package com.pancm.utils;

import java.math.BigInteger;

import java.net.InetAddress;

import java.net.UnknownHostException;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.sql.ResultSet;

import java.sql.ResultSetMetaData;

import java.sql.SQLException;

import java.sql.Timestamp;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.time.LocalDate;

import java.time.LocalDateTime;

import java.time.Period;

import java.time.format.DateTimeFormatter;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.Date;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

import java.util.Map.Entry;

import java.util.Random;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import org.apache.commons.codec.binary.Base64;

import com.alibaba.fastjson.JSON;

import com.alibaba.fastjson.JSONObject;

*/\*\**

*\**

*\** ***@Title:*** *MyTools*

*\** ***@Description:常用工具类***

*\** ***@Version:1.0.1***

*\** ***@author*** *pancm*

*\** ***@date*** *2017年9月26日*

*\*/*

public final class MyTools {

*/\*\* 时间格式包含毫秒 \*/*

private static final String *sdfm* = "yyyy-MM-dd HH:mm:ss SSS";

*/\*\* 普通的时间格式 \*/*

private static final String *sdf* = "yyyy-MM-dd HH:mm:ss";

*/\*\* 时间戳格式 \*/*

private static final String *sd* = "yyyyMMddHHmmss";

*/\*\* 检查是否为整型 \*/*

private static Pattern *p* = Pattern.*compile*("^\\d+$");

*/\*\**

*\* 判断String类型的数据是否为空 null,""," " 为true "A"为false*

*\**

*\** ***@return*** *boolean*

*\*/*

public static boolean isEmpty(String str) {

return (null == str || str.trim().length() == 0);

}

*/\*\**

*\* 判断String类型的数据是否为空 null,"", " " 为false "A", 为true*

*\**

*\** ***@return*** *boolean*

*\*/*

public static boolean isNotEmpty(String str) {

return !*isEmpty*(str);

}

*/\*\**

*\* 判断list类型的数据是否为空 null,[] 为 true*

*\**

*\** ***@return*** *boolean*

*\*/*

public static boolean isEmpty(List<?> list) {

return (null == list || list.size() == 0);

}

*/\*\**

*\* 判断list类型的数据是否为空 null,[] 为 false*

*\**

*\** ***@return*** *boolean*

*\*/*

public static boolean isNotEmpty(List<?> list) {

return !*isEmpty*(list);

}

*/\*\**

*\* 判断Map类型的数据是否为空 null,[] 为true*

*\**

*\** ***@return*** *boolean*

*\*/*

public static boolean isEmpty(Map<?, ?> map) {

return (null == map || map.size() == 0);

}

*/\*\**

*\* 判断map类型的数据是否为空 null,[] 为 false*

*\**

*\** ***@return*** *boolean*

*\*/*

public static boolean isNotEmpty(Map<?, ?> map) {

return !*isEmpty*(map);

}

*/\*\**

*\* 判断JSONObject类型的数据是否为空 null,[] 为true*

*\**

*\** ***@return*** *boolean*

*\*/*

public static boolean isEmpty(JSONObject json) {

return (null == json || json.size() == 0);

}

*/\*\**

*\* 判断json类型的数据是否为空 null,[] 为 false*

*\**

*\** ***@return*** *boolean*

*\*/*

public static boolean isNotEmpty(JSONObject json) {

return !*isEmpty*(json);

}

*/\*\**

*\* 字符串反转 如:入参为abc，出参则为cba*

*\**

*\** ***@param*** *str*

*\** ***@return***

*\*/*

public static String reverse(String str) {

if (*isEmpty*(str)) {

return str;

}

return *reverse*(str.substring(1)) + str.charAt(0);

}

*/\*\**

*\* 获取当前long类型的的时间*

*\**

*\** ***@return*** *long*

*\*/*

public static long getNowLongTime() {

return System.*currentTimeMillis*();

}

*/\*\**

*\* long类型的时间转换成 yyyyMMddHHmmss String类型的时间*

*\**

*\** ***@param*** *lo*

*\* long类型的时间*

*\** ***@return***

*\*/*

public static String longTime2StringTime(long lo) {

return *longTime2StringTime*(lo, *sd*);

}

*/\*\**

*\* long类型的时间转换成自定义时间格式*

*\**

*\** ***@param*** *lo*

*\* long类型的时间*

*\** ***@param*** *format*

*\* 时间格式*

*\** ***@return*** *String*

*\*/*

public static String longTime2StringTime(long lo, String format) {

return new SimpleDateFormat(format).format(lo);

}

*/\*\**

*\* String类型的时间转换成 long*

*\**

*\** ***@param*** *lo*

*\** ***@return*** *String*

*\** ***@throws*** *ParseException*

*\*/*

public static long stringTime2LongTime(String time, String format) throws ParseException {

if (*isEmpty*(format)) {

format = *sdf*;

}

if (*isEmpty*(time)) {

time = *getNowTime*(format);

}

SimpleDateFormat sd = new SimpleDateFormat(format);

Date date = sd.parse(time);

return date.getTime();

}

*/\*\**

*\* 格式化时间*

*\**

*\** ***@param*** *format1*

*\* 之前的 时间格式*

*\** ***@param*** *format2*

*\* 之后的 时间格式*

*\** ***@param*** *time*

*\* 时间*

*\** ***@return*** *String*

*\** ***@throws*** *ParseException*

*\*/*

public static String formatTime(String format1, String format2, String time) throws ParseException {

SimpleDateFormat d1 = new SimpleDateFormat(format1);

SimpleDateFormat d2 = new SimpleDateFormat(format2);

time = d2.format(d1.parse(time));

return time;

}

*/\*\**

*\* 时间补全 例如将2018-04-04补全为2018-04-04 00:00:00.000*

*\**

*\** ***@param*** *time*

*\* 补全的时间*

*\** ***@return***

*\*/*

public static String complementTime(String time) {

return *complementTime*(time, *sdfm*, 1);

}

*/\*\**

*\* 时间补全 例如将2018-04-04补全为2018-04-04 00:00:00.000*

*\**

*\** ***@param*** *time*

*\* 补全的时间*

*\** ***@param*** *format*

*\* 补全的格式*

*\** ***@param*** *type*

*\* 类型 1:起始;2:终止*

*\** ***@return***

*\*/*

public static String complementTime(String time, String format, int type) {

if (*isEmpty*(time) || *isEmpty*(format)) {

return null;

}

int tlen = time.length();

int flen = format.length();

int clen = flen - tlen;

if (clen <= 0) {

return time;

}

StringBuffer sb = new StringBuffer(time);

if (clen == 4) {

if (type == 1) {

sb.append(".000");

} else {

sb.append(".999");

}

} else if (clen == 9) {

if (type == 1) {

sb.append(" 00:00:00");

} else {

sb.append(" 23:59:59");

}

} else if (clen == 13) {

if (type == 1) {

sb.append(" 00:00:00.000");

} else {

sb.append(" 23:59:59.999");

}

}

return sb.toString();

}

*/\*\**

*\* 获取当前String类型的的时间 使用默认格式 yyyy-MM-dd HH:mm:ss*

*\**

*\** ***@return*** *String*

*\*/*

public static String getNowTime() {

return *getNowTime*(*sdf*);

}

*/\*\**

*\* 获取当前String类型的的时间(自定义格式)*

*\**

*\** ***@param*** *format*

*\* 时间格式*

*\** ***@return*** *String*

*\*/*

public static String getNowTime(String format) {

return new SimpleDateFormat(format).format(new Date());

}

*/\*\**

*\* 获取当前Timestamp类型的的时间*

*\**

*\** ***@return*** *Timestamp*

*\*/*

public static Timestamp getTNowTime() {

return new Timestamp(*getNowLongTime*());

}

*/\*\**

*\* 获取的String类型的当前时间并更改时间*

*\**

*\** ***@param*** *number*

*\* 要更改的的数值*

*\** ***@param*** *format*

*\* 更改时间的格式 如yyyy-MM-dd HH:mm:ss*

*\** ***@param*** *type*

*\* 更改时间的类型 时:h; 分:m ;秒:s*

*\** ***@return*** *String*

*\*/*

public static String changeTime(int number, String format, String type) {

return *changeTime*(number, format, type, "");

}

*/\*\**

*\* 获取的String类型时间并更改时间*

*\**

*\** ***@param*** *number*

*\* 要更改的的数值*

*\** ***@param*** *format*

*\* 更改时间的格式*

*\** ***@param*** *type*

*\* 更改时间的类型 。时:h; 分:m ;秒:s*

*\** ***@param*** *time*

*\* 更改的时间 没有则取当前时间*

*\** ***@return*** *String*

*\*/*

public static String changeTime(int number, String format, String type, String time) {

if (*isEmpty*(time)) { // 如果没有设置时间则取当前时间

time = *getNowTime*(format);

}

SimpleDateFormat format1 = new SimpleDateFormat(format);

Date d = null;

Calendar ca = null;

String backTime = null;

try {

d = format1.parse(time);

ca = Calendar.*getInstance*(); // 定义一个Calendar 对象

ca.setTime(d);// 设置时间

if ("h".equals(type)) {

ca.add(Calendar.*HOUR*, number);// 改变时

} else if ("m".equals(type)) {

ca.add(Calendar.*MINUTE*, number);// 改变分

} else if ("s".equals(type)) {

ca.add(Calendar.*SECOND*, number);// 改变秒

}

backTime = format1.format(ca.getTime()); // 转化为String 的格式

} catch (Exception e) {

e.printStackTrace();

}

return backTime;

}

*/\*\**

*\* 两个日期带时间比较 第二个时间大于第一个则为true，否则为false*

*\**

*\** ***@param*** *String*

*\** ***@return*** *boolean*

*\** ***@throws*** *ParseException*

*\*/*

public static boolean isCompareDay(String time1, String time2, String format) {

if (*isEmpty*(format)) {// 如果没有设置格式使用默认格式

format = *sdf*;

}

SimpleDateFormat s1 = new SimpleDateFormat(format);

Date t1 = null;

Date t2 = null;

try {

t1 = s1.parse(time1);

t2 = s1.parse(time2);

return t2.after(t1);// 当 t2 大于 t1 时，为 true，否则为 false

} catch (ParseException e) {

e.printStackTrace();

return false;

}

}

*/\*\**

*\* 获取几天之前的时间*

*\**

*\** ***@since*** *1.8*

*\** ***@param*** *day*

*\** ***@return***

*\*/*

public static String getMinusDays(int day) {

return *getMinusDays*(day, *sdf*);

}

*/\*\**

*\* 获取几天之前的时间*

*\**

*\** ***@since*** *1.8*

*\** ***@param*** *day*

*\** ***@param*** *format*

*\** ***@return***

*\*/*

public static String getMinusDays(int day, String format) {

return LocalDateTime.*now*().minusDays(day).format(DateTimeFormatter.*ofPattern*(format));

}

*/\*\**

*\* 获取几天之后的时间*

*\**

*\** ***@since*** *1.8*

*\** ***@param*** *day*

*\** ***@return***

*\*/*

public static String getPlusDays(int day) {

return *getPlusDays*(day, *sdf*);

}

*/\*\**

*\* 获取几天之后的时间*

*\**

*\** ***@since*** *1.8*

*\** ***@param*** *day*

*\** ***@param*** *format*

*\** ***@return***

*\*/*

public static String getPlusDays(int day, String format) {

return LocalDateTime.*now*().plusDays(day).format(DateTimeFormatter.*ofPattern*(format));

}

*/\*\**

*\* 获取几天之后的时间*

*\**

*\** ***@since*** *1.8*

*\** ***@param*** *day*

*\** ***@return***

*\*/*

public static String getPlusMonths(int month) {

return *getPlusMonths*(month, *sdf*);

}

*/\*\**

*\* 获取几月之后的时间*

*\**

*\** ***@since*** *1.8*

*\** ***@param*** *day*

*\** ***@param*** *format*

*\** ***@return***

*\*/*

public static String getPlusMonths(int month, String format) {

return LocalDateTime.*now*().plusMonths(month).format(DateTimeFormatter.*ofPattern*(format));

}

*/\*\**

*\* 增加月份*

*\**

*\** ***@param*** *time*

*\* 格式为yyyy-MM-dd*

*\** ***@param*** *month*

*\* 增加月份*

*\** ***@return***

*\*/*

public static String addPlusMonths(String time, int month) {

return LocalDate.*parse*(time).plusMonths(month).toString();

}

*/\*\**

*\* 时间相比得月份 如果是201711和201801相比，返回的结果是2 前面的时间要小于后面的时间*

*\**

*\** ***@param*** *month*

*\* 格式为yyyyMM*

*\** ***@param*** *toMonth*

*\* 格式为yyyyMM*

*\** ***@since*** *jdk 1.8*

*\** ***@return***

*\*/*

public static int diffMonth(String month, String toMonth) {

int year1 = Integer.*parseInt*(month.substring(0, 4));

int month1 = Integer.*parseInt*(month.substring(4, 6));

int year2 = Integer.*parseInt*(month.substring(0, 4));

int month2 = Integer.*parseInt*(month.substring(4, 6));

LocalDate ld1 = LocalDate.*of*(year1, month1, 01);

LocalDate ld2 = LocalDate.*of*(year2, month2, 01);

return Period.*between*(ld1, ld2).getMonths();

}

*/\*\**

*\* 判断是否为整型*

*\**

*\** ***@param*** *String*

*\** ***@return*** *boolean*

*\*/*

public static boolean isInteger(String str) {

Matcher m = *p*.matcher(str);

return m.find();

}

*/\*\**

*\* 自定义位数产生随机数字*

*\**

*\** ***@param*** *int*

*\** ***@return*** *String*

*\*/*

public static String random(int count) {

char start = '0';

char end = '9';

Random rnd = new Random();

char[] result = new char[count];

int len = end - start + 1;

while (count-- > 0) {

result[count] = (char) (rnd.nextInt(len) + start);

}

return new String(result);

}

*/\*\**

*\* 获取自定义长度的随机数(含字母)*

*\**

*\** ***@param*** *len*

*\* 长度*

*\** ***@return*** *String*

*\*/*

public static String random2(int len) {

int random = Integer.*parseInt*(*random*(5));

Random rd = new Random(random);

final int maxNum = 62;

StringBuffer sb = new StringBuffer();

int rdGet;// 取得随机数

char[] str = { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's',

't', 'u', 'v', 'w', 'x', 'y', 'z', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N',

'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', '0', '1', '2', '3', '4', '5', '6', '7', '8',

'9' };

int count = 0;

while (count < len) {

rdGet = Math.*abs*(rd.nextInt(maxNum));// 生成的数最大为62-1

if (rdGet >= 0 && rdGet < str.length) {

sb.append(str[rdGet]);

count++;

}

}

return sb.toString();

}

*/\*\**

*\* 获取本机ip*

*\**

*\** ***@return*** *String*

*\** ***@throws*** *UnknownHostException*

*\*/*

public static String getLocalHostIp() throws UnknownHostException {

return InetAddress.*getLocalHost*().getHostAddress();

}

*/\*\**

*\* Object 转换为 String*

*\**

*\** ***@param*** *list*

*\** ***@return*** *String*

*\*/*

public static String toString(Object obj) {

return JSON.*toJSONString*(obj);

}

*/\*\**

*\* JSON 转换为 JavaBean*

*\**

*\** ***@param*** *json*

*\** ***@param*** *t*

*\** ***@return*** *<T>*

*\*/*

public static <T> T toBean(JSONObject json, Class<T> t) {

return JSON.*toJavaObject*(json, t);

}

*/\*\**

*\* JSON 字符串转换为 JavaBean*

*\**

*\** ***@param*** *str*

*\** ***@param*** *t*

*\** ***@return*** *<T>*

*\*/*

public static <T> T toBean(String str, Class<T> t) {

return JSON.*parseObject*(str, t);

}

*/\*\**

*\* JSON 字符串 转换成JSON格式*

*\**

*\** ***@param*** *obj*

*\** ***@return*** *JSONObject*

*\*/*

public static JSONObject toJson(String str) {

if (*isEmpty*(str)) {

return new JSONObject();

}

return JSON.*parseObject*(str);

}

*/\*\**

*\* JavaBean 转化为JSON*

*\**

*\** ***@param*** *t*

*\** ***@return***

*\*/*

public static JSONObject toJson(Object t) {

if (null == t || "".equals(t)) {

return new JSONObject();

}

return (JSONObject) JSON.*toJSON*(t);

}

*/\*\**

*\* JSON 字符串转换为 HashMap*

*\**

*\** ***@param*** *json*

*\* - String*

*\** ***@return*** *Map*

*\*/*

@SuppressWarnings("rawtypes")

public static Map toMap(String json) {

if (*isEmpty*(json)) {

return new HashMap();

}

return JSON.*parseObject*(json, HashMap.class);

}

*/\*\**

*\* 将map转化为string*

*\**

*\** ***@param*** *m*

*\** ***@return***

*\*/*

@SuppressWarnings("rawtypes")

public static String toString(Map m) {

return JSONObject.*toJSONString*(m);

}

*/\*\**

*\* String转换为数组*

*\**

*\** ***@param*** *text*

*\** ***@return***

*\*/*

public static <T> Object[] toArray(String text) {

return *toArray*(text, null);

}

*/\*\**

*\* String转换为数组*

*\**

*\** ***@param*** *text*

*\** ***@return***

*\*/*

public static <T> Object[] toArray(String text, Class<T> clazz) {

return JSON.*parseArray*(text, clazz).toArray();

}

*/\*\**

*\* name1=value1&name2=value2格式的数据转换成json数据格式*

*\**

*\** ***@param*** *str*

*\** ***@return***

*\*/*

public static JSONObject str2Json(String str) {

if (*isEmpty*(str)) {

return new JSONObject();

}

JSONObject json = new JSONObject();

String[] str1 = str.split("&");

String str3 = "", str4 = "";

if (null == str1 || str1.length == 0) {

return new JSONObject();

}

for (String str2 : str1) {

str3 = str2.substring(0, str2.lastIndexOf("="));

str4 = str2.substring(str2.lastIndexOf("=") + 1, str2.length());

json.put(str3, str4);

}

return json;

}

*/\*\**

*\* json数据格式 转换成name1=value1&name2=value2格式*

*\**

*\** ***@param*** *str*

*\** ***@return***

*\*/*

@SuppressWarnings("rawtypes")

public static String json2Str(JSONObject json) {

if (*isEmpty*(json)) {

return null;

}

StringBuffer sb = new StringBuffer();

Iterator it = json.entrySet().iterator(); // 定义迭代器

while (it.hasNext()) {

Map.Entry er = (Entry) it.next();

sb.append(er.getKey());

sb.append("=");

sb.append(er.getValue());

sb.append("&");

}

sb.delete(sb.length() - 1, sb.length()); // 去掉最后的&

return sb.toString();

}

*/\*\**

*\* 将JDBC查询的数据转换成List类型*

*\**

*\** ***@param*** *ResultSet*

*\** ***@return*** *List*

*\** ***@throws*** *SQLException*

*\*/*

@SuppressWarnings({ "rawtypes", "unchecked" })

public static List convertList(ResultSet rs) throws SQLException {

if (null == rs) {

return new ArrayList<>();

}

List list = new ArrayList();

ResultSetMetaData md = rs.getMetaData();

int columnCount = md.getColumnCount();

while (rs.next()) {

JSONObject rowData = new JSONObject();

for (int i = 1; i <= columnCount; i++) {

rowData.put(md.getColumnName(i), rs.getObject(i));

}

list.add(rowData);

}

return list;

}

*/\*\**

*\* MD5加密*

*\**

*\** ***@param*** *message*

*\** ***@return***

*\*/*

public static String md5Encode(String message) {

byte[] secretBytes = null;

try {

secretBytes = MessageDigest.*getInstance*("md5").digest(message.getBytes());

} catch (NoSuchAlgorithmException e) {

throw new RuntimeException("没有md5这个算法！");

}

String md5code = new BigInteger(1, secretBytes).toString(16);// 16进制数字

// 如果生成数字未满32位，需要前面补0

int length = 32 - md5code.length();

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

md5code = "0" + md5code;

}

return md5code;

}

*/\*\**

*\* base64 加密*

*\**

*\** ***@param*** *str*

*\** ***@return***

*\*/*

public static String base64En(String str) {

Base64 base64 = new Base64();

byte[] encode = base64.encode(str.getBytes());

return new String(encode);

}

*/\*\**

*\* base64解密*

*\**

*\** ***@param*** *encodeStr*

*\** ***@return***

*\*/*

@SuppressWarnings("static-access")

public static String base64De(String encodeStr) {

Base64 base64 = new Base64();

byte[] decodeStr = base64.*decodeBase64*(encodeStr);

return new String(decodeStr);

}

*/\*\**

*\* 十进制转二进制*

*\**

*\** ***@param*** *n*

*\** ***@return***

*\*/*

public static String decToBinary(int n) {

String str = "";

while (n != 0) {

str = n % 2 + str;

n = n / 2;

}

return str;

}

*/\*\**

*\* 二进制转十进制*

*\**

*\** ***@param*** *n*

*\** ***@return***

*\*/*

public static int binaryToDec(char[] cs) {

return *binaryToDec*(cs);

}

*/\*\**

*\* 二进制转十进制*

*\**

*\** ***@param*** *n*

*\** ***@return***

*\*/*

public static int binaryToDec(String cs) {

return new BigInteger(new String(cs), 2).intValue();

}

*/\*\**

*\* 本方法的测试示例*

*\**

*\** ***@param*** *args*

*\*/*

@SuppressWarnings({ "rawtypes" })

public static void main(String[] args) {

/\*

\* String 和List 空数据判断

\*/

String str1 = "";

String str2 = " ";

String str3 = null;

String str4 = "a";

List list = null;

List<String> list2 = new ArrayList<String>();

List<Object> list3 = new ArrayList<Object>();

list3.add("a");

System.*out*.println("str1 :" + *isEmpty*(str1)); // str1 :true

System.*out*.println("str2 :" + *isEmpty*(str2)); // str2 :true

System.*out*.println("str3 :" + *isEmpty*(str3)); // str3 :true

System.*out*.println("str4 :" + *isEmpty*(str4)); // str4 :false

System.*out*.println("list :" + *isEmpty*(list)); // list :true

System.*out*.println("list2 :" + *isEmpty*(list2)); // list2 :true

System.*out*.println("list3 :" + *isEmpty*(list3)); // list3 :false

/\*

\* 时间

\*/

long start = *getNowLongTime*();

System.*out*.println("getNowTime():" + *getNowTime*()); // getNowTime():2017-09-26

// 17:46:44

System.*out*.println("getNowLongTime():" + *getNowLongTime*()); // getNowLongTime():1506419204920

System.*out*.println("getNowTime(sdfm):" + *getNowTime*(*sdfm*)); // getNowTime(sdfm):2017-09-26

// 17:46:44

// 920

System.*out*.println("当时时间向前推移30秒:" + *changeTime*(-30, *sdf*, "s")); // 2017-09-26

// 17:46:14

System.*out*.println("时间比较:" + *isCompareDay*(*getNowTime*(*sdfm*), *changeTime*(-30, *sdf*, "s"), "")); // 时间比较:false

System.*out*.println("getTNowTime():" + *getTNowTime*()); // getTNowTime():2017-09-26

// 17:46:44.921

System.*out*.println("LongTime2StringTime():" + *longTime2StringTime*(start, *sd*)); // LongTime2StringTime():20170926174644

/\*

\* 整型判断

\*/

String st = "258369";

String st2 = "258369A!@";

String st3 = "258 369 ";

System.*out*.println("st:" + *isInteger*(st)); // st:true

System.*out*.println("st2:" + *isInteger*(st2)); // st2:false

System.*out*.println("st3:" + *isInteger*(st3)); // st3:false

/\*

\* 字符串反转

\*/

String re = "abcdefg";

System.*out*.println("字符串反转:" + *reverse*(re)); // 字符串反转:gfedcba

/\*

\* 本机IP

\*/

try {

System.*out*.println("本机IP:" + *getLocalHostIp*()); // 本机IP:192.168.1.111

} catch (UnknownHostException e) {

e.printStackTrace();

}

/\*

\* 随机数

\*/

System.*out*.println("6位随机数:" + *random*(6)); // 6位随机数:222488

System.*out*.println("10位随机数:" + *random2*(10)); // 10位随机数:ZwW0pmofjW

/\*

\* JSON数据转换

\*/

String value = "name1=value1&name2=value2&name3=value3";

JSONObject json = new JSONObject();

json.put("name1", "value1");

json.put("name2", "value2");

json.put("name3", "value3");

System.*out*.println("value:" + value); // value:name1=value1&name2=value2&name3=value3

System.*out*.println("str2Json:" + *str2Json*(value)); // str2Json:{"name1":"value1","name2":"value2","name3":"value3"}

System.*out*.println("json:" + json.toJSONString()); // json:{"name1":"value1","name2":"value2","name3":"value3"}

System.*out*.println("json2Str:" + *json2Str*(json)); // json2Str:name3=value3&name1=value1&name2=value2

String jsonString = json.toJSONString();

System.*out*.println("jsonString:" + jsonString); // {"name1":"value1","name2":"value2","name3":"value3"}

System.*out*.println("toJson(jsonString):" + *toJson*(jsonString)); // toJson(jsonString):{"name1":"value1","name2":"value2","name3":"value3"}

System.*out*.println("long TO String" + *longTime2StringTime*(32472115200L));

System.*out*.println("long TO String" + *longTime2StringTime*(1513330097L));

String time1 = "2018-04-04";

String time2 = "2018-04-04 14:48:00";

String time3 = "2018-04-04 14:48:00.000";

System.*out*.println("时间补全:" + *complementTime*(time1, *sdfm*, 1));

System.*out*.println("时间补全:" + *complementTime*(time2, *sdfm*, 2));

System.*out*.println("时间补全:" + *complementTime*(time3, *sdfm*, 1));

String time4 = *addPlusMonths*(time1, 2);

System.*out*.println("增加之前的数据:" + time1 + "增加月份之后的数据:" + time4);

System.*out*.println("相差月份:" + *diffMonth*("201711", "201801"));

int l = 2;

String string = "10101";

System.*out*.println(l + " 十进制转二进制: " + *decToBinary*(l));

System.*out*.println(string + " 二进制转十进制: " + *binaryToDec*(string));

}

}