**4.1 Commons-lang API介绍**

# 4.1.1 StringUtils

*StringUtils是提供字符串操作的工具类。提供的方法如下：*

1 public static boolean isEmpty(String str);

**说明：**

如果参数str为NULL或者str.length() == 0 返回true

**对比：**JDK 中类String的方法public boolean isEmpty()

此方法通过判断私有变量count是否等于0来进行判断。

2 public static boolean isNotEmpty(String str)

**说明：**

判断给定参数是否不为空，其实现方式利用了方法一： !isEmpty(str);

**对比：**JDK中String类无此方法。

**3** public static boolean isBlank(String str)；

**说明：**

如果参数str为NULL或者其长度等于0，又或者其由空格组成，那么此方法都返回true。

**对比：**JDK中String类无此方法。

**4** public static boolean isNotBlank(String str)；

**说明：**利用方法三实现。

5 public static String trim(String str)；

**说明：**

去除字符串开头和结尾处的空格字符。如果参数str为null,则返回null.

**对比：**

利用JDK中String类的trim()方法。

**6** public static String stripStart(String str, String stripChars)；

**说明：**

去掉str前端的在stripChars中的字符

7 public static String stripEnd(String str, String stripChars)；

**说明：**

去掉str末端的在stripChars中的字符

8 public static int ordinalIndexOf(String str, String searchStr, int ordinal)

**说明：**

返回字符串search在字符串str中第ordinal次出现的位置。

如果str=null或searchStr=null或ordinal<=0则返回-1.

# 4.1.2 StringEscapeUtils

StringEscapeUtils这个类里提供了很多转义的方法，比如可以转成json、xml、html等格式。

1 public static String escapeJava(String str)；

**说明：**使用java String rules转义给定字符串str.

正确处理引号和控制字符（如，tab,backslash,cr,ff等）.

因此，一个tab符号转义成’\\t’.

2 public static String escapeJavaScript(String str)

**说明：**同方法1.唯一区别是，在javaScript中，单引号必须被转义。

# 4.1.3 ArrayUtils

1. final变量

EMPTY\_\*\_ARRAY : 根据\*处的类型，返回对应的长度为0的数组。注意原始变量及其包裹类。

INDEX\_NOT\_FOUND: 返回-1，表述数组下标未找到。

2．toString(Object, [,stringIfNull]): 如果为空，返回stringIfnULL

3. hashCode(Object): 使用HashCodeBuilder返回数组的hashcode

4. isEquals(Object1,Object2): 用EqualsBuilder返回两个数组比较的结果。

5. toMap(Object[]):作为参数的数组有两个选择，一是成员为Map.Entry，然后通过遍历该数组，把Map.Entry拆分并分别放入新生成Map的Key和Value中；二是成员为长度大于等于2的数组，位置0的元素作为key，位置1的元素作为value。除了以上两种情况，会抛出异常。

6.clone族

Object[]:如果不为空，则使用参数自己的clone方法处理。

Long[],int[],short[],char[],byte[],double[],float[],Boolean[]同上。

7. subarray族

Object[],start,end: Object[],start,end：如果不为空，start小于0，取0；end大于长度，取长度。如果start大于end，则用Array.newInstance返回一个空的对象数组，其类型由参数的getClass().getComponentType()决定。否则同样定义一个长度为end-start的数组，使用System.arraycopy进行数组的拷贝。

8. isSameLength族

略.

9. isSameType

Object1,Object2: 如果一方为空，抛出IllegalArgumentException.否则根据getClass().getName()比较两个参数的类名是否相同.

10. reverse族

Object[]: 从数组前后两个坐标i,j开始，交换数据，并向中间移动。当i>j时停止。

11. 下标查找族

11.1 indexOf

Object[],object[,startIndex]:Object[]为空，则返回-1,。如果startIndex小于0，则置为0.如果object为空，则从数组的第一个位置开始找第一个为空的数据。否则，利用Object.equals()方法正向遍历数组，获取匹配的下标并返回。否则返回-1.

# 4.1.4 DateUtils

**1 static final**

MILLIS\_PER\_SECOND = 1000；一秒等于1000毫秒

MILLIS\_PER\_MINUTE = 60 \* MILLIS\_PER\_SECOND; 一分钟等于多少毫秒

MILLIS\_PER\_HOUR = 60 \* MILLIS\_PER\_MINUTE;一小时等于多少毫秒

MILLIS\_PER\_DAY = 24 \* MILLIS\_PER\_HOUR;一天等于多少毫秒

**2 日期比较**

public static boolean isSameDay(Date date1, Date date2);

public static boolean isSameDay(Calendar cal1, Calendar cal2);

比较日期是否相同，忽略time

具体是通过比较Calendar.ERA、YEAR、DAY\_OF\_YEAR三个属性判断给定日期是否相同.

**3 时间比较**

public static boolean isSameInstant(Date date1, Date date2);

public static boolean isSameInstant(Calendar cal1, Calendar cal2);

比较时间是否相同.

具体是通过Date类中的getTime()方法的。

**4 ADD族**

public static Date addYears(Date date, int amount);

在给定日期date的基础上添加amount年,注意，函数返回新的对象;

以下同上。

addMonths();//月

addWeeks();//周

addDays();//日

addHours();//小时

addMinutes();//分钟

addSeconds();//秒

addMilliseconds();//毫秒

**5 set族**

public static Date setYears(Date date, int amount);

为date设置新的年份信息，并返回一个新的对象，对象date未发生改变.

以下方法同上。

setMonths();//设置月份

setDays();//设置日期

setHours();//设置小时

setMinutes();//设置分钟

setSeconds();//设置秒

setMilliseconds();//设置毫秒

**6 round族、truncate族、ceil族**

日期取整（日期精度调节，如调节至秒/分等）

Date round(Date date, int field);DateUtils.round()相当于数学中的四舍五入法取整

Date truncate(Date date, int field);DateUtils.truncate()相当与去余法取整。

Date ceiling(Date date, int field);DateUtils.ceiling()相当于向上取整。

能对Calendar类中几乎所有的field做日期取整，包括Calender.YEAR,Calendar.SECOND,Calendar.MINUTE,Calendar.HOUR,

Calendar.DAY\_OF\_MONTH,Calendar.MONTH.

# 4.1.5 DateFormatUtils

与SUN的SimpleDateFormat相比，其主要优点是：线程安全。

对应于SimpleDateFormat的format()的方法，是DateFormatUtils 的format系列方法，常用的就是：

public static java.lang.String format (java.util.Date date, java.lang.String pattern);

DateFormatUtils定义了很多内置的固定日期格式，均为FastDateFormat类型，比如 ISO\_DATE\_FORMAT。使用 FastDateFormat的format()方法可以直接将日期格式化为内置的固定格式。

public java.lang.String format (java.util.Date date)

# 4.1.6 RandomUtils

随机数据生成类,包括浮点,双精,布尔,整形,长整在内的随机数生成

RandomUtils.nextInt();采用默认的JVMRandom类，数值范围0~2147483647

nextInt(Random random)；也可以设置其他的random

还支持以下方法：

nextLong();

nextBoolean();

nextFloat();

nextDouble();

# 4.1.7 NumberUtils

为JDK中的Number类提供额外的功能。

**提供可复用的值为0,1的数值型的包装类。包括Long、Integer、Short、Byte、Double、Float。**

**toXXX(String str[,XXX])族**

将给定的字符串装换成XXX代表的数值类型，包括Long\Integer\Short\Byte\Double\Float。

如果指定了默认值XXX,那么当String为空，或是转换发生异常，则返回指定的默认值XXX。

**createXXX(String)族**

用给定的字符串创建XXX数值类型的对象。包括Float\Double\Integer\Long\BigInteger\BigDecimal

**XXX min(XXX[]) 族**

从类型为XXX的数组中找出最小的。XXX可为long\float\double\byte\short\int

时间复杂度是O(n).

**XXX max(XXX[]) 族**

同min族。

# 4.1.8 FieldUtils

通过反射技术来操作成员变量。

**1 getField族**

Field getField(Class<?> ,String [,Boolean]);

Field getDeclaredField(Class<?>,String [,Boolean]);

说明：

**getField:**  Gets an accessible Field by name breaking scope if requested. 当前类的接口和和父类都会被考虑。

**getDeclaredField :** an accessible Field by name respecting scope.

仅当前类被考虑。

1. **readField族**

readStaticField(Field[,boolean]);

readStaticField(Class<?>,String[,boolean]);

readDeclaredStaticField(Class<?>,String[,boolean]);

readField(Field,Object[,boolean]);

readField(Object,String[,boolean]);

readDeclaredField(Object,String[,boolean])

获取字段的值。区别是：带有Declared的仅考虑当前类，其他情况会考虑当前类实现的接口以及其父类。

1. **writeField族**

writeStaticField(Field,Object[,boolean]);

writeStaticField(Class<?>,String,Object[,boolean]);

writeDeclaredStaticField(Class<?>,String,Object[,boolean]);

writeField(Field,Object,Object[,boolean]);

writeField(Object, String, Object[,boolean]);

writeDeclaredField(Object, String, Object[,boolean]);

设置字段的值.

# 4.1.9 CharUtils

静态类，不需要创建

1 public static boolean isAscii(char ch)

用途：判断是否为ascii字符

实现：

public static Boolean isAscii(char ch){

return ch < 128;

}

2 public static boolean isAsciiAlpha(char ch)

用途：判断是否为ascii字母，即值在65到90或者97到122

实现：

public static Boolean isAsciiAlpha(char ch) {

return (ch >= ‘A’&& ch <= ‘Z’) || (ch >= ‘a’&& CH <= ‘z’)

}

3 public static Boolean isAsciiAlphaLower(char ch)

同2. 判断范围是 65 到 90，即a 到 z.

4 public static boolean isAsciiAlphaUpper(char ch);

同2.判断范围是97到122.即A 到Z.

5 public static boolean isAsciiAlphanumeric(char ch)

用途：判断是否为ascii字符数字，即值在48到57，65到90或者97到122.

实现：

Public static boolean isAsciiAlphanumeric(char ch) {

return (ch >= ‘A’&& cn <= ‘Z’) || (ch >= ‘a’&& ch <= ‘z’) || (ch >= ‘0’&& ch <= ‘9’)

}

6 public static Boolean isAsciiControl(char ch)

用途：判断是否为控制字符

实现：

public static boolean isAsciiControl(char ch) {

return ch < 32 || ch = 127;

}

7 public static boolean isAsciiNumeric(char ch)

用途：判断是否为数字

实现：

Public static Boolean isAsciiNumeric(char ch) {

Return ch >= ‘0’ && ch <= ‘9’;

}

8 public static Boolean isAsciiPrintable(char ch)

用途：判断是否可打印出得ascii字符

实现：

Public static Boolean isAsciiPrintable(char ch) {

return ch >= 32 && ch < 127;

}

9 public static int toIntValue(char ch)

用途：数字转换

实现：

Public static int toIntValue(char ch) {

if(isAsciiNumeric(ch) == false) {

throw new IllegalArgumentException(“the character” + ch + “is not in the range ‘0’- ‘9’”)

}

Return ch – 48;

}

10 public static String unicodeEscaped(char ch)

用途：将ch转换为unicode表示的字符串形式

实现：

public static String unicodeEscaped(char ch) {

if(ch < 0x10) {

return “[\\u000](file:///\\u000)” + Integer.toHexString(ch);

}else if(ch < 0x100){

retrun “[\\u00](file:///\\u00)” + Integer.toHexString(ch);

}else if(ch < 0x1000) {

Return “\\u0” + Integer.toHexString(ch);

}

Return “[\\u](file:///\\u)” + Integer.toHexString(ch);

}

# 4.1.10 BooleanUtils

1 negate(Boolean bool)

用法：否定指定的boolean值

2 isTrue(Boolean bool)

用法：检查一个boolean值是否为true,如果参数为null,返回false

3 isNotTrue(Boolean bool)

用法：检查一个boolean值是否为false,如果参数为null,返回true

4 isFalse(Boolean bool)

用法：检查一个boolean值是否为false,如果是 返回true.

如果检查的值为true或null返回false.

5 isNotFalse(Boolean bool)

用法：检查一个boolean值是否不为false,如果是返回true

6 toBoolean(Boolean bool)

用法：转换一个为null的boolean值,返回一个false.

\* <pre>

\* BooleanUtils.toBoolean(Boolean.TRUE) = true

\* BooleanUtils.toBoolean(Boolean.FALSE) = false

\* BooleanUtils.toBoolean(null) = false

\* </pre>

7 toBooleanDefaultIfNull(Boolean bool, boolean valueIfNull)

用法： 转换一个为null的boolean值,返回后面参数给定的boolean值.

\* <pre>

\* BooleanUtils.toBooleanDefaultIfNull(Boolean.TRUE, false) = true

\* BooleanUtils.toBooleanDefaultIfNull(Boolean.FALSE, true) = false

\* BooleanUtils.toBooleanDefaultIfNull(null, true) = true

\* </pre>

\*/

8 toBoolean(int value)

用法： 当参数为0是返回false,其它都返回true.

\* <pre>

\* BooleanUtils.toBoolean(0) = false

\* BooleanUtils.toBoolean(1) = true

\* BooleanUtils.toBoolean(2) = true

\* </pre>

9 toBooleanObject(int value)

用法： 当参数为0是返回Boolean.FALSE对象,其它都返回Boolean.TRUE.

\* <pre>

\* BooleanUtils.toBoolean(0) = Boolean.FALSE

\* BooleanUtils.toBoolean(1) = Boolean.TRUE

\* BooleanUtils.toBoolean(2) = Boolean.TRUE

\* </pre>

10 toBooleanObject(Integer value)

用法： 当参数为0是返回Boolean.FALSE对象,为null返回null

\* 其它则返回Boolean.TRUE.

\* <pre>

\* BooleanUtils.toBoolean(new Integer(0)) = Boolean.FALSE

\* BooleanUtils.toBoolean(new Integer(1)) = Boolean.TRUE

\* BooleanUtils.toBoolean(new Integer(null)) = null

\* </pre>

11 toBoolean(int value, int trueValue, int falseValue)

用法： \* 如果第一个参数和第二个参数相等返回true,

\* 如果第一个参数和第三个参数相等返回false,

\* 如果都没有相等的,返回一个IllegalArgumentException

\* \* <pre>

\* BooleanUtils.toBoolean(0, 1, 0) = false

\* BooleanUtils.toBoolean(1, 1, 0) = true

\* BooleanUtils.toBoolean(2, 1, 2) = false

\* BooleanUtils.toBoolean(2, 2, 0) = true

\* </pre>

# 4.1.11 ExceptionUtils

对异常的常见操作,获得堆栈,异常抛出方法名,错误链中对象数

public static Throwable getCause(Throwable [,String[]])；

用法：获取导致Throwable的Throwable. 可以设置自己制定的方法名称.

public static Throwable getRootCause(Throwable throwable);

用法：获取导致Throwable的 Root Throwable。

public static int getThrowableCount(Throwable throwable)

用法：统计异常链上的Throwable对象数量。

A throwable without cause will return 1；

A throwable with one cause will return 2 and so on

A null throwable will return 0.