

Web

歴史

1989 CERNで提案

素粒子物理実験関連の文書やデータを世界中で交換するため

Hypertext '91 conference

Mosaic 1993 - 最初のグラフィカルブラウザ

Netscape Navigator, Internet Explorer

W3C (World Wide Web Consortium) 1994

Architectural Overview

Web page を表示するシステム

相互のリンク (hyperlink) - hypertext (Vannevar Bush 1945)

browser - ページを表示

HTTP (HyperText Transfer Protocol) - TCP上で動作

static page - ファイルに格納されいつも同じ内容

dynamic page - プログラムで生成 / プログラムを含む

The Client Side

URL (Uniform Resource Locator) - ページの名前と場所を示す

例: `http://www.uec.ac.jp/index.html`

protocol (`http`)

ホストのDNS名 (`www.uec.ac.jp`)

パス名 (`index.html`)

リンクをクリックしたときの動作

1. ブラウザはURLを決定
2. DNSでサーバのIPアドレスを問い合わせ
3. DNSでIPアドレスを得る
4. そのIPアドレスの80番ポートへTCPコネクション
5. `/index.html` を要求するHTTPリクエストを送信
6. サーバはページをHTTPレスポンスとして返す
7. 受け取ったページの中に表示すべきURLが含まれていれば, それらについても同様の手順(2-6)を繰り返す (画像やスクリプト(後述)など)
8. ブラウザは受け取ったページを画面に表示
9. TCPコネクションを閉じる

URLは特定のホスト名を含んでいる - そうでない情報は表現できない
たとえば特定の文書や特定の本など

URN (Uniform Resource Name)

例: 本をISBNで指定 `urn:isbn:0451450523`

URI (Uniform Resource Identifier) - URL, URN の総称

プレインテキスト以外のページ (画像, 動画, PDF, DOC, ...)

MIME Type で区別 - HTTPレスポンス中に存在

(Multipurpose Internet Mail Extension)

ファイルの拡張子で区別することもある

ブラウザでの表示

Plug-in または Helper application

The Server Side

動作 (かっこ内は dynamic page のとき)

1. TCPコネクションをacceptする
2. 要求されたパス名を受け取る
3. ディスクからそのファイルを読み出す (あるいはプログラム起動)
4. そのファイルの内容(あるいは処理結果)をクライアントに送信
5. TCPコネクションの解放

HTTP (HyperText Transfer Protocol)

1991 HTTP 0.9 (と通称)

1996 HTTP 1.0 RFC 1945

1999 HTTP 1.1 RFC 2616

2015 HTTP/2 RFC 7540

TCP上で動作 - 本来はwebのためのアプリケーションプロトコル

しかし現代では他のサービスも利用 - 一種のトランスポートプロトコル

自由なポートの接続をフィルタする傾向が強まっていることも原因

例: antivirus software の更新
HDレコーダ内蔵のwebサーバでコマンド受付
プログラム同士の交信

connection

TCP上なので, メッセージ長制限, エラー, 輻輳の心配なし

メッセージ

HTTP 0.9

(接続)

GET <path>

<レスポンス: 指定したHTMLファイルの内容>

(接続終了)

HTTP 1.0 から

リクエスト、レスポンスにヘッダがつくようになった

ヘッダの終了は空行

一度の接続でリクエスト/レスポンス一対だけだったが

HTTP 1.1 から persistent connection

一度の接続で複数のリクエスト/レスポンスを送受信

TCPの接続設定のオーバーヘッドを減らせる

さらに, レスポンスを待たずに次のリクエストも可能 - pipeline
メインページに含まれる画像などを次々に要求できる

method

GET, HEAD, POST

ほかにも各種

status code - レスポンスで返ってくる

1xx Information

2xx Success

3xx Redirection

4xx Client Error

5xx Server Error

Message Headers (Wikipedia「Hypertext Transfer Protocol」に一覧あり)

リクエストでGETなどの後に続く付加情報 (request header)

レスポンスにも付加される (response header)

例:

User-Agent (Request) ブラウザに関する情報

If-Modified-Since (Request) ページが更新されているかどうか

Content-Type (Response) ページのMIMEタイプ

Caching

一度読み出したページをブラウザが保存しておく

HTTPヘッダで変更が確認できるので, 変化しなければキャッシュを利用

conditional GET

HTTPの観察

Webサーバからの応答を観察

telnet コマンドで要求を送信 (ブラウザは使わない)

```
-----  
HTTP 0.9 のリクエスト  
$ telnet pr.ice.uec.ac.jp 80  
Trying 192.168.1.4...  
Connected to pr.ice.uec.ac.jp.  
Escape character is '^]'. (ここでTCP接続完了)  
GET / (リクエストはこの一行)  
<HTML> (ここからはサーバのレスポンス; HTMLファイルそのまま)  
<HEAD>  
<TITLE>top page</TITLE>  
<meta http-equiv="Content-Type" content="text/html; charset=euc-jp">  
</HEAD>  
  
<BODY>
```

top page of pr.ice.uec.ac.jp

```
<hr>
<a href="~/terada/openhouse/"> openhouse on 7/14 </a> <br>
<a href="~/terada/chofusai/"> openhouse on 11/23 </a>
<hr>

<ul>
<li> <a href="http://sp.ice.uec.ac.jp/"> TERADA Lab.</a>
<li> <a href="~/terada/"> Minoru TERADA </a>
</ul>
</body>
</html>
```

(ここまででHTMLファイル終了)

Connection closed by foreign host. (サーバがTCP接続を切った)

HTTP 1.1 のリクエスト形式

\$ telnet pr.ice.uec.ac.jp 80

Trying 192.168.1.4...

Connected to pr.ice.uec.ac.jp.

Escape character is '^['.

GET / HTTP/1.1

Host: pr.ice.uec.ac.jp (request header)

(この空行でリクエスト完了)

HTTP/1.1 200 OK (ここからサーバのレスポンス)

Date: Thu, 10 Oct 2013 06:32:49 GMT (response header が並ぶ)

Server: Apache

Last-Modified: Wed, 10 Jul 2013 14:26:04 GMT

ETag: "1083f3-161-ebcb700"

Accept-Ranges: bytes

Content-Length: 353

Content-Type: text/html (MIMEを使ったデータの種別)

(この空行のあとが要求したHTMLファイル)

<HTML>

<HEAD>

<TITLE>top page</TITLE>

<meta http-equiv="Content-Type" content="text/html; charset=euc-jp">

</HEAD>

<BODY>

... 中略 ...

</body>

</html>

(サーバは次のリクエストを待っている)

Connection closed by foreign host. (サーバがタイムアウトして接続を切った)

Web Proxies

Web proxy - 通常は組織の入り口に設置

複数のユーザからのHTTPリクエストを受け, キャッシュする

ヒット率向上

フィルタリングも可能

NATを使用せずに外部のウェブページを利用可能

ブラウザの設定でproxyを使用

HTTPコネクションはすべて proxy にむかう

Cookies

ページの獲得はそれぞれ独立している - 状態をもたない

ログインセッションの概念がない

あるユーザからの一連のリクエストを他と識別したい

IPアドレスでは不十分

NAT だと同一, DHCP では一意でない(割り当てが変化する)

RFC 2109

クライアントがページを要求したとき, サーバがCookieを付加してページを返信

ブラウザは受け取ったCookieを保存

Cookie - 5つのフィールド

Domain - サーバのドメイン名

Path - サーバ内のディレクトリパス (サーバ内でのアプリケーションの区別)

Content - "name = value" 任意の文字列

Expires - ブラウザでの保存期限

指定なければブラウザ終了時まで - nonpersistent cookie

Secure - secure connection のときだけ返送

Cookieを使う

ブラウザは以後, 対応するドメイン名のサーバへのHTTPリクエストに

保存しているCookieをすべて返送

サーバはそれを自由に利用できる - たとえば

顧客番号

現在の買い物カートの内容

ユーザがカスタマイズした興味分野(スポーツとか)

訪問回数カウンタ

実演 (http://pr.ice.uec.ac.jp/~terada/learn/php/cookie0.php)

一度目のアクセスと二度目のアクセスで内容が違う

ブラウザを再起動すると一度目に戻る

cookieのデフォルトの保存期限はブラウザ終了時

ブラウザを再起動すると前のcookieは捨てられる

サーバ側でのcookieの利用法 - 上記の例の実装

セッションごとに変数のセットをファイルに保存して利用

-----cookie0.php-----

```
<?php session_start(); ?>
```

```
<html>
```

```
<body>
```

```
<p>012345</p>
```

```
<?php
```

```
if (isset($_SESSION['count'])) {
    echo "count = ", $_SESSION['count'];
} else {
    echo "count undefined.";
    $_SESSION['count'] = "";
}
```

```
$_SESSION['count'] .= "w";      (文字列の接続)
?>
```

```
<p>abcdef</p>
```

```
</body>
```

```
</html>
```

PHPで session_start() が呼び出されると,

クライアントからのリクエストにcookieがあるか確認

ある場合 - そのcookieにひもづいているファイルから変数群を読み出す

/var/lib/php/session/<PHPSESSIDの値>

本体を実行

変数群をそのファイルに格納しなおす

ない場合 - 新規のcookieを作成して送り返す(PHPSESSID)

本体を実行

変数群をそのファイルに格納する

2017-10-13

HTTPの観察

前述のtelnetコマンドでの観察では不十分

ローカル側が人間ではなくブラウザだから

ブラウザ側からのメッセージも確認するには

一種の proxy をはさむ

ブラウザからは proxy として設定

コネクションはすべてそのプログラムへ
到来したコネクションは、正しい proxy へ中継
その際に通信内容をプリントする

行頭の0: ブラウザ -> サーバ
行頭の1: サーバ -> ブラウザ

```
---(一回目)-----
% new connection: 0 Thu Oct 10 16:10:03 JST 2013 /192.168.1.11 49572 proxy-west.uec.ac.jp/1
30.153.8.66 8080
0:GET http://pr.ice.uec.ac.jp/~terada/learn/php/cookie0.php HTTP/1.1
0:Host: pr.ice.uec.ac.jp
0>User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101 Firefox/24.0
0:Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
0:Accept-Language: ja,en-us;q=0.7,en;q=0.3
0:Accept-Encoding: gzip, deflate
0:Connection: keep-alive (ここまで、ブラウザがリクエストに付加するヘッダ)
0:
1:HTTP/1.0 200 OK
1>Date: Thu, 10 Oct 2013 07:10:03 GMT
1:Server: Apache
1:X-Powered-By: PHP/5.1.6
1:Set-Cookie: PHPSESSID=93896v2s04h67mcqe7pla9ckj5; path=/
1:Expires: Thu, 19 Nov 1981 08:52:00 GMT
1:Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
1:Pragma: no-cache
1:Content-Length: 78
1:Content-Type: text/html
1:X-Cache: MISS from proxy-west.uec.ac.jp
1:Via: 1.1 proxy-west.uec.ac.jp:8080 (squid/2.7.STABLE7)
1:Connection: keep-alive
1:Proxy-Connection: keep-alive
1:
   (この空行がresponse headerとページの境界)
1:<html>
1:<body>
1:
1:<p>012345</p>
1:
1:count undefined. (Cookieのあるなしでこの行の反応を変えている)
1:<p>abcdef</p>
1:
1:</body>
1:</html>
---(二回目)-----
0:GET http://pr.ice.uec.ac.jp/~terada/learn/php/cookie0.php HTTP/1.1
0:Host: pr.ice.uec.ac.jp
0>User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101 Firefox/24.0
0:Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
0:Accept-Language: ja,en-us;q=0.7,en;q=0.3
0:Accept-Encoding: gzip, deflate
0:Cookie: PHPSESSID=93896v2s04h67mcqe7pla9ckj5 (前に受け取ったCookieを返送)
0:Connection: keep-alive
0:Cache-Control: max-age=0
0:
HTTP/1.0 200 OK
1>Date: Thu, 10 Oct 2013 07:10:18 GMT
1:Server: Apache
1:X-Powered-By: PHP/5.1.6
1:Expires: Thu, 19 Nov 1981 08:52:00 GMT
1:Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
1:Pragma: no-cache
1:Content-Length: 71
1:Content-Type: text/html
1:X-Cache: MISS from proxy-west.uec.ac.jp
1:Via: 1.1 proxy-west.uec.ac.jp:8080 (squid/2.7.STABLE7)
1:Connection: keep-alive
1:Proxy-Connection: keep-alive
1:
```

```
1:<html>
1:<body>
1:
1:<p>012345</p>
1:
1:count = w (Cookieのあるなしでこの行の反応を変えている)
1:<p>abcdef</p>
1:
1:</body>
1:</html>
```

サイトを越えたユーザトラッキング

サイトA のページa に広告画像(でなくてもいいが)を貼り付ける

<http://www.sneaky.com/382674902342.gif>

サイトB のページb に対しても同様に

<http://www.sneaky.com/193654919923.gif>

ページa をアクセスすると、画像にも要求がいく

その時、sneaky.com は Cookie を送信 (たとえば UserID=4627239101)

その後、ページb をアクセスしたとき、sneaky.com にはそのCookieが返送される

-> ページa, bを閲覧したことがわかる (そのユーザの行動が把握できる)

画像は1ピクセルでも構わない; それをクリックする必要もなし

ユーザが気づかないのは問題

自分のブラウザが保存しているCookieを確認しよう

ブラウザがCookieを受け取らないように設定することも可能

ただし、サイトの動作に支障がでるケース

third-party cookie をブロックする

主ページとは違うサイトからのCookie (上例の sneakey.com)