

[favorito \(20\)](#) [imprimir](#) [anotar](#) [marcar como lido](#) [dúvidas?](#)

Conceitos básicos sobre Expressões Regulares em Java

(9) (0)

Veja neste artigo os principais conceitos, características e exemplos de código sobre expressões regulares na programação Java. Saiba também como desenvolver padrões personalizados para sistemas.

Introdução

Em qualquer software, sempre existem campos que passam por algumas validações. O objetivo de ter essas validações é simplesmente o fato de que os dados que são informados podem ter o valor e forma determinados pelo sistema. Podemos destacar que essas validações, são como mecanismos de defesa do sistema em relação ao usuário, pois muitos dos problemas são relacionados à gravação de dados incorretos por parte do usuário.

Um exemplo é o sistema permitir apenas letras em alguns campos e o usuário acaba inserindo letras e

Receba notificações :)

[ASSINE](#)[FALE CONOSCO](#)[APP](#)



programa, assegurando que estes dados estejam em um determinado formato. Uma expressão consiste em caracteres literais e símbolos especiais.

Para criar uma expressão regular é necessário informar caracteres especiais usados no padrão da expressão. Esses caracteres são conhecidos como **metacaracteres**, indicando a ocorrência de números, letras entre outros caracteres no texto. Abaixo são exibidos alguns caracteres utilizados para construir a expressão regular.

Caractere	Descrição	Metacaractere
.	Busca qualquer caractere	
\d	Busca qualquer número	[0-9]
\D	Busca qualquer caractere que não seja número	[^0-9]
\w	Busca qualquer caractere de letras e números	[a-zA-Z_0-9]
\W	Busca qualquer caractere que não sejam letras e números	[^\w]
\s	Busca qualquer caractere de espaço em branco, tabulações	[\t\n\x0B\f\r]
\S	Busca qualquer caractere sem espaço em branco	[^\s]

Figura 1: Lista de caracteres

Método matches

O método **matches** especifica uma expressão regular e localiza o conteúdo do objeto String em que e sendo aplicada essa expressão. Para saber se essa correspondência foi atendida ou não, é retornado valor booleano (true ou false).

Nas listagens desse artigo, são mostrados alguns exemplos do que pode ser inicialmente trabalhado. Para fins de ensino, as variáveis estão sendo impressas para mostrar o valor de retorno de cada padrão utilizado.

Listagem 1: Exemplo simples da classe matches

```
public class TestaExpressoes {  
    public static void main(String[] args) {  
        boolean nome = "Maria".matches("Maria");  
        System.out.println("Retorno = "+nome);  
    }  
}
```

Receba notificações :)



Modificadores

Um modificador é um caractere adicionado depois de um o delimitador final, onde acaba mudando o jeito como a função irá tratar a expressão. Abaixo estão relacionados os modificadores que podem ser usados:

- (?i) - Ignora maiúsculas de minúsculas.
- (?m) - Trabalha com multilinhas.
- (?s) - Faz com que o caractere encontre novas linhas.
- (?x) - Permite inclusão de espaços e comentários.

Na listagem 2 foi usado o modificador (?i), que faz com que o resultado tenha o retorno verdadeiro (true), pois esse modificador tem o objetivo de ignorar letras maiúsculas e minúsculas.

Listagem 2: Modificador ignora letras maiúsculas e minúsculas

```
public class TestaExpressoes_Modificadores {  
    public static void main(String[] args) {  
        boolean sobreNome = "Silveira".matches("(?i)silveira");  
        System.out.println("Retorno = "+sobreNome);  
    }  
}
```

Receba notificações :)

Abaixo, nas listagem 3 e 4, são mostrados algumas validações de ocorrências através dos metacaracteres.

Listagem 3: Validações de caracteres

```
public class TestaMetacaractere {  
  
    public static void main(String[] args) {  
  
        boolean email = "@".matches(".");  
        System.out.println("Qualquer caractere: "+email);  
  
        boolean numero = numero = "a".matches("\\d");  
    }  
}
```



```
System.out.println("Possui número? "+numero);
```

```
boolean letrasNumeros = "A2".matches("\\w\\d");  
System.out.println("Possui letras e números? "+letrasNumeros);
```

```
boolean espaco = " ".matches("\\s");  
System.out.println("Possui espaço? "+espaco);
```

```
}
```

```
}
```

Listagem 4: Exemplos de ocorrências

```
public class TestaOcorrencias {  
    public static void main(String[] args) {  
        //Procura a ocorrencia de 1 caractere  
        boolean caractere = "E".matches(".");  
        System.out.println(caractere);  
  
        //Procura a ocorrência de 2 caracteres  
        caractere = "Ab".matches("..");  
        System.out.println(caractere);  
  
        //Validação de cep  
        String cep = "\\d\\d\\d\\d\\d\\d-\\d\\d\\d";  
        boolean valida = "99855-000".matches(cep);  
        System.out.println(valida);  
    }  
}
```

Receba notificações :)

Quantificadores

Em alguns momentos é preciso fazer com que alguns caracteres se repitam, sendo por esse motivo que os metacaracteres precisam ter um controlador, conhecido como quantificador. O quantificador é um caractere que consegue informar quantas vezes um metacaractere pode ser repetido. Na figura 2 há a descrição desses quantificadores.



X{n,m}	X pelo menos n mas não mais que m
X?	0 ou 1 vez
X*	0 ou mais vezes
X+	1 ou mais vezes
X{n}	X procura a ocorrência de um caractere n vezes

Figura 2: Lista dos quantificadores**Listagem 5:** Demonstração dos quantificadores

```
public class TestaQuantificador {  
    public static void main(String[] args) {  
        //Procura 2 dígitos no texto  
        boolean valor = "74".matches("\\d{2}");  
        System.out.println(valor);  
  
        //Procura mais de 2 dígitos no texto  
        valor = "211".matches("\\d{9,}");  
        System.out.println(valor);  
  
        //Procura dígitos entre os valores de 2 e 5  
        valor = "2121".matches("\\d{2,5}");  
        System.out.println(valor);  
  
        //Procura dígito entre 0 e 1 vezes  
        valor = "22".matches(".?");  
        System.out.println(valor);  
  
        //Procura dígito entre 0 e mais vezes  
        valor = "75411".matches(".*");  
        System.out.println(valor);  
  
        //Procura dígito entre 1 e mais vezes  
        valor = ".".matches(".+");  
        System.out.println(valor);  
  
        //Cria expressão regular resumida da data  
        String data = "02/05/1995";  
        valor = data.matches("\\d{2}/\\d{2}/\\d{4}");  
        System.out.println("Data: "+valor);  
    }  
}
```

Receba notificações :)



Metacaracteres de fronteira

Esses metacaracteres definem se a String começa ou termina com um determinado padrão.

Metacaractere	Objetivo
* ^	Inicia
* \$	Finaliza
*	Ou (condição)

Figura 3: Lista dos metacaracteres de fronteira

Listagem 6: Demonstração dos metacaracteres

```
public class Metacaractere_Fronteira {  
    public static void main(String[] args) {  
        //Começa na palavra Java, continua com qualquer caractere a partir do . 0  
        boolean palavra = "Java322".matches("^Java.*");  
        System.out.println(palavra);  
  
        //Termina com 322  
        //O ponto (.) Começa com qualquer caractere e busca 0 ou mais caracteres f  
        palavra = "Java322".matches(".*322$");  
        System.out.println(palavra);  
  
        //Pesquisa se uma palavra existe no texto  
        palavra = "Hello World Java".matches(".*Java.*");  
        System.out.println(palavra);  
  
        //Pesquisa os caracteres que estão depois da letra O e antes da palavra Java  
        palavra = "O mundo Java".matches("^O.*Java$");  
        System.out.println(palavra);  
  
        //Pesquisa pela palavra Inter ou Grêmio  
        boolean time = "Inter".matches("Inter|Grêmio");  
        System.out.println("Time: "+time);  
    }  
}
```

Receba notificações :)



Agrupadores

Tem como objetivo agrupar conjuntos de caracteres que tenham alguma das ações descritas na figura 4.

Metacaractere	Objetivo
* [...]	Agrupamento
* [a-z]	Alcance
* [a-e][i-u]	União
* [a-z&&[aeiou]]	Interseção
* [^abc]	Exceção
* [a-z&&[^m-p]]	Subtração
* \x	Fuga literal

Figura 4: Lista dos agrupadores

Listagem 7: Demonstração de agrupadores

```
public class TestaAgrupadores {  
  
    public static void main(String[] args) {  
  
        //Busca qualquer letra de a até z - faz diferença utilizar maiúsculas e mi  
        boolean palavra = "g".matches("[a-z]");  
        System.out.println(palavra);  
  
        //Verifica se foi escrita em letra maiúscula ou minúscula  
        palavra = "Java".matches("[jJ]ava");  
        System.out.println(palavra);  
  
        //Verifica caracteres de A até Z e a até z  
        palavra = "Sql".matches("[A-Z][a-z]*");  
        System.out.println(palavra);  
  
        //Não permite que comece com as letras a e i  
        palavra = "Oracle".matches("[^aei]racle");  
        System.out.println(palavra);  
  
        //Verifica se foi digitado o caractere "j" e "s"  
        //Retorna false por causa da letra "z" onde que o padrão esperava a letra "s"  
        palavra = "Objetoz".matches("Ob[j]eto[s]");  
    }  
}
```

Receba notificações :)

```
boolean email = javatest@com.de.matches("(\\w+\\.?\\w{2,3})\\.?(\\w{2,3})");  
System.out.println(email);
```

- <http://docs.oracle.com/javase/tutorial/essential/regex/>
- <http://docs.oracle.com/javase/1.4.2/docs/api/java/util/regex/Pattern.html>

Acessem o Space: <http://www.devmedia.com.br/ThiagoVaralloPalmeira>.

 (9) (0)

Receba notificações :)

DEVMEDIA



Anderson
0 pontos



Receba notificações :)