

OC-正则表达式-(第一篇)

Wang99

关注

2017-11-07 10:05:19 字数 1,923 阅读 849

一、正则表达式基本概念

1.什么是正则表达式 正则表达式，又称正规表达式，是对字符串操作的一种逻辑公式。正则表达式可以检测 给定的字符串是否符合我们定义的逻辑，也可以从字符串中获取我们想要的特定部分。它可以 迅速地用极简单的方式达到字符串的复杂控制。

2 什么是谓词 Cocoa框架中的NSPredicate用于查询，原理和用法都类似于SQL中的where，作用相当于数据库的过滤谓。我们可以编写简单的谓词语句，就可以从数组中过滤出我们想要的数据库。

二.正则表达式的使用

//正则表达式的使用

```
(void)useRegularExpressions
{
    /**pragma mark 1.正则表达式规则**** */
    /-----1.1 普通字符-----/
    // 字母、数字、汉字、下划线、非特殊定义的标点符号，都是“普通字符”。表达式中的普通字符，在匹配一个字符串的时候，匹配与之相同的一个字符。

    NSString *searchString_1_1 = @"abcde";
    NSString *regexString_1_1 = @"c";
    NSString *matchedString_1_1 = [searchString_1_1 stringByMatching:regexString_1_1];
    NSLog(@"matchedString_1_1-%@",matchedString_1_1); // 输出结果: c

    /-----1.2简单的转义字符-----/
    // 一些不便书写的字符，在前面加 "\",如: \n,\t 等

    NSString *searchString_1_2 = @"abc$de";
    // 记住在字符串中“$”需要用“\”表示
    NSString *regexString_1_2 = @"$d";
    NSString *matchedString_1_2 = [searchString_1_2 stringByMatching:regexString_1_2];
    NSLog(@"matchedString_1_2-%@",matchedString_1_2); // 输出结果: $d

    /-----1.3能够与 '多种字符' 匹配的表达式-----/
    /*
    ∴ 匹配除换行符以外的任意字符

    \w : 匹配字母或数字或下划线或汉字
    \s : 匹配任意的空白符
    \d : 匹配数字
    \b : 匹配单词的开始或结束
    */
    NSString *searchString_1_3 = @"abc123";
    // 正则表达式有一条规则:最先开始的匹配拥有最高的优先权
    NSString *regexString_1_3 = @"\d\d";
    NSString *matchedString_1_3 = [searchString_1_3 stringByMatching:regexString_1_3];
    NSLog(@"matchedString_1_3-%@",matchedString_1_3); // 输出结果: 12

    /-----1.4自定义能够匹配 '多种字符' 的表达式-----/
    // 使用方括号 [] 包含一系列字符，能够匹配其中任意一个字符。用 [^ ] 包含一系列字符，则能够匹配其中字符之外的任意一个字符。同样的道理，虽然可以匹配其中任意一个，但是只能是一个，不是多个。[]本身就隐含了“或”的关系，在[]中使用“|”表示“或”的关系是不对的，这样做只是多了一个普通字符“|”，用来匹配“|”字符本身，|也是同样道理。 如：

    /*
    [ab5@] : 匹配 "a" 或 "b" 或 "5" 或 "@"

    [^abc] : 匹配 "a","b","c" 之外的任意一个字符

    [f-k] : 匹配 "f"~"k" 之间的任意一个字母

    [^A-F0-3] : 匹配 "A"~"F","0"~"3" 之外的任意一个字符
    */
    NSString *searchString_1_4 = @"abc123";
    NSString *regexString_1_4 = @"[bcd][bcd]";
    NSString *matchedString_1_4 = [searchString_1_4 stringByMatching:regexString_1_4];
    NSLog(@"matchedString_1_4-%@",matchedString_1_4); // 输出结果: bc

    /-----1.5 修饰匹配次数的特殊符号-----/
    // 使用表达式再加上修饰匹配次数的特殊符号，那么不用重复书写表达式就可以重复匹配

    /*
    {n} : 表达式重复n次

    1 {m,n} : 表达式至少重复m次，最多重复n次
    2 {m} : 表达式至少重复m次
    3 ? : 表达式0次或者1次，相当于 {0,1}
    4 + : 表达式至少出现1次，相当于 {1,}
    5 * : 表达式不出现或出现任意次，相当于 {0,}

    */
    NSString *searchString_1_5 = @"It costs $12.5";
    NSString *regexString_1_5 = @"\d+\.?\d";
    NSString *matchedString_1_5 = [searchString_1_5 stringByMatching:regexString_1_5];
    NSLog(@"matchedString_1_5-%@",matchedString_1_5); // 输出结果: 12.5

    /-----1.6 其他一些代表抽象意义的特殊符号-----/
    // 一些符号在表达式中代表抽象的特殊意义

    /*
    ^ : 与字符串开始的地方匹配，不匹配任何字符

    1 $ : 与字符串结束的地方匹配，不匹配任何字符
    2 \b : 匹配一个单词边界，也就是单词和空格之间的位置，不匹配任何字符
    3

    */
    // 进一步说明: "b" 与 ""^" 和 "$" 类似，本身不匹配任何字符，但是它要求它在匹配结果中所处位置的左右两边，其中一边是 "w" 范围，另一边是非"w" 的范围
    NSString *searchString_1_6 = @"@@@abc";
    NSString *regexString_1_6 = @"\b\b";
    NSString *matchedString_1_6 = [searchString_1_6 stringByMatching:regexString_1_6];
    NSLog(@"matchedString_1_6-%@",matchedString_1_6); // 输出结果: @a
    pragma mark 2.正则表达式中的一些高级规则

    /-----2.1 匹配次数中的贪婪与非贪婪-----/
    // 1)在使用修饰匹配次数的特殊符号"{m,n}","{m}","?", "", "+","*"可以使同一个表达式能够匹配不同的次数,这种重复匹配不定次数的表达式在匹配过程中，总是尽可能多的匹配。如：
    NSString *regexString2_1_1 = @"\d{4}(\w+)(\d)";
    NSString *matchedString2_1_1 = [searchString_2_1_1
    stringByMatching:regexString2_1_1];
    NSLog(@"matchedString2_1_1-%@",matchedString2_1_1); // 输出结果: dxxxxxxd

    // 2)在修饰匹配次数的特殊符号后再加上一个 "?" 号，则可以使匹配次数不定的表达式尽可能少的匹配,这种匹配原则叫作 "非贪婪" 模式，也叫作 "勉强" 模式
    NSString *regexString2_1_2 = @"\d{4}(\w+)?(\d)";
    NSString *matchedString2_1_2 = [searchString2_1_1
    stringByMatching:regexString2_1_2];
    NSLog(@"matchedString2_1_2-%@",matchedString2_1_2); // 输出结果: dxxxxd

    /-----2.2 反向引用-----/
    // 使用小括号指定一个子表达式后，匹配这个子表达式的文本(也就是此分组捕获的内容)可以在表达式或其它程序中作进一步的处理。默认情况下，每个分组会自动拥有一个组号，规则是：从左向右，以分组的左括号为标志，第一个出现的分组的组号为1，第二个为2，以此类推
    NSString *searchString2_2 = @"Go go";
    NSString *regexString2_2 = @"\b(w+)(b|s+)(l)b";
    NSString *matchedString2_2 = [searchString2_2 stringByMatching:regexString2_2];
    NSLog(@"matchedString2_2-%@",matchedString2_2); // 输出结果: go go

    /-----2.3 零宽断言-----/
    // 零宽断言用于查找某些内容(但并不包括这些内容)之前或之后的东西，也就是说它们像\b^,$那样用于指定一个位置，这个位置应该满足一定的条件(即断言)，因此它们也被称为零宽断言

    /*
    1)捕获
    (exp) : 匹配exp,并捕获文本到自动命名的组里
    (?<name>exp) : 匹配exp,并捕获文本到名称为name的组里，也可以写成(?:name'exp)
    (?exp) : 匹配exp,不捕获匹配的文本，也不给此分组分配组号零宽断言,可节约性能,提高效率

    2)零宽断言
    (?=exp) : 匹配exp前面的位置
    (?<=exp) : 匹配exp后面的位置
    (?!exp) : 匹配后面跟的不是exp的位置
    (?<exp) : 匹配前面不是exp的位置注释(?#comment)这种类型的分组不对正则表达式的处理产生任何影响，用于提供注释让人阅读

    */
    NSString *searchString2_3 = @"I'm singing while you're dancing.";
    NSArray *regexString2_3 = @"\b(w+)?(ing|b)";
    NSArray *matchedString2_3 = [searchString2_3
    componentsMatchedByRegex:regexString2_3];
    NSLog(@"matchedString2_3-%@",matchedString2_3); // 输出结果: (sing,danc)

    // 3.其他通用规则

    /-----3.1-----/
    // 在表达式中，可以使用 "\xxx" 和 "\uXXXX" 表示一个字符 ("X" 表示一个十六进制数)

    /*
    1 \xxx : 编号在 0 ~ 255 范围的字符，比如：空格可以使用 "\x20" 表示
    2
    3 \uXXXX : 任何字符可以使用 "\u" 再加上其编号的4位十六进制数表示，比如: "\u4E20"

    */

    /-----3.2-----/
    // 在表达式 "ls", "ld", "lw", "lb" 表示特殊意义的同时，对应的大写字母表示相反的意义

    /*
    1 \S : 匹配所有非空白字符 ("s" 可匹配各个空白字符)
    2 \D : 匹配所有的非数字字符
    3 \W : 匹配所有的字母、数字、下划线以外的字符
    4
    5 \B : 匹配非单词边界，即左右两边都是 "w" 范围或者左右两边都不是 "w" 范围时的字符通顺

    */

    /-----3.3-----/
    // 在表达式中有特殊意义，需要添加 "" 才能匹配该字符本身的字符汇总

    /*
    1 ^ : 匹配输入字符串的开始位置，要匹配 "^" 字符本身，请使用 "\\^"
    2
    3 $ : 匹配输入字符串的结束位置，要匹配 "$" 字符本身，请使用 "\\$"
    4
    5 ( ) : 标记一个子表达式的开始和结束位置，要匹配小括号，请使用 ""和""

    "

    1 [ ] : 用来自定义能够匹配 '多种字符' 的表达式，要匹配中括号，请使用 ""和""

    "

    1 { } : 修饰匹配次数的符号，要匹配大括号，请使用 "\\{" 和 "\\}"
    2
    3 . : 匹配除了换行符 (\n) 以外的任意一个字符，要匹配小数点本身，请使用 "\\."
    4
    5 ? : 修饰匹配次数为 0 次或 1 次。要匹配 "?" 字符本身，请使用 "\\?"
    6
    7 + : 修饰匹配次数为至少 1 次。要匹配 "+" 字符本身，请使用 "\\+"
    8
    9 * : 修饰匹配次数为 0 次或任意次。要匹配 "*" 字符本身，请使用 "\\*"
    10
    11 | : 左右两边表达式之间 "或" 关系，匹配 "|" 本身，请使用 "\\|"
    12
    13 \ : 用在不同的字符串中时，用\"来进行转义，用在单引的字符串中时，用\"来进行转义

    */
    /** 注意:大部分在正则中有特殊意义、在匹配本身时需转义的字符，在[]内是不需要转义的,必须转义的只有 "\", "[和"]", 而"出现在[]开始位置,"-“前后构成范围区间时，需要转义，出现在其它位置不需要转义。如:1.$@"[[[]]?-"]*
    */
    /*
    }
    */
```

2人点赞 >

2赞

更多

iOS

更多精彩内容，就在简书APP



“小礼物走一走，来简书关注我”

赞赏支持

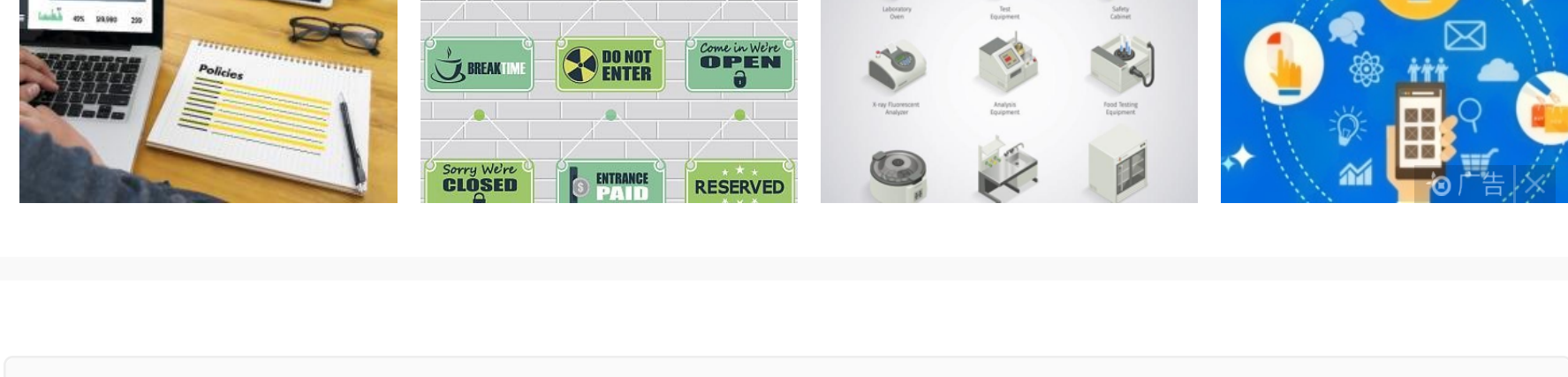
还没有人赞赏，支持一下

Wang99

总资产1 共写了1.5W字 获得10个赞 共16个粉丝

关注

客户关系管理系统



写下你的评论...

全部评论 0

只看作者

按时间倒序 按时间正序

推荐阅读

更多精彩内容 >

iOS中正则表达式的基本使用方法

声胡：本文基本是转载千叶的博客，里面稍微修改。一、通过第三方类库正则匹配在ios项目中可以借用第三方框架RegexKitLite...

黑夜0411

阅读 684 评论 0 赞 0


iOS中正则表达式的基本使用方法

一、第三方框架RegexKitLite的使用 在ios项目中可以借用第三方框架RegexKitLite来简化对正则...

贪吃的猫cx

阅读 206 评论 0 赞 3

留学院服装设计作品案例



正则表达式

正则表达式到底什么东西？字符是计算机软件处理文字时最基本的单位，可能是字母，数字，标点符号，空格、换行符，汉字等...

狮子撩歌

阅读 1,382 评论 0 赞 9

一.正则表达式的使用

一、第三方框架RegexKitLite的使用 在ios项目中可以借用第三方框架RegexKitLite来简化对正则...

蓝色的雪碧

阅读 604 评论 0 赞 0

iOS 正则表达式-详解 01 (简介)

联系人:石虎QQ: 1224614774昵称:瑞瑞呢叭叭叭...、正则表达式基本概念 1 什么是正则表达式 正则...

石虎132

阅读 304 评论 0 赞 8

推荐阅读

- JavaScript正则表达式
- 阅读 94
- R语言中的正则表达式
- 阅读 248
- Java教程 第18章 正则表达式
- 阅读 83
- C#字符常量
- 阅读 75
- 【使用Python处理文本2】
- 阅读 147



crm管理系统