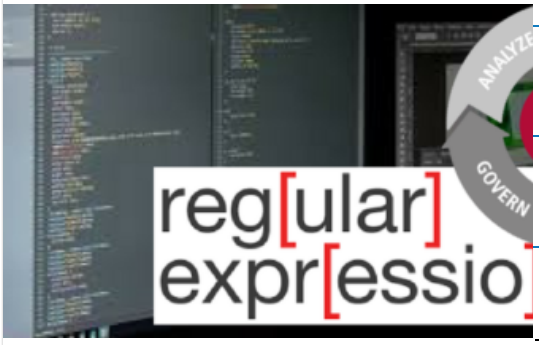
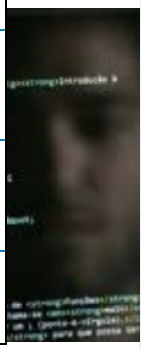
Posted by Vinay Chaturvedi on Feb 10, 2017 in Feat





Learn Regex in Salesforce with Examples

Regular expressions and its usage is a common topic in all projects where there's a need of maintaining quality data about our customers or organization based on the patterns we describe beforehand. Based on this theory, this blog Learn Regex in Salesforce with Examples is inclined to suggest few of the most common ways to implement Regex in Salesforce to refine our data and imposing validations.

"Regex or Regular Expressions are special texts or string for describing a predefined pattern as per the use case" for example if there's a field, named: (Mobile Number) so we are aware that here we must use numbers as input, how this can be validated, the answer is using regex or regular expressions.

"A regular expression, regex or regexp (sometimes called a rational expression) is, in theoretical computer science and formal language theory, a sequence of characters that define a search pattern. Usually this pattern is then used by string searching algorithms for "find" or "find and replace" operations on strings."

Find the recommended Regex documentation here for different patterns related to regex: Java Platform SE 6 syntax.

Before moving forward, I want to make sure that the readers of this blog are aware of various expressions used in regex, here are the one that you'll find in this blog.

These are a few common matchers:

- **Boundary Matchers**

  - *^ → The beginning of a line*
  - *$ → End of a line or string*
  - ** → Matches zero or more of the previous character*

Popular Recent Random

MON 30 10 Things To Know About Apex Triggers in Salesforce
Posted by Vinay Chaturvedi in Featured, Tutorials

TUE 23 Learn Salesforce Lightning with Examples – Part 1 (Create Records using Lightning Components)
Posted by Vinay Chaturvedi in Blog Series, Featured, Learn Salesforce Lightning with Examples, Tutorials

Archives

March 2017
February 2017
January 2017
December 2016
November 2016
October 2016
September 2016
August 2016
July 2016
June 2016
May 2016
April 2016

- **Character-class operators**

1. Literal escape \x
2. Grouping [...]
3. Range a-z
4. Union [a-e][i-u]
5. Intersection [a-z&&[AEIOU]]

- **Line terminators:** \n → Newline character

Regex can be used in various other ways as per our need and the use case, and this has its scope in maintaining data quality whether used in case of a Validation rule or a custom code, in case of a custom code regex can be used to validate some data before it's been saved, for example:

Using Matcher Class

"Matchers use Patterns to perform match operations on a character string"

Learn more about Matcher Class and its methods in Salesforce: Salesforce Matcher Class

Here's some sample code:

```
public class Validator
{
    public static Boolean checkEmail (String str)
    {
        return Pattern.compile('([a-zA-Z0-9_-.]+)
        @(([a-z]{1,3}.[a-z]{1,3}.[a-z]{1,3}.)|(([a-zA-Z0-9-]+.)+))
        ([a-zA-Z]{2,4}| [0-9]{1,3})').matcher(str).matches();
    }
}
```

Here I am using various regex expressions picking the use cases from day to day requirements, I promise this surely be very interesting topic to discuss and with ease to understand, so let's start:

Example 1

Here the user want to define a regex pattern that checks for whether the input string entered in the field is a valid Email Address.

```
([a-zA-Z0-9_-.] +) @(([a-z] {1,3} . [a-z] {1,3} . [a-z] {1,3} .)
|(([a-zA-Z0-9-] +.) +)) ([a-zA-Z] {2,4} | [0-9] {1,3})
```

Explanation :

Best approach to build & understand a pattern is to divide it into small groups like "([a-zA-Z0-9_\\-\\.] +).The first part of the regex defined before @, which includes all the characters from A-Z|0-9|a-z which can include "_" | "-" |"." In the pattern of input, same has been included in the pattern to follow, the best is to divide it based on your understanding.

Example 2

Here the user want to define a regex pattern that checks for whether the input string entered in the field is a String of Numbers.

```
^ [0-9] *$
```

Explanation :

In this example, a user can enter any series of numbers.

Example 3

Here the user want to define a regex that checks for whether the input string entered in the field is a valid 15-digit phone number.

```
[0-9] {15}
```

March 2016
February 2016
January 2016
December 2015
November 2015
October 2015
September 2015
August 2015
July 2015
June 2015
May 2015

Explanation:

- Here in this example as the input can only be numbers
- I have provided [0-9] so this means numbers ranging between 0-9 can be entered.
- The number of digits can be determined by the amount stated inside curly braces which here is {15}, so a fifteen-digit number can be entered inside this field.

Example 4

Here the user want to define a regex pattern that checks for whether the input string entered in the field is a valid Social Security number.

```
{3}-[0-9]{2}-[0-9]{4}
```

Explanation :

That's a simple regex, as we know that a SSN is "xxx-xx-xxxx" pattern of numbers so first 3 digits of numbers then a "-" with 2 digits of a number following "-" and at last 4-digit number to complete.

Example 5

Here the user want to define a regex pattern that checks for whether the input string entered in the field is a valid IP Address.

```
[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}
```

Explanation :

- Whole combination means, digit from 0 to 255 and follow by a dot ".", repeat 4 time and ending with no dot "." Valid IP address format is "0-255.0-255.0-255.0-255".

- Or say more complex version for the above:
  - ^([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.+ ([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.
  - + ([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\. + ([01]?\\d\\d?|2[0-4]\\d|25[0-5])$

Example 6

Here the user want to define a regex that checks for whether the input string entered in the phone field follows a predefined format, here in this case: (123) 456-7890.

```
D? [0-9] {3} D?) [s] [0-9] {3}- [0-9] {4}
```

Explanation :

In the first sequence, (123), "(\\D? [0-9] {3} has been defined followed with a space there after a space and a three digit number with a "Hyphen" and Then follows a four digit number at the end of the string.

Example 7

The user want to define a regex that checks for whether the input string entered is a Valid Date with yyyy-mm-dd in various formats yyyy-mm-dd | yyyy/mm/dd | yyyy.mm.dd etc. formats from 1900-01-01 through 2099-12-31.

```
^(19|20)dd[- /.](0[1-9]|1[012])[- /.](0[1-9]|[12][0-9]|3[01])$
```

Explanation :

- This pattern the first sequence is the combination of pattern which checks for a valid date in the form: yyyy-mm-dd and same with other two cases.
- Only specifics we need to look is what combination of literals validates your data or input best as per the use case.

Example 8

Here the user want to define a regex that checks for whether the input string entered is a Valid Date with yyyy-mm-dd in various formats yyyy-mm-dd | yyyy/mm/dd | yyyy.mm.dd etc. formats from 1900-01-01 through 2099-12-31.

```
Visa:  ^4[0-9]{12}(?:[0-9]{3})?$
MasterCard:  ^(?:5[1-5][0-9]{2}|222[1-9]|22[3-9][0-9]|2[3-6][0-9]{2}|27[01]
           [0-9]|2720)[0-9]{12}$
American Express:  ^3[47][0-9]{13}$
```

Explanation :

- All Visa card numbers start with a 4. All new cards have 16 digits, Old cards have 13, so starting digit is: 4 followed with numbers ranging between 0-9 for 12 digits and so on.
- For MasterCard numbers either start with the numbers 51 through 55 or with the numbers 2221 through 2720. All of them are 16 digits.
- And American Express card numbers start with 34 or 37 and have 15 digits.

Example 9

Here the user want to define a regex pattern that checks for whether the input string entered in the field is a valid URL.

```
/^(https?://)?([da-z.-]+).([a-z.]{2,6})([/w .-]*)*/?$/
```

Explanation :

- Start search for the beginning of the line with the ^ sign.
- This allows the URL to begin with “http://”, “https://”, or neither of them, ? after the “s” to allow URL’s that have http or https.
- For making this entire group optional a question mark is added to the end.
- Next is the domain name: one or more numbers, letters, dots, or hyphens followed by another dot then two to six letters or dots.
- The following section is the optional files and directories. Inside the group, we want to match any number of forward slashes, letters, numbers, underscores, spaces, dots, or hyphens, this group can be matched as many times as we want.
- Pretty much this allows multiple directories to be matched along with a file at the end.
- I have used the star instead of the question mark because the star says zero or more, not zero or one.
- If a question mark was to be used there, only one file/directory would be able to be matched, then a trailing slash is matched, but it can be optional.

Hope you Enjoyed the blog Learn Regex in Salesforce with Examples , stay tuned for more interesting content to follow.

2 Comments

Amit February 11, 2017

Nice post Vinay

REPLY

Vinay Chaturvedi March 3, 2017

Thanks Amit !

REPLY

Post a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

SUBMIT COMMENT

Notify me of follow-up comments by email.

Notify me of new posts by email.