## 1反射

JAVA反射机制是在运行状态中，对于任意一个类，都能够知道这个类的所有属性和方法；对于任意一个对象，都能够调用它的任意一个方法；这种动态获取的以及动态调用对象的方法的功能称为Java的反射机制。

三种方式获取class对象：

/第一种方式 通过对象**getClass**方法

Person person = new Person();

Class<?> class1 = person.getClass();

//第二种方式 通过**类的class属性**

class1 = Person.class;

try {

//第三种方式 通过**Class类的静态方法——forName()**来实现

class1 = Class.forName("com.whoislcj.reflectdemo.Person");

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

Field[] allFields = class1.**getDeclaredFields()**;//获取class对象的所有属性

Field[] publicFields = class1.**getFields()**;//获取class对象的public属性

**class对象动态生成**

//第一种方式 Class对象调用newInstance()方法生成

Object obj = class1.newInstance();

//第二种方式 对象获得对应的Constructor对象，再通过该Constructor对象的newInstance()方法生成

Constructor<?> constructor = class1.getDeclaredConstructor(new Class[]{String.class});//获取指定声明构造函数

obj = constructor.newInstance(new Object[]{"lcj"});

应用：

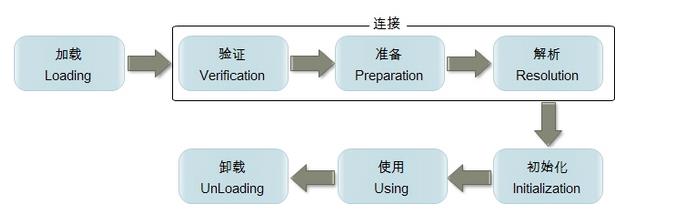
动态代理

Spring的AOP思想

## 2.类加载

Java虚拟机把描述类的数据从Class文件加载到内存，并对数据进行校验、转换解析和初始化，最终形成可以被虚拟机直接使用的Java类型，这就是虚拟机的加载机制。

它的整个生命周期包括了：加载（Loading）、验证（Verification）、准备（Preparation）、解析（Resolution）、初始化（Initialization）、使用（using）、和卸载（Unloading）七个阶段。其中验证、准备和解析三个部分统称为连接（Linking）



三个类加载器：

引导（Bootstrap）类加载器：负责将 <JAVA\_HOME>/lib路径下的核心类库或-Xbootclasspath参数指定的路径下的jar包加载到内存中

扩展（Extension）类加载器：负责加载<JAVA\_HOME>/lib/ext目录下或者由系统变量-Djava.ext.dir指定位路径中的类库

系统（System）类加载器（也称应用类加载器）：负责加载系统类路径java -classpath或-D java.class.path 指定路径下的类库

## 3.Java8新特性（十大）

1. **函数式接口和Lambda表达式**

函数式接口是只包含一个方法的抽象接口，通过@FunctionalInterface注解（非必须），将一个接口标注为函数式接口。函数式接口是为Java 8中的lambda而设计的，lambda表达式的方法体其实就是函数接口的实现。

Lambda表达式语法结构如下：

(parameters) -> expression 或者 (parameters) -> {statements;}

lambda表达式使代码变得简单，代替了匿名内部类

new Thread(new Runnable(){

@Override

public void run() {

for (int i = 0; i < 10; i++)

System.out.println("Without Lambda Expression");

}}).start();

2.**接口的默认方法**（default修饰）：接口里也可以写方法体了，实现该接口的类不再强制实现该方法，只需要在方法签名增加default签名并实现方法体；

接口可以定义**静态方法**，通过接口调用。实现类不需实现，也无法在实现类中直接调用。

3.**方法引用**

构造器引用。语法是Class::new，或者更一般的Class< T >::new，要求构造器方法是没有参数；

静态方法引用。语法是Class::static\_method，要求接受一个Class类型的参数；

特定类的任意对象方法引用。它的语法是Class::method。要求方法是没有参数的；

特定对象的方法引用，它的语法是instance::method。要求方法接受一个参数，与3不同的地方在于，3是在列表元素上分别调用方法，而4是在某个对象上调用方法，将列表元素作为参数传入；

**4.重复注解**