IO实现注册登录案例

2018年7月22日 20:29

由于已经分好包，因此只需对实现类进行修改即可

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

import cn.itcast.dao.UserDao;
import cn.itcast.pojo.User;

/**
 * 这是用户操作的具体实现类（IO版）
 *
 * @author TJtulong
 * @version V1.1
 */
public class UserDaoImpl implements UserDao {
	private static File file = new File("user.txt");

	// 静态代码块(代码运行即生成文件)
	static {
		try {
			file.createNewFile();
		} catch (IOException e) {
			System.out.println("创建文件失败");
		}
	}

	@Override
	public boolean isLogin(String username, String password) {
		BufferedReader br = null;
		boolean flag = false;
		try {
```

```
            br = new BufferedReader(new FileReader(file));
            String line = null;
            while ((line = br.readLine()) != null) {
                String[] datas = line.split("=");
                if (datas[0].equals(username) && datas[1].equals(password)) {
                    flag = true;
                }
            }
        } catch (FileNotFoundException e) {
            System.out.println("找不到信息所在文件");
        } catch (IOException e) {
            System.out.println("用户登陆失败");
        } finally {
            if (br != null) {
                try {
                    br.close();
                } catch (IOException e) {
                    System.out.println("用户登录释放资源失败");
                }
            }
        }
        return flag;
    }

    @Override
    public void regist(User user) {
        /*
         * 自定义规则：用户名=密码
         */
        BufferedWriter bw = null;
        try {
            // 为了保证数据是追加写入，必须加true
            bw = new BufferedWriter(new FileWriter(file, true));
            String s = user.getName() + "=" + user.getPassword();
            bw.write(s);
            bw.flush();
        } catch (IOException e) {
            System.out.println("用户注册失败");
            // e.printStackTrace();
        } finally {
```

```
                if (bw != null) {
                    try {
                        bw.close();
                    } catch (IOException e) {
                        System.out.println("用户注册释放资源失败");
                    }
                }
            }

        }

}
```

基本数据类型操作

2018年7月23日 23:16

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

/*
 * 数据输入流：DataInputStream
 *                              DataInputStream(InputStream in)
 * 数据输出流：DataOutputStream
 *                              DataOutputStream(OutputStream out)
 */
public class DateStreamDemo {
	public static void main(String[] args) throws IOException {
		write();

		read();
	}

	private static void read() throws IOException {
		DataInputStream dis = new DataInputStream(new
		FileInputStream("dos.txt"));

		byte b = dis.readByte();
		short s = dis.readShort();
		int i = dis.readInt();
		long l = dis.readLong();
		float f = dis.readFloat();
		double d = dis.readDouble();
		char c = dis.readChar();
		boolean bb = dis.readBoolean();

		dis.close();

		System.out.println(b);
		System.out.println(s);
```

```
            System.out.println(i);
            System.out.println(l);
            System.out.println(f);
            System.out.println(d);
            System.out.println(c);
            System.out.println(bb);
        }

        private static void write() throws IOException {
            DataOutputStream dos = new DataOutputStream(new
            FileOutputStream("dos.txt"));

            dos.writeByte(10);
            dos.writeShort(100);
            dos.writeInt(1000);
            dos.writeLong(10000);
            dos.writeDouble(12.56);
            dos.writeFloat(12.34F);
            dos.writeChar('a');
            dos.writeBoolean(true);

            dos.close();
        }
}
```

内存操作流

2018年7月24日 10:01

```
/*
 * 内存操作流：用于处理临时存储信息的，程序结束，数据就从内存中消失
 * 字节数组：(无中文)
 *                 ByteArrayInputStream
 *                 ByteArrayOutputStream
 * 字符数组：(有中文)
 *                 CharArrayReader
 *                 CharArrayWriter
 * 字符串：
 *                 StringReader
 *                 StringWriter
 */
public class ByteArrayStreamDemo {
        public static void main(String[] args) throws IOException {
                // 创建输出流对象
                ByteArrayOutputStream baos = new ByteArrayOutputStream();
                // 写数据
                for (int x = 0; x < 10; x++) {
                        baos.write(("hello" + x).getBytes());
                }
                // 不需要close()

                byte[] bys = baos.toByteArray();
                // 读数据
                ByteArrayInputStream bais = new ByteArrayInputStream(bys);

                int by = 0;
                while ((by = bais.read()) != -1) {
                        System.out.println((char) by);
                }
        }
}
```

打印流

2018年7月24日 10:23

```
import java.io.IOException;
import java.io.PrintWriter;

/*
 * 打印流
 * 字节打印流		PrintStream
 * 字符打印流		PrintWriter
 *
 * 打印流的特点：	只有写数据，没有读取数据
 *				可以操作任意类型的数据
 *				如果启动了自动刷新能够自动刷新
 *				该流是可以直接操作文本文件的
 *
 * 流分为两种：		基本流：能够直接读写文件
 *				高级流：在基本流上提供一些其它功能
 */
public class PrintWriterDemo {
	public static void main(String[] args) throws IOException {
		// 作为write的子类使用
		PrintWriter pw = new PrintWriter("pw.txt");

		pw.write("hello");
		pw.write("world");
		pw.write("java");

		pw.close();
	}
}
```

打印流自动刷新

2018年7月24日 11:12

```
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

/*
 * 可以操作任意类型的数据
 * print()
 * println()
 * 启动自动刷新：必须用println()方法
 *                  不仅仅自动刷新，还能够自动换行
 */
public class PrintWriterDemo2 {
        public static void main(String[] args) throws IOException {
                // PrintWriter pw =new PrintWriter("pw.txt");
                // 启用自动刷新
                PrintWriter pw = new PrintWriter(new FileWriter("pw.txt"), true);

                /*
                 * pw.print(true); pw.print(100); pw.print("hello");
                 */

                pw.println(true);
                pw.println(100);
                pw.println("hello");

                pw.close();
        }
}
```

复制文件

2018年7月24日 11:14

```
/*
 * 需求：复制文本文件
 */
public class CopyFileDemo {
	public static void main(String[] args) throws IOException {
		BufferedReader br = new BufferedReader(new FileReader("pw.txt"));

		PrintWriter pw = new PrintWriter(new FileWriter("dos.txt"), true);

		String line = null;
		while ((line = br.readLine()) != null) {
			pw.println(line);// 一句顶三句
		}

		br.close();
		pw.close();
	}
}
```

标准输入输出流

2018年7月24日 11:27

```
import java.io.PrintStream;

/*
 * 标准输入输出流
 * System类中的两个成员变量
 *                  public static final InputStream in:标准输入流
 *                  public static final PrintStream out:标准输出流
 *
 *                  InputStream is = System.in
 *                  PrintStream ps = System.out
 */
public class SystemInDemo {
        public static void main(String[] args) {
                System.out.println("helloworld");

                // 获取标准输出流对象
                PrintStream ps = System.out;// 打印流
                ps.println("helloworld");
        }
}
```

三种方法实现键盘录入

2018年7月24日 11:40

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

/*
 * System.in 标准输入流，从键盘获取数据
 * 键盘录入数据
 *                  1.main方法中的args接收参数
 *                  2.Scanner()  JDK1.5以后
 *                  3.通过字符缓冲流包装标准输入流
 *                  BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
 */
public class SystemInDemo {
        public static void main(String[] args) throws IOException{
                //获取标准输入流
                InputStream is = System.in;

                //一次获取一行数据readline()
                //把字节流转换为字符流
                /*InputStreamReader isr = new InputStreamReader(is);
                BufferedReader br = new BufferedReader(isr);*/

                BufferedReader br=new BufferedReader(new
                InputStreamReader(System.in));
                System.out.println("请输入一个字符串");
                String line = br.readLine();
                System.out.println(line);
        }
}
```

输出语句改进

2018年7月24日 12:00

```
public class SystemOutDemo2 {
	public static void main(String[] args) throws IOException {
		// 获取标准输出流
		PrintStream ps = System.out;
		OutputStream os = ps;

		OutputStreamWriter osw = new OutputStreamWriter(os);
		BufferedWriter bw = new BufferedWriter(osw);

		bw.write("hello");
		bw.write("world");
		bw.write("java");
		bw.flush();
		bw.close();
	}
}
```

基本没用

随机访问流

2018年7月24日 15:09

父类是object

```
/*
 * 随机访问流：
 * 支持对文件的随机访问及写入
 *
 * public RandomAccessFile(String name,String mode)
 * 第一个参数是文件路径，第二个参数是操作文件的模式
 * 模式有四种，最常用的为"rw"，既可以写也可以读数据
 */
public class RandomAccessFileDemo {
	public static void main(String[] args) throws IOException {
		// write();
		read();
	}

	private static void read() throws IOException {
		RandomAccessFile raf = new RandomAccessFile("raf.txt", "rw");

		int i = raf.readInt();
		System.out.println(i);
		System.out.println("当前文件的指针位置是" + raf.getFilePointer());// 4

		char ch = raf.readChar();
		System.out.println(ch);
		System.out.println("当前文件的指针位置是" + raf.getFilePointer());// 6

		String s = raf.readUTF();
		System.out.println(s);
		System.out.println("当前文件的指针位置是" + raf.getFilePointer());// 14=6+3
		*2+2

		// 直接读取a
		raf.seek(4);
		char ch1 = raf.readChar();
		System.out.println(ch); // a
	}
```

```
    private static void write() throws IOException {
        RandomAccessFile raf = new RandomAccessFile("raf.txt", "rw");

        raf.writeInt(100);
        raf.writeChar('a');
        raf.writeUTF("中国");

        raf.close();
    }
}
```

合并文件并复制

2018年7月24日 15:30

```
/*
 * a.txt+b.txt--->c.txt
 */
public class SequenceInputStreamDemo {
	public static void main(String[] args) throws IOException {
		InputStream s1 = new FileInputStream("dos.txt");
		InputStream s2 = new FileInputStream("pw.txt");

		// 合并流
		SequenceInputStream sis = new SequenceInputStream(s1, s2);

		BufferedOutputStream bos = new BufferedOutputStream(new
		FileOutputStream("copy.txt"));

		byte[] bys = new byte[1024];
		int len = 0;
		while ((len = sis.read(bys)) != -1) {
			bos.write(bys);
		}

		bos.close();
		sis.close();
	}
}
```

多个文件合并

2018年7月24日 15:46

```
/*
 * 多个文件合并
 * a.txt+b.txt+c.txt--->copy.txt
 */
public class SequenceInputStreamDemo2 {
	public static void main(String[] args) throws IOException{
		/*
		 * SequenceInputStream(Enumeration e)
		 * Enumeration是Vector中的一个方法的返回值
		 * Enumeration<E> elements
		 */

		Vector<InputStream> v = new Vector<InputStream>();
		InputStream s1 = new FileInputStream("a.txt");
		InputStream s2 = new FileInputStream("b.txt");
		InputStream s3 = new FileInputStream("c.txt");

		v.add(s1);
		v.add(s2);
		v.add(s3);

		Enumeration<InputStream> en = v.elements();
		SequenceInputStream sis = new SequenceInputStream(en);

		BufferedOutputStream bos = new BufferedOutputStream(new
		FileOutputStream("copy.txt"));

		byte[] bys = new byte[1024];
		int len = 0;
		while ((len = sis.read(bys)) != -1) {
			bos.write(bys);
		}

		bos.close();
		sis.close();
```

```
        }
    }
```

序列化流

2018年7月24日 15:58

```
/*
 * 序列化流：把对象按照流一样的方式存入文本文件或者在网络中传输 ObjectOutputStream
 * 对象--->流数据
 * 反序列化：ObjectInputStream
 *
 * 修改对象会报错：InvalidClassException
 * 序列化接口标记值id改变
 * 使id值变为一个固定的值
 */
public class ObjectStreamDemo {
	public static void main(String[] args) throws IOException, ClassNotFoundException {
		write();
		read();
	}

	private static void read() throws IOException, ClassNotFoundException {
		ObjectInputStream ois = new ObjectInputStream(new FileInputStream("oos.txt"));
		Object obj = ois.readObject();// 多态
		ois.close();
		System.out.println(obj);
	}

	private static void write() throws IOException {
		ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("oos.txt"));

		Student s = new Student("王若潇", 22);

		oos.writeObject(s);

		oos.close();
	}
}
```

```
/*
 * NotSerializableException未序列化异常
 * 必须启用序列化接口Serializable
 * 该接口没有方法，没有方法的接口被成为标记接口
 *
 * 一个类中有多个成员变量，如何让部分成员不被序列化
 * transient关键字声明不需要序列化的成员变量
 */
public class Student implements Serializable {
	private static final long serialVersionUID = 6210569037427281918L;
	private String name;
	private transient int age;// age=0
```

Properties类

2018年7月25日 22:37

```
/*
 * Properties:属性集合类，是一个可以和IO流相结合使用的集合类
 * Properties可以保存在流中或从流中加载键和值都为字符串
 *
 * 是Hashtable的子类，是Map集合
 */
public class PropertiesDemo {
	public static void main(String[] args) {
		Properties prop = new Properties();
		System.out.println("prop="+prop);

		prop.put("it002", "hello");
		prop.put("it001", "world");
		prop.put("it003", "java");

		System.out.println("prop="+prop);

		//遍历集合
		Set<Object> set = prop.keySet();
		for(Object key:set) {
			Object value = prop.get(key);
			System.out.println(key+"----"+value);
		}
	}
}
```

Properties的特殊功能

2018年7月25日 23:01

```
import java.util.Properties;
import java.util.Set;

/*
 * 特殊功能:
 *		添加元素
 *		public Object setProperty(String key,String value);
 *		获取元素
 *		public String getProperty(String key);
 *		获取所有的键的集合
 *		public Set<String> stringPropertyNames();
 */
public class PropertiesDemo2 {
	public static void main(String[] args) {
		Properties prop = new Properties();

		prop.setProperty("it002", "hello");
		prop.setProperty("it001", "world");
		prop.setProperty("it003", "java");

		Set<String> set = prop.stringPropertyNames();
		for (String key : set) {
			Object value = prop.get(key);
			System.out.println(key + "----" + value);
		}
	}
}
```

Properties与IO流

2018年7月25日 23:01

```
/*
 * public void load(Reader reader):把文件中的数据读取到集合中
 * public void store(Writer writer,String comment):把集合中的数据存储到文件中
 */
public class PropertiesDemo3 {
	public static void main(String[] args) throws IOException {
		// load();
		store();
	}

	private static void store() throws IOException {
		Properties prop = new Properties();

		prop.setProperty("it002", "hello");
		prop.setProperty("it001", "world");
		prop.setProperty("it003", "java");

		Writer w = new FileWriter("prop.txt");
		prop.store(w, "helloworld");
		w.close();
	}

	private static void load() throws IOException {
		Properties prop = new Properties();

		// public void load(Reader reader)
		// 文件中的数据必须是键值对形式
		Reader r = new FileReader("prop.txt");
		prop.load(r);
		r.close();

		System.out.println("prop=" + prop);
	}
}
```

NIO

2018年7月26日 10:36

了解

```
/*
 * NIO包在JDK4出现，提高了IO流的操作效率
 *
 * JDK7后的NIO
 * Path：路径
 * Paths：有一个静态方法返回路径
 *                 public static Psth get(URI uri)
 * Files：提供了静态方法供我们使用
 */
public class NIODemo {
	public static void main(String[] args) throws FileNotFoundException, IOException {
		//复制文件
		Files.copy(Paths.get("dos.txt"), new FileOutputStream("copy.txt"));

		ArrayList<String> array = new ArrayList<>();
		array.add("hello");
		array.add("world");
		array.add("java");

		Files.write(Paths.get("array.txt"), array,Charset.forName("GBK"));
	}
}
```