**《Java ProgramingⅡ》实验报告**

**班级：软件工程2班**

**学号：162210702207**

**姓名：陈浩明**

目录

[第1章 实验一 java 语言基本技能 2](#_Toc518121163)

[1.1 实验目的 2](#_Toc518121164)

[1.2 实验内容 2](#_Toc518121165)

[1.3 实验过程和结果分析 3](#_Toc518121166)

[1.4 实验内容总结 4](#_Toc518121167)

[第2章 实验二 JDBC数据库编程 5](#_Toc518121168)

[2.1 实验目的 5](#_Toc518121169)

[2.2 实验内容 5](#_Toc518121170)

[2.3 实验过程和结果分析 5](#_Toc518121171)

[2.4 实验内容总结 7](#_Toc518121172)

[第3章 实验三 Java web application 7](#_Toc518121173)

[3.1 实验目的 7](#_Toc518121174)

[3.2 实验内容 7](#_Toc518121175)

[3.3 实验过程和结果分析 7](#_Toc518121176)

[3.4 实验内容总结 10](#_Toc518121177)

# 实验一 java 语言基本技能

## 实验目的

掌握java 语言的一些基本技能。

## 实验内容

* static域和方法的理解和使用
* this理解和使用
* String不可变类的理解和使用
* StringBuilder可变类的理解和使用

## 实验过程和结果分析

**public class Experience\_1 {**

**private String string;**

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

**Experience\_1 experience\_1=new Experience\_1("experience\_1");**

**Experience\_1 experience\_2=new Experience\_1("experience\_2");**

**long t1=Experience\_1.testString();**

**long t2=Experience\_1.testStringBuilder();**

**System.out.println("使用String相加字符串10000次的时间: "+t1+"ms");**

**System.out.println("使用StringBuilder相加字符串10000次的时间: "+t2+"ms");**

**}**

**static {**

**System.out.println("静态初始化Experience\_1");**

**}**

**public Experience\_1(String string) {**

**this.string=string;**

**System.out.println("创建Experience\_1对象");**

**}**

**//测试String相加字符串10000次**

**public static long testString() {**

**String str1=new String();**

**long time1=System.currentTimeMillis();**

**for(int i=0;i<10000;i++) {**

**str1+=i;**

**}**

**long time2=System.currentTimeMillis();**

**return time2-time1;**

**}**

//测试StringBuilder相加字符串10000次

public static long testStringBuilder() {

StringBuilder sb=new StringBuilder();

long time1=System.currentTimeMillis();

for(int i=0;i<10000;i++) {

sb.append(i);

}

long time2=System.currentTimeMillis();

return time2-time1;

}

}

实验结果:

静态初始化Experience\_1

创建Experience\_1对象

创建Experience\_1对象

String对字符串相加执行1万次的时间: 164ms

StringBuilder对字符串相加执行1万次的时间: 1ms

## 实验内容总结

1. this关键字指的是本身对象，属于对象的变量。
2. String类的底层实现是一个定长的数组，这就是真正的不可变，当String字符串相加的时候，其实是调用数组的复制，将2个数组拷贝到新的数组。
3. StringBuilder的底层实现也是一个数组，但是这个数组具有扩容机制，当检测到数组要超过上限的时候，会按当前某个倍数扩容。同时，StringBuilder是线程不安全的，StringBuffer是线程安全的两者的区别是，StringBuffer多了一个synchronized关键字

# 实验二 JDBC数据库编程

## 实验目的

学会使用JDBC存取典型关系数据库，比如MySql数据，对数据进行适当的处理。

## 实验内容

* 驱动程序装载
* 数据库连接
* 数据库数据的存取
* 数据库数据的处理

## 实验过程和结果分析

JDBC连接数据库

•创建一个以JDBC连接数据库的程序，包含7个步骤：

1、JDBC所需的四个参数（user，password，url，driverClass）

（1）user用户名

（2）password密码

（3）URL定义了连接数据库时的协议、子协议、数据源标识。

•书写形式：协议：子协议：数据源标识

协议：在JDBC中总是以jdbc开始

子协议：是桥连接的驱动程序或是数据库管理系统名称。

数据源标识：标记找到数据库来源的地址与连接端口。

例如：（MySql的连接URL）

jdbc:mysql:

//localhost:3306/test?useUnicode=true&characterEncoding=gbk ;

useUnicode=true：表示使用Unicode字符集。如果characterEncoding设置为

gb2312或GBK，本参数必须设置为true 。characterEncoding=gbk：字符编码方式。

（4）driverClass连接数据库所需的驱动。

2、加载JDBC驱动程序：

在连接数据库之前，首先要加载想要连接的数据库的驱动到JVM（Java虚拟机），

这通过java.lang.Class类的静态方法forName(String className)实现。

例如：

try{

//加载MySql的驱动类

Class.forName("com.mysql.jdbc.Driver") ;

}catch(ClassNotFoundException e){

System.out.println("找不到驱动程序类 ，加载驱动失败！");

e.printStackTrace() ;

}

成功加载后，会将Driver类的实例注册到DriverManager类中。

3、创建数据库的连接

•要连接数据库，需要向java.sql.DriverManager请求并获得Connection对象，

该对象就代表一个数据库的连接。

•使用DriverManager的getConnectin(String url , String username ,

String password )方法传入指定的欲连接的数据库的路径、数据库的用户名和

密码来获得。

例如：

//连接MySql数据库，用户名和密码都是root

String url = "jdbc:mysql://localhost:3306/test" ;

String username = "root" ;

String password = "root" ;

try{

Connection con =

DriverManager.getConnection(url , username , password ) ;

}catch(SQLException se){

System.out.println("数据库连接失败！");

se.printStackTrace() ;

}

4、创建一个preparedStatement

•要执行SQL语句，必须获得java.sql.Statement实例，Statement实例分为以下3

种类型：

1、执行静态SQL语句。通常通过Statement实例实现。

2、执行动态SQL语句。通常通过PreparedStatement实例实现。

3、执行数据库存储过程。通常通过CallableStatement实例实现。

具体的实现方式：

Statement stmt = con.createStatement() ;

PreparedStatement pstmt = con.prepareStatement(sql) ;

CallableStatement cstmt =

con.prepareCall("{CALL demoSp(? , ?)}") ;

5、执行SQL语句

Statement接口提供了三种执行SQL语句的方法：executeQuery 、executeUpdate

和execute

1、ResultSet executeQuery(String sqlString)：执行查询数据库的SQL语句

，返回一个结果集（ResultSet）对象。

2、int executeUpdate(String sqlString)：用于执行INSERT、UPDATE或

DELETE语句以及SQL DDL语句，如：CREATE TABLE和DROP TABLE等

3、execute(sqlString):用于执行返回多个结果集、多个更新计数或二者组合的

语句。

具体实现的代码：

ResultSet rs = stmt.executeQuery("SELECT \* FROM ...") ;

int rows = stmt.executeUpdate("INSERT INTO ...") ;

boolean flag = stmt.execute(String sql) ;

6、遍历结果集

两种情况：

1、执行更新返回的是本次操作影响到的记录数。

2、执行查询返回的结果是一个ResultSet对象。

• ResultSet包含符合SQL语句中条件的所有行，并且它通过一套get方法提供了对这些

行中数据的访问。

• 使用结果集（ResultSet）对象的访问方法获取数据：

while(rs.next()){

String name = rs.getString("name") ;

String pass = rs.getString(1) ; // 此方法比较高效

}

（列是从左到右编号的，并且从列1开始）

7、处理异常，关闭JDBC对象资源

操作完成以后要把所有使用的JDBC对象全都关闭，以释放JDBC资源，关闭顺序和声

明顺序相反：

1、先关闭requestSet

2、再关闭preparedStatement

3、最后关闭连接对象connection

if(rs != null){ // 关闭记录集

try{

rs.close() ;

}catch(SQLException e){

e.printStackTrace() ;

}

}

if(stmt != null){ // 关闭声明

try{

stmt.close() ;

}catch(SQLException e){

e.printStackTrace() ;

}

}

if(conn != null){ // 关闭连接对象

try{

conn.close() ;

}catch(SQLException e){

e.printStackTrace() ;

}

}

## 实验内容总结

1. 首先下载MySQL的JDBC驱动包,然后导入项目中
2. 使用Class.forName(“com.mysql.jdbc.Driver”)注册驱动
3. 通过DriverManager获取connection
4. 使用connection.createStatement()得到Statement
5. 通过Statement执行CRUD
6. 以上过程完全可以封装成一个工具类,降低耦合,同时也避免了代码的冗余

# 实验三 Java web application

## 实验目的

了解和认识Java web application的涉及的核心技术、组成和开发过程

## 实验内容

实现一个简单的学生信息管理：学生信息（学号、姓名、性别和年龄等）输入；学生信息的查询；

* 客户端设计与实现
  + HTML
  + CSS
  + Javascript
* 服务器端设计与实现
  + Servlet
  + Java
  + JDBC

## 实验过程和结果分析

//MyServlet.java

package test.servlet;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\* Servlet implementation class MyServlet

\*/

@WebServlet("/MyServlet")

public class MyServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public MyServlet() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String str=request.getParameter("num");

response.setContentType("text/html;charset=UTF-8");

String name=getName(str);

PrintWriter out=response.getWriter();

out.write(name);

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

doGet(request, response);

}

//根据学号获取用户名

public String getName(String num) {

String str="";

//连接数据库

Connection connection;

ResultSet set;

try {

Class.forName("com.mysql.jdbc.Driver");

connection=DriverManager.getConnection("jdbc:mysql://localhost:3306/student","root","123456");

Statement statement=connection.createStatement();

String sql="select name from student\_info where num='"

+ num

+ "'";

set=statement.executeQuery(sql);

while(set.next()) {

str=set.getString("name");

}

set.close();

connection.close();

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return str;

}

}

//student\_info.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Student\_info</title>

</head>

<body>

<form action="MyServlet" method="GET">

学号<input type="text" name="num">

<input type="submit" value="submit">

</form>

<form action="MyServlet" method="POST">

学号:<input type="text" name="num">

姓名:<input type="text" name="name">

性别:<input type="text" name="sex">

年龄:<input type="text" name="age">

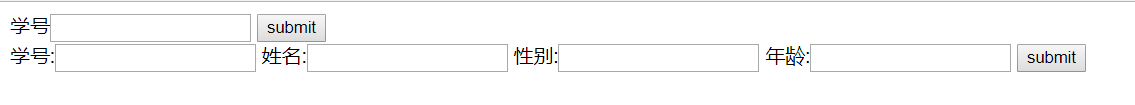
<input type="submit" value="submit">

</form>

</body>

</html>

实验结果:



## 实验内容总结

1. 客户端的设计:

1. 使用HTML5设计客户端的整体框架

第一部分为提交组,包含学号,姓名,性别和年龄的输入以及一个提交按钮

第二部分为查询组,包含学号的输入以及一个提交按钮

2. 使用CSS设计HTML文档的样式

3. 使用JavaScript设计HTML控件的行为,如点击按钮后向服务器发送数据并返回信息

2. 服务端的设计:使用Servlet承载HTML页面;使用JDBC连接mysql数据库,当有用户使用提交组的按钮提交学号,姓名,性别,年龄信息时,将其插入至数据库,当有用户使用查询组的按钮查询学生信息时,查询保存在数据库的信息并显示于HTML页面上