0712技术整理-Spring

> title: 0712技术整理
>
> author：Bucket
>
> content：Spring

Spring配置

```
<bean id="" class="" scope="" init-method="" destroy-method="" factory-bean = ""
factory-method=""></bean>
```

依赖注入

- 构造方法

```
<bean id="userService" class="com.xxx.service.impl.UserServiceImpl">
 <constructor-arg name="userDao" ref="userDao"/>
</bean>
```

- set方法

```
<bean id="userService" class="com.xxx.service.impl.UserServiceImpl"> <property
name="userDao" ref="userDao"/>
</bean>
```

- P命名空间注入

```
<bean id="userService" class="com.xxx.service.impl.UserServiceImpl" p:userDao-
ref="userDao"/>
```

首先需要引入P命名空间

```
xmlns:p="http://www.springframework.org/schema/p"
```

- 注入数据类型
  - 普通数据类型

```
<bean id="user" class="com.xxx.domain.User"> <property name="username"
value="jack"/> <property name="age" value="18"/> </bean>
```

username和age是User类中的两个参数。

  - 集合数据类型

```
<bean id="userDao" class="com.xxx.dao.impl.UserDaoImpl"> <property name="list">
<list><value>aaa</value> <ref bean="user"></ref> </list> </property> </bean>
```

在list中插入一个user对象。

DbUtils

```
QueryRunner queryRunner = new QueryRunner(DataSource dataSource);
int update(); 执行增、删、改语句
T query(); 执行查询语句
ResultSetHandler<T> 这是一个接口，主要作用是将数据库返回的记录封装到实体对象
```

Spring注解开发

使用注解简化配置

在applicationContext.xml中配置组件扫描，作用是指定哪个包及其子包

下的Bean需要进行扫描以便识别使用注解配置的类、字段和方法。

- bean实例化

```
<bean id="userDao" class="com.lagou.dao.impl.UserDaoImpl"></bean>
```

使用@Compont或@Repository标识UserDaoImpl需要Spring进行实例化。

```
// @Component(value = "userDao") @Repository // 如果没有写value属性值，Bean的id为：类名首字母小写
public class UserDaoImpl implements UserDao { }
```

- bean实例化

  使用@Compont或@Repository标识UserDaoImpl需要Spring进行实例化。
- 属性依赖注入

  使用@Autowired或者@Autowired+@Qulifier或者@Resource进行userDao的注入
- @Value

  使用@Value进行字符串的注入，结合SPEL表达式获得配置参数
- @Scope

  使用@Scope标注Bean的范围
- Bean生命周期

  使用@PostConstruct标注初始化方法，使用@PreDestroy标注销毁方法

注解替代配置：

* 非自定义的Bean的配置：<bean>

* 加载properties文件的配置：<context:property-placeholder>

* 组件扫描的配置：<context:component-scan>

* 引入其他文件：<import>

注解 说明

@Configuration 用于指定当前类是一个Spring 配置类，当创建容器时会从该类上加载注解

@Bean 使用在方法上，标注将该方法的返回值存储到 Spring 容器中

@PropertySource 用于加载 properties 文件中的配置

@ComponentScan 用于指定 Spring 在初始化容器时要扫描的包

@Import 用于导入其他配置类

纯注解整合DbUtils

- spring核心配置类
- 数据库配置信息类
- 测试代码

Spring整合Junit

1. 导入spring集成Junit的坐标
2. 使用@Runwith注解替换原来的运行器
3. 使用@ContextConfiguration指定配置文件或配置类
4. 使用@Autowired注入需要测试的对象
5. 创建测试方法进行测试

```
@RunWith(SpringJUnit4ClassRunner.class) @ContextConfiguration(classes =
{SpringConfig.class}) public class SpringJunitTest {
  @Autowired
  private AccountService accountService; //测试查询

                                        @Test

public void testFindById() {
                                                              Account account =
accountService.findById(3);

System.out.println(account);

                                        }
}
```

Spring&JdbcTemplate

JdbcTemplate

对象创建与基本操作

```
JdbcTemplate jdbcTemplate = new JdbcTemplate(DataSource dataSource);
int update(); 执行增、删、改语句
List<T> query(); 查询多个
T queryForObject(); 查询一个
new BeanPropertyRowMapper<>(); 实现ORM映射封装
```

Spring整合JdbcTemplate

1. 创建java项目，导入坐标

   编写pom文件

2. 编写操作的对象(Account)实体类

   查询数据所对应的实体，包含必要的数据定义

3. 编写AccountDao接口和实现类

   对于数据库操作的接口和实现方法

4. 编写AccountService接口和实现类

   向外提供服务端功能调用接口

5. 编写spring核心配置文件

6. 编写测试代码

   面向功能的单个测试