Spring，day04，主讲：汤小洋

一、注解简介

Spring提供了一系列的注解来替代配置文件，简化配置

实际开发中，建议使用注解+配置的形式

二、IoC注解

1. 扫描包

```
 <!-- 扫描包，可以配置多个 -->
<context:component-scan base-package="ioc"/>
<context:component-scan base-package="com.itany.dao.impl" />
<context:component-scan base-package="com.itany.service.impl" />
<context:component-scan base-package="com.itany.action" />
```

2. 常用注解

2.1 组件的定义

@Component 定义Bean组件，添加到IoC容器中，不区分组件类型

@Repository 表示Dao组件

@Service 表示Service组件

@Controller 表示Action组件

2.2 数据装配

注解方式的数据装配是直接使用属性进行注入，不是使用**setter**方法，所以可以没有**setter**方法

简单类型

```
 @Value("666")
private int num;

@Value("true")
private Boolean flag;

@Value("${jdbc.username}")
private String username;

@Value("java.lang.String")
private Class clazz;

@Value("classpath:ioc/spring.xml")
private Resource resource;
```

```
<!-- 读取属性文件 -->
<!--<bean
class="org.springframework.beans.factory.config.PropertyPlaceholderConfi
gurer">-->
<!--<property name="location" value="classpath:ioc/info.properties"/>-->
<!--</bean>-->
<context:property-placeholder location="classpath:ioc/info.properties"/>
```

其他bean的引用

```
/**
 * 方式1: 使用@Authwired，Spring提供
 *       自动装配，默认按byType，如果有多个同类型的bean，则按byName
 *       结合@Qualifier按指定byName注入
 * 方式2: @Resource，JavaEE提供
 */
//@Autowired
//@Qualifier("ob")
@javax.annotation.Resource
  private OtherBean otherBean;
```

集合的装配

```
 //集合的装配，使用@Resource注解，按byName注入
@javax.annotation.Resource(name="as")
  private Integer[] arrays;
```

```xml
<!-- 集合类型的装配 -->
<util:list id="as">
  <value>1</value>
  <value>2</value>
  <value>3</value>
</util:list>

<util:list id="lists">
  <ref bean="otherBean"/>
  <ref bean="otherBean"/>
  <ref bean="ob"/>
  <bean class="ioc.OtherBean">
    <property name="msg" value="嘿嘿"/>
  </bean>
</util:list>
```

2.3 生命周期

```java
//相当于init-method=""
@PostConstruct
public void init(){
  System.out.println("SpringBean.init");
}

//相当于destroy-method=""
@PreDestroy
public void destroy(){
  System.out.println("SpringBean.destroy");
}
```

2.4 实例化时机

```java
@Lazy
```

2.5 scope作用域

```java
@Scope("prototype")
```

三、AOP注解

1. 配置Advice

定义增强类，添加@Component和@Aspect注解

2. 配置Pointcut并指定通知类型

@Pointcut(切点表达式)

@Before(切点方法())

@AfterReturning

@AfterThrowing

@Around

3. 织入

```
<!--
                自动创建代理并织入切面
                        proxy-target-class，取值：
                                false：使用jdk动态代理，默认值
                                true：使用cglib
-->
<aop:aspectj-autoproxy proxy-target-class="true"/>
```