

Core Ruby 41st Batch

Home ► PORPC101-41C ► 26 April - 2 May ► *PARALLEL TRACK* Understand HTTP concepts

Navigation

Home

■ My home

Site pages

My profile

Current course

PORPC101-41C

Participants

General


5 April - 11 April

12 April - 18 April


19 April - 25 April


26 April - 2 May


 Week 4: Tutorial


 Week 4:
Exercises

 Week 4: Forum

 Some useful
Ruby methods

 Playfair
Instructions

 Playfair Cipher
Forum

 ***PARALLEL
TRACK*
Understand
HTTP concepts**

 Week 4: Quiz

3 May - 9 May

10 May - 16 May

17 May - 23 May

24 May - 30 May

31 May - 6 June

PARALLEL TRACK Understand HTTP concepts

When you start building web applications, you would be dealing with HTTP a lot.

What's HTTP?

HTTP means HyperText Transfer Protocol. A protocol is just a convention for computer dialogs. It assumes that the computers participating in the conversation have a way of sending messages to each other.

HTTP uses a simple conversation pattern: the client connects to the server, initiates the dialog by asking the server for something, the server then tries to provide the client with an answer (back).

An example of a simple HTTP request and response would be the client asking "give me the file A" and the server answering "I have it, here is the content of file A: (...content of file A...)".

But in reality the protocol uses its own simplified language which looks more like "GET /fileA" answered by "200 OK (...content of file A...)".

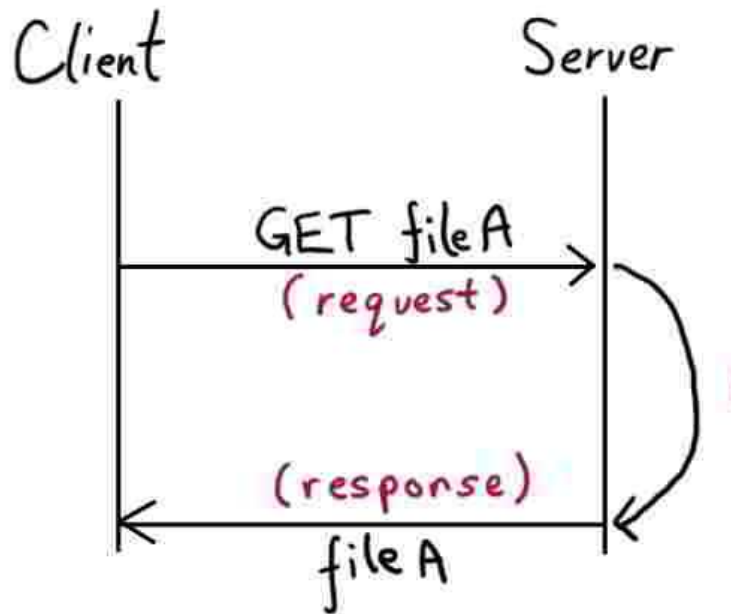
7 June - 13 June

My courses

Settings

Course administration

My profile settings



HTTP

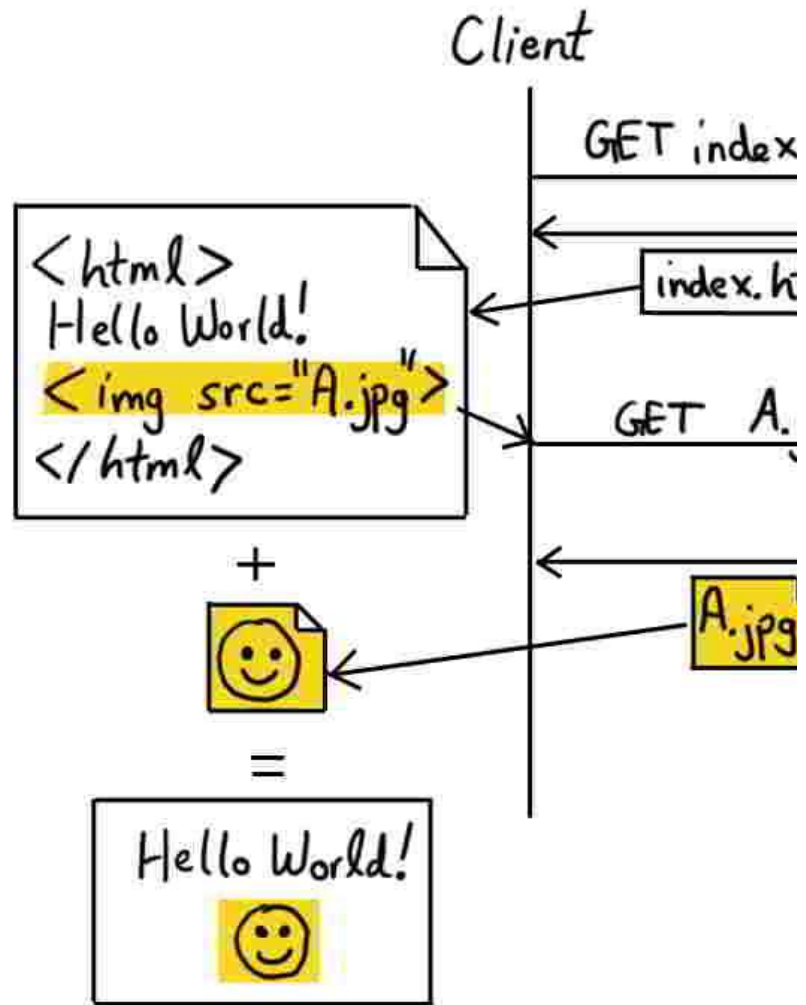
Loading a web page

When a browser loads a web page, it does a GET request for the URL requested. The content of the page is usually in a special text format called HTML that allows links, images and various style effects.

The browser analyses the HTML in order to display it to the user. But doing so, it may find that it needs more content to render the page correctly, like images.

When that happens, the browser initiates one HTTP request for each image.

When all the necessary information is downloaded, it is combined into the page the user sees on his screen.



HTTP

HTTP request methods (verbs)

GET

A resource is some chunk of information that can be identified by a URL (it's the R in URL - Uniform Resource Locator). The most common kind of resource is a file, but a resource may also be a dynamically-generated query result, a document that is available in several languages, or something else.

GET is the most common HTTP method; it says "give me this resource". Method names are always uppercase.

A typical request looks like this:

```
GET /path/to/file/index.html HTTP/1.0
```

The path is the part of the URL after the host name, also called the request URI (a URI is like a URL, but more general).

The HTTP version always takes the form "HTTP/x.x", uppercase.

POST

A POST request is used to send data to the server to be processed in some way. A POST request is different from a GET request in the following ways:

- There's a block of data sent with the request, in the message body.
- The request URI is not a resource to retrieve; it's usually a program to handle the data you're sending.
- The HTTP response is normally program output, not a static file.

PUT

A PUT request uploads a document from either the local file system or a remote HTTP server to the destination HTTP server.

DELETE

A DELETE request deletes a resource (or makes it unavailable) for future references.

HTTP response codes

After a client (such as a web browser) sends a request corresponding to one of the HTTP verbs, the web server responds with a numerical code indicating the HTTP status of the response. For example, a status code of 200 means "success", and a status code of 301 means "permanent redirect". If you install curl (when you installed the Bash shell, curl was also installed), a command-line client that can issue HTTP requests, you can see this directly at, e.g., www.satishtalim.com (where the --head flag prevents curl from returning the whole page). Open a Bash shell and type:

```
$ curl --head www.satishtalim.com
HTTP/1.1 200 OK
.
.
.
```

Here satishtalim.com indicates that the request was successful by returning the status 200 OK. In contrast, google.com is permanently redirected (to www.google.com, naturally), indicated by status code 301 (a "301 redirect"):

```
$ curl --head rubylearning.org
HTTP/1.1 301 Moved Permanently
Location: http://www.google.com/
.
```

•
•

Last modified: Friday, 15 June 2012, 7:32 PM

You are logged in as [C. O'Keefe](#) ([Logout](#))



© 2008-2013 RubyLearning.org : A Ruby Learning Hub