Adrian Pędziwiatr 208316

Lingwistyka Matematyczna: Laboratorium

Projekt 4 - Wyrażenia regularne

1. Adres IP z maską podsieci (rozdzielone spacją)

String ip\_regexp = "0\*(25[025]|2[024]\\d|1?\\d\\d?)(\\.0\*(25[025]|2[024]\\d|1?\\d\\d?)){3}";

String mask\_onlyOnes = "(255\\.)";

String mask\_leadingOnes = "(255|254|252|248|240|224|192|128|0+)";

String mask\_regexp = "((" + mask\_onlyOnes + "{3}" + mask\_leadingOnes + ")|" +

"(" + mask\_onlyOnes + "{2}" + mask\_leadingOnes + "\\.0+)|" +

"(" + mask\_onlyOnes + mask\_leadingOnes + "(\\.0+){2})|" +

"(" + mask\_leadingOnes + "(\\.0+){3}))";

Pattern.compile("^" + ip\_regexp + " " + mask\_regexp + "$");

W przypadku adresu IP, dozwolony zakres liczb to 1-255, co dostało wyrażone poprzez 25[025]|2[024]\\d|1?\\d\\d?. Występują cztery takie człony oddzielone kropką. {3} Oznacza powtórzenie 3-krotne liczby poprzedzone separującą kropką.

Adres MAC jest bardziej skomplikowany. W zapisie binarnym najpierw musi być ciąg jedynek, a następnie ciąg zer. mask\_onlyOnes jest odpowiedzialny za oktet zawierający jedynie jedynki. mask\_leadingOnes jest odpowiedzialny za oktet zawierający ciąg jedynek, a następnie zer. \\.0+ jest odpowiedzialny za oktet zawierając jedynie zera. mask\_regexp przewiduje każdą możliwą kombinacje złożoną z tych możliwości.

1. Adres e-mail

Pattern.compile("^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+(\\.[a-zA-Z0-9-.]\*)\\.[a-z]+$");

[a-zA-Z0-9\_.+-]+ wskazuje na ciąg składający się z co najmniej jednego znaku z zestawu: liter, liczb, oraz znaków podkreślenie, plus, i minus. Następnie następuje znak małpy. Po znaku mały następuje (sub)domena wyrażona za pomocą [a-zA-Z0-9-]+. Dalej (\\.[a-zA-Z0-9-.]\*) oznacza, że domena może się składać z sub domen oddzielonych kropką.

Na samym końcu znajduje się domena najwyższego rzędu .[a-z]+.

1. Dodawanie liczb całkowitych

Pattern.compile("^\\d+(?:\\s\*\\+\\s\*\\d+)\*$");

Dodawanie liczb całkowitych rozpoczyna się od dowolnej liczby \d+. Następnie znajduje się znak +, oraz kolejne cyfry - ta część może być wielokrotnie powtórzona (?:\\s\*\\+\\s\*\\d+)\*.

1. Dodawanie liczb zespolonych

Pattern.compile("^\\s\*\\d+(?:\\.\\d+)\*[ij]?+(?:\\s\*\\+\\s\*\\d+(?:\\.\\d+)\*[ij]?+)\*\\s\*$");

Przy liczbach zespolonych wziąłem pod uwagę liczby rzeczywiste. Po liczbie może się znaleźć kropka, i kolejne cyfry (?:\\.\\d+). Następnie [ij]? oznacza, że po liczbie może się znaleźć litera "i" lub "j" wskazująca na liczbę zespoloną.

1. Wybrane tagi html

Pattern.compile("^<td\\s\*" +

"( colspan=\"\\d+\")?\\s\*" +

"( rowspan=\"\\d+\")?\\s\*" +

"( headers=\"[a-z][a-z0-9]\*(,[a-z][a-z0-9]\*)\*\")?" +

"\\s\*>\\s\*" +

"(\\s\*<span>([^<>]\*)<\\/span>\\s\*)\*" +

"\\s\*<\\/td>$");

Wybrałem tag td. Ten tag może posiadać, zgodnie z HTML5, trzy atrybuty. Pierwsza dwa z nich: colspan , rowspan przyjmują jako wartość liczbę d+\. Trzeci headers natomiast może zawierać pewną liczbę identyfikatorów oddzielonych przycinkiem [a-z][a-z0-9]\*(,[a-z][a-z0-9]\*)\*\")?. Każdy z tych parametrów jest opcjonalny, i może być poprzedzony dowolną liczbą spacji. Następie tag td może zawierać dowolną liczbę tagów span, a tam znajduje się pewien ciąg nie zawierający nawiasów ostrych [^<>]\*