# 官网

https://www.hutool.cn/

# 参考文档

https://doc.hutool.cn/pages/index/#%F0%9F%93%9A%E7%AE%80%E4%BB%8B

# maven依赖

<dependency>  
 <groupId>cn.hutool</groupId>  
 <artifactId>hutool-http</artifactId>  
 <version>5.8.26</version>  
</dependency>

# 树型结构

## TreeNode类

# 类型转换

## Convert类

### public static List<?> toList(Object value)

转换为ArrayList，元素类型默认Object

# 日期时间

## DateUtil类

### 基本概念

针对日期时间操作提供一系列静态方法

### 方法

#### public static DateTime date()

返回当前时间

#### public static String now()

当前时间，格式 yyyy-MM-dd HH:mm:ss

#### public static String today()

当前日期，格式 yyyy-MM-dd

#### public static DateTime parse(CharSequence dateCharSequence)

将日期字符串转换为DateTime对象

#### public static DateTime parse(CharSequence dateStr, String format)

#### public static DateTime parseDate(CharSequence dateString)

#### public static String format(Date date, DateFormat format)

根据特定格式格式化日期

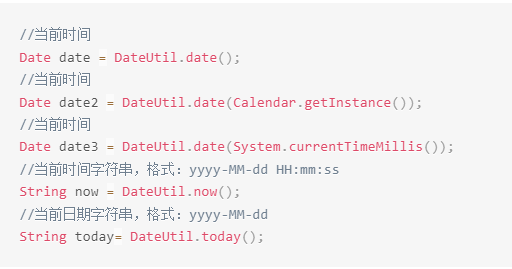
#### public static DateTime beginOfDay(Date date)

获取某天的开始时间

#### public static DateTime endOfDay(Date date)

获取某天的结束时间

### Date、long、Calendar之间的相互转换



### 字符串转日期

DateUtil.parse方法会自动识别一些常用格式，包括：

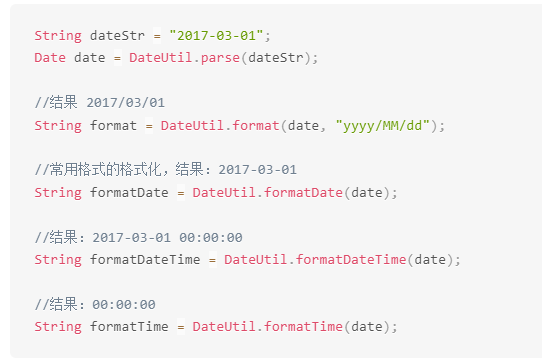




自定义日期格式转化



### 格式化日期输出

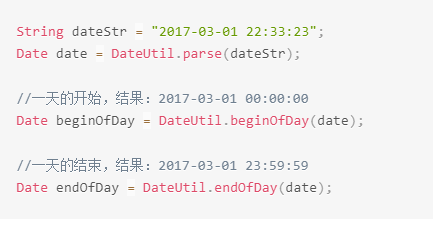


### 获取Date对象的某个部分



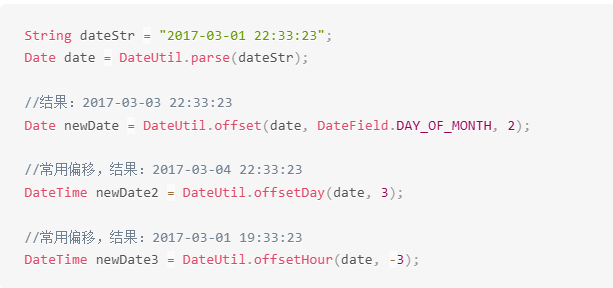
### 开始和结束时间

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：



### 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。



针对当前时间，提供了简化的偏移方法（例如昨天、上周、上个月等）：



### 日期时间差

### 日期范围

## DateTime类

### 说明

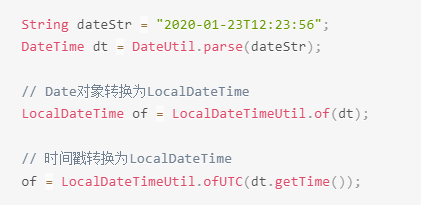
**DateTime**类继承于java.util.Date类，为Date类扩展了众多简便方法，这些方法多是DateUtil静态方法的对象表现形式，使用DateTime对象可以完全替代开发中Date对象的使用。

## LocalDateTimeUtil类

### 说明

从Hutool的5.4.x开始，Hutool加入了针对JDK8+日期API的封装，此工具类的功能包括LocalDateTime和LocalDate的解析、格式化、转换等操作。

### 日期转换



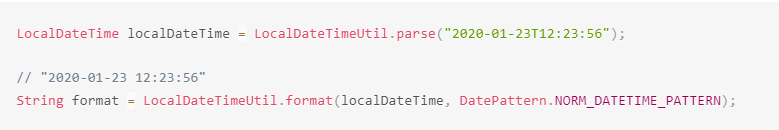
### 日期字符串解析



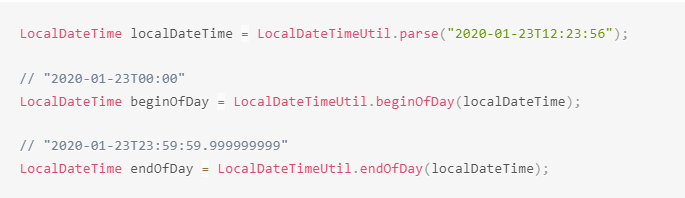
解析支持LocalDate



### 日期格式化



### 一天的开始和结束



# IO流相关

IoUtil

FileUtil

FileTypeUtil

# 电话号码工具类

## PhoneUtil

### 方法

#### public static boolean isMobile(CharSequence value)

验证是否为手机号码（中国）

#### public static boolean isTel(CharSequence value)

验证是否为座机号码（中国）

#### public 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（香港）

# 字符串工具（重点）

## StrUtil类

### 方法

#### public static String nullToEmpty(CharSequence str)

当给定字符串为null时，转换为Empty（空字符串）

#### public static boolean hasBlank(CharSequence... strs)

#### public static boolean isBlank(CharSequence str)

#### public static boolean isAllNotBlank(CharSequence... args)

#### public static String format(CharSequence template, Object... params)

#### public static String toCamelCase(CharSequence name)

将下划线方式命名的字符串转换为驼峰式

#### public static String upperFirst(CharSequence str)

大写首字母

#### public static List<String> splitTrim(CharSequence str, CharSequence separator)

# 对象工具类（重点）

## ObjectUtil类

### 方法

#### public static boolean isNull(Object obj)

#### public static boolean isNotNull(Object obj)

#### public static <T> T defaultIfNull(T source, Supplier<? extends T> defaultValueSupplier)

#### public static boolean equal(Object obj1, Object obj2)

比较两个对象是否相等

#### public static boolean contains(Object obj, Object element)

对象中是否包含元素

#### public static boolean isBasicType(Object object)

# 数字工具

## NumberUtil类

### 加减乘除

NumberUtil.add 针对数字类型做加法

NumberUtil.sub 针对数字类型做减法

NumberUtil.mul 针对数字类型做乘法

NumberUtil.div 针对数字类型做除法，并提供重载方法用于规定除不尽的情况下保留小数位数和舍弃方式

以上四种运算都会将double转为BigDecimal后计算，解决float和double类型无法进行精确计算的问题。这些方法常用于商业计算。

### 保留小数

NumberUtil.round 方法主要封装BigDecimal中的方法来保留小数，返回BigDecimal，这个方法更加灵活，可以选择四舍五入或者全部舍弃等模式。

NumberUtil.roundStr 方法主要封装String.format方法,舍弃方式采用四舍五入

### 校验数字

NumberUtil.isNumber 是否为数字

NumberUtil.isInteger 是否为整数

NumberUtil.isDouble 是否为浮点数

## MathUtil类

# JavaBean

## BeanUtil类

### 基本概念

### 方法

#### public static boolean isBean(Class<?> clazz)

是否为Bean对象

#### public static PropertyDescriptor getPropertyDescriptor(Class<?> clazz, final String fieldName)

获得指定字段的描述

#### public static <T> T fillBean(T bean, ValueProvider<String> valueProvider, CopyOptions copyOptions)

填充Bean的核心方法

#### public static <T> T toBean(Object source, Class<T> clazz)

用于Map转换为Bean

#### public static <T> T toBeanIgnoreCase(Object source, Class<T> clazz, boolean ignoreError)

对象或Map转Bean，忽略字段转换时发生的异常

#### public static Map<String, Object> beanToMap(Object bean, String... properties)

#### public static <T> List<T> copyToList(Collection<?> collection, Class<T> targetType)

复制集合中的Bean属性，这个方法遍历集合中的每个Bean，复制属性后加入到一个新的List中。

#### public static <T> T copyProperties(Object source, Class<T> tClass, String... ignoreProperties)

按照Bean对象属性创建对应的Class对象，并忽略某些属性

source 源Bean对象

tClass 目标Class

ignoreProperties 不拷贝的的属性列表

#### public static void copyProperties(Object source, Object target, String... ignoreProperties)

复制Bean对象属性

ignoreProperties 不拷贝的的属性列表

#### public static void copyProperties(Object source, Object target, boolean ignoreCase)

复制Bean对象属性

source：源Bean对象

target：目标Bean对象

ignoreCase 是否忽略大小写

#### public static void copyProperties(Object source, Object target, CopyOptions copyOptions)

复制Bean对象属性

source 源Bean对象

target 目标Bean对象

copyOptions 拷贝选项

# 反射工具

## 基本概念

Java的反射机制，可以让语言变得更加灵活，对对象的操作也更加“动态”，因此在某些情况下，反射可以做到事半功倍的效果。Hutool针对Java的反射机制做了工具化封装，封装包括：

1、获取构造方法

2、获取字段

3、获取字段值

4、获取方法

5、执行方法（对象方法和静态方法）

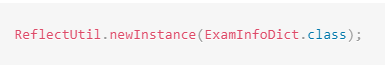
## 获取某个类的所有方法



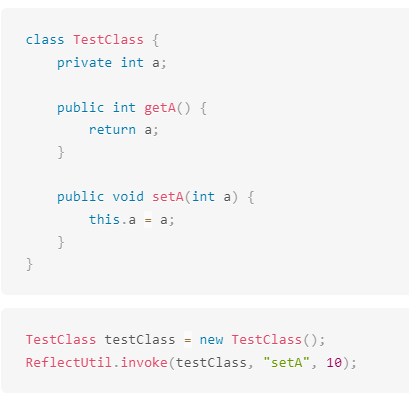
## 获取某个类的指定方法



## 实例化对象



## 执行方法



# JSON

## JSONUtil类

### 方法

#### public static JSONArray parseArray(String jsonStr)

#### public static <T> List<T> toList(JSONArray jsonArray, Class<T> elementType)

将JSONArray转换为Bean的List，默认为ArrayList

#### public static JSONObject parseObj(Object obj)

JSON字符串转JSONObject对象 此方法会忽略空值，但是对JSON字符串不影响

## JSONArray类

### 基本概念

JSONArray 是 Hutool 中用于处理 JSON 数组的类，类似于 org.json 或 com.alibaba.fastjson 中的 JSONArray。

### 构造函数

#### public JSONArray()

#### public JSONArray(int initialCapacity)

### 方法

# 泛型类型工具

# 枚举工具

# 集合工具

## CollUtil类

### 方法

#### public static <T> ArrayList<T> newArrayList(Collection<T> collection)

新建一个ArrayList集合

#### public static boolean isEmpty(Collection<?> collection)

集合是否为空

#### public static boolean isNotEmpty(Collection<?> collection)

集合是否为非空

#### public static <T> List<T> sub(List<T> list, int start, int end)

截取集合中的部分元素

#### public static <T> String join(Iterable<T> iterable, CharSequence conjunction)

以 conjunction 为分隔符将集合转换为字符串。

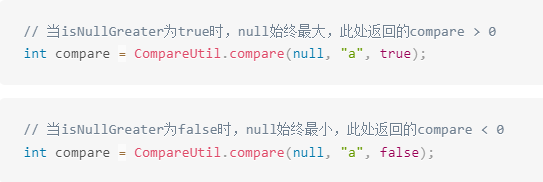
## ListUtil类

### 方法

# 比较工具

## 基本说明

在JDK提供的比较器中，对于null的比较没有考虑，Hutool封装了相关比较，可选null是按照最大值还是最小值对待。



## CompareUtil类

### public static <T extends Comparable<? super T>> int compare(T c1, T c2)

# 压缩工具

# 身份证工具

# 扩展HashMap

Dict类

# 控制台打印

# 字段验证器

# 双向查找Map

# 图片工具

# 配置文件

# 日志工厂

# 缓存工具

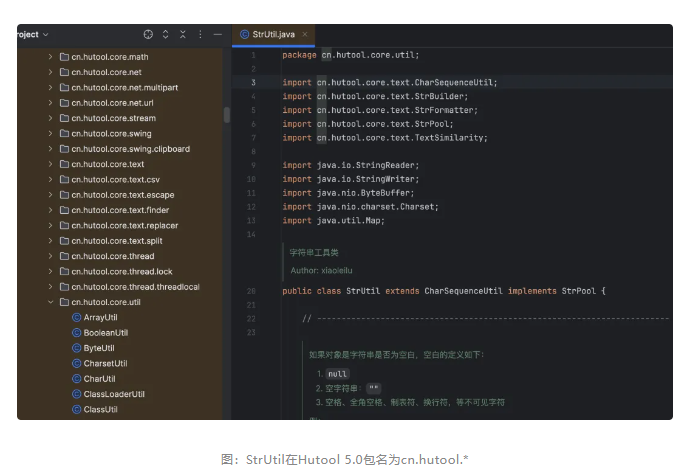
# ServletUtil类

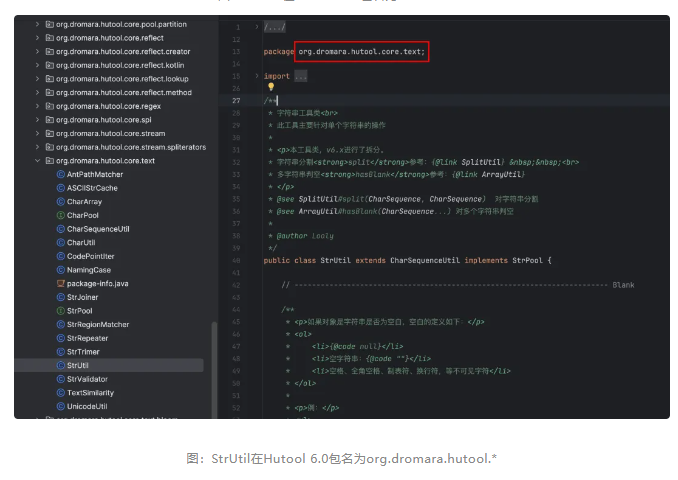
# 升级到6.0遇到的问题

今天本来是想尝鲜体验一下Hutool新版本都更新了什么新功能，于是从5.x.x升到到了6.0.0，然后就出现了各种包名不存在的错误，我当时就蒙了。



回想当年Mybatis-Plus从2.0升级到3.0时所有包名及方法都需要改一遍的痛苦经历仍然历历在目，至今还有很多项目在使用Mybatis-Plus2.0，因为升级修改的太多了，真的太痛苦了，干脆就不升级了。难道Hutool要重蹈Mybatis-plus的覆辙了？如果你使用了Hutool，以后想要升级到6.0时，那么你将要把所有的包名都改一遍，因为Hutool从5.0升级到6.0后包名全改了。





## 修改的内容

1、包名修改，包名变更为org.dromara.hutool

2、重新整理规整，内部工具类重新整理规整，减少无用模块，工具类归类到对应package中。

3、重构Http模块，被人诟病的http模块做重构，采用门面模式，支持包括HttpUrlConnection、HttpClient4、HttpClient5、OkHttp3等库。

4、性能优化，在能力范围之内尽量做性能优化，不跟其他高性能库“攀比”。

5、做减法，相比5.x版本做减法，大部分工作是删掉一些重复代码和无用的重载，使用上可能会增加代码量，但是相比减少了歧义。

6、统一构造方法，构建一种对象不再使用混乱的createXXX、newXXX、ofXXX等名字，而是统一使用of或者ofXXX。

## 使用建议

1、重度依赖，这种项目最好就不要升级了，继续使用5.0，作者说了5.0也会继续维护。

2、轻度依赖，如果只是用了一些简单的工具类，全局替换包名升级还是可以的

3、新项目，我们需要思考有没有必要使用Hutool，是不是可以使用JDK17？很多新功能JDK都自带了，自己项目组是不是也应该维护起属于自己团队的工具类。