spring

一. 简介

spring是jave EE开发的必备技能

降低开发、框架耦合

spring版本到达5.0,spring framework是最基础的配置

1.1 IOC

使用对象时，由主动new来交给spring进行处理

对象的创建和控制权交给外部，叫做控制反转

spring提供容器，成为IOC容器

在容器中的对象统称为bean

1.2 DI 依赖注入

在容器中建立bean与bean之间依赖关系的整个过程称为依赖注入

1.3 IOC入门

1. 创建maven工程

2. pom文件导入相关依赖

```xml
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.3.23</version>
</dependency>

<!--测试工具包-->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13.2</version>
    <scope>test</scope>
</dependency>
```

3. 在resources目录下创建applicationContext.xml文件

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

</beans>
```

4. 创建主要的类

```java
public interface BookMapper {
    public void save();
}
```

```java
public class BookMapperImpl implements BookMapper {
    @Override
    public void save() {
        System.out.println("bookMapper");
    }
}
```

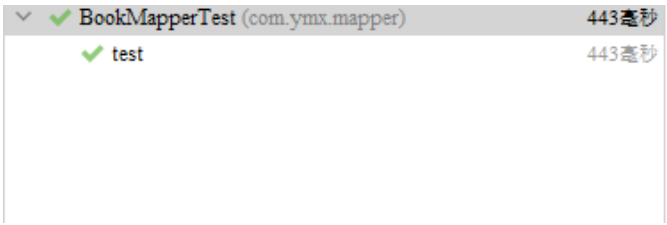
5. 配置bean

```xml
<bean id="bookMapper" class="com.ymx.mapper.impl.BookMapperImpl"/>
```

5. 测试

```
@Test
public void test(){
    ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");
    BookMapper mapper = (BookMapper) context.getBean("bookMapper");
    mapper.save();
}
```

6. 测试结果




```
"D:\jdk 17\bin\java.exe" ...
bookMapper

进程已结束,退出代码0
```

1.4 DI入门

1. 创建主要的类

```
public interface BookService {
    public void save();
}
```

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class BookServiceImpl implements BookService {

    private BookMapper bookMapper;
    @Override
    public void save() {
        System.out.println("bookService");
        bookMapper.save();
    }
}
```

2. 配置bean

```
<bean id="bookService" class="com.ymx.service.impl.BookServiceImpl">
    <property name="bookMapper" ref="bookMapper"/>
</bean>
```

3. 测试

```
@Test
public void test(){
    ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");
    BookService service = (BookService) context.getBean("bookService");
    service.save();
}
```

4. 结果

1.5 分析测试代码

获取ioc容器

```
ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");
```

获取bean

```
BookMapper mapper = (BookMapper) context.getBean("mapper");
```

执行

```
mapper.save();
```

二. bean

2.1 配置

```
<!--
    bean标签标识配置bean
    id属性表示为名字
    class属性表示定义类型-->
    <bean id="bookMapper" class="com.ymx.mapper.impl.BookMapperImpl"/>
```

2.1.1 别名

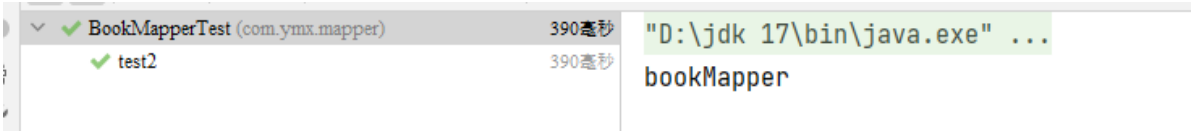
```
<bean id="bookMapper" class="com.ymx.mapper.impl.BookMapperImpl" name="mapper
mapper2 mapper3"/>
```

通过name属性进行赋值，可以赋值多个，用, ; 和空格都可以分割

1. 测试

```
@Test
public void test2(){
    // 测试别名
    ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");
    BookMapper mapper = (BookMapper) context.getBean("mapper");
    mapper.save();
}
```

2. 结果



2.1.2 作用范围配置

spring创建的bean默认为单例模式，简单来说就是创建的对象是否为同一个

1. 测试

```
@Test
public void test3(){
    // 测试别名
    ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");
    BookMapper mapper = (BookMapper) context.getBean("mapper");
    BookMapper mapper2 = (BookMapper) context.getBean("mapper");
    System.out.println(mapper);
    System.out.println(mapper2);
}
```

2. 结果

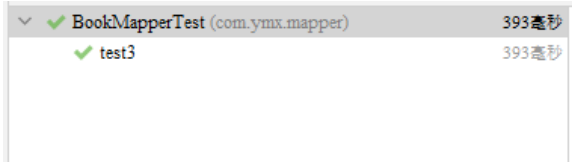

```
"D:\jdk 17\bin\java.exe" ...
com.ymx.mapper.impl.BookMapperImpl@a2431d0
com.ymx.mapper.impl.BookMapperImpl@a2431d0
```

3. 配置xml文件

```
<bean id="bookMapper" class="com.ymx.mapper.impl.BookMapperImpl" name="mapper
mapper2 mapper3" scope="prototype"/>
```

默认scope为singleton单例模式

4. 重新测试




```
"D:\jdk 17\bin\java.exe" ...
com.ymx.mapper.impl.BookMapperImpl@a2431d0
com.ymx.mapper.impl.BookMapperImpl@1cbb87f3
```

spring适合创建需要重复使用的对象

2.2 实例化

spring构造默认使用的方法

添加一段无参的构造方法

```
public class BookMapperImpl implements BookMapper {
    public BookMapperImpl() {
        System.out.println("book mapper constructor is running");
    }
    @Override
    public void save() {
        System.out.println("bookMapper");
    }
}
```

输出

发现会输出book mapper constructor is running

无需修改xml文件

2.3 生命周期

bean从创建到消亡的完整过程

控制bean的生命周期

2.3.1 第一种方法

1. 添加功能

```
public void init(){
    System.out.println("init");
}
public void destory(){
    System.out.println("destory");
}
```

2. xml配置

```
<bean id="bookMapper" class="com.ymx.mapper.impl.BookMapperImpl" name="mapper
mapper2 mapper3" scope="prototype" init-method="init" destroy-method="destory"/>
```

3. 测试

```
@Test
public void test4(){
    ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");
    context.registerShutdownHook();
    BookMapper mapper = (BookMapper) context.getBean("bookMapper1");
    mapper.save();
}
```

2.3.2 第二种方法

```
public class BookMapperImpl3 implements BookMapper , InitializingBean,
DisposableBean {

    @Override
    public void save() {
        System.out.println("bookMapper");
    }

    @Override
    public void destroy() throws Exception {
        System.out.println("s destroy");
    }

    @Override
    public void afterPropertiesSet() throws Exception {
        System.out.println("s init");
    }
}
```

通过这种方式不运行也可以执行