13_先看看 Spring IOC 依赖查找的方式有哪些？

1、开篇

上周介绍了互联网教育系统的项目环境，在正式开始编码之前我们需要补充一些 Spring IOC 和 Spring AOP 的知识。包括 Spring IOC 依赖查询方式，Spring IOC 依赖注入方式，Spring IOC 依赖来源，JDK 动态代理和 CGLib 动态代理以及一些常用的语法特性。今天主要介绍 Spring IOC 依赖查找，有如下内容：

- Spring IOC 依赖查找
- 根据 Bean 名称查找
- 根据 Bean 类型查找

2、Spring IOC 依赖查找

众所周知 Spring IOC 是用来创建和管理 Bean 实例的，Spring IOC 容器针对 Bean 创建实例以后，会被其他类使用。当一个实例调用另外一个实例的时候，就会对调用的实例产生依赖，虽然这个依赖的实例由 Spring IOC 容器创建并管理，但如果要使用这个 Bean 实例需要通过某种途径，其中一种就成为依赖查询。依赖查询的种类很多，这里介绍最简单的两种查找方式：根据 Bean 名称查询、根据 Bean 类型查询。

3、根据 Bean 名称查找

首先，我们创建一个类并且通过配置文件对其进行实例化。

```java
public class User {
    private Long id;
    private String name;

    public Long getId(){
        return id;
    }
    public String getName(){
        return name;
    }
    public void setName(String name){
        this.name = name;
    }
    @Override
    public String toString() {
        return super.toString();
    }
}
```

图 1 创建 User 类

如图 1 所示，创建一个 User 类包括 id 和 name 字段，分别对其设置 get 和 set 方法，同时设置 toString 方法打印实例化以后的 id 和 name 的信息。

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

    <bean id="user" class="org.spring.ioc.overview.dependency.domain.User">
        <property name="id" value="1"/>
        <property name="name" value="小明"/>
    </bean>

</beans>
```

图 2 定义 Bean 并且对其实例化

如图 2 所示，在 resources 目录下面建立 META-INF/dependency-lookup.xml 配置文件，在 bean 节点中设置 bean 实例对应的 id 为 user，同时在 class 属性中定义 class 所在的 namespace。之后在 property 节点中通过对 name，value 赋值来实例化 user 实例。

```java
public class DependencyLookup {
    public static void main(String[] args) {
        BeanFactory beanFactory = new ClassPathXmlApplicationContext("classpath:/META-INF/dependency-lookup.xml");
        lookupRealtime(beanFactory);
    }

    private static void lookupRealtime(BeanFactory beanFactory) {
        User user = (User) beanFactory.getBean("user");
        System.out.println(user);
    }
}
```

图 3 实时查找

如图 3 所示，创建 DependencyLookup 类，添加 main 方法，通过 new ClassPathXmlApplicationContext，并且制定实例化 bean 的 xml 文件路径，classpath:/META-INF/dependency-lookup.xml。也就是刚才实例化 user bean 的 xml 文件路径。随后调用 lookupRealTime 方法，传入 BeanFactory 对象，利用 BeanFactory 中的 getBean 方法，找到 xml 文件中配置的 bean id 为 user 的实例。最终通过 println 方法打印实例中的内容。

4、根据 Bean 类型查找

说完了根据 Bean 名字查找，再来看看根据 Bean 类型查找，顾名思义就是根据 Bean 实例的类型进行查找，其实现也很简单。

```
/**
 * 根据类型查找单个bean
 * @param beanFactory
 */
private static void lookupSingleByType(BeanFactory beanFactory){

    User user2 = beanFactory.getBean(User.class);
    System.out.println("根据类型时实查找单一" + user2.toString());
}
/**
 * 查找多个bean
 */
private static void lookupCollectionByType (BeanFactory beanFactory){
    if (beanFactory instanceof ListableBeanFactory) {
        ListableBeanFactory listableBeanFactory = (ListableBeanFactory) beanFactory;
        Map<String, User> users = listableBeanFactory.getBeansOfType(User.class);
        System.out.println("查找多个bean" + users);
    }
}
```

图 4 根据 Bean 类型查找

如图 4 所示，依旧是在 main 函数中增加了两个方法分别是 lookupSingleByType 和 lookupCollectionByType。lookupSingleByType 方法是根据类型查找单一实例，依旧是使用 BeanFactory 中的 getBean 方法传入的参数与前面不同，传入的是 User.class 也就是 User 的类型。此时通过 println 方法也可以打印 User 实例的内容，说明能够找到 User 实例。lookupCollectionByType 方法实现了查找多个 Bean 实例，它将传入的 BeanFactory 转化成 ListableBeanFactory，使用其包含的 getbeansOfType 方法，传入 User.class 获得包含 User 实例的列表，并且打印列表中所有实例的内容。

5、总结

本节课介绍了 Spring IOC 依赖查找的定义，以及其包含的两种最为常用的查找 Bean 实例的方式：根据 Bean 名称查找和根据 Bean 类型查找。下节课，会围绕 Spring IOC 依赖注入方式给大家做讲解，下期见，拜拜。