Working with CSV

Raphael Alampay
DEVELOPER

@happyalampay github.com/ralampay

Overview

- What is CSV?
- Reading from CSV Files
- Writing to CSV Files

CSV Fundamentals

What is a CSV file?

Comma Separated Value file

Just a txt file!

Represents data in a tabular fashion

Uses a character “,” to separate values

Uses a .csv extension

Tabular Data

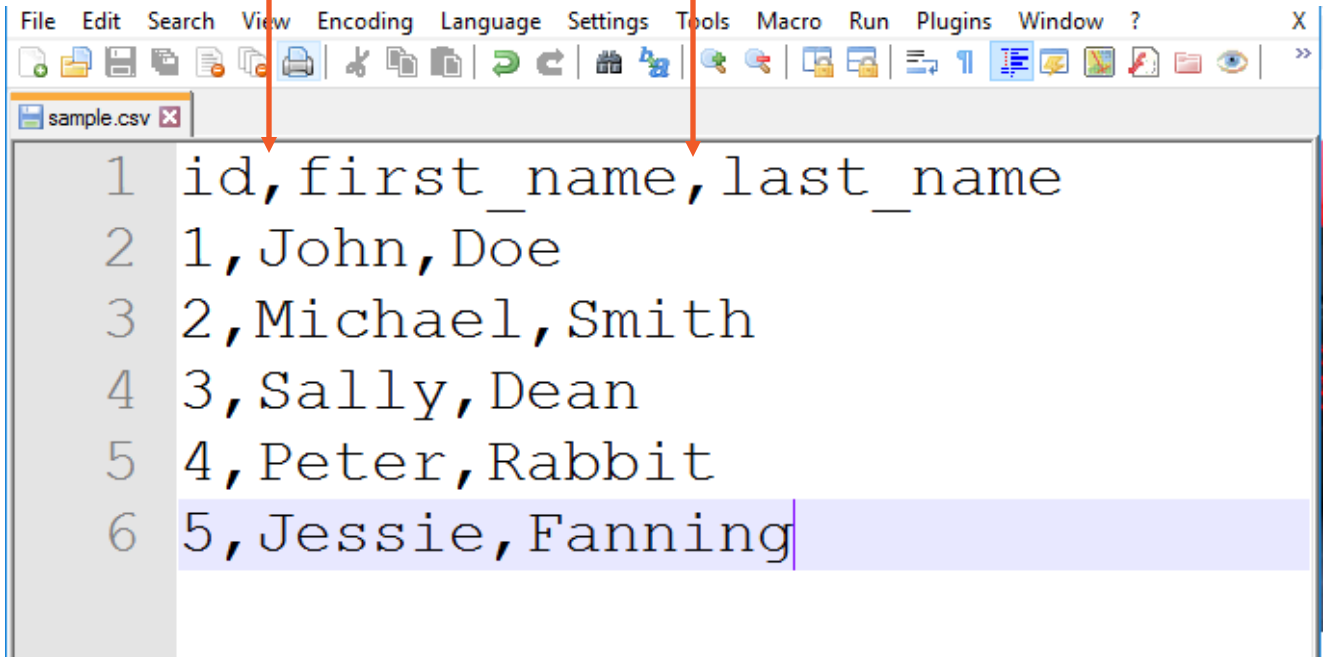
Headers

Data

id	first_name	last_name
1	John	Doe
2	Michael	Smith
3	Sally	Dean
4	Peter	Rabbit
5	Jessie	Fanning

CSV Data

“,” as the delimiter



Open in Spreadsheet

	A	B	C
1	id	first_name	last_name
2	1	John	Doe
3	2	Michael	Smith
4	3	Sally	Dean
5	4	Peter	Rabbit
6	5	Jessie	Fanning

Demo

Opening a CSV File

Saving a CSV File

Reading from CSV Files

Ways to Programmatically Read CSV

Read as Txt File

Treat it like a normal text file and extract content as string

Ruby's CSV Library

Programmatically represent tabular data as an Array

```
file_location = “sample.csv”

file = File.open(file_location)

content = file.read

puts content
```

- Variable representing the file location
- Open the file and return a reference
- Extract contents of file
- Display content

```
"id,first_name,last_name\n
1,John,Doe\n
2,Michael,Smith\n
3,Sally,Dean\n
4,Peter,Rabbit\n
5,Jessie,Fanning"
```

\n is used to create a new line for each row except the last

What content looks like

String representation of the content

```
require ‘csv’

content = File.read(“sample.csv”)

data = CSV.parse(content)

data.each do |d|

  puts “ID: #{d[0]} First Name:
#{d[1]} Last Name: #{d[2]}”

end
```

- Require the library
- Read the file
- Parse the content
- Loop through each row
- Access each attribute value

```
ID: id First Name: first_name Last Name: last_name
ID: 1 First Name: John Last Name: Doe
ID: 2 First Name: Michael Last Name: Smith
ID: 3 First Name: Sally Last Name: Dean
ID: 4 First Name: Peter Last Name: Rabbit
ID: 5 First Name: Jessie Last Name: Fanning
```

Header as data!

What the output looks like

As iterated through the array

```
require ‘csv’

content = File.read(“sample.csv”)

data = CSV.parse(content, headers: true)

data.each do |d|

  puts “ID: #{d[‘id’]} First Name:
#{d[‘first_name’]} Last Name:
#{d[‘last_name’]}”

end
```

- Require the library
- Read the file
- Parse the content and pass headers: true
- Loop through each row
- Access each attribute value as a hash using the header’s key

```
ID: 1 First Name: John Last Name: Doe

ID: 2 First Name: Michael Last Name: Smith

ID: 3 First Name: Sally Last Name: Dean

ID: 4 First Name: Peter Last Name: Rabbit

ID: 5 First Name: Jessie Last Name: Fanning
```

What the output looks like

As iterated through an array of hashes

CSV data as string

Delimiter to use

```
CSV.parse("1\tJohn\tDoe", headers: false, col_sep: "\t")

CSV.parse("1 John Doe", headers: false, col_sep: " ")

CSV.parse("1|John|Doe", headers: false, col_sep: "|")
```

Reading CSV Data

Using Different Delimiter

CSV file location

```
data = CSV.read(“sample.csv”, headers: true, col_sep: “,”)
```

Reading a CSV file

The most convenient way

Demo

Reading from a File

Iterating through Data

Writing to CSV Files

Ways to Programmatically Write CSV

Write Text

Construct the string and write to text file

Ruby's CSV Library

Programmatically represent tabular data as an Array

```
data = [
    [“id”, “first_name, “last_name”],
    [1, “John”, “Doe”],
    [2, “Michael”, “Smith”],
    [3, “Sally”, “Dean”],
    [4, “Peter”, “Rabbit”],
    [5, “Jessie”, “Fanning”]
]

content = data.map{ |d| d.join(“,”)
}.join(“\n”)


File.write(“sample.csv”, content)
```

- Represent the data as an array of arrays
- Each element of the array represents a row in the CSV file
- Map the top level array
- Join each element with the delimiter “,” before joining the mapped array with a new line “\n”
- Write to file

```
require ‘csv’
CSV.open(“sample.csv”, “w”) do |c|
    c << [“id”, “first_name, “last_name”]
    c << [1, “John”, “Doe”]
    c << [2, “Michael”, “Smith”]
    c << [3, “Sally”, “Dean”]
    c << [4, “Peter”, “Rabbit”]
    c << [5, “Jessie”, “Fanning”]
end
```

- Require the CSV library
- Open a file in write mode
- Append each row to the block argument

```
require ‘csv’

content = CSV.generate do |c|

    c << [“id”, “first_name, “last_name”]

    c << [1, “John”, “Doe”]

    c << [2, “Michael”, “Smith”]

    c << [3, “Sally”, “Dean”]

    c << [4, “Peter”, “Rabbit”]

    c << [5, “Jessie”, “Fanning”]

end


File.write(“sample.csv”, content)
```

- Require the CSV library
- Store the generated content as a string
- Append each row to the block argument
- Write content to CSV file

Demo

Programmatically Represent Tabular Data

Write to a CSV File

Summary

CSV is just text!

Delimiter to separate columnar data

Built in Ruby libraries