**Groovy正则表达式使用**

// 定义正则表达式，里面的特殊字符会自动转义

~// 定义正则表达式，会将字符串编译成Pattern

=~ 将左边的字符串局部匹配右边的正则表达式，并返回Matcher

==~ 将左边的字符串全局匹配右边的正则表达式，并返回boolean

//语法“//”定义正则表达式，对于出现在表达式中特殊字符，会自动进行转义

def p1 = /a\d$@/;

println p1;// /a\d$@/

println p1.class;//类型还是字符串

println 'a\\d$@' == p1;//true

// "//"中可以使用GString

def sname = "name";

println (/$sname/ == "$sname");//true

println (/$sname/ == sname);//true

println "============================";

//~：用在字符串之前，会将字符串编译成Pattern

def p2 = ~/groovy\d+/;

println p2;// groovy\d+

println p2.class; //java.util.regex.Pattern

println "============================";

//=~：将操作符左边的字符串跟右边的Pattern进行局部匹配，返回值为Matcher

def matcher = ("hello groovy88 test" =~ p2);

println matcher;//java.util.regex.Matcher[pattern=a region=0,3 lastmatch=]

println matcher.class;//java.util.regex.Matcher

println matcher[0] == "groovy88";//没有匹配到matcher[0]为null,所以这里在校验有没有匹配到时可以通过getCount()来判断

println matcher.getCount();

println "============================";

//用法跟=~类似，只是这里进行的精确匹配，即左边的整个字符串跟左边的模式进行匹配，==~的结果跟Matcher.matches()的结果是一样的。返回值为Boolean

def matcher2 = ("hello groovy88 test" =~ /groovy\d+/);//局部匹配返回Matcher

def matcher3 = ("hello groovy88 test" ==~ /groovy\d+/);//全局匹配返回boolean

println matcher2[0];//groovy88

println matcher3;//false

println "==========matches==================";

//Matcher类的matches方法只有在全匹配才返回true

def matcher4 = ("groovy123" =~ /groovy\d+/);

println matcher4.matches();//true

matcher4 = ("testgroovy123" =~ /groovy\d+/);

println matcher4.matches();//false