HTTP-Requests

Ni ska påbörja ett projekt där ni skriver en Webbapplikations-server liknande Sinatra.

Det en webbläsare skickar till servern är bara strängar, formaterade enligt HTTP-standarden.

Läs HTTP Messages - Requests på MDN för en förklaring av strukturen.

TL;DR

HTTP-requests är strängar enligt följande format:

Requesten består av flera rader. Varje rad avgränsas av en radbrytning (`\n`)

Första raden innehåller HTTP-metod, resurs och version, med ett blanksteg (space) mellan varje "attribut".

```
GET '/fruits' HTTP/1.1\n
```

I exemplet ovan är HTTP-*metoden* `GET` , *resursen* `/fruits` och *version* `HTTP/1.1`

Efter första raden följer en eller flera headers-rader.

Headers-raderna är strukturerade enligt följande: först kommer namnet på headern, följt av ett kolon (`:`). Efter kolonet är ett mellanrum, sen följer värdet för headern.

```
GET '/fruits' HTTP/1.1\n
Host: developer.mozilla.org\n
Accept-Language: fr\n
\n
```

I exemplet ovan är finns följande headers: `Host` med värdet `developer.mozilla.org` och `Accept-Language` med värdet `fr`.

Det finns ingen lista över hur många, eller vilka headers en webbläsare får skicka.

Efter alla headers-rader kommer en tom rad.

Efter den tomma raden *kan* det komma rader med data från HTML-formulär eller annan data.

Uppgift

Er server ska kunna ta emot `HTTP-Request`-strängar och omvandla dem till ruby-objekt, för att sedan kunna bestämma vad servern ska göra.

För att minimera felkällor kommer vi till en början att ha fejkade http-request lagrade i filer som vi kan använda när vi utvecklar vår server.

Använd de bifogade textfilerna (`get-index.request.txt`, `get-examples.request.txt` , `get-fruits-with-filter.request.txt`, `post-login-request.txt`) som *exempelrequests* när ni skriver er kod. (Längre fram i kursen kommer vi ordna så er kod tar emot riktiga requests istället).

Kodstruktur

Studera requestens textsträngs uppbyggnad och fundera på hur du ska dela upp textsträngen i mindre bitar som du sedan kan behandla på olika sätt för att plocka ut och lagra relevant data.

Ett tips är att använda `String#split`. Tänk på att radbrytning är ett tecken (`\n`).

Tänk på att `String#split` kan ta mer än 1 tecken som argument (`"Superduperbäst".split('duper') #=> ['Super', 'bäst']`). Du kommer behöva använda `String.split` flera gånger. Fundera på i vilken ordning du ska dela upp strängen i mindre strängar, som du sen kan behandla var för sig.

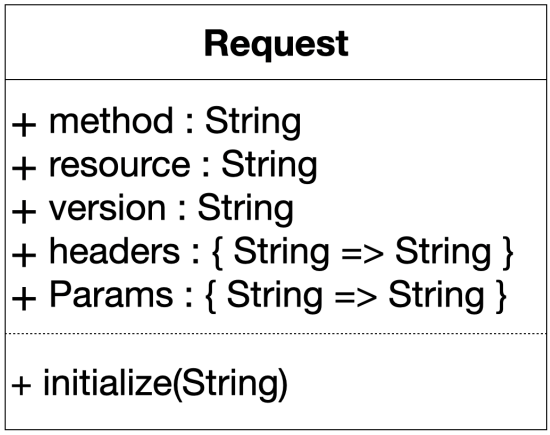
Eftersom `String#split` returnerar en `Array` kan du använda *Array desctructuring* för att skriva tydligare/smidigare kod.

```
#Array destructuring
first_name, lastname = ['Banan', 'Paj']
p first_name #=> 'Banan'
p last_name  #=> 'Paj'

words = ['Lorem', 'ipsum', 'dolor', 'sit', 'amet']
first, second = words
p first  #=> # 'Lorem'
p second #=> 'ipsum'

words = ['Lorem', 'ipsum']
first, second, third = words
p first  #=> 'Lorem'
p second #=> 'ipsum'
p third  #=> nil

words = ['Lorem', 'ipsum', 'dolor', 'sit', 'amet']
first, second, *rest = words
p first  #=> # 'Lorem'
p second #=> 'ipsum'
p rest   #=> ['dolor', 'sit', 'amet']
```

Läs mer om Array destructuring här

Skapa en klass (`Request`) med en *konstruktor* som tar emot *innehållet* i filerna (en String), och skapar ett objekt som innehåller datan i requesten.

Objektet ska kunna returnera `method` (en String), `resource` (en String), `version` (en String), `headers` (en Hash med Strings som Key och Value), och `params` (en Hash med Strings som Key och Value). Se exempel nedan. Se klassdiagrammet nedan



Ni kan med fördel lägga till privata metoder och attribut i klassen om det leder till tydligare/bättre kod.

Er kod ska fungera med åtminstone för requestsen nedan, men egentligen för alla giltiga HTTP-requests.

1. get-index.request.txt

Filens innehåll ser ut som följer:

```
GET / HTTP/1.1\n
Host: developer.mozilla.org\n
Accept-Language: fr\n
\n
```

Ditt program ska fungera enligt nedan (`#=>` används för att *illustrera* vad som ska skrivas ut):

```
request_string = File.read('get-index.request.txt')

request = Request.new(request_string)

p request.method   #=> 'GET'
p request.resource #=> '/'
p request.version  #=> 'HTTP/1.1'
p request.headers  #=> {'Host' => 'developer.mozilla.org', 'Accept-Language' => 'fr'}
p request.params   #=> {}
```

2. get-examples.request.txt

Filens innehåll ser ut som följer:

```
GET /examples HTTP/1.1\n
Host: example.com\n
User-Agent: ExampleBrowser/1.0\n
Accept-Encoding: gzip, deflate\n
Accept: */*\n
\n
```

Ditt program ska fungera enligt nedan (radbrytningarna i headers-utskriften är för tydlighet i visningen av exemplet och ert program ska/behöver inte göra radbrytningarna):

```
request_string = File.read('get-examples.request.txt')

request = Request.new(request_string)

p request.method   #=> 'GET'
p request.resource #=> '/examples'
p request.version  #=> 'HTTP/1.1'
p request.headers  #=> {'Host' => 'example.com',
#                       'User-Agent' => 'ExampleBrowser/1.0',
#                       'Accept-Encoding' => 'gzip, deflate',
#                       'Accept' => '*/*'}
p request.params   #=> {}
```

3. get-fruits-with-filter.request.txt

Filens innehåll ser ut som följer:

```
GET /fruits?type=bananas&minrating=4 HTTP/1.1\n
Host: fruits.com\n
User-Agent: ExampleBrowser/1.0\n
Accept-Encoding: gzip, deflate\n
Accept: */*\n
\n
```

Ditt program ska fungera enligt nedan:

```
request_string = File.read('get-fruits-with-filter.request.txt')

request = Request.new(request_string)

p request.method   #=> 'GET'
p request.resource #=> '/fruits?type=bananas&minrating=4'
p request.version  #=> 'HTTP/1.1'
p request.headers  #=> {'Host' => 'fruits.com',
#                       'User-Agent' => 'ExampleBrowser/1.0',
#                       'Accept-Encoding' => 'gzip, deflate',
#                       'Accept' => '*/*'}
p request.params   #=> {'type' => 'bananas', 'minrating' => '4'}
```

4. post-login.request.txt

Filens innehåll ser ut som följer:

```
POST /login HTTP/1.1\n
Host: foo.example\n
Content-Type: application/x-www-form-urlencoded\n
Content-Length: 39\n
\n
username=grillkorv&password=verys3cret!
```

Ditt program ska fungera enligt nedan:

```
request_string = File.read('post-login.request.txt')

request = Request.new(request_string)

p request.method   #=> 'POST'
p request.resource #=> '/login'
p request.version  #=> 'HTTP/1.1'
p request.headers  #=> {'Host' => 'foo.example',
#                       'Content-Type' => 'application/x-www-form-urlencoded',
#                       'Content-Length' => '39',
p request.params   #=> {'username' => 'grillkorv', 'password' => 'verys3cret!'}
```