Recursion in Regex

or

How parse IBM Datastage dsx file?

Today we'll talk about recursion in regular expressions

Inspired by

Parsing JSON with a single regex

brian d _foy

http://blogs.perl.org/users/brian_d_foy
/2013/10/parsing-json-with-a-single-regex.html

and Buddy Burden

http://blogs.perl.org/users/buddy_burden
/2013/06/slideshows-in-vroom-so-noted.html

Actually I am delighted by Brian D Foy as

He plays with regexes

```
use v5.10;
$_ = <<'HERE';
Amelia said "I am a camel"
HERE

say "Matched [$+{said}]!" if m/
           ( ['"] )
           (?<said>.*?)
           ( ['"] )
           /x;
```

Here we know that% $ + this hash contains named

brackets

```
use v5.10;
$_ = <<'HERE';
Amelia said "I am a camel"
HERE

say "Matched [$+{said}]!" if m/
           ( ['"] )
           (?<said>.*?)
           ( \1 )
           /x;
```

```
use v5.10;
$_ = <<'HERE';
Amelia said "I am a camel"
HERE

say "Matched [$+{said}]!" if m/
           ( ['"] )
           (?<said>.*?)
           (?1)
           /x;
```

```
use v5.10;
$_ = <<'HERE';
Amelia said "I am a camel'
HERE

say "Matched [$+{said}]!" if m/
           ( ['"] )
           (?<said>.*?)
           (?1)
           /x;
```

```
use v5.18;
$_ = <<'HERE';
He said 'Amelia said "I am a camel"'
HERE

say "Matched [$+{said}]!" if m/
           (?<said>              #$1
           (?<quote>['"])
                  (?:
                      [^'"]++
                      |
                      (?<said> (?1) )
                  )*
           \g{quote}
           )
           /x;
```

```
use v5.18;
$_ = <<'HERE';
Out "Top 'Middle "Bottom" Middle' Out"
HERE

say "Matched [$+{said}]!" if m/
           (?<said>              #$1
           (?<quote>['"])
                  (?:
                      [^'"]++
                      |
                      (?R)
                  )*
           \g{quote}
           )
           (?{say "Inside regex: $+{said}"})
           /x;
```

```
use v5.18;
$_ = <<'HERE';
Out "Top 'Middle "Bottom" Middle' Out"
HERE


say "Matched [$+{said}]!" if m/
         (?(DEFINE)
             (?<QUOTE> ['"])
             (?<NOT_QUOTE> [^'"])
         )
         (?<said>
         (?<quote>(?&QUOTE))
                 (?:
                    (?&NOT_QUOTE)++
                    |
                    (?R)
                 )*
          \g{quote}
         )
         (?{ say "Inside regex: $+{said}" })
         /x;
```

```
use v5.18;
$_ = <<'HERE';
Out "Top 'Middle "Bottom" Middle' Out"
HERE

my @matches;

say "Matched [$+{said}]!" if m/
         (?(DEFINE)
             (?<QUOTE> ['"])
             (?<NOT_QUOTE> [^'"])
         )
         (?<said>
         (?<quote>(?&QUOTE))
                 (?:
                    (?&NOT_QUOTE)++
                    |
                    (?R)
                 )*
          \g{quote}
         )
         (?{ push @matches, $^N })
         /x;

say "\n".join '|',@matches;
```

```
use v5.18;
$_ = <<'HERE';
Out "Top 'Middle "Bottom" Middle' Out"
HERE

my @matches;

say "Matched!" if m/
         (?(DEFINE)
             (?<QUOTE_MARK> ['"])
             (?<NOT_QUOTE_MARK> [^'"])
             (?<QUOTE>
                 (
                      (?<quote>(?&QUOTE_MARK))

                      (?:
                           (?&NOT_QUOTE_MARK)++
                           (?&QUOTE)
                      )*
                      \g{quote}
           )
          (?{ push @matches, $^N })
      )
     )
   (?&QUOTE)
         /x;
```

```
use v5.18;
$_ = <<'HERE';
Out "Top 'Middle "Bottom" Middle' Out"
HERE

my @matches;

say "Matched!" if m/
         (?(DEFINE)
             (?<QUOTE_MARK> ['"])
             (?<NOT_QUOTE_MARK> [^'"])
             (?<QUOTE>
                 (
                      (?<quote>(?&QUOTE_MARK))

                      (?:
                           (?&NOT_QUOTE_MARK)++
                           (?&QUOTE)
                      )*
                      \g{quote}
           )
          (?{ [ @{$^R}, $^N ] })
      )
     )
   (?&QUOTE) (?{ @matches=@{ $^R } })
         /x;
```

```
It is a dsx file:
BEGIN HEADER
   CharacterSet "CP1251"
   ExportingTool "IBM Websphere DataStage Export"
   ServerName "YAPC"
   ToolInstanceID "Russia"
END HEADER
BEGIN DSJOB
   Identifier "Parse_DSX"
   DateModified "2024-09-08"
   TimeModified "13.03.02"
   BEGIN DSRECORD
      Identifier "ROOT"
      OLEType "CJobDefn"
      Readonly "0"
      Name "Parse_DSX"
      ControlAfterSubr "0"
      Parameters "CParameters"
      MetaBag "CMetaProperty"
      BEGIN DSSUBRECORD
         Owner "APT"
         Name "AdvancedRuntimeOptions"
         Value "#DSProjectARTOptions#"
      END DSSUBRECORD
      BEGIN DSSUBRECORD
         Owner "APT"
         Name "ClientCodePage"
         Value "1251"
      END DSSUBRECORD
      NULLIndicatorPosition "0"
      OrchestrateCode =+=+=+=
#################################################################
#### STAGE: T201
## Operator
transform
## Operator options
-flag run
-name 'Russia_YAPC_Ryasan_2015'

## General options
[ident('T201'); jobmon_ident('T201')]
## Inputs
0< [] 'Ryazan_client:L201.v'
## Outputs
0> [] 'T201:L01.v'
1> [] 'T201:L202.v'
;


=+=+=+=
      IsTemplate "0"
      NLSLocale ",,,,"
      JobType "3"
      ValidationStatus "0"
      RecordPerformanceResults "0"
   END DSRECORD
END DSJOB
```

```
use Modern::Perl;

use File::Slurp qw(read_file write_file);
use Data::Dumper qw(Dumper);
my $filename = 'short_example.dsx';

my $data           = read_file($filename);
my $header_and_job = split_by_header_and_job($data);
my $header_fields  = split_fields_by_new_line( $header_and_job->{header} );
say Dumper $header_fields;

sub split_by_header_and_job {
    my $data = shift;
    local $/ = '';    # Paragraph mode
    my %header_and_job = ();
    my @fields         = ();

    #@fields = (
    $data =~ /
(?<header>
BEGIN[ ]HEADER
.*?
END[ ]HEADER
)
.*?
(?<job>
BEGIN[ ]DSJOB
.*?
END[ ]DSJOB )
```

```
    /xsg
          ;
        %header_and_job = %+;
        return \%header_and_job;
    }

    sub split_fields_by_new_line {
        my ($curr_record)     = @_;
        my %fields_and_values = ();
        my @fields            = ();
        while (
            $curr_record =~ m/
            (?<name>\w+)[ ]"(?<value>.*?)(?<!\\)"|
            ((?<name>\w+)[ ]\Q=+=+=+=\E
            (?<value>.*?)
            \Q=+=+=+=\E)
            /xsg
          )
        {
            my $name       = $+{name};
            my $value      = $+{value};
            my %hash_value = ();
            push @fields, \%hash_value;
        }
        return \@fields;
    }
```

```
use Modern::Perl;

use File::Slurp qw(read_file write_file);
use Data::Dumper qw(Dumper);
my $filename = 'short_example.dsx';

my $data           = read_file($filename);
my $header_and_job = split_by_header_and_job($data);
my $header_fields  = split_fields_by_new_line( $header_and_job->{header} );
say Dumper $header_fields;

sub split_by_header_and_job {
    my $data = shift;
    local $/ = '';    # Paragraph mode
    my %header_and_job = ();
    my @fields         = ();

    #@fields = (
    $data =~ /
(?<header>
BEGIN[ ]HEADER
.*?
END[ ]HEADER
)
.*?
(?<job>
BEGIN[ ]DSJOB
.*?
END[ ]DSJOB )
```

```
    /xsg
          ;
        %header_and_job = %+;
        return \%header_and_job;
    }

    sub split_fields_by_new_line {
        my ($curr_record)     = @_;
        my %fields_and_values = ();
        my @fields            = ();
        while (
            $curr_record =~ m/
             (?(DEFINE)
                 (?<QUOTE> ["])
                 (?<LONG_QUOTE> \Q=+=+=+=\E)
                 (?<ALL_QUOTE> &QUOTE|&LONG_QUOTE)
             )
            (?<name>\w+)[ ]
            (?&QUOTE)
            (?<value>.*?)
            (?<!\\)
            (?&QUOTE)
            |
            ((?<name>\w+)[ ]
            (?&LONG_QUOTE)
            (?<value>.*?)
            (?&LONG_QUOTE)
            )
            /xsg
          )
        {
            my $name       = $+{name};
            my $value      = $+{value};
            my %hash_value = ();
            $hash_value{$name} = $value;
            push @fields, \%hash_value;
        }
        return \@fields;
    }
```

This presentaion

* https://github.com/mishin/presentation/tree/master/regex_recursion

The End