HTTP を学ぼう

斎藤 祐一郎[*1]

2013 年 12 月 16 日

[*1] http://www.koemu.com/

目次

第 1 章

はじめに

今回の講義では、HTTP [?] を学びます。

HTTP とは、インターネット上で通信する際の「約束」の一つです。インターネットを利用したソフトウェアを開発する時は、自分が開発したものとは違うソフトウェア同士で通信する事が珍しくありません。そこで、プロトコルという約束を決める事で、お互いが正しく通信できるようにしているのです。

HTTP は、正式には Hypertext Transfer Protocol といい、インターネットを通じて HTML(Hypertext Markup Language [?]) ファイルを転送 (Transfer) するための約束 (Protocol) のことです。今日、HTTP の知識はインターネットを利用するソフトウェア開発において、無くてはならないものとなっています。

…とはいっても、これだけではよくわからないですよね。そこで、どのようなものなのか、これから一緒に探って行きましょう。

第 2 章

身近な HTTP

2.1　ブラウザで見てみる

HTTP は、皆さんにとってとても身近なものです。では、生活にどのように関わっているのでしょうか。

まずは、PC で Yahoo! Japan のトップページを見てみます。

図 2.1　Yahoo! Japan のトップページ

では、iPhone ではどうでしょうか。

図 2.2　Yahoo! Japan のトップページ

「なんだ、ブラウザに Web ページが表示されているだけじゃないか！」と考えた人もいると思います。

そうです、それで正しいのです。HTML は Hypertext Transfer Protocol、即ち HTML を転送するために生まれた通信の約束なのです。

2.2　ブラウザ以外で見てみる

さて、HTTP は「HTML ファイルを転送するための約束」となると、転送されたデータの使い道は、ブラウザである必要は全くないのです。こちらの画面、iPhone を使った事がある人だと見覚えがありませんか？

図 2.3　App Store のページ

「あれ、ブラウザではないけど、どうして？」と思うかもしれません。それに、HTTP なのに HTML ページを転送していないじゃないか！と思われるかもしれません。

でも、これも実は正しいのです。現在、HTTP は HTML ファイルを転送するだけではなく、ネトゲ[*1]をはじめとしてネットワークを通じて情報をやり取りするためにも用いられています。

だから、HTTP は必ずブラウザだけで利用しなければならない、という事は無いのです。

[*1] ネットワークゲーム・オンラインゲーム

第 3 章

他にもあるプロトコル

3.1 どんなプロトコルがあるのだろう？

インターネットで通信をするにあたって、プロトコルと言う通信の約束がある事がわかりました。その中に、HTTP があります。

実は、HTTP 以外にもプロトコルがあります。それは、何でしょうか？考えて、挙げてみましょう。

-
-
-
-
-
-
-
-
-
-

3.2 どんなプロトコルがあるのだろう？

いくつでてきましたでしょうか？ここで、よく使われるプロトコルを挙げてみます。

■FTP [?] File Transfer Protocol。ファイルをやり取りするためのプロトコル。最近はあまり使われなくなりました。

■SSH [?] Secure SHell。暗号化通信を用いてサーバと直接接続するためのプロトコル。サーバをメンテナンスするとき等によく使われます。

■SMTP [?] Simple Mail Transfer Protocol。メールを送信するためのプロトコルです。Thunderbird からメールを送信する時、SMTP が使われているんですよ。

■DNS [?] Domain Name System。「ドメイン」からインターネット上の IP アドレスを探す時に使う DNS サーバと通信するためのプロトコルです。とても身近なものですが、説明はまた別の機会に。

■POP3 [?] Post Office Protocol Version 3。メールサーバからメールを受信するためのプロトコルです。こちらも、皆さんよく使われていると思います。

■NTP [?] Network Time Protocol。時計合わせを行うためのプロトコルです。コンピュータの時計が合っているのは、このプロトコルの活躍があります。

■TCP [?] Transmission Control Protocol。先ほどまで挙げてきたプロトコルを支えるプロトコルです。正しいデータが流れるよう、通信の内容を管理しています。

■UDP [?] User Datagram Protocol。TCP の兄弟ですが、こちらはデータが正しく到達したかは管理しません。即時性を求めるデータを転送する時に使われる場合があります。

第 4 章

どうやってやりとりしているの？

4.1　サーバとユーザエージェント

皆さんはインターネット上に「サーバ」があることをご存知だと思います。また、サーバからデータを受け取る端末があります。HTTP ではこれを「ユーザ エージェント」と呼びます。

さて、人同士で互いの意思を伝え合う時、会話をしますよね。多くの人は、それを「声」という形で行います。

では、「サーバ」と「ユーザ エージェント」の間では、どのような会話…コンピュータ同士ですから通信が行われているでしょうか？ちょっと考えて、吹き出しの中、および通信方向に矢印を書き込んでみましょう。

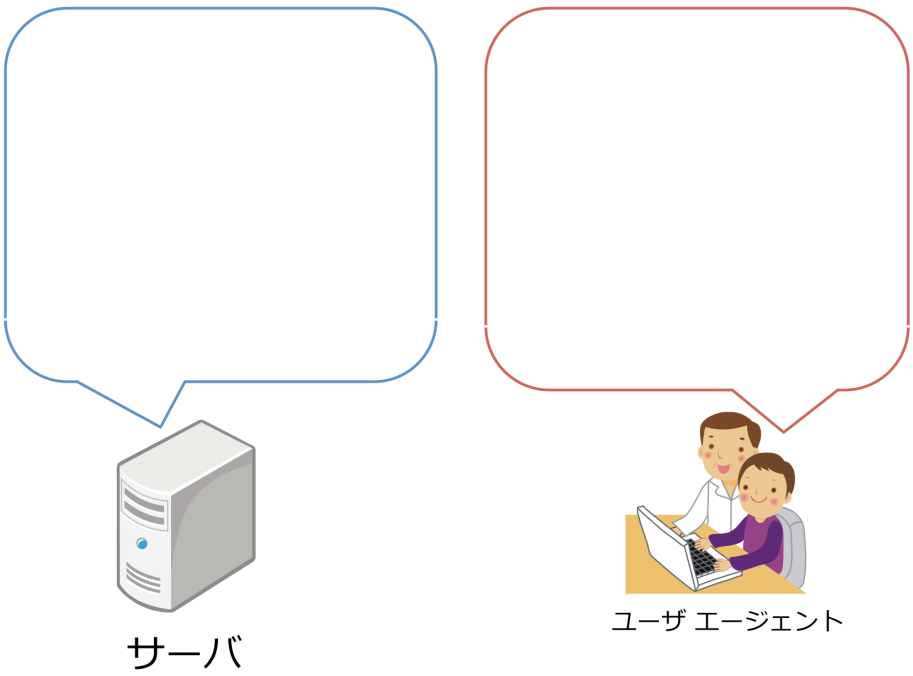

図 4.1　どんな通信が行われているのでしょう！？

いかがでしょうか？みんなで見合わせて話し合ってみましょう。

4.2　リクエストとレスポンス

HTTP の通信は、次の 2 つの通信で説明する事ができます。

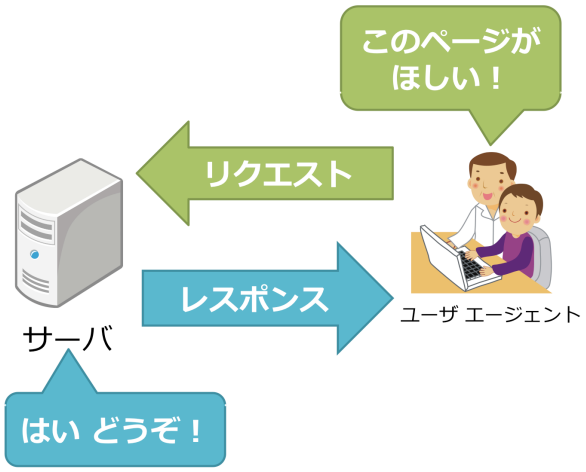

図 4.2　リクエストとレスポンス

ユーザ エージェント側から発せられる通信を「リクエスト」(日本語: 要求)、サーバから返される通信を「レスポンス」(日本語: 回答) といいます。
これを、日常の会話のモデルに当てはめて考えてみましょう。

表 4.1　リクエストとレスポンス

	人	HTTP 通信
リクエスト	「あなたは何歳ですか？」	「トップページをください」
レスポンス	「32 歳です」	トップページのデータが届く

単純ですね。でも、これが HTTP 通信の大切な基本です。

第 5 章

通信の時に何がやり取りされているの？

5.1　実際に通信してみよう！

いきなりですが、Windows の「コマンドプロンプト」を開いて、サーバと通信してみましょう。

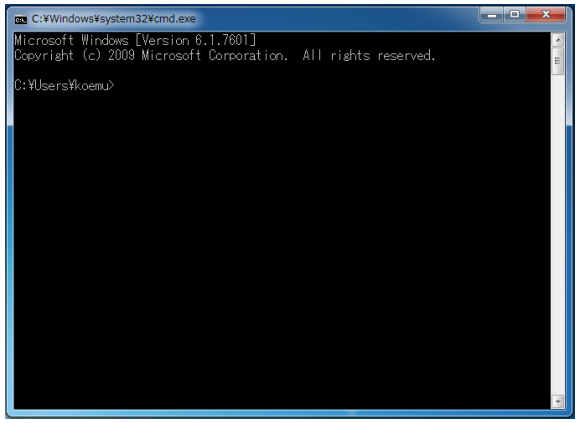

図 5.1　ブラウザで表示

上記の画面が開けたら、上記の文字を入力して Enter を押してください。

```
1 C¥:> telnet www.koemu.com 8888
```

続いて、次の文字が表示されるか確認してください。

```
1 Trying 219.94.232.204...
2 Connected to www.koemu.com.
3 Escape character is '^]'.
```

そして、次の文字を入力します。文字が見えない人がいるかもしれませんが、目をつぶっていると思って、がんばって打ち込んでみてください！

```
1 GET / HTTP/1.0
```

最後に、Enter を 2 回押したら…何がでるかを確かめてみてください。

5.2 どのような通信が行われているのだろう？

改めて、先ほど打ち込んだ内容について確認してみましょう。
最初に、以下の内容を打ち込みました。

```
C¥:> telnet www.koemu.com 8888
```

これは、つなぐサーバのアドレスを入力しています。
次に、画面に次の情報が表示されました。

```
Trying 219.94.232.204...
Connected to www.koemu.com.
Escape character is '^]'.
```

1 行ずつ追いかけてみましょう。
1 行目は、「Trying 219.94.232.204...」は「IP アドレス 219.94.232.204 にあるサーバに接続を試みています。」です。
2 行目は、「Connected to www.koemu.com.」は「www.koemu.com に接続しました。」となっています。うまく接続できたみたいですね。
3 行目は、「Escape character is '^]'.」→「エスケープ文字 (特別な意味がある文字) は'^]'(ESC キー) です」とでています。これは接続する事とは直接関係ありません。
そうです、ここで早速「リクエスト」と「レスポンス」のやり取りが行われているのです。初めに打ち込んだ文字が、サーバに接続しますというリクエスト、返ってきた 3 行の文字列がサーバからのレスポンスです。
続いて、次の文字を打ち込みました。

```
GET / HTTP/1.0
```

これは、サーバに「ルートディレクトリ (フォルダの中の一番上にある所) にあるファイルが欲しい」というリクエストです。この書き方は、HTTP で定義された書き方に則っています。
そして、サーバから返ってきたレスポンスがこちら。

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/plain
Content-Length: 12
Connection: close

Hello World
Connection closed by foreign host.
```

8 行のレスポンスが返ってきました。このレスポンスも、HTTP に則っています。
1 行目、「HTTP/1.1 200 OK」ですが、これはサーバが「正しく処理できました」と返事をしています。200 という数字は「ステータスコード」という名前があり意味もあるのですが、後ほど説明します。
2 行目、「X-Powered-By: Express」は、サーバ側で「Express」というプログラムがこの処理を管理している事を示します。ただし、ユーザエージェントに対して意味があるレスポンスではありません。このようなレスポンスには、たいてい「X-」で始まる名前がついています。

3 行目、「Content-Type: text/plain」は、コンテントタイプと言って、どのようなファイルをレスポンスとして返すかを示しています。今回は text/plain、すなわち皆さんがプログラム等を作る時に書いているテキスト形式のデータである事を伝えてきています。

4 行目、「Content-Length: 12」は、返すファイルのバイト数を示します。

5 行目、「Connection: close」は、接続を終えた事を意味します。close 以外もあるのですが、今回は説明を省略します。

6 行目、ここが空白です。実は、この空白には意味があります。空白行手前までが「ヘッダ」といって、主にサーバがどのような処理をしたかを示す情報が含まれています。

7 行目、「Hello World」とあります。これが、レスポンスとして返したいファイルの中身です。今回はテキスト形式ですので、そのままテキスト形式の”Hello World”という文字がでています。また、この長さは、4 行目の「Content-Length」のバイト数と一致します。

8 行目、「Connection closed by foreign host.」とありますが、これはサーバへ接続するためのソフトが発したメッセージで、レスポンスではありません。すなわち、レスポンスは 1 行前までとなります。

ここで知っていただきたい事があります。

- HTTP には、リクエスト・レスポンス、どちらにも書き方の約束が決まっている。
- レスポンスの中身は、1 行空けた手前をヘッダ、その先にリクエストされたデータが入っている。

5.3　ブラウザでやってみよう

先ほど、サーバと HTTP で通信した事を、ブラウザでやってみましょう。
Internet Explorer などのブラウザを立ち上げて、次の URI を入力し、Enter を押してください。

- http://www.koemu.com:8888/

その結果、次のページが表示されます。

図 5.2　ブラウザで表示

先ほど、コマンドプロンプトで確認した 2 回目のレスポンスの、5 行目だけでて来ていると思います。

そうです、ブラウザでは先ほど入力・受け取った内容のほとんどを、プログラムで処理しているのです。ブラウザは偉大ですね。

もちろん、自分自身で HTTP のリクエスト・レスポンスを自動処理するソフトウェアを作ることもできます。

5.4 他のリクエストはどうだろう

先ほどと同様に、サーバに接続してみます。

```
1 C¥:> telnet www.koemu.com 8888
```

そして、次の文字が表示されるか確認してください。

```
1 Trying 219.94.232.204...
2 Connected to www.koemu.com.
3 Escape character is '^]'.
```

今回は、/(デフォルトのファイル) ではなく、"hoge.html"を取り出すリクエストを出してみましょう。

```
1 GET /hoge.html HTTP/1.0
```

Enter を 2 回押したら…何がでるかを確かめてみてください。

5.5 えっ、ページが無い？

あれ…先ほどと様子が違いますね。

```
1 HTTP/1.1 404 Not Found
2 X-Powered-By: Express
3 Content-Type: text/plain
4 Connection: close
5
6 Cannot GET /hoge.html
7 Connection closed by foreign host.
```

では、先ほどと違っている部分を確認して行きましょう。

1 行目。「HTTP/1.1 404 Not Found」とあります。これは「ファイルが無い」事を意味しています。そう、"hoge.html"はサーバには存在しないのです。あと、ステータスコードが 200 ではなく 404 になっています。

6 行目、本来ファイルの中身が表示される部分に「Cannot GET /hoge.html」、即ち「hoge.html を取得できません」と表示されています。これは、サーバ側であらかじめ定義された文章が返されるのが一般的です。

リクエストは必ずうまく行くものではなく、失敗する場合もある事を覚えておきましょう。

5.6 エラーの状況をブラウザで確認してみよう

では、エラーはブラウザではどのように見えるのでしょうか。

Internet Explorer などのブラウザを立ち上げて、次の URI を入力し、Enter を押してください。

- http://www.koemu.com:8888/hoge.html

その結果、次のページが表示されます。

図 5.3　エラーページがブラウザによって変更される例

先ほどのサーバのレスポンスでは「Cannot GET /hoge.html」とでていましたが、ブラウザにはでていません。これは、ブラウザがステータスコードを確認して、ユーザエージェントを利用している人がわかりやすいように代替のメッセージを出しているためです[*1]。

[*1] 「Cannot GET /hoge.html」と出してくれるブラウザもあります

第 6 章

ステータスコードを知ろう

6.1　ステータスコードは 3 桁の数字

さて、先ほどから 200, 404 などのステータスコードがでてきています。これは、サーバが、ユーザエージェントからのリクエストが正しく処理できたか否かを示す数値で、レスポンスの中に含まれています。

では、どのようなステータスコードが定められているでしょうか。自分がサーバの開発者だったらどのようなステータスコードを定めますか？または、ユーザエージェント側のソフトウェアを開発するなら、どのようなステータスコードが用意されているといいでしょうか。考えてみましょう。

-
-
-
-
-
-
-
-
-

書き終わったら、友達同士で見せ合ってみましょう。

6.2　代表的なステータスコード

ステータスコードは、先にも述べた通り定義されています。
まず、数値には付番の原則が決まっています。

- 100 番台 — 情報[*1]
- 200 番台 — 成功
- 300 番台 — ページが移動した
- 400 番台 — クライアント側のエラー
- 500 番台 — サーバ側のエラー

代表的な HTTP ステータスコードを説明します。

- 200 — OK
- 301 — Moved Permanently: ファイルが移動しました
- 307 — Temporary Redirect: ファイルが一時的に移動しました
- 401 — Unauthorized: 認証エラー (パスワードを間違えた等)
- 403 — Forbidden: 許可が無い (特定の人のみアクセスできる場所)
- 404 — Not Found: ファイルが無い
- 500 — Internal Server Error: サーバ上でエラーが発生した

また、ステータスコードを見ると、プログラム側でも容易に成功・失敗を解釈する事もできます。プログラムを開発する時、エラーの有無を確認する時に使ってみてください。
全てのステータスコードは、RFC に定義されています。時間がある時に読んでみましょう。

- http://www.studyinghttp.net/cgi-bin/rfc.cgi?2616

[*1] あまり使われません

第 7 章

きょうのまとめ

では、今日学んだ事を復習しましょう。

1. HTTP はインターネットを通じてファイルを転送する約束を定義したもの。
2. 通信には「リクエスト」と「レスポンス」がある。
3.「リクエスト」には、主に取り出したいページファイルのアドレスを送っている。
4.「レスポンス」には、ページのデータの他に、リクエストの成否が含まれている。
5.「ステータスコード」はレスポンスに含まれており、リクエストがサーバ上でどのように処理されたかを知る最も単純な方法である。

普段の TENTO の人とは違って、勉強っぽくなりちょっと大変だったかもしれません。

プログラムを書く事と同時に、ソフトウェアを取り巻く約束事の理解を深めて、より高度なソフトウェアが開発できるよう、取り組んで行きましょう！

今日の講義、おつかれさまでした。

参考文献