");
        System.out.println("--------------------------------");
        Book book1 = repository.findByTitle("The Da Vinci Code");
        book1.setPrice(5);
        // Update Book
        repository.save(book1);
        book1 = repository.findByTitle("The Da Vinci Code");
        System.out.println(book1);
        // Delete Book
        repository.delete(book1);
        System.out.println("Book found with findByType('Novel'):");
        System.out.println("--------------------------------");
        for (Book book : repository.findByType("Novel")) {
            System.out.println(book);
        }
    }
}
```

Lancer MongoDB

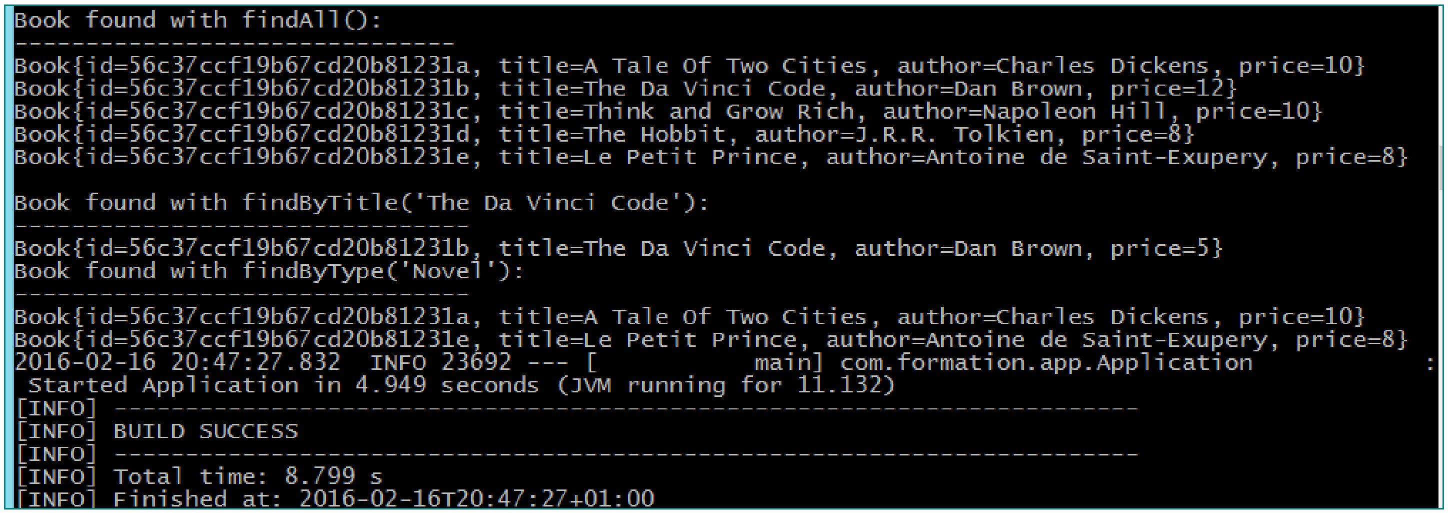
Executer le code



```
2016-02-16 20:17:31.289  INFO 24256 --- [           main] com.formation.app.Application            :
2016-02-16 20:17:31.296  INFO 24256 --- [           main] com.formation.app.Application            :
2016-02-16 20:17:31.452  INFO 24256 --- [           main] s.c.a.AnnotationConfigApplicationContext :
2016-02-16 20:17:35.196  INFO 24256 --- [           main] o.s.j.e.a.AnnotationMBeanExporter        :
Collection deleted
Book found with findAll():
-------------------------------
Book{id=56c375cf19b62e15d29270c9, title=A Tale Of Two Cities, author=Charles Dickens, price=10}
Book{id=56c375cf19b62e15d29270ca, title=The Da Vinci Code, author=Dan Brown, price=12}
Book{id=56c375cf19b62e15d29270cb, title=Think and Grow Rich, author=Napoleon Hill, price=10}
Book{id=56c375cf19b62e15d29270cc, title=The Hobbit, author=J.R.R. Tolkien, price=8}
Book{id=56c375cf19b62e15d29270cd, title=Le Petit Prince, author=Antoine de Saint-Exupery, price=8}

Book found with findByTitle('The Da Vinci Code'):
--------------------------------
Book{id=56c375cf19b62e15d29270ca, title=The Da Vinci Code, author=Dan Brown, price=5}
Book found with findByType('Novel'):
--------------------------------
Book{id=56c375cf19b62e15d29270c9, title=A Tale Of Two Cities, author=Charles Dickens, price=10}
Book{id=56c375cf19b62e15d29270cd, title=Le Petit Prince, author=Antoine de Saint-Exupery, price=8}
2016-02-16 20:17:35.587  INFO 24256 --- [           main] com.formation.app.Application            :
2016-02-16 20:17:35.588  INFO 24256 --- [       Thread-2] s.c.a.AnnotationConfigApplicationContext :
2016-02-16 20:17:35.591  INFO 24256 --- [       Thread-2] o.s.j.e.a.AnnotationMBeanExporter        :
```

Test en ligne de commande

```
mvn spring-boot:run
```

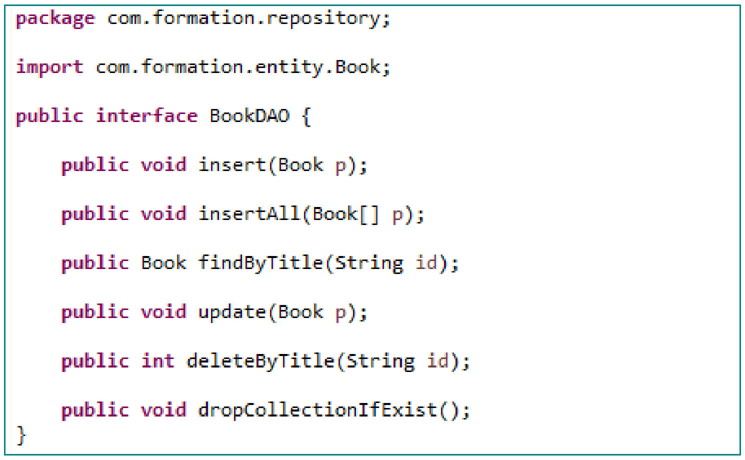


```
Book found with findAll():
-------------------------------
Book{id=56c37ccf19b67cd20b81231a, title=A Tale Of Two Cities, author=Charles Dickens, price=10}
Book{id=56c37ccf19b67cd20b81231b, title=The Da Vinci Code, author=Dan Brown, price=12}
Book{id=56c37ccf19b67cd20b81231c, title=Think and Grow Rich, author=Napoleon Hill, price=10}
Book{id=56c37ccf19b67cd20b81231d, title=The Hobbit, author=J.R.R. Tolkien, price=8}
Book{id=56c37ccf19b67cd20b81231e, title=Le Petit Prince, author=Antoine de Saint-Exupery, price=8}

Book found with findByTitle('The Da Vinci Code'):
--------------------------------
Book{id=56c37ccf19b67cd20b81231b, title=The Da Vinci Code, author=Dan Brown, price=5}
Book found with findByType('Novel'):
--------------------------------
Book{id=56c37ccf19b67cd20b81231a, title=A Tale Of Two Cities, author=Charles Dickens, price=10}
Book{id=56c37ccf19b67cd20b81231e, title=Le Petit Prince, author=Antoine de Saint-Exupery, price=8}
2016-02-16 20:47:27.832  INFO 23692 --- [           main] com.formation.app.Application            :
 Started Application in 4.949 seconds (JVM running for 11.132)
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 8.799 s
[INFO] Finished at: 2016-02-16T20:47:27+01:00
```

Spring Data MongoTemplate

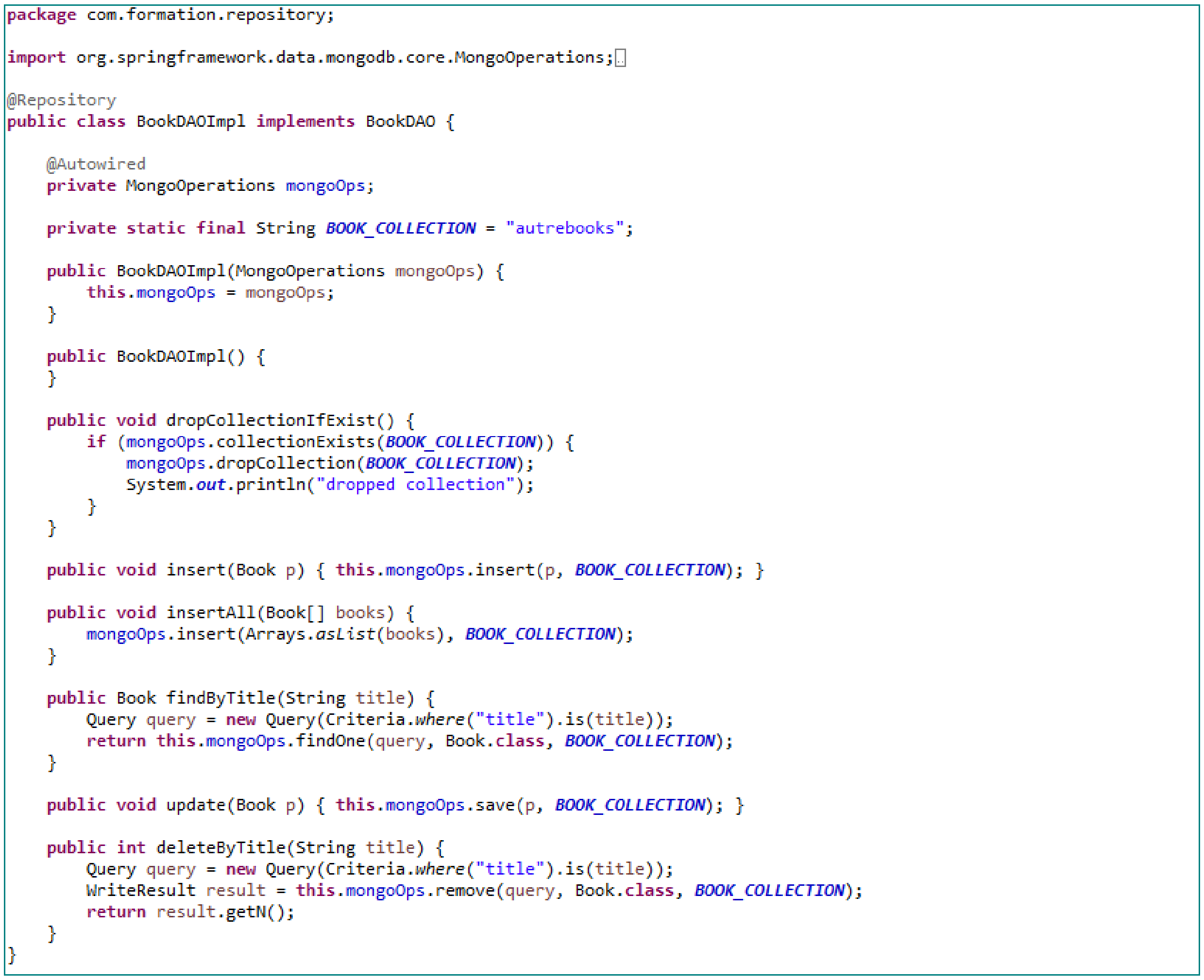
Création des DAOs

BookDAO.java



```java
package com.formation.repository;

import com.formation.entity.Book;

public interface BookDAO {

    public void insert(Book p);

    public void insertAll(Book[] p);

    public Book findByTitle(String id);

    public void update(Book p);

    public int deleteByTitle(String id);

    public void dropCollectionIfExist();
}
```

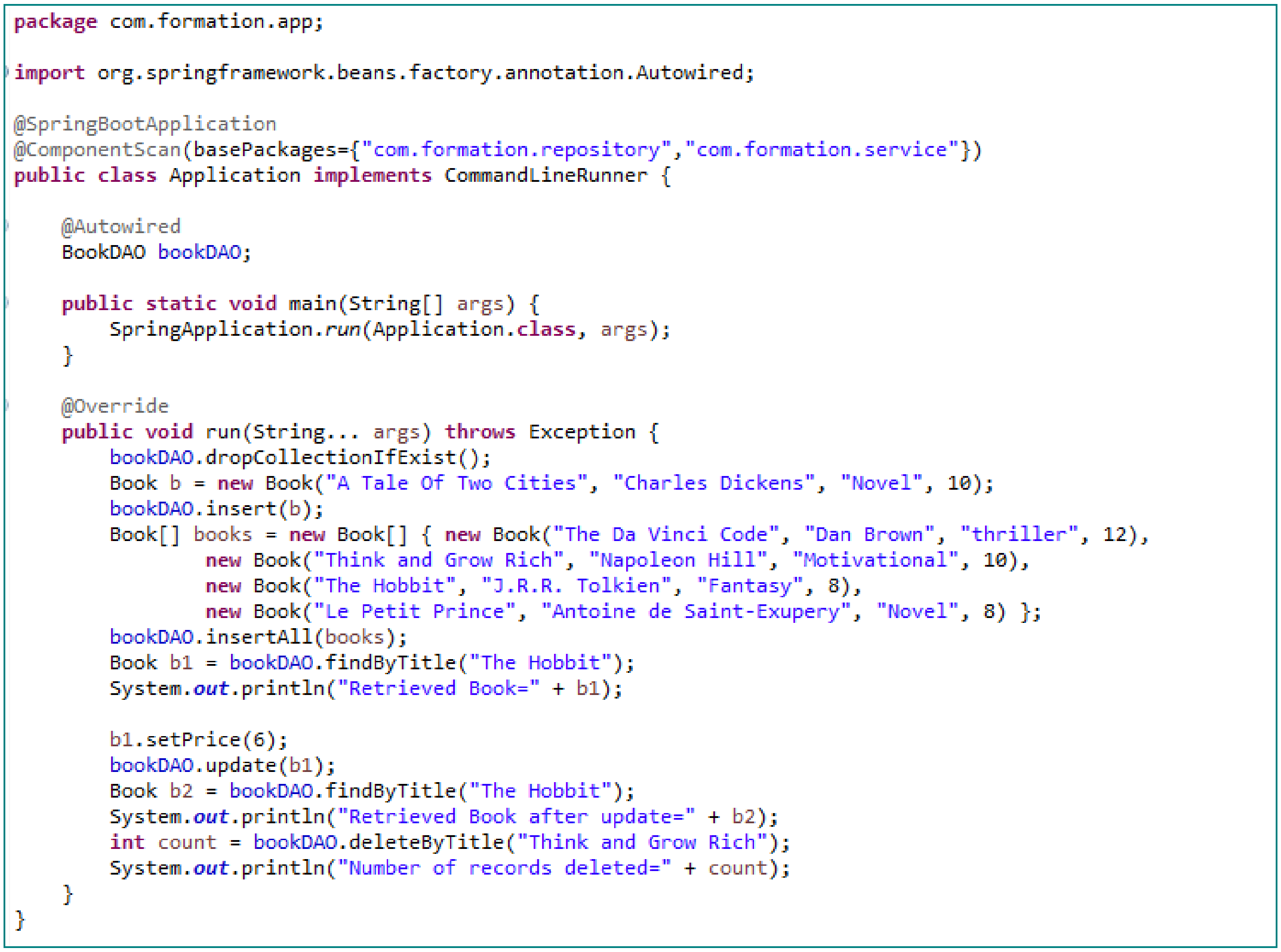
BookDAOImpl.java



```java
package com.formation.repository;

import org.springframework.data.mongodb.core.MongoOperations;

@Repository
public class BookDAOImpl implements BookDAO {

    @Autowired
    private MongoOperations mongoOps;

    private static final String BOOK_COLLECTION = "autrebooks";

    public BookDAOImpl(MongoOperations mongoOps) {
        this.mongoOps = mongoOps;
    }

    public BookDAOImpl() {
    }

    public void dropCollectionIfExist() {
        if (mongoOps.collectionExists(BOOK_COLLECTION)) {
            mongoOps.dropCollection(BOOK_COLLECTION);
            System.out.println("dropped collection");
        }
    }

    public void insert(Book p) { this.mongoOps.insert(p, BOOK_COLLECTION); }

    public void insertAll(Book[] books) {
        mongoOps.insert(Arrays.asList(books), BOOK_COLLECTION);
    }

    public Book findByTitle(String title) {
        Query query = new Query(Criteria.where("title").is(title));
        return this.mongoOps.findOne(query, Book.class, BOOK_COLLECTION);
    }

    public void update(Book p) { this.mongoOps.save(p, BOOK_COLLECTION); }

    public int deleteByTitle(String title) {
        Query query = new Query(Criteria.where("title").is(title));
        WriteResult result = this.mongoOps.remove(query, Book.class, BOOK_COLLECTION);
        return result.getN();
    }
}
```

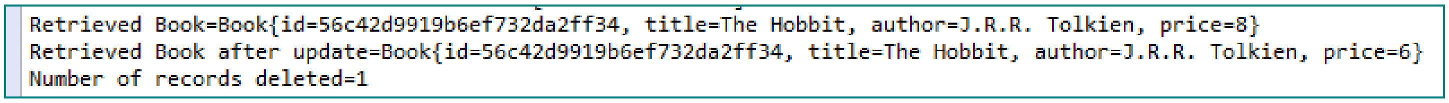
Test

Transformer Application.java comme suit :



```java
package com.formation.app;

import org.springframework.beans.factory.annotation.Autowired;

@SpringBootApplication
@ComponentScan(basePackages={"com.formation.repository","com.formation.service"})
public class Application implements CommandLineRunner {

    @Autowired
    BookDAO bookDAO;

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        bookDAO.dropCollectionIfExist();
        Book b = new Book("A Tale Of Two Cities", "Charles Dickens", "Novel", 10);
        bookDAO.insert(b);
        Book[] books = new Book[] { new Book("The Da Vinci Code", "Dan Brown", "thriller", 12),
                new Book("Think and Grow Rich", "Napoleon Hill", "Motivational", 10),
                new Book("The Hobbit", "J.R.R. Tolkien", "Fantasy", 8),
                new Book("Le Petit Prince", "Antoine de Saint-Exupery", "Novel", 8) };
        bookDAO.insertAll(books);
        Book b1 = bookDAO.findByTitle("The Hobbit");
        System.out.println("Retrieved Book=" + b1);

        b1.setPrice(6);
        bookDAO.update(b1);
        Book b2 = bookDAO.findByTitle("The Hobbit");
        System.out.println("Retrieved Book after update=" + b2);
        int count = bookDAO.deleteByTitle("Think and Grow Rich");
        System.out.println("Number of records deleted=" + count);
    }
}
```

Sortie



```
Retrieved Book=Book{id=56c42d9919b6ef732da2ff34, title=The Hobbit, author=J.R.R. Tolkien, price=8}
Retrieved Book after update=Book{id=56c42d9919b6ef732da2ff34, title=The Hobbit, author=J.R.R. Tolkien, price=6}
Number of records deleted=1
```

SpringData-MongoTemplate + SpringBootRest

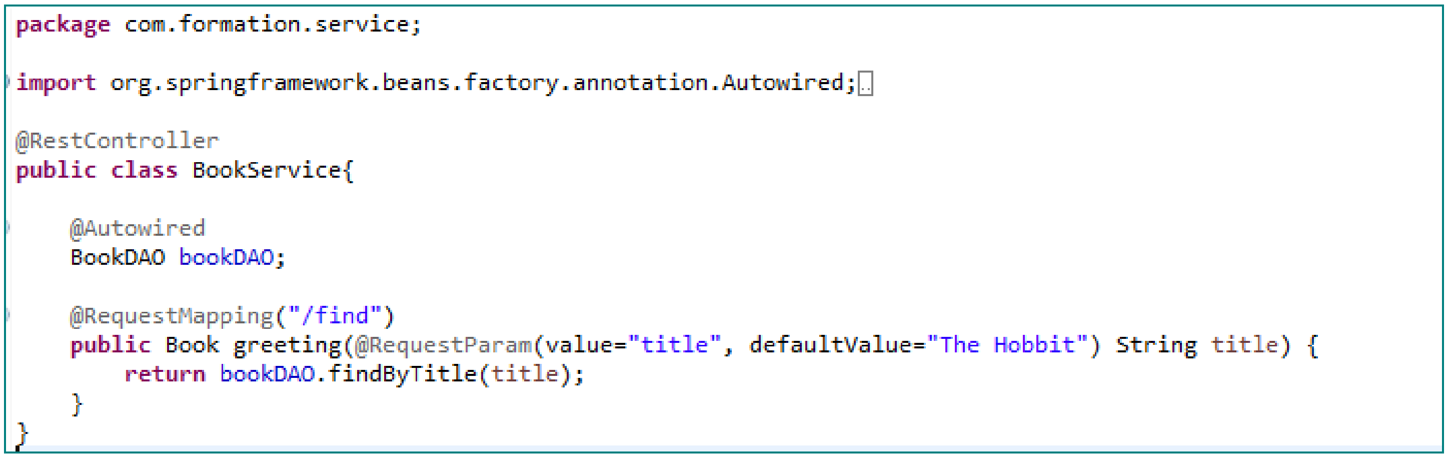
Configuration du fichier pom.xml

Editer le fichier pom.xml et ajouter la dépendance ci-dessous :

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

Création du service Rest

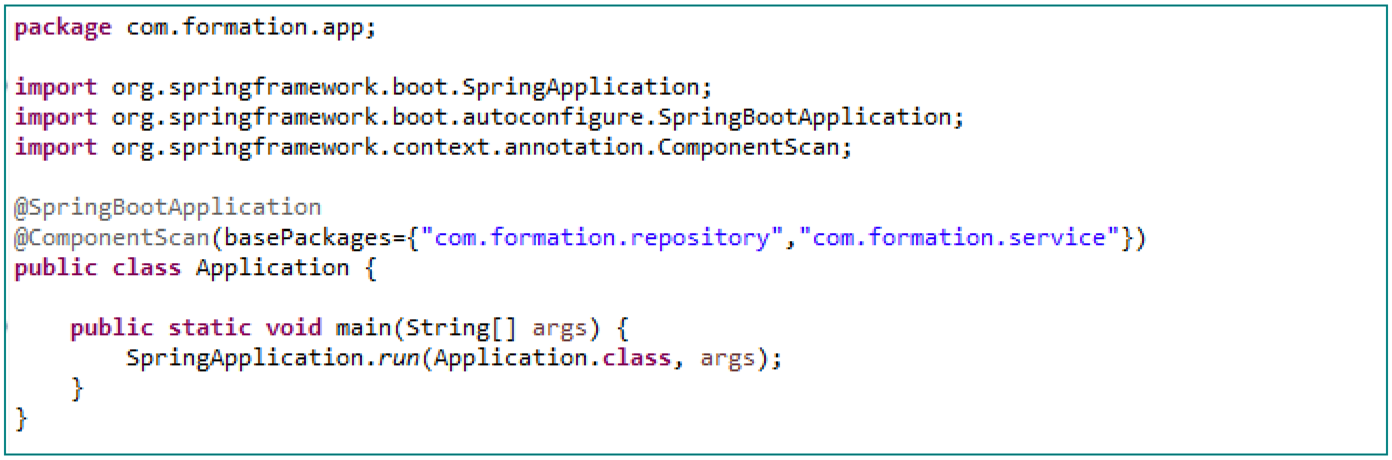
Créez la classe suivante :



```
package com.formation.service;

import org.springframework.beans.factory.annotation.Autowired;

@RestController
public class BookService{

    @Autowired
    BookDAO bookDAO;

    @RequestMapping("/find")
    public Book greeting(@RequestParam(value="title", defaultValue="The Hobbit") String title) {
        return bookDAO.findByTitle(title);
    }
}
```

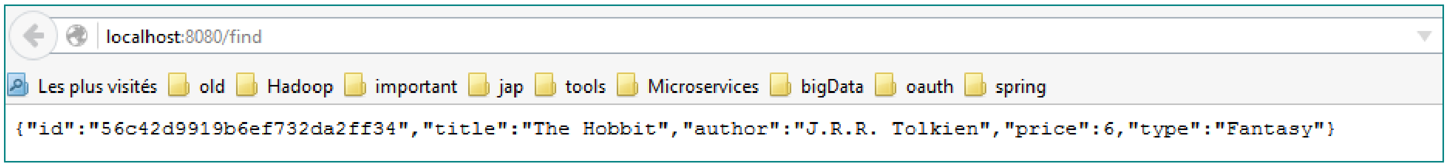
Test

Transformer la classe Application.java comme suit :



```
package com.formation.app;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication
@ComponentScan(basePackages={"com.formation.repository","com.formation.service"})
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

Lancer l'application.
Ouvrir un navigateur et tapez l'url : http://localhost:8080/find



```
{"id":"56c42d9919b6ef732da2ff34","title":"The Hobbit","author":"J.R.R. Tolkien","price":6,"type":"Fantasy"}
```

Deploiement externe sous forme de war

Configuration du fichier pom.xml

Editer le fichier pom.xml et transformer/ajouter comme suit :

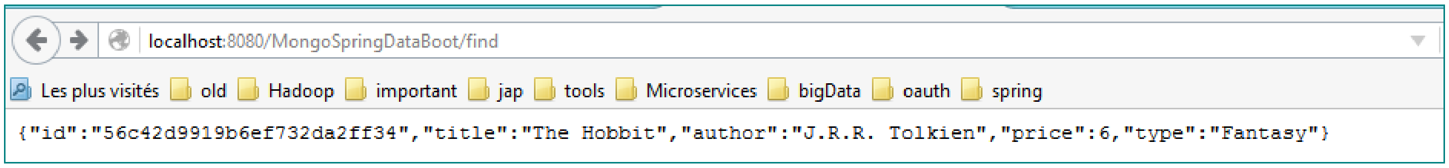
```
…
<packaging>war</packaging>
…
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-tomcat</artifactId>
        <scope>provided</scope>
</dependency>
…
```

Modification de la classe Application.java

Editer et transformer comme suit :

```
package com.formation.app;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.boot.context.web.SpringBootServletInitializer;
import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication
@ComponentScan(basePackages={"com.formation.repository","com.formation.service"})
public class Application extends SpringBootServletInitializer  {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
        return application.sources(Application.class);
    }
}
```

Exporter le projet en war Eclipse ➔ Export ➔ war …
Déployer dans Tomcat. Lancer Tomcat.
Ouvrir un navigateur et tapez l'url : http://localhost:8080/MongoSpringDataBoot/find



```
{"id":"56c42d9919b6ef732da2ff34","title":"The Hobbit","author":"J.R.R. Tolkien","price":6,"type":"Fantasy"}
```