# 反射

类 class

字段，属性

方法，函数

构造函数

动态的获得指定一个类，jvm加载类的时候，自动产生一个对象 Class 指定的对象

当前这个类的静态属性 class

对象中的一个方法.getClass

这个类的类对象

Class.forName 获取当前这个类的类对象。同时他还有动态得到对象功能

Class.forName(你想要实例化的类的名字)

这个名字可以动态传递

配和接口一起使用

修改配置文件即可

接口配和反射机制 实现一个工厂模式

处理字段时候，私有修饰符可以操作，可以设置和获取值

所有的类都继承一个类，Object类

方法的调用

对象.方法() 调用

方法.invoke(对象)唤醒

反射机制

字段 get set

不仅方法调用，还有参数，返回值

java int integer 类型其实是两种不同的类型

但是在转换的时候，可以自动转换，自动装箱和拆箱机制

getmethods 并不会获取构造函数

通过一个特殊的方法来获取

构造函数的反射调用

new 构造函数名字()

通过工厂模式，掌握反射机制的作用

原理 概念

字段 设置和获取

函数 调用 invoke 唤醒

构造函数（实例化对象）

接口目前学了，有啥作用？？

工厂模式

管理对象的，代码中，得到对象，不要自己new 从工厂中获取。

对象换实现了，

实现，结合接口，反射机制，配置文件

升级了

开发模式 提高开发效率，提高团队合作效率，降低维护成本。

增加项目的风险

软件春天的到来。

# 注解

注释 给程序员看的

注解也是注释，是给成员看，程序看到注释会做出相应的处理，也是代码中的一部分，但是又不是像代码一样编写

注解无处不在，作用在类中任何一个部位，表示这些部位的含义

java.lang

java.util

java.text

java.io

java.sql

java.net

java.lang.reflect

java.annotation

java.time

javadoc 生成文档 帮助文档 api

请求映射

<bean name=”/a.do” class=”com.action.UserAction” method=”show”/>

<bean name=”/a2.do” class=”com.action.UserAction” method=”show2”/>

<bean name=”/a3.do” class=”com.action.UserAction” method=”show3”/>

@Component

class UserAction{

@requestmapping(“a.do”)

public void show(){

}

@requestmapping(“a1.do”)

public void show2(){

}

@requestmapping(“a2.do”)

public void show3(){

}

}

提供了2种

元注解

jdk自带的

int String byte

自定义注解

自定义一个类 User Dog 等

包，类，方法，属性，参数，局部变量

target就是定义你在哪个地方使用这个注解

什么时候用到注解

什么阶段

源码阶段

编译阶段 class字节码

运行阶段

手动写了一个简答的orm框架

mybatis框架 使用xml配置文件

注解

写在代码中

@insert(“insert into user(id,name,age) values(#{id},#{name},#age{})”)

独立的配置文件

<insert id=’save’ >

into user(id,name,sex) values(#{id},#{name},#age{}

</insert>

springmvc 中官方也推介使用注解

注解刚出来使用的时候，配置文件。

争议 ，替代xml配置文件

官方重点维护注解版的框架

vue

vue框架

vivio

int String

元注解

自定义的时候 ，使用元注解

什么是注解

2种

怎么去解析注解 反射机制

类 方法 参数