

What to Expect When You're Expected to Parse a CSV

Ruby Jam

Tony Drake

Indy.rb, July 2018

A brief history...

- Ruby 1.8
 - CSV sucked
 - FasterCSV gem implemented “better” and faster CSV parsing
- Ruby 1.9 and later
 - FasterCSV merged into Ruby core
 - CSV == FasterCSV

The basics

`CSV.foreach(filepath, options={})`

`CSV.read(filepath, options={})`

`CSV.open(filepath, mode="rb", options={})`

`CSV.new(data, options={})`

Reading a file line by line

Reading a whole file at once

Writing to a file

Generic constructor


**DON'T
USE
THIS**

Options are optional

Option	Default	Notes
:col_sep	“” ,	Separator character
:row_sep	auto	How each row is terminated (\r\r / \n
:quote_char	‘ “ ‘	The quoting character
:converters / :header_converters	nil	Lambdas / symbols to covert values from strings to Ruby objects
:headers	false	Assumes first row is header row and maps values accordingly
:return_headers	false	Include the header row as the first row of data
:skip_blanks	false	Skip over blank rows
:force_quotes	false	(For writing) Always wrap values in quotes
:skip_lines	nil	Skip row if regex matches the raw data

Writing a file

```
>> CSV.open(filepath, "wb", force_quotes: true) do |csv|  
..   csv << ["Head1", "Head2"]  
..   (1..4).each do |x|  
..     csv << [x, x+1]  
..   end  
.. end
```



```
"Head1", "Head2"  
"1", "2"  
"2", "3"  
"3", "4"  
"4", "5"
```

Reading a File

Head1,Head2

1,2

2,3

```
>> CSV.foreach(filepath) do |row|  
  ..      puts row.inspect  
  ..      end  
["Head1", "Head2"]  
["1", "2"]  
["2", "3"]  
=> nil
```

Reading a File (Parsing header row)

```
>> CSV.foreach(filepath, headers: true) do |row|  
  ..      puts row.inspect  
  ..      end  
#<CSV::Row "Head1": "1" "Head2": "2">  
#<CSV::Row "Head1": "2" "Head2": "3">
```

```
>> CSV.foreach(filepath, headers: true) do |row|  
  ..      puts row["Head1"].inspect  
  ..      end  
"1"  
"2"
```

Reading a File (Providing your own headers)

1,2

2,3

```
>> CSV.foreach(filepath,  
..   headers: ["A", "B"]) do |row|  
..   puts row["B"].inspect  
..   end  
"2"  
"3"  
=> nil
```


Reading a File (Pipe delimited)

Head1	Head2
1	2
2	3

```
>> CSV.foreach(filepath,  
..   headers: true,  
..   col_sep: "|") do |row|  
..   puts row.inspect  
..   end  
#<CSV::Row "Head1":"1" "Head2":"2">  
#<CSV::Row "Head1":"2" "Head2":"3">  
=> nil
```

Reading a File

```
>> CSV.foreach(filepath,  
..     headers: true,  
..     converters: [:numeric]) do |row|  
..     puts row["Head1"].inspect  
.. end  
1  
2
```

Reading a File

```
>> CSV.foreach(filepath,  
..     headers: true,  
..     converters: ->(value) { "#{value}-a" }) do |row|  
..     puts row["Head1"].inspect  
.. end  
"1-a"  
"2-a"
```

Reading a File?

Super Special Report

Client: Big Business, LLC

Generated: 9/9/1999

Account, Foo, Bar, Fizz, Buzz

123,456,789,000

123,456,789,000



Header row all the way down here?!?!?

CSV.new()

- Basic constructor
- Used internally by .foreach .read, and .open (for formatting)
- Takes IO input
 - Raw string
 - StringIO object
 - File handle
- CSV is just a glorified wrapper for File in Ruby!

Read the file!

May raise EOFError, so rescue this

```
file = File.open(filepath)
```

```
begin
```

```
end until(file.readline.strip.match(/^Account/))
```

File#readline moves the cursor one line passed the header, so we must rewind and re-read to get to the spot we want

```
header_row = file.lineno - 1
```

```
file.rewind
```

```
header_row.times{ file.readline }
```

Give CSV.new the opened handle with the cursor right before the header row. Code the rest as normal

```
CSV.new(file, headers: true).each do |row|
```

```
  puts row.inspect
```

```
end
```

```
file.close
```

Always close your files!

```
>> file = File.open(filepath)
=> #<File:/Users/tony/Desktop/bar.csv>
>>
>> begin
..   end until(file.readline.strip.match(/^Account/))
=> nil
>>
>> header_row = file.lineno - 1
=> 5
>> file.rewind
=> 0
>> header_row.times{ file.readline }
=> 5
>>
>> CSV.new(file, headers: true).each do |row|
..   puts row["Account"]
..   end
123
123
=> nil
>>
>> file.close
=> nil
```

Final Notes

- It's all or nothing - Your files must be valid
 - Consistent line endings
 - Must have actual line endings
 - Properly escaped values
- Read the documentation!
 - <http://ruby-doc.org/stdlib-2.5.1/libdoc/csv/rdoc/CSV.html>

The end

Twitter: @t27duck

Slides: github.com/t27duck/showandtell