Validatio Velectrica: Email Input Validation I

§

Made with Obsidian

Type amulet | Category data-engineering | Technologies RegEx | Website Post Link

Input validation lets us ensure only properly formed data is entering the workflow in an information system, preventing malformed data from reaching the database and triggering malfunction of various downstream components. RegEx is a powerful pattern-matching engine designed to search and match complex patterns: From email addresses to IP addresses to phone numbers to infinitely complex text patterns.

In this Amulet, we'll use regular expressions to match some email addresses.

We'll be using a text file containing email addresses which can be found here.

§

Table of Contents

§

Problem Statement

We are presented with a set of email addresses containing different structures. A sample of the complete set can be found below:

```
dougie.jones@luckyseven.com
rodion.raskolnikov@stpetersburgphilosophers.ru
dunya.raskolnikova@moscowfeminists.ru
semyon.marmeladov@petersburgtragedies.ru
petr.luzhin@urbanwealth.ru
arkady.svidrigailov@noblesdelight.ru
john.tavner@milwaukeemusicians.com
tom.tavner@unseenheroes.us
agnes.tavner@quietkeepers.com
```

Our job is to match all the valid email addresses, using whichever flavor suits us best. A typical email address is composed of the following parts:

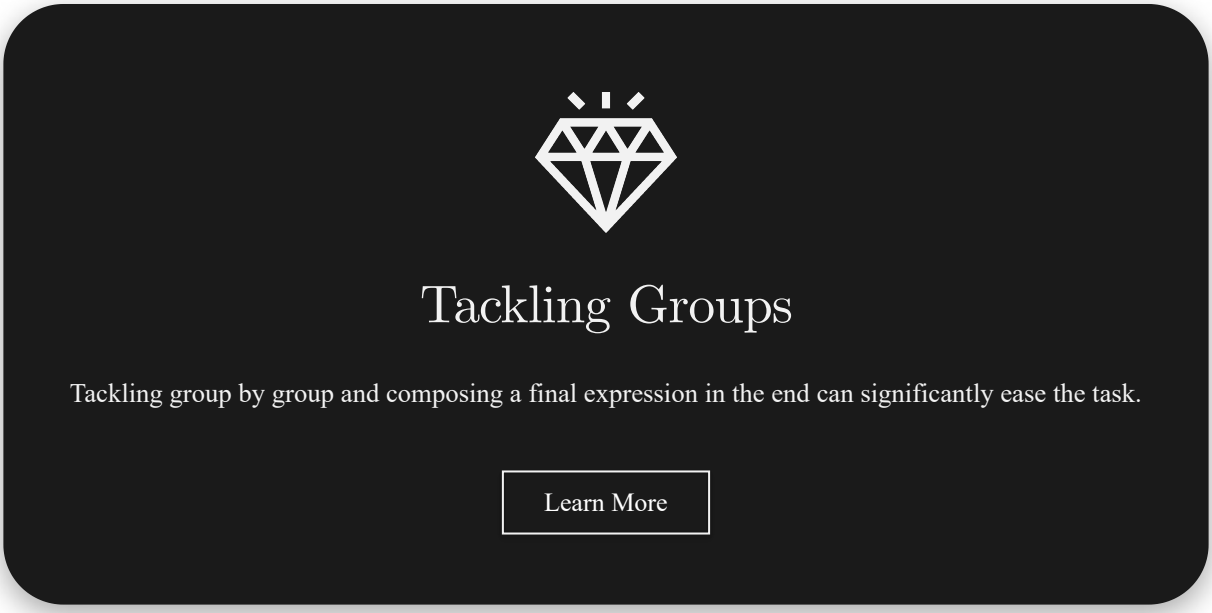
- A unique username: `dougie.jones`

- A domain name: `luckyseven`
- A top-level domain name: `.com`

§

Requirements

- Only the valid addresses must be matched.
- Each address must be a complete and single match, meaning we must not do partial matches on fragments of the address.
- Our expression has to be flexible, meaning it has to account for all possible and top-level domains.
- We must name each part of the email address with its respective section. That means:
  - Username
  - Domain name
  - Top-level domain name



The RegEx expression can be written in any flavor using any tool. However, the solution only includes the expression in PCRE2 flavor (*PHP >=7.3*).

§

Unlocking the Amulet

Let us break down this problem into groups:

- The first group should include the entire username, including any dots or other valid separators we might encounter.

- The second group should include the domain name. This can vary, since there are custom domains, so we cannot just limit our RegEx to the commercial ones, such as google, yahoo, etc.
- The third group should include the top-level domain, which should also not be limited to known domains, since every day, new ones are created. For example, `.ai` did not exist a few years back.

1. The first group

We can start with specifying a beginning-of-line metacharacter, followed by the capturing group name, and any alphanumerical character along with some special characters:

Code

```
^(?<username>[A-Za-z0-9-_.]+)\@
```

We can then repeat this using the `+` metacharacter, which matches the previous token between one and unlimited times.

At the end of our group, we include the `@` symbol including an escape character to make sure that it's not taken as a special character.

2. The second group

We can continue with the domain name, which will practically include any alphanumerical character as well as dash `-` and underscore `_` characters. Other special characters are not allowed.

Code

```
(?<domain_name>[A-Za-z0-9-_]+)
```

3. The third group

We can end our expression by including the top-level domain name, and an end-of-line metacharacter. The top-level domain can consist of multiple groups. For example, `.gov.au` is a valid option, and we need to account for that. This can be managed using an optional metacharacter `?` at the end of the second specification.

Code

```
(?<top_level_domain_name>\.[A-Za-z0-9_]+)(\.[A-Za-z0-9_]+)?$
```

As we can see, we limit the types of special characters we can use, since top-level domains do not accept characters other than alphanumerical or underscores.

4. Composing the complete expression

In the end, we get something like such:

Code

```
^(?<username>[A-Za-z0-9-_.]+)\@(?<domain_name>[A-Za-z0-9-_]+)(?<top_level_domain_name>\.[A-Za-z0-9_]+)(\.[A-Za-z0-9_]+)?$
```

Which, if plotted using a RegEx railroad diagram visualizer (*Regex-Vis was used for this example*), results in something like such:

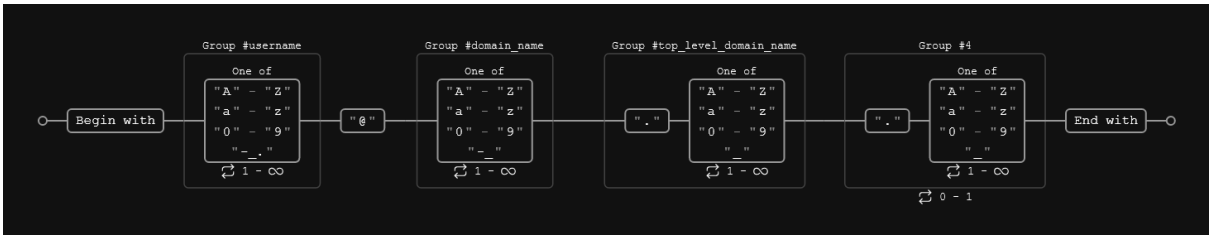


Figure 1: Railroad Diagram for Complete Expression

§

References

- Groups and backreferences, Mozilla
- GeeksForGeeks, PHP | Regular Expressions

§

Copyright