# 第七单元 工具类编程

# 【授课重点】

1. 工具类编程思想
2. 字符串工具类
3. 日期工具类
4. 文件工具类
5. 流处理工具类

# 【考核要求】

1. 创建maven工程编写工具类
2. 利用git对工具类进行版本管理
3. 字符串常用处理
4. 日期常用处理
5. 文件常用处理
6. 流的常用处理

# 【教学内容】

## 7.1 课程导入

项目开发工具类演示

开发工具类的用途

## 7.2 准备工作

### 7.2.1 创建Maven工程

### 7.2.2 git版本管理

## 7.3 字符串工具类

### 7.3.1 创建字符串工具类

### 7.3.2 判空

判断源字符串是否有值，空引号也算没值；

null != src && src.length() \> 0;  
  
null != src && src.trim().length() \> 0;

### 7.3.3 是否有值

判断源字符串是否有值，空引号和空格也算没值

参见上一条

### 7.3.4 验证手机号码

使用正则表达式。

String regex =  
"\^((13[0-9])\|(14[5\|7])\|(15([0-3]\|[5-9]))\|(17[013678])\|(18[0,5-9]))\\\\d{8}\$";  
  
Pattern p = Pattern.compile(regex);  
  
Matcher m = p.matcher(phone);  
  
boolean isMatch = m.matches();  
  
Return isMatch ;  
  
###

### 7.3.5 验证邮箱

String regEx1 =  
"\^([a-z0-9A-Z]+[-\|\\\\.]?)+[a-z0-9A-Z]\@([a-z0-9A-Z]+(-[a-z0-9A-Z]+)?\\\\.)+[a-zA-Z]{2,}\$";  
  
Pattern p = Pattern.compile(regEx1);  
  
Matcher m = p.matcher(email);  
  
Return m;

### 7.3.6 验证全为字母

String regEx1 = “[a-zA-Z]?”;

### 7.3.7 获取n位随机英文字符串

StringBuilder sb = \*\*new\*\* StringBuilder();  
  
Random random = \*\*new\*\* Random();  
  
\*\*for\*\*(\*\*int\*\* i=0;i\<10;i++) {  
  
\*\*int\*\* t = 'A' + random.nextInt(26);  
  
sb.append((\*\*char\*\*)t);  
  
}  
  
System.\*out\*.append("sb is " + sb);

### 7.3.8 获取n位随机英文和数字字符串

定义一个长度为36的char类型数组，然后随机获取下表，组成字符串。

### 7.3.9 获取n个随机中文字符串

String str = "";  
  
 int hightPos; //  
  
 int lowPos;  
  
 Random random = new Random();  
  
 hightPos = (176 + Math.\*abs\*(random.nextInt(39)));  
  
 lowPos = (161 + Math.\*abs\*(random.nextInt(93)));  
  
 byte[] b = new byte[2];  
  
 b[0] = (Integer.\*valueOf\*(hightPos)).byteValue();  
  
 b[1] = (Integer.\*valueOf\*(lowPos)).byteValue();  
  
 try {  
  
 str = new String(b, "GBK");  
  
 } catch (Exception e) {  
  
 e.printStackTrace();  
  
 System.\*out\*.println("错误");  
  
 }  
  
 return str.charAt(0);  
  
##

## 7.4 日期工具类

### 7.4.1 创建创建类

### 7.4.2 计算年龄

//由出生日期获得年龄  
  
public static int getAge(Date birthDay) throws Exception {  
  
Calendar cal = Calendar.getInstance();  
  
if (cal.before(birthDay)) {  
  
throw new IllegalArgumentException(  
  
"The birthDay is before Now.It's unbelievable!");  
  
}  
  
int yearNow = cal.get(Calendar.YEAR);  
  
int monthNow = cal.get(Calendar.MONTH);  
  
int dayOfMonthNow = cal.get(Calendar.DAY\_OF\_MONTH);  
  
cal.setTime(birthDay);  
  
int yearBirth = cal.get(Calendar.YEAR);  
  
int monthBirth = cal.get(Calendar.MONTH);  
  
int dayOfMonthBirth = cal.get(Calendar.DAY\_OF\_MONTH);  
  
int age = yearNow - yearBirth;  
  
if (monthNow \<= monthBirth) {  
  
if (monthNow == monthBirth) {  
  
if (dayOfMonthNow \< dayOfMonthBirth) age--;  
  
}else{  
  
age--;  
  
}  
  
}  
  
return age;  
  
}

### 7.4.2 计算剩余天数

时间戳相减除以3600\*24

### 7.4.3 判断是否为当天

\*\*

> public static boolean\*\* isToday(Date date) {  
  
> SimpleDateFormat fmt = \*\*new\*\* SimpleDateFormat("yyyy-MM-dd");  
  
> \*\*if\*\* (fmt.format(date).toString().equals(fmt.format(\*\*new\*\*  
> Date()).toString())) {  
  
> // 格式化为相同格式  
  
> \*\*return true\*\*;  
  
> } \*\*else\*\* {  
  
> \*\*return false\*\*;  
  
> }  
  
> }

### 7.4.4 判断是否在本周

public static boolean\*\* isThisWeek(Date date) {  
  
SimpleDateFormat format = \*\*new\*\* SimpleDateFormat("yyyyMMdd");  
  
Calendar firstDayOfWeek = Calendar.\*getInstance\*(Locale.\*getDefault\*());  
  
firstDayOfWeek.setFirstDayOfWeek(Calendar.\*MONDAY\*);  
  
\*\*int\*\* day = firstDayOfWeek.get(Calendar.\*DAY\_OF\_WEEK\*);  
  
firstDayOfWeek.add(Calendar.\*DATE\*, -day+1+1);// 后面的+1是因为从周日开始  
  
// 本周一的日期  
  
System.\*out\*.println(format.format(firstDayOfWeek.getTime()));  
  
Calendar lastDayOfWeek = Calendar.\*getInstance\*(Locale.\*getDefault\*());  
  
lastDayOfWeek.setFirstDayOfWeek(Calendar.\*MONDAY\*);  
  
day = lastDayOfWeek.get(Calendar.\*DAY\_OF\_WEEK\*);  
  
lastDayOfWeek.add(Calendar.\*DATE\*, 7-day+1);  
  
// 本周星期天的日期  
  
System.\*out\*.println(format.format(lastDayOfWeek.getTime()));  
  
\*\*return\*\* (date.getTime()\<lastDayOfWeek.getTime().getTime() &&  
date.getTime()\>firstDayOfWeek.getTime().getTime() );  
  
> }

### 7.4.5 判断是否在本月

参见是否为当天

### 7.4.6 给定时间对象，初始化到该月初的1日0时0分0秒0毫秒

Calendar c = Calendar.*getInstance*();

c.setTime(src);

// 让当前月份加1

c.add(Calendar.*MONTH*, 1);

// 获取月初

Date monthStart = *getDateByInitMonth*(c.getTime());

// 用月初初始化日历类

c.setTime(monthStart);

// 用月初时间 -1 秒

c.add(Calendar.*SECOND*, -1);

**return** c.getTime();

### 7.4.7 给定时间对象，设定到该月最一天的23时59分59秒999毫秒

/\*

\* 方法2：(10分)

\*  
给任意一个时间对象，返回该时间所在月的最后日23时59分59秒，需要考虑月大月小和二月特殊情况。

\* 例如一个Date对象的值是2019-05-18 11:37:22，则返回的时间为2019-05-31 23:59:59

\* 例如一个Date对象的值是2019-02-05 15:42:18，则返回的时间为2019-02-28 23:59:59

\*/

**public static** Date getDateByFullMonth(Date src){

//**TODO** 实现代码

Calendar c = Calendar.*getInstance*();

c.setTime(src);

c.add(Calendar.*MONTH*, 1);

c.set(Calendar.*DAYOFMONTH*, 1);

c.set(Calendar.*HOUR*, 0);

c.set(Calendar.*MINUTE*, 0);

c.set(Calendar.*SECOND*, 0);

c.add(Calendar.*SECOND*, -1);

**return** c.getTime();

}

### 7.4.8 时间比较

date.compareTo(*anotherDate*);

## 7.5 文件工具类

### 7.5.1 创建类

### 7.5.2 获取文件扩展名

String fileName = file.getOriginalFilename();

//求出扩展名

String suffix = fileName.substring(fileName.lastIndexOf("."));

//生成新的文件名

String newFileName = UUID.*randomUUID*().toString()+ suffix;

### 7.5.3 删除文件

如果是目录，则下面的文件和所有子目录中的文件都要删除

使用递归。涉及内容。判断目录的存在性，判断是否为目录或文件，删除。

### 7.5.4 获取操作系统用户目录

String usrHome = System.getProperty("user.home");

其他常用

"file.separator" File separator (e.g., "/")

"java.class.path" Java classpath

"java.class.version" Java class version number

"java.home" Java installation directory

"java.vendor" Java vendor-specific string

"java.vendor.url" Java vendor URL

"java.version" Java version number

"line.separator" Line separator

"os.arch" Operating system architecture

"os.name" Operating system name

"path.separator" Path separator (e.g., ":")

"user.dir" User's current working directory

"user.home" User home directory

"user.name" User account name

### 7.5.5 返回文件以指定单位大小表示

方法一：

if (file.exists() && file.isFile()) {

String fileName = file.getName();

System.out.println("文件"+fileName+"的大小是："+file.length());

}

方法二：

if(file.exists() && file.isFile()){

String fileName = file.getName();

fis = new FileInputStream(file);

System.out.println("文件"+fileName+"的大小是："+fis.available()+"\n");

}

## 7.6 流工具类

### 7.6.1 创建类

### 7.6.2 关闭流

fis.close();

### 7.6.2 复制流

InputStream inStream = **new** FileInputStream(realPath+filename);/

**while** ((len = inStream.read(b)) > 0)

response.getOutputStream().write(b, 0, len);

inStream.close();

### 7.6.3 读取文本文件

#### 7.6.3.1 按行读取文本文件

/\*\*

\* 按行读取文件

\*/

**public static void** ReadFileByLine(String filename) {

File file = **new** File(filename);

InputStream is = **null**;

Reader reader = **null**;

BufferedReader bufferedReader = **null**;

**try** {

is = **new** FileInputStream(file);

reader = **new** InputStreamReader(is);

bufferedReader = **new** BufferedReader(reader);

String line = **null**;

**while** ((line = bufferedReader.readLine()) != **null**) {

System.*out*.println(line);

}

} **catch** (FileNotFoundException e) {

e.printStackTrace();

} **catch** (IOException e) {

e.printStackTrace();

} **finally** {

**try** {

**if** (**null** != bufferedReader)

bufferedReader.close();

**if** (**null** != reader)

reader.close();

**if** (**null** != is)

is.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

#### 7.6.3.2 按字节读取文件

/\*\*

\* 按字节读取文件

\*

\* **\@param** filename

\*/

**public static void** ReadFileByBytes(String filename) {

File file = **new** File(filename);

InputStream is = **null**;

**try** {

is = **new** FileInputStream(file);

**int** index = 0;

**while** (-1 != (index = is.read())) {

System.*out*.write(index);

}

} **catch** (FileNotFoundException e) {

e.printStackTrace();

} **catch** (IOException e) {

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

e.printStackTrace();

} **finally** {

**try** {

**if** (**null** != is)

is.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

System.*out*.println("-----------------------------------");

**try** {

is = **new** FileInputStream(file);

**byte**[] tempbyte = **new byte**[1000];

**int** index = 0;

**while** (-1 != (index = is.read(tempbyte))) {

System.*out*.write(tempbyte, 0, index);

}

} **catch** (FileNotFoundException e) {

e.printStackTrace();

} **catch** (IOException e) {

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

e.printStackTrace();

} **finally** {

**try** {

**if** (**null** != is)

is.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

#### 7.6.3.3 按字符读取文件

/\*\*

\* 按字符读取文件

\*

\* **\@param** filename

\*/

**public static void** ReadFileByChar(String filename) {

File file = **new** File(filename);

InputStream is = **null**;

Reader isr = **null**;

**try** {

is = **new** FileInputStream(file);

isr = **new** InputStreamReader(is);

**int** index = 0;

**while** (-1 != (index = isr.read())) {

System.*out*.print((**char**) index);

}

} **catch** (FileNotFoundException e) {

e.printStackTrace();

} **catch** (IOException e) {

e.printStackTrace();

} **finally** {

**try** {

**if** (**null** != is)

is.close();

**if** (**null** != isr)

isr.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

#### 7.6.4 写入文本文件

* 通过BufferedWriter写文件

/\*\*

\* 通过BufferedWriter写文件

\*

\* **\@param** filename

\*/

**public static void** Write2FileByBuffered(String filename) {

File file = **new** File(filename);

*FileOutputStream* fos = **null**;

*OutputStreamWriter* osw = **null**;

*BufferedWriter* bw = **null**;

**try** {

**if** (!file.exists()) {

file.createNewFile();

}

fos = **new** *FileOutputStream*(file);

osw = **new** *OutputStreamWriter*(fos);

bw = **new** *BufferedWriter*(osw);

bw.write("Write2FileByBuffered");

} **catch** (FileNotFoundException e) {

e.printStackTrace();

} **catch** (IOException e) {

e.printStackTrace();

} **finally** {

**if** (**null** != bw) {

**try** {

bw.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

**if** (**null** != osw) {

**try** {

osw.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

**if** (**null** != fos) {

**try** {

fos.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

}

* 通过FileWriter写文件

/\*\*

\* 通过FileWriter写文件

\*

\* **\@param** filename

\*/

**public static void** Write2FileByFileWriter(String filename) {

File file = **new** File(filename);

*FileWriter* fw = **null**;

**try** {

**if** (!file.exists()) {

file.createNewFile();

}

fw = **new** *FileWriter*(file);

fw.write("Write2FileByFileWriter");

} **catch** (IOException e) {

e.printStackTrace();

} **finally** {

**if** (**null** != fw) {

**try** {

fw.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

}

#### 7.6.5 网络文件下载

**public static void** download(String realPath,HttpServletRequest  
request,HttpServletResponse response,String filename) **throws**  
FileNotFoundException {

/\* // 下载本地文件

String fileName = "Operator.doc".toString(); // 文件的默认保存名

\*/ // 读到流中

InputStream inStream = **new** FileInputStream(realPath+filename);//  
文件的存放路径

// 设置输出的格式

response.reset();

response.setContentType("bin");

response.addHeader("Content-Disposition", "attachment; filename=\"" + filename

* "\"");

// 循环取出流中的数据

**byte**[] b = **new byte**[1024];

**int** len;

**try** {

**while** ((len = inStream.read(b)) > 0)

response.getOutputStream().write(b, 0, len);

inStream.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}