17_再看看 Spring IOC 依赖注入有哪些方式？

1、开篇

上节课介绍 Spring IOC 依赖查找的两种简单方式，分别是根据 Bean 名字查找和根据 Bean 类型进行查找。今天继续 Spring IOC 的介绍，看看 Spring IOC 如何进行依赖注入的，包括以下内容：

- Spring IOC 依赖注入
- 根据 Bean 名称注入
- 根据 Bean 类型注入

2、Spring IOC 依赖注入

Spring IOC 依赖注入也称为自动装载，当两个类存在依赖关系，通过配置<bean>元素的 autowire 属性自动装配 <bean> 标签的方式获取依赖类的实例。这里介绍两种常见的注入方式：

根据 Bean 名称注入，使用 byName 标签，此选项将检查容器并根据名字查找与属性完全一致的 Bean，并将其与属性自动装配。

根据 Bean 类型注入，使用 byType 标签。如果容器中存在一个与制定属性类型相同的 Bean，那么将与该属性自动装配；如果存在多个该类型 Bean，那么抛出异常，并指出不能使用 byType 方式进行自动状态；如果没有找到相匹配的 Bean，则什么事情都不发生。

上面介绍的概念有点抽象，下面通过代码示例给大家介绍。

3、根据 Bean 名称注入

依旧沿用上节课使用的 User 类作为例子。

```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 User 类

如图 1 所示，User 类包含 id 和 name 两个字段，并且包含对应的 get 和 set 方法。

```java
public class TestUtil {

    private User user;
    public User getUser() {
        return user;
    }
    public void setUser(User user) {
        this.user = user;
    }
}
```

图 2 TestUtil 中依赖 User 类

如图 2 所示，TestUtil 类中依赖 User 类，其中定义了 User 的 get 和 set 方法，由于在 TestUtil 中直接使用 User 的实例，因此需要对 User 进行实例化，这个实例化的工作是交给 Spring IOC 容器实现的。因此需要告诉 Spring IOC 容器 TestUtil 与 User 的依赖关系，并且在 TestUtil 实例化的时候自动装配 User 的实例，这样才能完成它们之间的依赖注入。

```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>

    <bean id="testUtil" class="org.spring.ioc.overview.dependency.domain.TestUtil"  autowire="byName"/>


</beans>
```

图 3 配置文件更新

如图 3 所示，我们将上节课用到的配置文件进行更新，加入一个新的 bean 节点，id 为“testUtil”并且使用了对应的 class namesapce 为“org.spring.ioc.overview.dependency.domain.TestUtil”，需要注意的是这里引入了“autowire”属性并且设置为“byName”，意思是根据 Bean 名称进行依赖注入。这样当 testUtil 类实例化的时候能够通过这个配置知道去依赖注入 User 的实例了。接下来在看测试代码。

```java
public static void main(String[] args) {
    BeanFactory beanFactory = new ClassPathXmlApplicationContext("classpath:/META-INF/dependency-lookup.xml");
    TestUtil testUtil = (TestUtil) beanFactory.getBean("testUtil");
    User user = testUtil.getUser();
    System.out.println(user);
}
```

图 4 测试代码

如图 4 所示，创建 main 函数依旧使用 ClassPathXMLApplicationContext 制定 xml 文件的地址，这里的 xml 的文件名字没有修改。通过 BeanFactory 中的 getBean 方法获取“testUtil”类的实例，这里使用的是 Spring IOC 依赖查找，也就

是根据 Bean 名称获取对应的实例。随后，根据产生的 testUtil 实例中的 getUser 方法生成 User 的实例，此时就用到了 Spring IOC 的依赖注入，也就是自动装配。注意此时没有使用 BeanFactory 中的 getBean 方法获取 User 的实例，而是将 XML 文件中“testUtil”bean 节点的“autowire”属性并且设置为“byName”，从而实现 testUtil 对于 User 实例的自动装配。在获取 User 实例之后，在通过 println 方法打印实例内容：“User{id=1, name='小明'}”。

4、根据 Bean 类型注入

有了根据 Bean 名称注入的介绍，介绍 Bean 类型注入会简单许多，上面代码基本没有修改，主要回到 XML 配置文件中。

```
<?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>

    <bean id="testUtil" class="org.spring.ioc.overview.dependency.domain.TestUtil"  autowire="byType"/>


</beans>
```

图 5 配置文件修改 byType

如图 5 所示，将原来 bean 节点中 autowire 标签的“byName”修改为“byType”即可。在实例化时 Spring IOC 容器会查找与制定类型相同的 Bean 进行实例化。此例中 testUtil 会去找 User 类型的 Bean 实例。后面的测试过程和 byName 一样这里就不再赘述了。

5、总结

这一节课介绍了 Spring IOC 依赖注入的两种方式，分别是根据 Bean 名称注入和根据 Bean 类型注入。在上节课代码基础上增加了 TestUtil 类，同时修改了 XML

文件，在增加的新 bean 节点中使用了 autowire 属性（byName/byType）完成自动装配的功能。

下节课会介绍 Spring IOC 依赖来源有哪些？，下期见，拜拜。