01

Expressões gananciosas

Transcrição

Vamos continuar com os grupos, mas ver um novo exemplo. Praticar é tudo com Regex! O nosso alvo agora é uma tag HTML, escolhemos um cabeçalho (`<h1>`) da nossa página e a tarefa é extrair o conteúdo. O alvo é:

```
<h1 class="text-left">Expressões regulares</h1>
```

Mãos à obra para definir a regex!

Ganancioso ou preguiçoso?

Nossa regex começa com os valores literais `<h1` seguido por qualquer caractere (`.`), uma ou mais vezes. Assim garantimos que todos os atributos da tag serão encontrados. No final colocamos o fechamento da tag (`>`):

```
<h1.+>
```

Ao testar percebemos que a tag inteira foi selecionada e não só a primeira parte:

Como assim? Nossa regex é gananciosa por padrão e selecionou todos os caracteres até o último `>` . O *meta-char*, que na verdade é ganancioso, é o `+` , igualmente `*` e `{}` são também assim, e sempre selecionam o máximo de caracteres possíveis, se não for configurado diferente. Ou seja, podemos dizer que não queremos "ganância" e sim preguiçoso ou hesitante. Isso se faz, novamente pelo caractere `?` :

```
<h1.+?>
```

Isso faz que a regex só seleciona até o primeiro `>` :

Para entender melhor, um bom teste pode ser testar a regex: `<h1.{1,10}` gananciosa, e depois a preguiçoso: `<h1.{1,10}?`. A primeira seleciona *10* caracteres depois do `<h1`, a segunda apenas *1* caractere.

Continuando com a elaboração da regex, vamos definir o conteúdo dentro do parágrafo, aproveitando as classes de caracteres já vistas:

```
<h1.+?>([\w\sõãí.]+)
```

Dentro dos colchetes, podemos declarar mais caracteres do alfabeto português, mas para o nosso texto isso já é suficiente. Por fim falta selecionar o fechamento da tag:

```
<h1.+?>([\w\sõãí.]+)</h1>
```

Repare que já usamos um grupo para receber o conteúdo do parágrafo de volta.