Expressões Regulares

Âncoras

- `^` : início da linha/string
- `$` : fim da linha/string

Quantificadores

- `a*` : “a” zero ou mais vezes
- `a+` : “a” uma ou mais vezes
- `a?` : “a” zero ou uma vez
- `a{m}` : “a” exatamente m vezes
- `a{m,}` : “a” pelo menos m vezes
- `a{m,n}` : “a” de m a n vezes

Operador OU e Colchetes

- `a|b` : “a” ou “b”
- `[abc]` : “a” ou “b” ou “c”
- `[A-Z]` : de A a Z (maiúsculo)
- `[A-Za-z]` : de A a Z (maiúsculo ou minúsculo)
- `[0-9]` : de 0 a 9 (equivalente a \d)
- `[\^ab]` : nem “a” nem “b” (negação)

Classes

- `\d` : dígito (0 a 9)
- `\w` : um caracter (letras, numeros ou underscore)
- `\s` : um espaço (espaço, tabulação ou quebra de linha)
- `.` : qualquer coisa (CUIDADO!)
- OBS: As versões em maísculas negam o padrão. Ex: `\D` : qualquer coisa que não é um dígito.

Caracteres Especiais

`^.[$()|\*+?`

É preciso usar um `\` antes destes caracteres utilizá-los de forma literal.

OBS: Dentro de colchetes, não é necessário.

Ex: `\(`

Alguns outros significados especiais:

- `\t` : tabulação
- `\n` : quebra de linha

Grupos

- `(abc)` : captura “abc” como um grupo, para ser usado futuramente na expressão com `\\n` ,
  onde `n` é o número do grupo

> **Exemplo**:
>
> `(abc) \1` : captura “abc abc”

> `(\d)\1` : captura “11”, “22”, “33”, etc

Gananciosos (greedy) ou Preguiçosos (lazy)?

Os quantificadores `\*`, `+` e `{}` são gananciosos por padrão, ou seja, tentam casar com a maior string possível.
Assim `<.*>` casa com a string inteira `"<tag>valor</tag>"`.

Se adicionarmos o lazy operator `?`, transformamos o operador em preguiçoso, e ele tentará casar com a menor string possível.
Assim `<.*?>` aplicado à string `"<tag>valor</tag>"` casa com as strings `"<tag>"` e `"</tag>"`.