单例模式实验

**【实验目的及要求】**

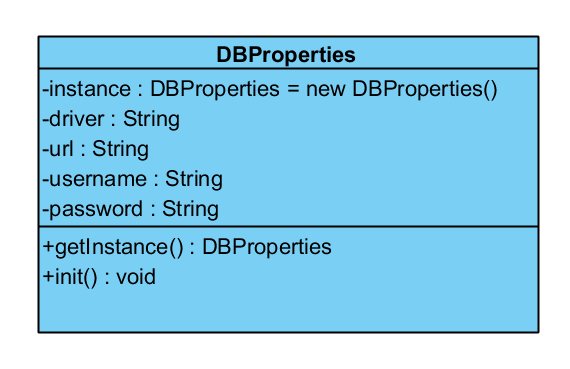
1. 熟悉单例模式的定义；
2. 掌握单例模式角色及作用，熟悉单例模式结构图；
3. 掌握单例模式编程。

**【实验内容】**

**在软件项目中经常使用属性配置文件“properties”设置一些运行参数，例如db.properties文件中设置访问数据库的有关参数（假设有driver，url，username，password四个属性参数）。这些配置参数在项目运行中是全局共享的，程序中只需一份数据。请使用单例模式实现属性文件的读取操作。**

**【实验步骤】**

1. **属性文件的类图**

****

1. **实现代码**
2. **db.properties属性文件**

**Eclipse软件在项目文件夹下，Idea软件在resources文件夹下创建db.properties属性文件，文件内容为**

driver=mysql.driver=com.mysql.cj.jdbc.Driver

url=jdbc:mysql://localhost:3306/mysql

username=root

password=root

1. **Singleton类(DBProperties)**

**package** patter.singleton;

**import** java.io.BufferedInputStream;

**import** java.io.FileInputStream;

**import** java.io.IOException;

**import** java.util.Properties;

**public** **class** DBProperties {

//私有静态成员变量，存储唯一实例

Private static final **DBProperties instance = new DBProperties()**;

**private** String driver = **null**;

**private** String url = **null**;

**private** String username = **null**;

**private** String password = **null**;

// 成员变量的getter/setter方法 , 此处省略

// 私有构造方法

**private** DBProperties() {

init();

}

// 读取属性配置文件中的参数值

**public** **void** init() {

Properties properties = **new** Properties();

**try** {

// 使用InPutStream流读取properties文件

BufferedInputStream bin = **new** BufferedInputStream(**new** FileInputStream("db.properties"));

properties.load(bin);

// 获取key对应的value值

driver = properties.getProperty("driver");

url = properties.getProperty("url");

username = properties.getProperty("username");

password = properties.getProperty("passord");

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

//公有静态成员方法，返回唯一实例

**Private static DBProperties getInstance(){**

**return instance;**

**}**

}

1. **测试类Test**

**package** patter.singleton;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

//创建2个DBProperties对象

DBProperties properties1,properties2;

properties1 = DBProperties.*getInstance*();

properties2 = DBProperties.*getInstance*();

//判断属性文件对象是否相同

**if** (properties1 == properties2) {

System.***out***.println("属性文件对象具有唯一性！");

}

// 获取参数值

System.***out***.println("driver="+properties1.getDriver()

+",url="+properties1.getUrl()

+",username="+properties1.getUsername()

+",password="+properties1.getPassword());

}

}

**【思考问题】**

1. 你实现的程序是饿汉式单例模式还是懒汉式单例模式？这两种模式有什么差异。

我实现的程序是饿汉式单例模式。

**饿汉式单例模式**

**实例化时机：** 在类加载时就会创建唯一的实例，保证在整个程序生命周期中，实例只有一个。

**实现方式：** 类加载时就创建实例，确保类被初始化时实例已经存在。

**优点：** 线程安全，创建实例时不会出现并发问题。

**缺点：** 即使没有使用该实例，也会在类加载时创建，这可能会导致资源的浪费。

**懒汉式单例模式**

**实例化时机：** 只有在需要实例时才创建单例对象，因此在程序启动时不会立即创建实例。

**实现方式：** 使用时检查是否已经存在实例，如果不存在，则创建。

**优点：** 延迟加载，节省资源，实例只会在需要时创建。

**缺点：** 如果多线程环境下没有妥善处理，可能会导致创建多个实例，因此需要额外考虑线程安全问题。

1. 如何修改前面代码，实现另一种单例模式。

public class DBProperties {

// 私有静态成员变量，存储唯一实例（懒汉式，延迟加载）

Private volatile static DBProperties instance=null;

private String driver = null;

private String url = null;

private String username = null;

private String password = null;

// 成员变量的getter/setter方法，省略

// 私有构造方法

private DBProperties() {

init();

}

// 读取属性配置文件中的参数值

public void init() {

Properties properties = new Properties();

try {

// 使用InputStream流读取properties文件

BufferedInputStream bin = new BufferedInputStream(new FileInputStream("db.properties"));

properties.load(bin);

// 获取key对应的value值

driver = properties.getProperty("driver");

url = properties.getProperty("url");

username = properties.getProperty("username");

password = properties.getProperty("password");

} catch (IOException e) {

e.printStackTrace();

}

}

// 公有静态成员方法，返回唯一实例（懒汉式实现，保证线程安全）

public static DBProperties getInstance() {

// 如果实例为null，才创建一个新实例

if (instance == null) {

// 使用synchronized确保多线程下的线程安全

synchronized (DBProperties.class) {

// 双重检查锁定，防止多次创建实例

if (instance == null) {

instance = new DBProperties();

}

}

}

return instance;

}

}

package patter.singleton;

public class Test {

public static void main(String[] args) {

// 创建2个DBProperties对象

DBProperties properties1, properties2;

properties1 = DBProperties.getInstance();

properties2 = DBProperties.getInstance();

// 判断属性文件对象是否相同

if (properties1 == properties2) {

System.out.println("属性文件对象具有唯一性！");

}

// 获取参数值

System.out.println("driver=" + properties1.getDriver()

+ ",url=" + properties1.getUrl()

+ ",username=" + properties1.getUsername()

+ ",password=" + properties1.getPassword());

}

}

**【总结】**

|  |  |
| --- | --- |
| 实验总结  （不少于100字） | 在本次实验中，我深入学习了单例模式的定义、角色及作用，并掌握了其结构图和编程实现。单例模式是一种设计模式，旨在确保类只有一个实例，并提供全局访问点。通过实验，我了解了单例模式在实际项目中的应用，尤其是如何通过饿汉式和懒汉式实现单例模式，确保资源的共享与线程安全。在编程过程中，我熟悉了如何使用静态变量、静态方法以及构造函数的私有化来保证实例唯一性，并通过实际编码加深了对单例模式的理解。通过对结构图和代码的结合，能够更清晰地理解单例模式的工作原理与使用场景。 |
| 学号 | 22154A206 |
| 姓名 | 冯艳芳 |
| 成绩 |  |