# 一、正则表达式（RegExp）简介

正则表达式并非一门专用的语言，它用于在一个文件或字符里查找或替换文本。

它具有两种标准：基本的正则表达式(BRE)和扩展的正则表达式(ERE)，其中ERE包括BRE的功能和其他概念。

原理

简单的说，正则表达式是一种用于模式匹配和替换的强有力工具。正则表达式可以通过使用一系列的特殊字符构建匹配模式，然后把匹配模式与数据件、程序输入以及Web页面的表单输入等目标对象进行比较，根据比较对象中是否包含匹配模式，执行相应的程序，防注入系统都是利用这个原理来编写的。

应用

正则表达式最普通的应用可能就要算是用于验证用户在线输入的邮件地址的格式是否正确。如果通过正则表达式验证用户电子邮件地址的格式正确，用户所填写的表单信息将被正常处理，反之，如果用户输入的电子邮件地址与正则表达式的模式不匹配，将会弹出提示信息，要求用户重新输入正确的邮件地址。

组成

正则表达式就是由普通字符(例如字母a到z)以及特殊字符(称为元字符)组成的文字模式，这些模式就是描述在查找文字主体时待匹配的一个或多个字符串。正则表达式作为一个模板，将某个字符模式与所搜索的字符串进行匹配。**正则表达式使用斜杠进行分隔**，例如，/abc/就定义了一个简单的正则表达式。

正则表达式的用途主要有以下三点：

(1)、测试字符串是否符合某个模式。

可以对一个输入字符串进行测试，看这个字符串是否符合电话号码的模式要求或信用卡号码的模式，这种需求称为数据有效性验证。

(2)、替换文本。

可以在文档中使用正则表达式来标识特定的文字，然后将其删除或替换成别的文字，防注入系统就是利用的这个功能实现的。

(3)、根据匹配模式从字符串中提取一个子字符串。

可以用来在文本或输入字段中查找特定的文字，我们的记事本就有这个功能。

# 二、正则表达式的构建

在javascript中是一个对象，它主要用于字符串的模式匹配。创建正则表达式有两种方法：文字量方法和使用构造函数RegExp()。

文字量方法创建正则表达式

使用文字量创建正则表达式的方法为：将文字量的正则表达式赋值给一个变量。正则表达式包含在两个斜杠之间的一个或多个字符，在后一个斜杠后面，可以指定一个或多个选项。例如，/abc/、/ab+c/、/abc/1、/abc/gi都是正则表达式。使用文字量方法创建正则表达式的一般格式为：

*Var varname = /pattern/flags*

其中，varname是变量名称，这个变量用于保存新创建的正则表达式；pattern为指定匹配模式的正则表达式；flags是零个或多个选项，有效选项及其意义为：

● i——忽略大小写，即进行字符串匹配时，不区分大小写。

● g——全局匹配，即匹配字符串中出现的所有模式。

● m——进行多行匹配。

正则表达式的一些示例：

Var hacker = /脚本黑客/;

Var reobj = /love me/ig;

/^JavaScript/

/19[0-9][0-9]\*/

构造函数RegExp()创建正则表达式

正则表达式的构造函数是RegExp()，可以利用这个构造函数创建正则表达式。RegExp()可以使用一个或两个参数：第一个参数指定正则表达式，这是正则表达式的字符串表示方法，例如，"脚本黑客"代表文字量的正则表达式/脚本黑客/；第二个参数是可选参数，它指定正则表达式的选项。其标记字符与文字量正则表达式使用的标记字符相同，也是i、g或m，并且意义也相同。

使用构造函数RegExp()创建正则表达式的一般格式为：

*Var varname = new RegExp("pattern"[, "flags"])*

其中，varname是变量名称，该变量用于保存新创建的正则表达式；pattern为指定匹配模式的正则表达式；flags是零个或多个可选项，方括号表示flags参数可以省略。在这里，pattern和flags参数都可以是字符串变量。例如：

*Var country = new RegExp("脚本黑客");*

*Var reobj = new RegExp("脚本黑客", "ig")*

正则表达式对象的方法

正则表达式对象RegExp提供两种方法：test()和exec()。这两个函数的语法分别为：

**test(string)**

功能：测试字符串string是否包含了匹配该正则表达式的子串，如果包含了这样的子串，那么函数返回ture，否则返回false。

**exec(string)**

功能：在字符串string中进行匹配搜索，并将结果保存在一个数组中返回；如果没有找到匹配的子串，那么返回null。

**(1)、test()**

正则表达式对象RegExp的test()方法用于检测字符串中是否包含正则表达式指定的模式，包含指定的模式返回ture，否则返回false。完成检测后，RegExp对象的lastIndex属性包含了下一次字符串搜索的开始位置。如果进行全局搜索，那么lastIndex属性的值最后一个匹配字符串后面一个字符的位置。使用test()方法进行正则表达式匹配测试的过程为：

**①**、使用一个变量，将正则表达式赋值给这个变量

**②**、使用该正则表达式的test()方法，检测字符串中是否包含该正则表达式指定的模式。

**例**：

*var mastring = "精通脚本黑客";*

*var hack = /黑客/ //文字量方法创建正则表达式*

*var script = new RegExp("黑客"); //构造函数方法创建正则表达式*

*script.test(mastring); //在字符串mastring中搜索模式/黑客/*

另外也可以直接使用文字量正则表达式的test()方法:

*/黑客/.test("精通脚本黑客");*

举个完整例子来具体说明，代码如下：

<html>

<head>

<title>test()方法</title>

<script language = "JavaScript">

var myString="精通脚本黑客";

var myregex = new RegExp("黑客"); // 创建正则表达式

if (myregex.test(myString)){ //判断条件是否成立

alert("找到了指定的模式！");

}

else{

alert("未找到指定的模式。");

}

</script>

</head><body></body>

</html>

**(2)、exec()**

exec()方法忽略了正则表达式中的g选项。执行exec()方法后，正则表达式对象的lastIndex属性的值是我们在前一次匹配字符串后面的第一个字符的位置，不过要注意的是这里是从零开始计数的。所以，利用这个特性，可以得到字符串中与正则表达式相匹配的所有子串。exec()方法返回的数组对象有一个**扩展属性**，一般的数组是没有这个特性的，称为index，它给出了匹配字符串的开始位置。此外，该数组还有另外一个**扩展属性**，称为input，它给出了被搜索字符串。

示例，里面用到了上面的所有知识点，代码如下：

<html>

<head>

<title>正则表达式对象exec()方法应用演示 </title>

<script language = "JavaScript">

var myString="aaa 111 bbb 222 ccc 1111 222ddd";

var regex = /111/; //创建正则表达式对象

var array=regex.exec(myString); //使用 <html>

<head>

<title>正则表达式对象exec()方法应用演示 </title>

<script language = "JavaScript">

var myString="aaa 111 bbb 222 ccc 1111 222ddd";

var regex = /111/; //创建正则表达式对象

var array=regex.exec(myString); //使用exec()方法

if (array){

var str="找到了匹配子串!"+"\n返回数组的值为:"+array+"\n数组元素个数："

+array.length+"\n被搜索的字符串为："+array.input

+"\n匹配子串的开始位置为："+array.index

+"\n匹配子串后面第一个字符的位置为："+regex.lastIndex;

alert(str);

}

else{

alert("未找到匹配子串！！");

}

</script>

<body></body>

</html>

正则表达式的字符串对象的常用方法

|  |  |
| --- | --- |
| 方法 | 功能 |
| Match(regex) | 在一个数组中返回与正则表达式regex相匹配的子串 |
| **Replace(regex,replacement)** | 将字符串中regex指示的子串替换为replacement |
| Search(regex) | 查找正则表达式regex指示的模式在字符串中的开始位置 |
| Split(regex) | 使用正则表达式regex分割字符串，并将分割结果以数组形式返回 |

上面的四个方法与黑客技术紧密相关的要数Replace(regex,replacement)了，通用防注入系统就是利用这个方法来完成的。其他的方法不是那么的密切，所以这里就详细说一下Replace()方法，字符串对象的Replace()方法用于搜索字符串，并用另一个字符串替换搜索到的子串。在指定搜索模式时，可以将其指定为一个字符串，也可以是一个正则表达式。Replace()的调用格式为：

afterString =

beforeString.replace(regular\_expression,replacement\_value);

其中，afterString是完成替换后的字符串；beforeString是替换前的字符串；regular\_expression是指定匹配模式的正则表达式；replacement\_value是替换后的内容。

下面演示一下Replace()方法的效果，示例代码如下：

<html>

<head>

<title>字符串对象replace()方法演示</title>

</head>

<body bgcolor="lightgrey">

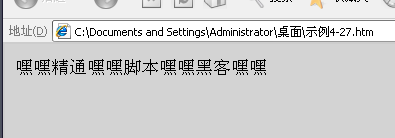
<font size="+1">

<font face="宋体">

<script language = "JavaScript">

<!--

var oldString="哈哈精通哈哈脚本哈哈黑客哈哈"

var regex = /哈哈/g; //一个全局替换的正则表达式

var newString=oldString.replace(regex, "嘿嘿");

document.write(newString +"<br>");

// -->

</script>

</body>

</html>

这个例子中是把oldstring中的"哈哈"替换成"嘿嘿"，然后输出。

# 三、防注入程序

下面介绍一个由VBScript编写的SQL通用防注入系统，代码如下：

1 < % //ASP程序的开始

2 On error resname next //出错后继续执行

3 Dim Sqlstr,error,num //定义数据库查询语句Sqlstr，出错误标志error，计数器num

4 Sq lstr=request.servervariables("query\_string") //request是ASP中的一个对象，servervariablevs是request对象的一个集合，它是获得客户端请求的HTTP报头值，已经一些Web服务器环境变量值的集合。这里是将服务器提交的查询语句赋给Sqlstr

5 dim sqlkey(18) //定义了查询关键字数组sqlkey，共19个元素

6 sqlkey(0)="net user" //从这开始到24行是定义过滤的关键字

7 sq lkey(1)="xp\_cmdshell"

8 sqlkey(2)="/add"

9 sq lkey(3)="exec%20master.dbo.xp\_cmdshell"

10 sqlkey(4)="net localgroup administrators"

11 sqlkey(5)="select"

12 sq lkey(6)="count"

13 sqlkey(7)="asc"

14 sqlkey(8)="char"

15 sqlkey(9)="m id"

16 sq lkey(10)="'"

17 sq lkey(11)=";"

18 sqlkey(1 2)=""""

19 sq lkey(13)="insert"

20 sq lkey(14)="delect"

21 sq lkey(15)="drop"

22 sq lkey(16)="truncate"

23 sq lkey(17)="from"

24 sq lkey(19)="%"

25 error=false //置出错标志为否，即默认不出错。ubound()函数是返回数组的指示最大可用下标，这里为18

26 for num=0 to ubound(deStr) //为一个for循环依次检查数组中的每一个字符

27 if instr(Sqlstr,sqlkey(num))<>0 then // instr()函数是返回某字符串在另一字符串中第一次出现的位置，这里起的作用是查询提交连接中如有19个关键字则置为error为真。

28 error=true

29 e nd if

30 next

31 if error then

32 R esponse.Write("您输入的数据中含有攻击字符，请不要尝试攻击！"); //如果error为真，就输出括号内字符串

33 resp onse.end

34 e nd if

35 % >

大致思路：首先编写一个数组，里面为SQL攻击的关键字。取得每一个从客户端输入的数据，然后利用for循环和instr函数检测客户端输入的数据中是否含有数组中的SQL攻击关键字，如果有就输入“您输入的数据中含有攻击字符，请不要尝试攻击！”，否则系统就正常运行。