

## 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](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.

- `$`: string end by.

- `\`: scape character.

## Building patterns. Metacharacters asociated to [ ]

---

- **?** : 0-1 occurences of the expression.
- **\*** : 0-n occurences of the expression.
- **+** : 1-n occurences 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/essential/regex/test\\_harness.html](https://docs.oracle.com/javase/tutorial/essential/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	" "
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
<code>^</code>	Beginning of a line
<code>\A</code>	Beginning of the input
<code>\$</code>	End of a line
<code>\Z</code>	End of the input for the final terminator
<code>\b</code>	Word boundary
<code>\B</code>	Non word boundary

# Examples

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

`^[\w-]+(\.[\w-]+)*@[A-Za-z0-9]+(\.[A-Za-z0-9]+)*(\.[A-Za-z]{2,})$`

# Rubular.com

The screenshot shows the Rubular website, a Ruby regular expression editor. The browser's address bar displays the URL `rubular.com/r/zG0TiSX2zk`. The page features a dark theme with yellow text for the logo and headings.

**Rubular**  
a Ruby regular expression editor

**Your regular expression:**

```
/ ^[\w-]+(\. [\w-]+)*@[A-Za-z0-9]+(\. [A-Za-z0-9]+)*(\. [A-Za-z]{2,})$ /
```

**Your test string:**

```
gregorio.rubio@upm.es
```

**Match result:**

```
gregorio.rubio@upm.es
```

**Match groups:**

```
1. .rubio
2.
3. .es
```

Wrap words ☒ Show invisibles ☐

Your regex is available at: <https://rubular.com/r/RDXL9UzFR54cCI>

[make permalink](#) [clear fields](#)

**Regex quick reference**

<code>[abc]</code>	A single character of: a, b, or c	<code>.</code>	Any single character	<code>(...)</code>	Capture everything enclosed
<code>^[abc]</code>	Any single character except: a, b, or c	<code>\s</code>	Any whitespace character	<code>(a b)</code>	a or b

**SPONSOR** Authentic Jobs — Your new development career awaits. Check out the latest listings.

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

# Regex101.com

Online regex tester and debugger

regex101.com/r/iwNrtL/1/

regular expressions

SAVE & SHARE

- Update Regex ctrl+s
- Fork Regex
- Delete Regex

FLAVOR

- PCRE (PHP) ✓
- ECMAScript (JavaScript)
- Python
- Golang

TOOLS

- Code Generator
- Regex Debugger

REGULAR EXPRESSION v1

1 match, 27 steps (~320ms)

TEST STRING

gregorio.rubio@upm.es

EXPLANATION

- asserts position at start of a line
- Match a single character present in the list below
- Quantifier — Matches between one and unlimited times, as many times as possible, giving back as needed (greedy)
- Matches any word character (equal to [a-zA-Z0-9\_])
- matches the character . literally (case sensitive)
- 1st Capturing Group (\.[\w-]+)
- Quantifier — Matches between zero and unlimited times, as many times as possible, giving back as needed (greedy)
- A repeated capturing group will only capture the last iteration. Put a capturing group around the repeated group to capture all iterations or use a non-capturing group instead if you're not interested in the data

MATCH INFORMATION

Match	Full match	Group 1	Group 2
Match 1	gregorio.rubio@upm.es	.rubio	.es

QUICK REFERENCE

Search reference		
All Tokens	A single character of: a, b or c	[abc]
Common Tokens ✓	A character except: a, b or c	[^abc]
General Tokens	A character in the range: a-z	[a-z]
Anchors	A character not in the range: a-z	[^a-z]
Meta Sequences	A character in the range: a-z or A-Z	[a-zA-Z]
Quantifiers	Any single character	.
Group Constructs	Any whitespace character	\s
Character Classes	Any non-whitespace character	\S
	Any digit	\d
	Any non-digit	\D

SPONSOR

Grow with the All-in-One Marketing Platform

More users are downloading your app! Now what? Market to them based on how they use it. That's what.

SUBSTITUTION

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

# Regexper.com

Online regex tester and debugger x Regexper

regexper.com/#%5E%5B%5Cw-%5D%2B%28%5C.%5B%5Cw-%5D%2B%29\*%40%5BA-Za-z0-9%5D%2B%28%5C.%5BA-Za-z0-9%5D%2B%29\*%28%5C.%5BA-... ☆ | G

## REGEXPER

You thought you only had two problems... [Changelog](#) [Documentation](#) [Source on GitLab](#)

```
^[\w-]+(\.[\w-]+)*@[A-Za-z0-9]+(\.[A-Za-z0-9]+)*(\.[A-Za-z]{2,})$
```

Display [Download SVG](#) // [Download PNG](#) // [Permalink](#)

Created by [Jeff Avallone](#) // Generated images licensed: [\(cc\) BY](#)

Permalink: [https://regexper.com/#%5E%5B%5Cw-%5D%2B%28%5C.%5B%5Cw-%5D%2B%29\\*%40%5BA-Za-z0-9%5D%2B%28%5C.%5BA-Za-z0-9%5D%2B%29\\*%28%5C.%5BA-Za-z%5D%7B2%2C%7D%29%24%0A](https://regexper.com/#%5E%5B%5Cw-%5D%2B%28%5C.%5B%5Cw-%5D%2B%29*%40%5BA-Za-z0-9%5D%2B%28%5C.%5BA-Za-z0-9%5D%2B%29*%28%5C.%5BA-Za-z%5D%7B2%2C%7D%29%24%0A)



# 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>

### ■ 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.

- **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 parametrized.

`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 the description of the error.
  - `getIndex()`: return the error index.
  - `getPattern()`: return the erroneous 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>

## Regular Expressions

---