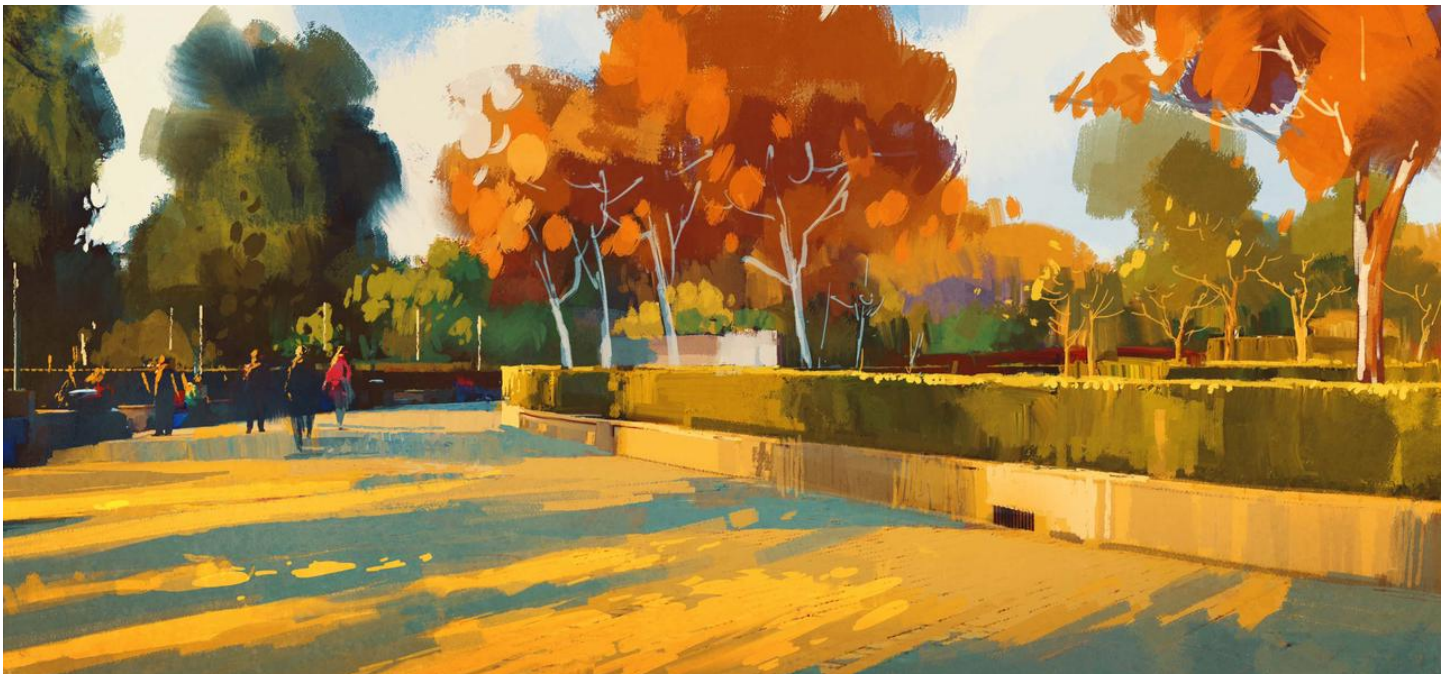

Hutool 常用工具类

介绍

Hutool是一个的Java工具类库，通过静态方法封装，降低相关API的学习成本，提高工作效率，它节省了开发人员对项目中公用类和公用工具方法的封装时间，同时可以最大限度的避免封装不完善带来的bug。

安装

Maven:在项目的pom.xml的dependencies中加入以下内容:

pom.xml

```
<dependency>
    <groupId>cn.hutool</groupId>
    <artifactId>hutool-all</artifactId>
```

```
    <version>5.8.37</version>
</dependency>
```

官网

https://hutool.cn/

常用工具类

1. 树形结构工具类 TreeUtil

构建Tree示例

代码块

```
package com.sl.transport.common.util;

import cn.hutool.core.bean.BeanUtil;
import cn.hutool.core.collection.CollUtil;
import cn.hutool.core.lang.tree.Tree;
import cn.hutool.core.lang.tree.TreeNode;
import cn.hutool.core.lang.tree.TreeNodeConfig;
import cn.hutool.core.lang.tree.TreeUtil;
import cn.hutool.json.JSONUtil;
import lombok.extern.slf4j.Slf4j;
import org.junit.jupiter.api.Test;

import java.util.List;

@Slf4j
class TreeUtilTest {

    @Test
    void treeTest() {
        // 构建node列表
        List<TreeNode<String>> nodeList = CollUtil.newArrayList();

        nodeList.add(new TreeNode<>("1", "0", "系统管理", 5));
        nodeList.add(new TreeNode<>("11", "1", "用户管理", 222222));
        nodeList.add(new TreeNode<>("111", "11", "用户添加", 0));
        nodeList.add(new TreeNode<>("2", "0", "店铺管理", 1));
        nodeList.add(new TreeNode<>("21", "2", "商品管理", 44));
```

```
        nodeList.add(new TreeNode<>("221", "2", "添加商品", 2));

        //配置
        TreeNodeConfig treeNodeConfig = new TreeNodeConfig();
        // 自定义属性名
        treeNodeConfig.setWeightKey("weight");
        treeNodeConfig.setIdKey("id");
        //构造树结构
        List<Tree<String>> treeNodes = TreeUtil.build(nodeList, "0",
treeNodeConfig,
                (treeNode, tree) -> {
                    tree.putAll(BeanUtil.beanToMap(treeNode));
                });

        log.info("treeNodes {}", JSONUtil.toJsonStr(treeNodes));
    }
}

```

输出结果

代码块

```
[{
        "id": "2",
        "parentId": "0",
        "name": "店铺管理",
        "weight": 1,
        "children": [{
                "id": "221",
                "parentId": "2",
                "name": "添加商品",
                "weight": 2
        }, {
                "id": "21",
                "parentId": "2",
                "name": "商品管理",
                "weight": 44
        }]
}, {
        "id": "1",
        "parentId": "0",
        "name": "系统管理",
        "weight": 5,
        "children": [{
                "id": "11",
```

```
                "parentId": "1",
                "name": "用户管理",
                "weight": 222222,
                "children": [{
                        "id": "111",
                        "parentId": "11",
                        "name": "用户添加",
                        "weight": 0
                }]
        }]
}]

```

2. Bean转换工具类 BeanUtil

isEmpty和isNotEmpty

判断对象是否为null或者所有属性都为null

代码块

```
// 将user中所有属性赋值给UserDTO已有属性
UserDTO userDTO = BeanUtil.copyProperties(user, UserDTO.class);

//bean转map
Map<String, Object> map = BeanUtil.beanToMap(person)

//map转bean

// 方式一
UserDTO userDTOC = BeanUtil.fillBeanWithMap(userMap, new UserDTO, false);
// 方式二
UserDTO userDTO = BeanUtil.fillBeanWithMap(userMap, UserDTO.class,
false).newInstance();
```

3. 类型转换工具类 Convert

Convert类可以说是一个工具方法类，里面封装了针对Java常见类型的转换，用于简化类型转换。Convert类中大部分方法为toXXX，参数为Object，可以实现将任意可能的类型转换为指定类型。同时支持第二个参数defaultValue用于在转换失败时返回一个默认值

代码块

```
//转换为字符串
int a = 1;
String aStr = Convert.toStr(a);
```

```
//转换为指定类型数组
String[] b = { "1", "2", "3", "4" };
Integer[] intArray = Convert.toIntArray(b);
//转换为日期对象
String dateStr = "2022-09-28";
Date date = Convert.toDate(dateStr);
//转换为列表
String[] strArr = {"a", "b", "c", "d"};
List<String> strList = Convert.toList(String.class, strArr);
Object[] a = {"a", "你", "好", "", 1};
List<?> list = Convert.convert(List.class, a);
//从4.1.11开始可以这么用
List<?> list = Convert.toList(a);

```

4. 表格操作工具类 ExcelUtil

表格数据导入

代码块

```
@PostMapping("/upload")
public Result upload(MultipartFile file) throws IOException {
    //Hutool 工具类 ExcelUtil 创建一个 Excel 读取器，将文件输入流传递给它
    //调用读取器的 readAll 方法，将 Excel 文件中的所有数据读取并转换为 Type 类型的对象列表
    List<Type> infoList = ExcelUtil.getReader(file.getInputStream()).readAll(Type.class);
    if (!CollectionUtil.isEmpty(infoList)) {
        for (Type type : infoList) {
            try {
                typeService.add(type);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
    return Result.success();
}
```

tip: 实体类上面需要加@Alias("分类名称"), 否则没法自动映射

表格数据导出

代码块

```
@GetMapping("/export")
public void export(HttpServletResponse response) throws IOException {
    List<Type> all = typeService.findAll();
    if (CollectionUtil.isEmpty(all)) {
        throw new CustomException("没有数据");
    }
    List<Map<String, Object>> list = new ArrayList<>();
    for (Type type : all) {
        Map<String, Object> row = new LinkedHashMap<>();
        row.put("分类名称", type.getName());
        row.put("分类描述", type.getDescription());
        list.add(row);
    }

    //创建一个 Excel 写入器
    ExcelWriter writer = ExcelUtil.getWriter(true);
    //将转换后的数据列表 list 写入到 Excel 文件中，true 表示自动合并表头。
    writer.write(list, true);

    response.setContentType("application/vnd.openxmlformats-officedocument.spreadsheetml.");
    response.setHeader("Content-Disposition", "attachment;filename=type.xlsx");

    //获取响应的输出流，用于将 Excel 文件数据发送给客户端。
    ServletOutputStream out = response.getOutputStream();
    // 将 Excel 写入器中的数据刷新到输出流中
    writer.flush(out, true);
    //关闭 Excel 写入器
    writer.close();
}
```

5. 日期时间工具类 DateUtil（LocalDateTimeUtil）

Date、long、Calendar之间的相互转换

代码块

```
//获取当前时间
Date date = DateUtil.date();
//Calendar类型时间转为Date
Date date2 = DateUtil.date(Calendar.getInstance());
//时间戳转为Date
Date date3 = DateUtil.date(System.currentTimeMillis());
//当前时间字符串，格式：yyyy-MM-dd HH:mm:ss
String now = DateUtil.now();
//当前日期字符串，格式：yyyy-MM-dd
```

```
String today= DateUtil.today();
// 时间格式转为时间戳
Date date1 = DateUtil.parse("2023-06-21");
long time = date1.getTime();
System.out.println("time = " + time);

```

开始和结束时间

有的时候我们需要获得每天的开始时间、结束时间，每月的开始和结束时间等等，DateUtil也提供了相关方法：

代码块

```
//字符串类型的日期转换为data类型的
String dateStr = "2017-03-01 22:33:23";
Date date = DateUtil.parse(dateStr);

//一天的开始，结果：2017-03-01 00:00:00
Date beginOfDay = DateUtil.beginOfDay(date);

//一天的结束，结果：2017-03-01 23:59:59
Date endOfDay = DateUtil.endOfDay(date);

```

日期时间偏移

日期或时间的偏移指针对某个日期增加或减少分、小时、天等等，达到日期变更的目的。Hutool也针对其做了大量封装

代码块

```
String dateStr = "2017-03-01 22:33:23";
Date date = DateUtil.parse(dateStr);

//结果：2017-03-03 22:33:23
Date newDate = DateUtil.offset(date, DateField.DAY_OF_MONTH, 2);

//常用偏移，结果：2017-03-04 22:33:23
DateTime newDate2 = DateUtil.offsetDay(date, 3);

//常用偏移，结果：2017-03-01 19:33:23
DateTime newDate3 = DateUtil.offsetHour(date, -3);

```

星座和属相

```
代码块
1   // "摩羯座"
2   String zodiac = DateUtil.getZodiac(Month.JANUARY.getValue(), 19);
3
4   // "狗"
5   String chineseZodiac = DateUtil.getChineseZodiac(1994);
```

其他

```
代码块
1   //年龄
2   DateUtil.ageOfNow("1990-01-30");
3
4   //是否闰年
5   DateUtil.isLeapYear(2017);
6
7   //获得指定月份的总天数
8   DateUtil.lengthOfMonth(6,false);
```

6. 数组工具类 ArrayUtil

1. 判断空

```
代码块
1   int[] a = {};
2   int[] b = null;
3   ArrayUtil.isEmpty(a);
4   ArrayUtil.isEmpty(b);
```

2. 判断非空

```
代码块
1   int[] a = {1,2};
2   ArrayUtil.isNotEmpty(a);
```

3. 调整大小

使用 `ArrayUtil.resize` 方法生成一个新的重新设置大小的数组。

4. 合并数组

`ArrayUtil.addAll` 方法采用可变参数方式，将多个泛型数组合并为一个数组。

5. 克隆

数组本身支持clone方法，因此确定为某种类型数组时调用 `ArrayUtil.clone(T[])` ,不确定类型的使用 `ArrayUtil.clone(T)` ，两种重载方法在实现上有所不同，但是在使用中并不能感知出差别。

6. 拆分数组

`ArrayUtil.split` 方法用于拆分一个byte数组，将byte数组平均分成几等份，常用于消息拆分

7. 过滤

`ArrayUtil.filter` 方法用于过滤已有数组元素，只针对泛型数组操作，原始类型数组并未提供。 方法中Filter接口用于返回boolean值决定是否保留。

8. 编辑

修改元素对象,此方法会修改原数组

代码块

```
Integer[] a = {1, 2, 3, 4, 5, 6};
// [1, 20, 3, 40, 5, 60]
ArrayUtil.edit(a, t -> (t % 2 == 0) ? t * 10 : t);

```

9. 是否包含元素

`ArrayUtil.contains` 方法只针对泛型数组，检测指定元素是否在数组中

10. 包装和拆包

在原始类型元素和包装类型中，Java实现了自动包装和拆包，但是相应的数组无法实现，于是便是用 `ArrayUtil.wrap` 和 `ArrayUtil.unwrap` 对原始类型数组和包装类型数组进行转换。

11. 判断对象是否为数组

`ArrayUtil.isArray` 方法封装了 `obj.getClass().isArray()`

12. 转为字符串

`ArrayUtil.toString` 通常原始类型的数组输出为字符串时无法正常显示，于是封装此方法可以完美兼容原始类型数组和包装类型数组的转为字符串操作

`ArrayUtil.join` 方法使用间隔符将一个数组转为字符串，比如[1,2,3,4]这个数组转为字符串，间隔符使用“-”的话，结果为 1-2-3-4，join方法同样支持泛型数组和原始类型数组

13. toArray

`ArrayUtil.toArray` 方法针对ByteBuffer转数组提供便利。

7. 集合工具类 CollUtil

代码块

```
import cn.hutool.core.collection.CollUtil;
import java.util.List;

public class CollUtilExample {
    public static void main(String[] args) {
        List<String> list1 = CollUtil.newArrayList("a", "b", "c");
        List<String> list2 = CollUtil.newArrayList("d", "e");

        // 合并集合
        List<String> combined = CollUtil.union(list1, list2);
        System.out.println(combined);  // ["a", "b", "c", "d", "e"]

        // 去重集合
        List<String> listWithDuplicates = CollUtil.newArrayList("a", "b", "a", "c");
        List<String> uniqueList = CollUtil.removeDuplicate(listWithDuplicates);
        System.out.println(uniqueList);  // ["a", "b", "c"]

        // 判断集合是否为空
        boolean isEmpty = CollUtil.isEmpty(list1);
        System.out.println(isEmpty);  // false
    }
}
```

8. 唯一ID工具类 IdUtil

唯一ID生成器的工具类，涵盖了：

- UUID
- ObjectId（MongoDB）

- Snowflake（Twitter）

UUID全称通用唯一识别码（universally unique identifier），JDK通过java.util.UUID提供了 Leach-Salz 变体的封装。在Hutool中，生成一个UUID字符串方法如下

代码块

```
//生成的UUID是带-的字符串，类似于：a5c8a5e8-df2b-4706-bea4-08d0939410e3
String uuid = IdUtil.randomUUID();

//生成的是不带-的字符串，类似于：b17f24ff026d40949c85a24f4f375d42
String simpleUUID = IdUtil.simpleUUID();
```

全局唯一性限制

- 简单的 id 通常是在单个系统或数据库中自增长的整数，在分布式系统或多个系统集成的环境中，很难保证在不同系统之间的唯一性。而UUID能确保在全球范围内的唯一性，可用于跨系统、跨网络的资源标识。

安全性问题

- 自增的 id 容易被猜测和遍历，存在一定的安全风险。例如，攻击者可能通过猜测 id 值来获取敏感信息。UUID的随机性和唯一性使其难以被猜测和伪造，能提高系统的安全性。

可读性与业务无关性

- 简单的 id 可能与业务逻辑相关，如订单系统中订单 id 可能按照一定规则与业务流程关联。这在业务复杂或系统扩展时可能带来限制。UUID是无意义的字符串，与业务逻辑无关，不会因业务变化而受到影响，具有更好的通用性和适应性。

性能考虑

- 在大规模数据处理中，自增 id 在插入数据时可能导致数据库索引的频繁更新，影响性能。UUID虽然占用空间比简单 id 大，但在某些数据库中，可通过优化策略来提高性能，而且在分布式环境下的整体性能表现更优

9. 身份证工具类 IdcardUtil

工具中主要的方法包括：

isValidCard 验证身份证是否合法

convert15To18 身份证15位转18位

getBirthByIdCard 获取生日

getAgeByIdCard 获取年龄

getYearByIdCard 获取生日年

getMonthByIdCard 获取生日月

getDayByIdCard 获取生日天

getGenderByIdCard 获取性别

getProvinceByIdCard 获取省份

10. JWT工具类 JWTUtil

JWT创建

代码块

```
Map<String, Object> map = new HashMap<String, Object>() {
    private static final long serialVersionUID = 1L;
    {
        put("uid", Integer.parseInt("123"));
        put("expire_time", System.currentTimeMillis() + 1000 * 60 * 60 * 24 *
15);
    }
};

JWTUtil.createToken(map, "1234".getBytes());
```

JWT解析

代码块

```
String rightToken = "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9." +
    "eyJzdWIiOiIxMjM0NTY3ODkwIiwiYWRtaW4iOnRydWUsIm5hbWUiOiJsb29seSJ9." +
    "U2aQkC2THYV9L0fTN-yBBI7gmo5xhmvMhATtu8v0zEA";

final JWT jwt = JWTUtil.parseToken(rightToken);

jwt.getHeader(JWTHeader.TYPE);
jwt.getPayload("sub");
```

JWT验证

```
String token = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9." +
    "eyJ1c2VyX25hbWUiOiJhZG1pbiIsInNjb3BlIjpbImFsbCJdLCJleHAiOjE2MjQwMDQ4MjIsInVzZXJJZCI6MSwiYXV0aG9yaXRpZXMiOlsiUk9MRV_op5LoibLkuozlj7ciLCJzeXNfbWVudV8xIiwiUk9MRV_op5LoibLkuIDlj7ciLCJzeXNfbWVudV8yIl0sImp0aSI6ImQ0YzVlYjgwLTA5ZTctNGU0ZC1hZTg3LTVkNGI5M2FhNmFiNiIsImNsaWVudF9pZCI6ImhhbmR5LXNob3AifQ." +
        "aixF1eKlAKS_k3ynFnStE7-IRGiD5YaqznvK2xEjBew";

JWTUtil.verify(token, "123456".getBytes());
```

11. 二维码工具类 QrCodeUtil

引入zxing

考虑到Hutool的非强制依赖性，因此zxing需要用户自行引入：

代码块

```
<dependency>
    <groupId>com.google.zxing</groupId>
    <artifactId>core</artifactId>
    <version>3.3.3</version>
</dependency>
```

生成二维码

代码块

```
// 生成指定url对应的二维码到文件，宽和高都是300像素
QrCodeUtil.generate("https://blog.csdn.net/abst122", 300, 300,
FileUtil.file("d:/qrcode.jpg"));
```

自定义参数（since 4.1.2）

1. 基本参数设定
   通过QrConfig可以自定义二维码的生成参数，例如长、宽、二维码的颜色、背景颜色、边距等参数，使用方法如下：

代码块

```
QrConfig config = new QrConfig(300, 300);
// 设置边距，既二维码和背景之间的边距
config.setMargin(3);
// 设置前景色，既二维码颜色（青色）
config.setForeColor(Color.CYAN.getRGB());
```

```
// 设置背景色（灰色）
config.setBackColor(Color.GRAY.getRGB());

// 生成二维码到文件，也可以到流
QrCodeUtil.generate("https://blog.csdn.net/abst122", config,
FileUtil.file("e:/qrcode.jpg"));
```

2. 附带logo小图标

代码块

```
QrCodeUtil.generate(//
    "https://blog.csdn.net/abst122", //二维码内容
    QrConfig.create().setImg("e:/logo_small.jpg"), //附带logo
    FileUtil.file("e:/qrcodeWithLogo.jpg")//写出到的文件
);
```

4. 调整纠错级别

很多时候，二维码无法识别，这时就要调整纠错级别。纠错级别使用zxing的ErrorCorrectionLevel枚举封装，包括：L、M、Q、H几个参数，由低到高。低级别的像素块更大，可以远距离识别，但是遮挡就会造成无法识别。高级别则相反，像素块小，允许遮挡一定范围，但是像素块更密集。

代码块

```
QrConfig config = new QrConfig();
// 高纠错级别
config.setErrorCorrection(ErrorCorrectionLevel.H);
QrCodeUtil.generate("https://blog.csdn.net/abst122", config,
FileUtil.file("e:/qrcodeCustom.jpg"));
```

识别二维码

代码块

```
// decode -> "https://blog.csdn.net/abst122"
String decode = QrCodeUtil.decode(FileUtil.file("d:/qrcode.jpg"));
```

12. 电话号码工具类 PhoneUtil

常用方法:

static boolean isMobile(CharSequence value)验证是否为手机号码（中国大陆）

static boolean isMobileHk(CharSequence value)验证是否为手机号码（中国香港）

static boolean isMobileMo(CharSequence value)验证是否为手机号码（中国澳门）

static boolean isMobileTw(CharSequence value)验证是否为手机号码（中国台湾）

static boolean isPhone(CharSequence value)验证是否为座机号码+手机号码（CharUtil中国）+ 400 + 800电话 + 手机号号码（中国香港）

代码块

```
if(!phone){
        return Wrapper.fail("输入的手机号存在非法字符，请重新输入。");
}
```

13. 随机数工具类 RandomUtil

`RandomUtil.randomInt` 获得指定范围内的随机数

例如我们想产生一个[10, 100)的随机数，则：

代码块

```
int c = RandomUtil.randomInt(10, 100);
```

`RandomUtil.randomBytes` 随机bytes，一般用于密码或者salt生成

代码块

```
byte[] c = RandomUtil.randomBytes(10);
```

`RandomUtil.randomEle` 随机获得列表中的元素

`RandomUtil.randomEleSet` 随机获得列表中的一定量的不重复元素，返回LinkedHashSet

代码块

```
Set<Integer> set = RandomUtil.randomEleSet(CollUtil.newArrayList(1, 2, 3, 4, 5, 6), 2);
```

此外，

`RandomUtil.randomString` 获得一个随机的字符串（只包含数字和字符）

`RandomUtil.randomNumbers` 获得一个只包含数字的字符串

`RandomUtil.weightRandom` 权重随机生成器，传入带权重的对象，然后根据权重随机获取对象

14. 敏感词工具类 SensitiveUtil

代码块

```
List<String> sensitiveWords = new ArrayList<>();
sensitiveWords.add("敏感词1");
sensitiveWords.add("敏感词2");
sensitiveWords.add("敏感词3");

SensitiveUtil.init(sensitiveWords);
String text = "这是一段包含敏感词的文本，包含关键字：敏感词1、敏感词2、敏感词3";
String filteredText = SensitiveUtil.sensitiveFilter(text);
System.out.println(filteredText);
```

代码块

```
import java.util.ArrayList;
import java.util.List;
public class SensitiveUtil {
    private static List<String> sensitiveWords = new ArrayList<>();
    public static void init(List<String> words) {
        sensitiveWords.addAll(words);
    }
    public static String sensitiveFilter(String text) {
        for (String word : sensitiveWords) {
            text = text.replace(word, "*".repeat(word.length()));
        }
        return text;
    }
}
```

更多精彩尽在 https://hutool.cn/