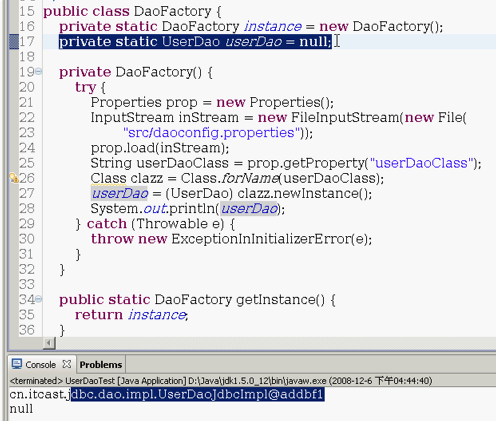
使用DAO工厂模式

1. 利用Java的反射机制：例子可以参看java中的反射机制笔记里的例子。MainBoard分别操作USB、Video、Camera等设备，通过manifest.Properties文件存放键值对，从而加载各个应用类，通过接口指向这些对象，然后通过主板操作这些对象。
2. 试题：下面代码的结果中，**为何再一次输出userDao时，结果为null**？（System.out.println(userDao);没有截屏）



答：因为在创建对象的时候，类中的所有的静态代码块都会按照顺序执行一遍，这样的话，先执行第一句：**private static DaoFactory instance = new DaoFactory(); 由于是new对象，固然要先执行下面的构造方法，这样userDAO就会指向新的对象，但是执行完后又执行了第二句，赋予了userDao 为null，最终使userDAO为null。**

1. 自创题目：

public class LLLLLL {

public static void main(String[] args) {

DDD ddd = DDD.getInstance();

System.out.println(ddd.getTTT());

}

}

class DDD{

private static DDD ddd = new DDD();

private static TTT ttt = null;

private DDD() {

ttt = new TTT();

}

public static DDD getInstance() {

return ddd;

}

public TTT getTTT() {

return ttt;

}

}

class TTT{

}

输出结果为：null。

将 private static DDD ddd = new DDD();

private static TTT ttt = null;换一下位置，改为

private static TTT ttt = null;

private static DDD ddd = new DDD();

结果就是。**赋予了对象。**

**置于为什么？**

**根本原因就是执行的顺序。静态的东西肯定会执行，且**private static TTT ttt = null;

**这句话是在执行完构造方法之后执行的。**

**类加载进虚拟机，需要按照次序执行静态内容，故先执行第一句，然后调用了构造方法，后又执行了第二句，故为null。**

1. DAO工厂模式的特点是：

只需要修改配置文件，改变对应的类名，从而加载不同的类，就可以利用Factory制造出不同的类的对象，从而进行访问操作。不必要去修改代码。

1. 由于配置文件的路径可能会发生改变，所以需要通过其他方法来实现，文件不受路径的影响。

只要是文件路径不发生变化，根据文件名称就可以找到此文件。

但是必须保证文件的路径必须在classpath中。

这种方法测试未通过，不知道到底可不可以。

方法：利用Class的getClassLoader（）方法的getResourceAsStream方法加载，返回一个输入流。

InputStream in = MultiPICs.class.getClassLoader().getResourceAsStream("manifest.properties");