Procesamiento de Información en Aplicaciones Telemáticas

Regular Expressions

Bibliography

- Libros y tutoriales:
 - T. Nield, An Introduction to Regular Expressions.
 O'Reilly Media, Inc. 2021
 - https://docs.oracle.com/javase/tutorial/essential/regex/
 - https://www.tutorialspoint.com/java/java_regular_expressions.htm
- Cheat sheets
 - https://regexlib.com/CheatSheet.aspx
 - https://cheatography.com/davechild/cheat-sheets/regular-expressions/pdf/

Concept.

- It is a special sequence of characters to match or find other strings or sets of strings, using a specialized syntax held in a pattern. E.g.:
 - Find how many times a string is repeated in a text.
 - Check if a text string has a certain structure.
 - Validate if an email is well written.
 - **...**
- Can be used to search, edit, or manipulate text and data.
- Java provides the java.util.regex package for pattern matching with regular expressions. Its classes are Pattern, Matcher and PatternSyntaxException.

Building patterns.

Metacharacters:

$$[] {} ()?* + ^ $. =]$$

- Special characters to build patterns.
- They are non interpreted characters.
- Use:
 - []: set of characters of the expression.
 - **{}:** lenght of the expresión.
 - (): group of a part of a expression.
 - string starts by.
 - •s: string end by.
 - : scape character.

Building patterns. Metacharacters asociated to []

- ?: 0-1 occurrences of the expression.
- *: 0-n occurences of the expression.
- +: 1-n occurrences of the expression.
- : any character except newline
- -: separate values.
- choose between two values (or).
- : it does not have

Executing examples

- From this slide, examples of regular expressions can be checked in different ways:
 - Using the java code of https://docs.oracle.com/javase/tutorial/esential/regex/test_harness.html
 - Using any online evaluator. E.g.:
 - http://www.rubular.com/
 - https://regex101.com/
 - http://www.regexper.com/

Character Classes

| RegEx | Description | Valid | Not Valid |
|-------------|--|---------------|-----------|
| [abc] | a, b, or c (simple class) | a / b /c | g / f / 6 |
| [^abc] | Any character except a, b, or c (negation) | G / h / = | a / b / c |
| [a-zA-Z] | a through z, or A through Z, inclusive (range) | a / A / H | 7 / & / = |
| [a-zA-Z0-9] | a through z, A through Z, or 0 through 9 inclusive (range) | a / f / H / 6 | - / % |

Predefined Character Classes

| RegEx | Description | Valid | Not Valid |
|-------|-----------------------------|-----------------|-------------------|
| | Any character | a / : / @ | |
| \d | Digit [0-9] | 1/2/9 | a / b / c |
| \D | Non digit [^0-9] | a / & / = | 7 / 4 / 1 |
| \s | whitespace | [\t\n\x0B\f\r] | e/l/q |
| \S | Non whitespace | d / e / f | $[\t\n\x0B\f\r]$ |
| \w | Word character [a-zA-Z_0-9] | a/R/8/_ | &/%/\$ |
| \W | Non word character | &/%/\$ | a / R / 8 |

Quantifiers

| RegEx | Description | Valid | Not Valid |
|--------|----------------------|----------------------|-------------|
| X? | X once | a | ab |
| X* | X zero or more times | aa | none |
| X+ | X one or more times | aa | w w |
| X{n} | X, exactly, n times | X{3} aaa | aa / aaaa |
| X{n, } | X, at least, n times | X{3, } aaaa | a / aa |
| X{n,m} | X of n to m times | x{3, 5} aaa /aaaa | aa / aaaaaa |

Boundary Matchers

| RegEx | Description |
|-------|---|
| ^ | Beginning of a line |
| \A | Beginning of the input |
| \$ | End of a line |
| \Z | End of the input for the final terminator |
| \b | Word boundary |
| \B | Non word boundary |

Examples

| Description | String | Regular expression |
|---|------------------------------|---|
| The string matchs exactly with the pattern piat | piat | piat |
| The string contains piat | Esto es piat de tercer curso | .*piat.* |
| The string starts by piat | piat es obligatoria | ^piat.* |
| The string ends by piat | Es obligatoria piat | .*piat\$ |
| The string starts by piat o Piat | piat | ^[pP]iat.* |
| The string contains only several times p i a t | piatpita | (p i a t)+ |
| La cadena después de una p no va una t | Asigpiat / asigp / Asigptaa | .*p(?!t).* |
| La cadena no acaba por un digito | Esta cadena tiene 28 digitos | .*[^\d]\$ |
| Dirección de correo electrónico | gregorio.rubio@upm.es | ^[\w-]+(\.[\w-]+)*@[A- Za-z0-9]+(\.[A-Za- z09]+)*(\.[A-Za-z]{2,})\$ |

Capturing groups

Parts of the expression can be grouped in parentheses to create capture groups.

Regex: $([a-zA-Z\s]^*)(\d+)(.*)$

Test line: This order was placed for QT3000! OK?

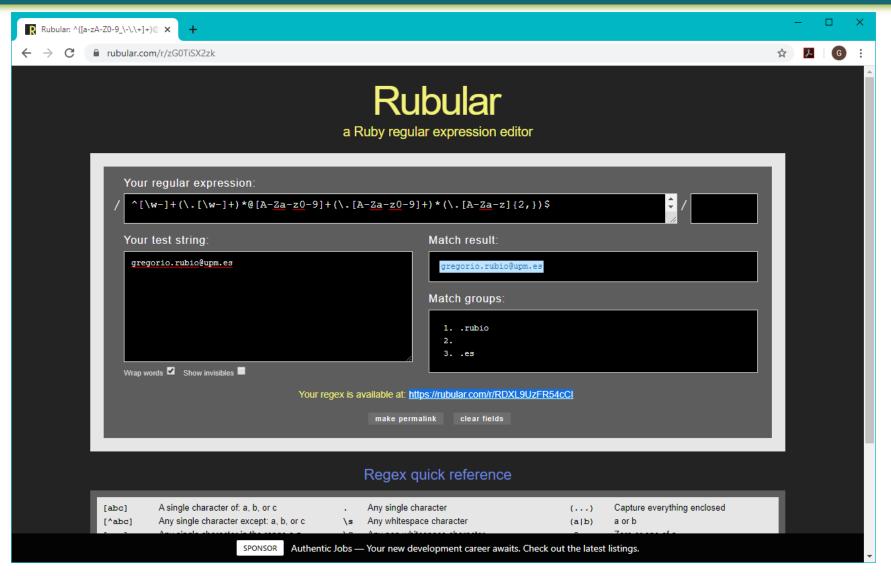
Groups:

- Group 0: *This order was placed for QT3000! OK?*
- Group 1: This order was placed for QT
- Group 2: *3000*
- Group 3: ! OK?

Regular Expressions editors and debuggers

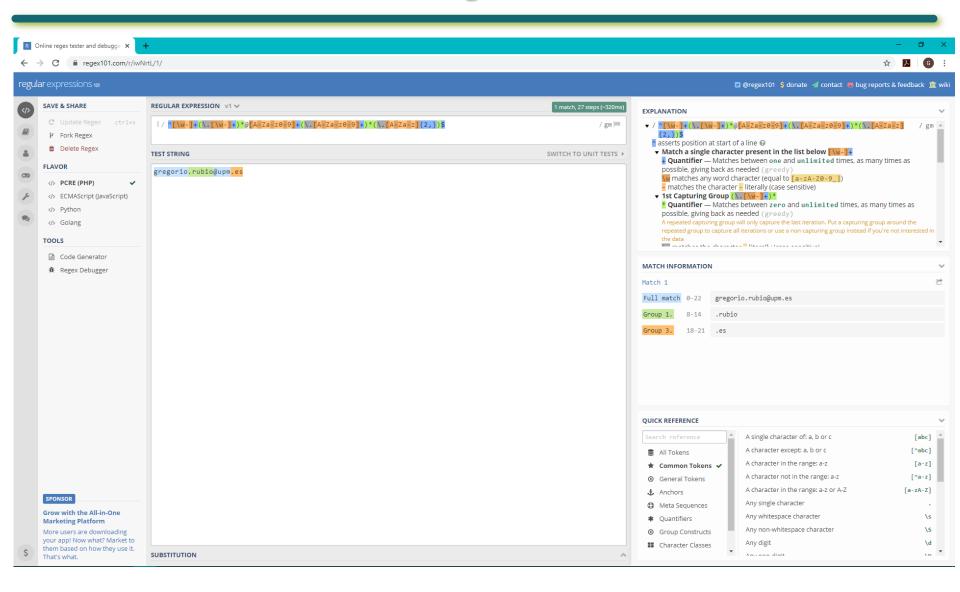
- http://www.rubular.com : simple evaluator of a regular expression
- https://regex101.com: evaluator of a regular expression including a detailed explanation.
- http://www.regexper.com: provides a syntax diagram of a regular expression.
- Next three slides provide the result in each editor of the expression that represents the pattern of the e-mail:
- $(-1)^{(w-)+(\cdot,[w-]+)*@[A-Za-z0-9]+(\cdot,[A-Za-z0-9]+)*(\cdot,[A-Za-z]{2,})$

Rubular.com



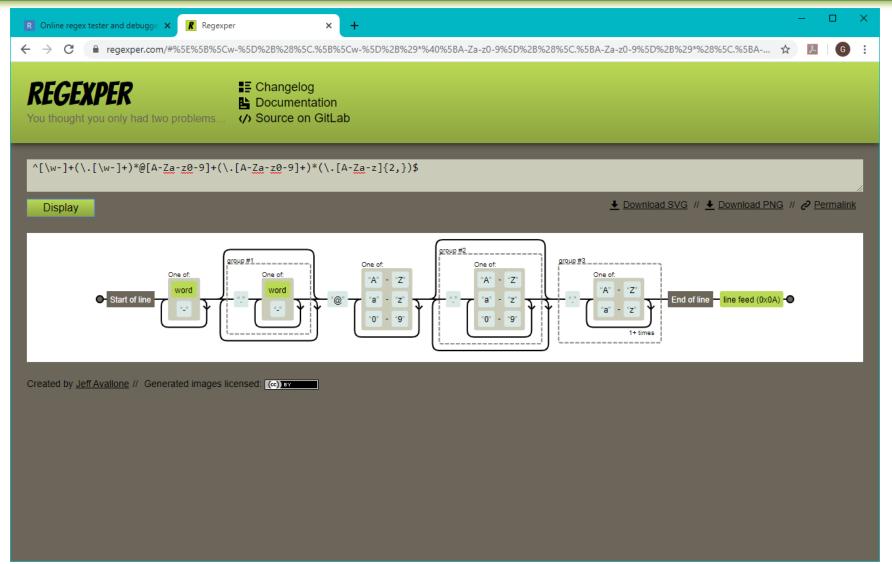
Permalink: https://rubular.com/r/RDXL9UzFR54cCI

Regex101.com



Permalink: https://regex101.com/r/iwNrtL/1

Regexper.com



Regex and java

- Java provides the java.util.regex package for pattern matching with regular expressions.
- The java.util.regex package consists of:
 - Classes:
 - Pattern
 - Matcher
 - Interface:
 - MatchResult
 - Exception:
 - PatternSyntaxException
- Javadoc:

https://docs.oracle.com/javase/8/docs/api/java/util/regex/package
-summary.html

java.util.regex classes

Pattern:

- It is a compiled representation of a regular expression.
- The Pattern class provides no public constructors.
- To create a pattern, you must first invoke one of its public static compile() methods, which will then return a Pattern object. These methods accept a regular expression as the first argument.

Matcher:

- It is the engine that interprets the pattern and performs match operations against an input string.
- The Matcher class, provides no public constructors.
- You obtain a Matcher object by invoking the matcher() on a Pattern object.

java.util.regex classes

- PatternSyntaxException:
 - A PatternSyntaxException object is an unchecked exception that indicates a syntax error in a regular expression pattern.

Pattern class method

Method: compile(). To create a pattern, invoking public static compile () methods. This methods accept a regular expression as the first argument it will return a Pattern object.

Pattern pattern = Pattern.compile (regexExp);

- Compile () method can be parametized.
 Pattern pattern = Pattern.compile (regexExp,
 PATTERN.CASE_INSENSITIVE);
- See parameters at

https://docs.oracle.com/javase/tutorial/essential/regex/pattern.html

Pattern class method

Method: split(). To get the text that falls on either side of the regular expression (splitChars), in the input sequence (InputString).

```
String splitChars = [-_.:];

Pattern pattern = Pattern.compile (splitChars);

String[] items= pattern.split(inputString);
```

See code example at

https://docs.oracle.com/javase/tutorial/essential/regex/pattern.html

(some) Matcher class methods.

- Index methods:
 - start(): return de start index of the match.
 - end(): return de offset of the match character matched.
- Study methods:
 - find(): find the next subsequence of the input sequence that matches the pattern.
 - matches(): match the entire substring against the pattern.
- Replacement methods:
 - replaceAll(): replace every subsequence mached in the input sequence with the replacement string given.
- See all methods at https://docs.oracle.com/javase/tutorial/essential/regex/matcher.html

PatternSyntaxException class methods.

- getDescription(): return th description of the error.
- getIndex(): return the error index.
- getPattern(): return the erroneus pattern.
- getMessage(): return a multi-line description of the syntax error.
- See all methods at https://docs.oracle.com/javase/tutorial/essential/regex/pse.html

Procesamiento de Información en Aplicaciones Telemáticas

Regular Expressions