## Spring Framework

### **4.1 Ioc概述：**

IoC（Inverse of Control，控制反转）是Spring容器的内核，AOP、声明式事务等功能在此基础上开花结果。

#### **4.1.2　IoC的类型：**

从注入方法上看，IoC主要可以划分为3种类型：***构造函数注入、属性注入和接口注入***。Spring支持构造函数注入和属性注入。

### **4.2　相关Java基础知识：**

Java语言允许通过程序化的方式间接对Class进行操作。Class文件由类装载器装载后，在JVM中将形成一份描述Class结构的元信息对象，通过该元信息对象可以获知Class的结构信息，如构造函数、属性和方法等。Java允许用户借由这个与Class相关的元信息对象间接调用Class对象的功能，这就为使用程序化方式操作Class对象开辟了途径。

#### **4.2.1　简单实例（反射）**

*// 1.通过类加载器获取Car类对象* ClassLoader loader = Thread.*currentThread*().getContextClassLoader();  
 Class clazz = loader.loadClass(**"com.smart.jkzr.Car"**);  
*// 2.获取类的默认构造器，并实例化对象* Constructor constructor = clazz.getDeclaredConstructor((Class[]) **null**);  
 Car car = (Car) constructor.newInstance();  
*// 3.通过反射方法设置属性* Method setBrand = clazz.getMethod(**"setBrand"**, String.**class**);  
 setBrand.invoke(car,**"红旗轿车"**);  
 Method setColor = clazz.getMethod(**"setColor"**, String.**class**);  
 setColor.invoke(car,**"red"**);  
 Method setMaxSpeed = clazz.getMethod(**"setMaxSpeed"**, **int**.**class**);  
 setMaxSpeed.invoke(car,200);  
 car.introduce();

#### **4.2.2　类装载器ClassLoader**

类装载器就是寻找类的节码文件并构造出类在JVM内部表示对象的组件。在Java中，类装载器把一个类装入JVM中，需要经过以下步骤：

（1）装载：查找和导入Class文件。

（2）链接：执行校验、准备和解析步骤，其中解析步骤是可以选择的。

① 校验：检查载入Class文件数据的正确性。

② 准备：给类的静态变量分配存储空间。

③ 解析：将符号引用转换成直接引用。

（3）初始化：对类的静态变量、静态代码块执行初始化工作。

类装载工作由ClassLoader及其子类负责。ClassLoader是一个重要的Java运行时系统组件，它负责在运行时查找和装入Class字节码文件。

**实战经验:**

每个类在JVM中都拥有一个对应的java.lang.Class对象，它提供了类结构信息的描述。数组、枚举、注解及基本Java类型（如int、double等），甚至void都拥有对应的Class对象。Class没有public的构造方法。Class对象是在装载类时由JVM通过调用类装载器中的defineClass()方法自动构造的。

#### **4.2.3　Java反射机制**

Class反射对象描述类语义结构，可以从Class对象中获取构造函数、成员变量、方法类等类元素的反射对象，并以编程的方式通过这些反射对象对目标类对象进行操作。这些反射对象类在java.reflect包中定义。下面介绍3个主要的反射类。

在访问private或protected成员变量和方法时，必须通过setAccessible(boolean access)方法取消Java语言检查，否则将抛出IllegalAccessException。如果JVM的安全管理器设置了相应的安全机制，那么调用该方法将抛出SecurityException。

### **4.3　资源访问利器**

#### **4.3.1　资源抽象接口**

实战经验

用Resource操作文件时，如果资源配置文件在项目发布时会被打包到JAR中，那么不能使用Resource#getFile()方法，否则会抛出FileNotFoundException。但可以使用Resource#getInputStream()方法读取。

错误的读取方式：

(new DefaultResourceLoader()).getResource("classpath:conf/sys.properties").getFile()

正确的读取方式：

(new DefaultResourceLoader()).getResource("classpath:conf/sys.properties").getInputStream()

这个问题在实际的项目开发过程中很容易被忽视，因为在项目开发时，资源配置文件一般是在文件夹下的，所以Resource#getFile()是可以正常工作的。但在发布时，如果资源配置文件被打包到JAR中，这时getFile()就无法读取了，从而造成部署实施的时候出现意想不到的问题。因此，我们建议尽量以流的方式读取，避免环境不同造成的问题。

作业：自己搭建一套springmvc+mybatis的框架

## Spring Boot

作业：自己搭建一套springboot+jda的框架

## Spring Cloud

作业：自己搭建一个电商网站