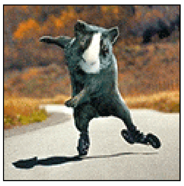
Net::HTTP Cheat Sheet

By Peter Cooper / January 16, 2010



Norwegian Rubyist August Lilleaas has been busy putting together a ton of examples of using the Net::HTTP Ruby library that comes with most Ruby distributions. I asked him if it'd be okay to put some of them directly on Ruby Inside for reference purposes and he said "No problem!"

It's worth noting that Net::HTTP has been superseded in many areas by libraries like John Nunemaker's HTTParty and Paul Dlx's high performance Typhoeus, but as part of the standard library, Net::HTTP is still a popular option though it doesn't have the easiest API to remember.

Here's a selection of August's examples for some of the most common operations. Want to see *all* of the examples and follow any updates made to them? Check out August's net-http-cheat-sheet GitHub repo.

Standard HTTP Request

```
require "net/http"
require "uri"

uri = URI.parse("http://google.com/")

# Shortcut
response = Net::HTTP.get_response(uri)

# Will print response.body
Net::HTTP.get_print(uri)

# Full
http = Net::HTTP.new(uri.host, uri.port)
response = http.request(Net::HTTP::Get.new(uri.request_uri))
```

Basic Auth

```
require "net/http"
require "uri"

uri = URI.parse("http://google.com/")

http = Net::HTTP.new(uri.host, uri.port)
request = Net::HTTP::Get.new(uri.request_uri)
request.basic_auth("username", "password")
response = http.request(request)
```

Dealing with response objects

```
require "net/http"
require "uri"

uri = URI.parse("http://google.com/")
```

```
http = Net::HTTP.new(uri.host, uri.port)
request = Net::HTTP::Get.new(uri.request_uri)

response = http.request(request)

response.code             # => 301
response.body             # => The body (HTML, XML, blob, whatever)
# Headers are lowercased
response["cache-control"] # => public, max-age=2592000
```

POST form request

```
require "net/http"
require "uri"

uri = URI.parse("http://example.com/search")

# Shortcut
response = Net::HTTP.post_form(uri, {"q" => "My query", "per_page" => "50"}

# Full control
http = Net::HTTP.new(uri.host, uri.port)

request = Net::HTTP::Post.new(uri.request_uri)
request.set_form_data({"q" => "My query", "per_page" => "50"})

response = http.request(request)
```

File upload - input type="file" style

```
require "net/http"
require "uri"

# Token used to terminate the file in the post body. Make sure it is not
# present in the file you're uploading.
BOUNDARY = "AaB03x"

uri = URI.parse("http://something.com/uploads")
file = "/path/to/your/testfile.txt"

post_body = []
post_body < < "--#{BOUNDARY}rn"
post_body < < "Content-Disposition: form-data; name="datafile"; filename="#
post_body < < "Content-Type: text/plainrn"
post_body < < "rn"
post_body < < File.read(file)
post_body < < "rn--#{BOUNDARY}--rn"

http = Net::HTTP.new(uri.host, uri.port)
request = Net::HTTP::Post.new(uri.request_uri)
request.body = post_body.join
request["Content-Type"] = "multipart/form-data, boundary=#{BOUNDARY}"

http.request(request)
```

SSL/HTTPS request

Update: There are some good reasons why this code example is bad. It introduces a potential security vulnerability if it's essential you use the server certificate to verify the identity of the server you're connecting to. There's a fix for the issue though!

```
require "net/https"
require "uri"

uri = URI.parse("https://secure.com/")
http = Net::HTTP.new(uri.host, uri.port)
http.use_ssl = true
http.verify_mode = OpenSSL::SSL::VERIFY_NONE

request = Net::HTTP::Get.new(uri.request_uri)

response = http.request(request)
response.body
response.status
response["header-here"] # All headers are lowercase
```

SSL/HTTPS request with PEM certificate

```
require "net/https"
require "uri"

uri = URI.parse("https://secure.com/")
pem = File.read("/path/to/my.pem")
http = Net::HTTP.new(uri.host, uri.port)
http.use_ssl = true
http.cert = OpenSSL::X509::Certificate.new(pem)
http.key = OpenSSL::PKey::RSA.new(pem)
http.verify_mode = OpenSSL::SSL::VERIFY_PEER

request = Net::HTTP::Get.new(uri.request_uri)
```

REST methods

```
# Basic REST.
# Most REST APIs will set semantic values in response.body and response.cod
require "net/http"

http = Net::HTTP.new("api.restsite.com")

request = Net::HTTP::Post.new("/users")
request.set_form_data({"users[login]" => "quentin"})
response = http.request(request)
# Use nokogiri, hpricot, etc to parse response.body.

request = Net::HTTP::Get.new("/users/1")
response = http.request(request)
# As with POST, the data is in response.body.

request = Net::HTTP::Put.new("/users/1")
request.set_form_data({"users[login]" => "changed"})
response = http.request(request)

request = Net::HTTP::Delete.new("/users/1")
response = http.request(request)
```

There are more in August's repo if you want to keep browsing..