Java反射

获取反射对象方式

- Class.forName(“类的路径”)：当你知道该类的全路径名时，你可以使用该方法获取 Class 类对象　　Class clz = Class.forName("java.lang.String");
- 类名.class。这种方法只适合在编译前就知道操作的 Class　　Class clz = String.class;
- 对象名.getClass()
  ```
  String str = new String("Hello");
  Class clz = str.getClass();
  ```
- 如果是基本类型的包装类，可以调用包装类的Type属性来获得该包装类的Class对象

API类型

Class 类：反射的核心类，可以获取类的属性，方法等信息

- 创建此 Class 对象表示的类的新实例
  - newInstance()　JDK1.8
  - getDeclaredConstructor().newInstance()　JDK17
- 获取对象信息
  - Class<?>[] getClasses()
  - Class<?>[] getDeclaredClasses()
    这些对象反映声明为此 Class 对象所表示的类的成员的所有类和接口。这包括公共、受保护、默认（包）访问以及类声明的私有类和接口，但不包括继承的类和接口。如果类没有将任何类或接口声明为成员，或者如果此 Class 对象表示基元类型、数组类或 void，则此方法返回长度为 0 的数组
- 获取类属性
  - Field[] getFields()
  - Field[] getDeclaredFields()
    这些对象反映由此 Class 对象表示的类或接口声明的所有字段。这包括公共、受保护、默认（包）访问和私有字段，但不包括继承的字段
- 获取类方法
  - Method[] getMethods()
  - Method[] getDeclaredMethods()
- 获取构造函数
  - getConstructors()
  - getDeclaredConstructors()
- 获取类注解对象类型　AnnotatedType getAnnotatedSuperclass()
- 获取对象注解内容属性　Annotation[] getAnnotations()

Field 类：Java.lang.reflec 包中的类，表示类的成员变量，可以用来获取和设置类之中的属性值

- 获取注解的标签属性
  - <T extends Annotation> T getAnnotation(Class<T> annotationClass)
    返回此元素对指定类型的批注（如果 存在此类批注），否则为 null。
    请注意，此方法返回的任何批注都是声明批注。
    抛出：
    NullPointerException –因为：
    1.5
  - Annotation[] getDeclaredAnnotations()
    返回 直接存在于 此元素上的批注。此方法忽略继承的批注。如果此元素上没有 直接存在的 注释，则返回值是长度为 0 的数组。此方法的调用方可以自由修改返回的数组;它对返回给其他调用方的数组没有影响。
    请注意，此方法返回的任何批注都是声明批注。
- 获取注解返回值类型
  - <T extends Annotation> T[] getAnnotationsByType(Class<T> annotationClass)
    返回与此元素 关联的 批注。如果没有与此元素 关联的 注释，则返回值是长度为 0 的数组。此方法与 getAnnotation(Class) 此方法之间的区别在于，此方法检测其参数是否为 可重复的注释类型 （JLS @jls 9.6），如果是，则尝试通过“查看”容器注释来查找该类型的一个或多个注释。此方法的调用方可以自由修改返回的数组;它对返回给其他调用方的数组没有影响。
    请注意，此方法返回的任何批注都是声明批注。
    抛出：
    NullPointerException –
    因为：
    1.8
  - AnnotatedType getAnnotatedType()
    返回一个 AnnotatedType 对象，该对象表示使用类型来指定此字段所表示的字段的声明类型。
    返回：
    一个对象，表示此字段表示的字段的声明类型
    因为：1.8

Method 类：Java.lang.reflec 包中的类，表示类的方法，它可以用来获取类中的方法信息或者执行方法

- 获取注解的标签属性　Annotation[][] getParameterAnnotations()
- 获取注解返回值类型　AnnotatedType getAnnotatedReturnType()
- 执行方法　Object invoke(Object obj, Object... args)

Constructor 类：Java.lang.reflec 包中的类，表示类的构造方法

AccessibleObject类：安全检查类

- 权限注入setAccessible()
- JDK1.8

常用反射场景举例

- JDBC 的数据库的连接
  1. 通过Class.forName()加载数据库的驱动程序 （通过反射加载，前提是引入相关了Jar包）
  2. 通过 DriverManager 类进行数据库的连接，连接的时候要输入数据库的连接地址、用户名、密码
  3. 通过Connection 接口接收连接
- Spring 通过 XML 配置模式装载 Bean 的过程
  1. 将程序内所有 XML 或 Properties 配置文件加载入内存中
  2. Java类里面解析xml或properties里面的内容，得到对应实体类的字节码字符串以及相关的属性信息
  3. 使用反射机制，根据这个字符串获得某个类的Class实例
  4. 动态配置实例的属性

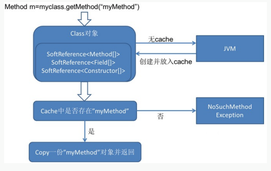
反射机制原理

1.反射获取类实例 Class.forName()，并没有将实现留给了java,而是交给了jvm去加载！主要是先获取 ClassLoader, 然后调用 native 方法，获取信息，加载类则是回调 java.lang.ClassLoader。最后，jvm又会回调 ClassLoader 进类加载

2.newInstance() 主要做了三件事：

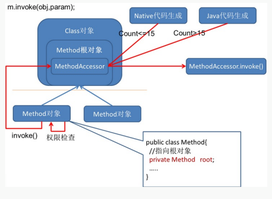
1. 权限检测，如果不通过直接抛出异常；
2. 查找无参构造器，并将其缓存起来；
3. 调用具体方法的无参构造方法，生成实例并返回。

3.获取Method对象



每次getMethod获取到的Method对象都持有对根对象的引用，因为一些重量级的Method的成员变量（主要是MethodAccessor），我们不希望每次创建Method对象都要重新初始化，于是所有代表同一个方法的Method对象都共享着根对象的MethodAccessor，每一次创建都会调用根对象的copy方法复制一份

4.调用invoke()方法



调用Method.invoke之后，会直接去调MethodAccessor.invoke
MethodAccessor就是上面提到的所有同名method共享的一个实例，由ReflectionFactory创建

详细介绍：https://www.cnblogs.com/yougewe/p/10125073.html