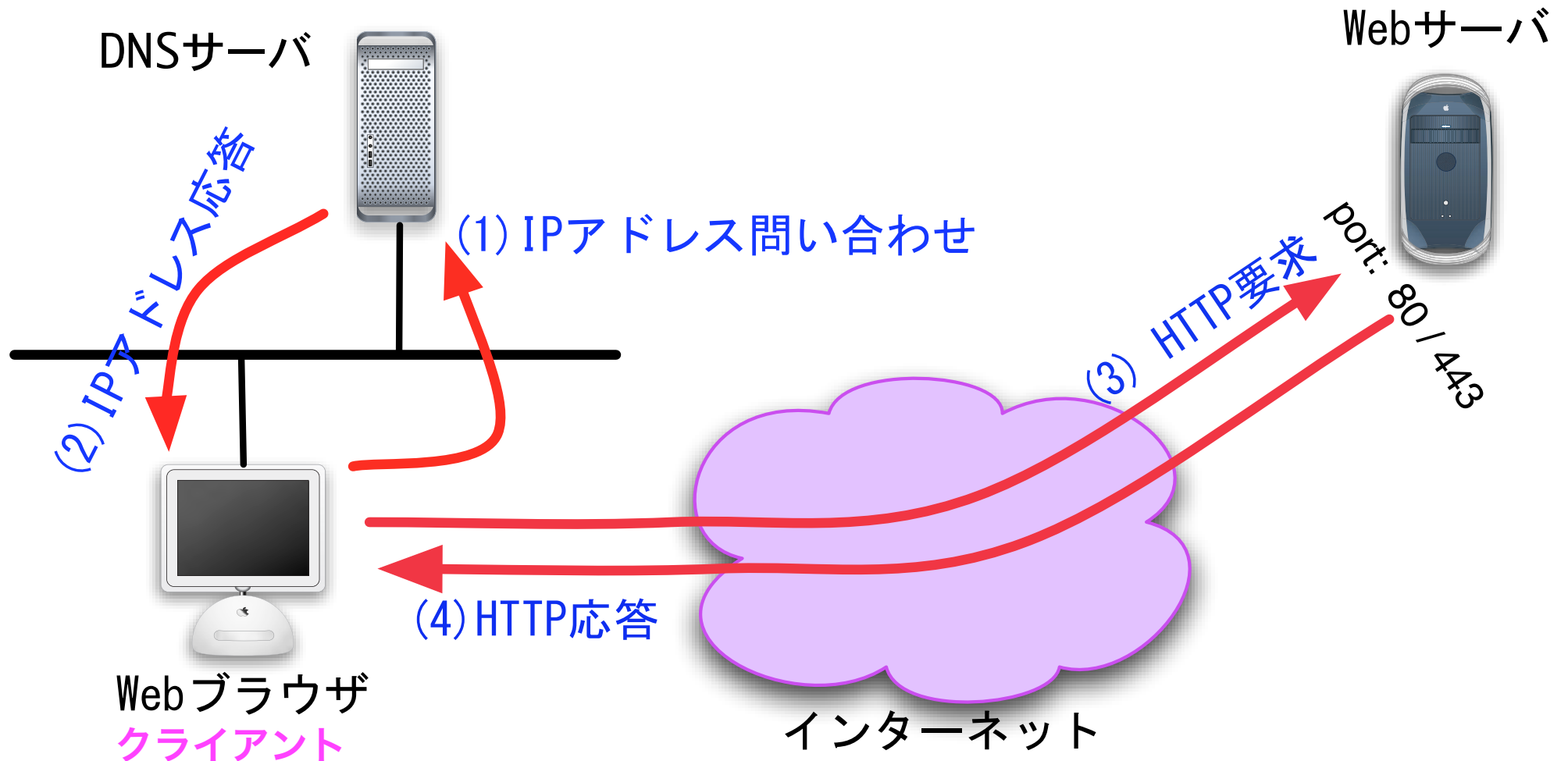


Webを実現するHTTP

World Wide Webの通信



WWWの成立に必要な 3 要素

- **URL/URI**(Uniform Resource Locator/Identifier)
 - ネットワーク上の情報を特定・取得するための方法（スキーム）
- **HTTP**(Hyper Text Transfer Protocol)
 - Webクライアントとサーバ間の通信規約
- **HTML**(Hyper Text Markup Language)
 - 情報をハイパーテキストとして表現する記述法.

URL(Uniform Resource Locator)

インターネット上にあるサーバにある情報資源を取得するために、そのアクセス方式と位置を記述

URL = スキーム:その方式の固有部分

スキーム例

http Hyper Text Transfer Protocol

`http://www.asahi.com/sports/`

mailto 電子メールの送信

`mailto:umeko@gm.tsuda.ac.jp`

ftp ファイル転送

`ftp://ftp.isi.edu/in-notes/rfc-index.txt`

telnet 遠隔端末アクセス

`telnet://remote.sarukani.ac.jp`

HTML(Hyper Text Markup Language)

- ハイパーテキストを記述する表現様式
- 互いにリンクし合い共有される情報となる
- **SGML**を祖先とするマークアップ言語
 - 指定した文字列に**タグ**により付加的情報を付与
 - **WYSIWYG方式**とは違い、文書レイアウトとは独立 (**リフロー型**)
 - W3C(<http://w3c.org/>)が中心となり開発
 - 現在 HTML5 + CSS3を策定中
- 電子書籍フォーマット**EPUB3**の基盤技術を提供

HTMLの例

ドキュメントタイプ宣言

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ja" xml:lang="ja">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"
    <title>太郎のページ</title>
  </head>
  <body>
    <h1>ようこそ太郎のページ</h1>
    <ol>
      <li><a href="movie/index.html">映画、大好き</a></li>
      <li><a href="sport/index.html">スポーツでリフレッシュ</a></li>
      <li><a href="http://www.sarukani.ac.jp/">サル蟹大学</a></li>
    </ol>
  </body>
</html>
```

ヘッダ部

HTML部
ボディ部

HTTP(Hyper Text Transfer Protocol)

1996年 HTTP1.0仕様 (RFC1945)

Tim Berners-LeeがURLとともに考案

- WebブラウザとWebサーバ間の通信規約
- 標準ポート : 80 (SSL/TSL使用時: 443)
- 主なアプリケーション
 - Webクライアント
 - Internet Explorer, Firefox, Safari, Google Chrome
 - Webサーバ
 - Apache, IIS

HTTPの役割

- Webサーバ側
 - URLで指定されたファイルをクライアントに送信
 - Webアプリケーションのバックヤードとの通信
- Webクライアント側
 - 送られたデータがHTMLファイルのときは、それを解釈して、整形表示する
 - 目的ファイルをURLで指定して、Webサーバに要求
 - ハイパーリンクに関連付けされた部分をクリックすると、サーバへ指定されたURLによる情報取得のための要求を送信

HTTPの概要

- 要求メッセージ（クライアントからサーバへ送信）
 - リクエストヘッダ
 - 空行（\r\n）
 - リクエストボディ
- 応答メッセージ（サーバからクライアントへ送信）
 - レスポンスヘッダ
 - 空行（\r\n）
 - レスポンスボディ

HTTPの代表的メソッド

GET

指定したURLリソースを取得する

HEAD

指定したURL取得の結果応答のヘッダのみ取得

POST

サーバに、要求ボディに載せてデータを送る

HTMLで form="POST"で指定したフォーム入力を送るとき

PUT

サーバに対し、（要求ボディにデータを載せて）
ファイルをアップロードする

HTTP要求と応答

要求

メソッド パス名 HTTP/version

応答

HTTP/version ステータス番号 応答メッセージ

補足メッセージとして、OK や Not Found など
status番号を説明する

HTTP応答Statusとメッセージ(1)

分類	番号	メッセージ	説明
情報	100	Continue	処理を継続。続きのリクエストを送信して。
	101	Switching Protocols	指定したプロトコルに変更し再要求。
成功	200	OK	成功。
	201	Created	新しいコンテンツが作成された。
	202	Accepted	要求は受理されたが、処理は完了してない。
	203	Non-Authoritative Information	サーバーが返したものと異なるが、処理は成功。
	204	No Content	コンテンツはないが、処理は成功。
	205	Reset Content	現在のコンテンツ（画面）を破棄。
	206	Partial Content	コンテンツを一部のみ返却。
	207	Partial Content	コンテンツを一部のみ返却。
転送	300	Multiple Choices	入手方法について複数の選択肢がある。
	301	Moved Permanently	別の場所に移動した。
	302	Found	別の場所に見つかった。そちらを参照。
	303	See Other	Location ヘッダで指定された他の場所を見て。
	304	Not Modified	更新されてない。
	305	Use Proxy	Location ヘッダで指定したプロキシを使用。
	306	(Unused)	未使用。
	307	Temporary Redirect	別の場所に一時的に移動。

HTTP応答Statusとメッセージ(2)

分類	番号	メッセージ	説明
Client Error	400		要求が不正。
	401	Unauthorized	認証されてない。
	402	Payment Required	支払いが必要。
	403	Forbidden	アクセスが認められてない。
	404	Not Found	見つからない。
	405	Method Not Allowed	指定したメソッドはサポートされていない。
	406	Not Acceptable	許可されてない。
	407	Proxy Authentication Required	プロキシ認証が必要。
	408	Request Timeout	リクエストがタイムアウト。
	409	Conflict	リクエストがコンフリクト(衝突・矛盾)。
	410	Gone	要求されたコンテンツは無くなった。
	411	Length Required	Content-Length ヘッダを付加して要求。
	412	Precondition Failed	If-... ヘッダで指定された条件に合致しない。
	413	Request Entity Too Large	要求されたエンティティが大きすぎる。
	414	Request-URI Too Long	要求された URI が長すぎる。
	415	Unsupported Media Type	サポートされていないメディアタイプ。
	416	Requested Range Not Satisfiable	要求されたレンジが不正。
	417	Expectation Failed	Expect ヘッダで指定された拡張要求は失敗。
Server Error	500	Internal Server Error	サーバーで予期しないエラーが発生。
	501	Not Implemented	実装されてない。
	502	Bad Gateway	ゲートウェイが不正。
	503	Service Unavailable	サービスは利用可能でない。
	504	Gateway Timeout	ゲートウェイがタイムアウト。
	505	HTTP Version Not Supported	このHTTPバージョンはサポートされてない。

仮想ホスト

HTTP/1.1 で仮想ホストをサポート

1 台のサーバーで複数のWebサイトをサポートすることが可能

HTTP/1.1 クライアント

Hostヘッダでホスト名を送信せねばならない

```
GET /index.html HTTP/1.1  
Host: www.google.com
```

サーバーは、仮想ホストに対応したコンテンツを応答する

HTTP要求メッセージの例

超簡単版

GET /index.html HTTP/1.1

リクエスト行

Host: www.google.co.jp

リクエストヘッダ

<空行>

<空行>

<空行>=\r\n

ブラウザのHTTP要求メッセージ例

本当はもう少し沢山送っている(^^;

GET /index.html HTTP/1.1 リクエスト行

Accept: image/gif, image/jpeg, */* ヘッダ

Accept-Language: ja

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/6.0

Host: www.google.com

Connection: Keep-Alive

<空行>

<メッセージbody (POSTのとき)>

HTTP応答メッセージの例(1)

Webページ取得に成功

ステータスライン

HTTP/1.1 200 OK

Cache-control: private

Content-Type: text/html

Server: GWS/2.1

Content-length: 2232

Date: Sun, 13 Jun 2004 07:45:45 GMT

応答ヘッダ

応答ボディ

<html>

<head>

<meta http-equiv="content-type" content="text/html;
charset =Shift_JIS">

<title>Google</title>

. . . .

HTTP応答メッセージの例(2)

Webページ取得に失敗

ステータスライン

HTTP/1.1 404 Not Found

Content-Type: text/html

Server: GWS/2.1

Content-length: 1219

Date: Sun, 13 Jun 2004 08:10:01 GMT

応答ヘッダ

<html>

<head>

<title>404 Not Found</title>

...

応答ボディ

telnetでHTTPを確かめる

```
% telnet www.google.com 80
```

```
Trying 74.125.153.103...
Connected to www.l.google.com.
Escape character is '^['.
```

```
GET /index.html HTTP/1.1
```

```
Host: www.google.com
```

```
HTTP/1.1 405 Method Not Allowed
Content-Type: text/html; charset=UTF-8
Content-Length: 11826
Date: Thu, 16 Jun 2011 04:00:40 GMT
Server: GFE/2.0
```

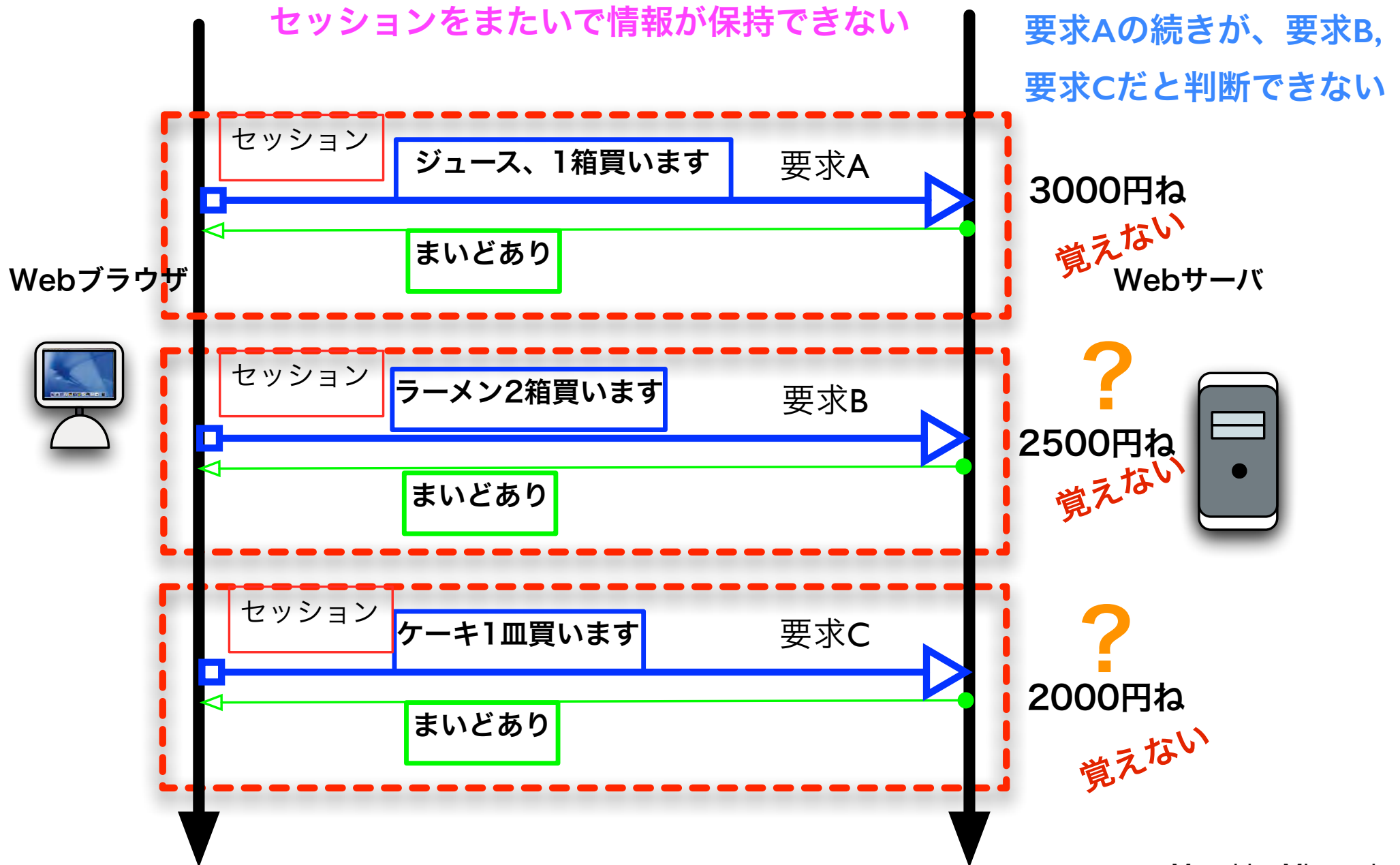
```
<!DOCTYPE html>
<html lang=en>
  <meta charset=utf-8>
  <title>Error 405 (Method Not Allowed)!!1</title>
  <style>...
```

```
.....
```

HTTPの特徴と問題点

- 非常に単純
 - HTTP要求とその応答が1つのセッション
- ステートレス (stateless)
 - 次の通信は前のセッション結果と何ら関係を持たない
 - 処理結果は残さずに、その都度廃棄される
- トランザクション処理では工夫が必要
 - 状態を維持できないために、関連する複数の処理を一つの処理単位としてまとめることができず工夫が必要
 - ユーザ認証をした上でのページ移動
 - 複数選んだ商品の購入決算

ステートレスなHTTP通信



http daemonのログ

[/var/log/apache/info/access_log](#)

from	date	method	resource	version	result	size
172.19.0.xxx - -	[21/Jun/2013:20:58:24 +0900]	"GET	/center/keiji.html	HTTP/1.1"	200	3227
172.19.0.yyy - -	[21/Jun/2013:20:58:25 +0900]	"GET	/center/green.gif	HTTP/1.1"	304	-
172.19.0.zzz - -	[21/Jun/2013:20:58:24 +0900]	"GET	/center-cgi/center-wsroomtoday.cgi	HTTP/1.1"	200	13009
133.99.177.aaa - -	[21/Jun/2013:21:01:24 +0900]	"GET	/center-cgi/center-wsroomtoday.cgi	HTTP/1.1"	200	13009
133.99.177.bbb - -	[21/Jun/2013:21:01:27 +0900]	"GET	/center/green.gif	HTTP/1.1"	304	-
133.99.177.ccc - -	[21/Jun/2013:21:01:27 +0900]	"GET	/center/keiji.html	HTTP/1.1"	200	3227
mdddd.tsuda.ac.jp - -	[21/Jun/2013:21:06:00 +0900]	"GET	/	HTTP/1.1"	200	7437
172.19.0.ffff - -	[21/Jun/2013:21:08:24 +0900]	"GET	/center/keiji.html	HTTP/1.1"	200	3227
172.19.0.ggg - -	[21/Jun/2013:21:08:25 +0900]	"GET	/center/green.gif	HTTP/1.1"	304	-
172.19.0.hhh - -	[21/Jun/2013:21:08:24 +0900]	"GET	/center-cgi/center-wsroomtoday.cgi	HTTP/1.1"	200	13009
172.19.0.ggg - -	[21/Jun/2013:21:08:24 +0900]	"GET	/center-cgi/center-wsroomtoday.cgi	HTTP/1.1"	200	13009
vpn.tsuda.ac.jp - -	[21/Jun/2013:21:10:51 +0900]	"GET	/	HTTP/1.1"	200	7437
vpn.tsuda.ac.jp - -	[21/Jun/2013:21:11:04 +0900]	"GET	/HOME/lecture/	HTTP/1.1"	200	10563
ex.tsuda.ac.jp - -	[21/Jun/2013:21:11:05 +0900]	"GET	/library/	HTTP/1.0"	200	636
vpn.tsuda.ac.jp - -	[21/Jun/2013:21:11:05 +0900]	"GET	/~mizutani/	HTTP/1.1"	200	6210
vpn.tsuda.ac.jp - -	[21/Jun/2013:21:11:06 +0900]	"GET	/~mizutani/js/update.js	HTTP/1.1"	200	21
vpn.tsuda.ac.jp - -	[21/Jun/2013:21:11:06 +0900]	"GET	/~mizutani/basic.css	HTTP/1.1"	200	1208
vpn.tsuda.ac.jp - -	[21/Jun/2013:21:11:06 +0900]	"GET	/~mizutani/icons/green.gif	HTTP/1.1"	200	57
vpn.tsuda.ac.jp - -	[21/Jun/2013:21:11:06 +0900]	"GET	/~mizutani/icons/blue.gif	HTTP/1.1"	200	

クッキー(cookie) の利用

- トランザクション処理を可能
 - クッキーを使って状態を維持
- クッキーの仕組み (RFC2965/6265)
- サーバからクッキーをブラウザに渡す
 - HTTPヘッダに埋め込むかJavascriptで利用
- ユーザ情報をクッキーに書き込む
- 同じURL（やサーバ）に接続するときはクッキー情報をサーバに送信
- WebクライアントとWebサーバ間でお互いを認識し、継続的なデータ交換を可能にする

Cookieファイルの場所

Safari MacOS

~/Library/Cookies/Cookies.plist Safari5.xまで

~/Library/Cookies/Cookies.binarycookies Safari6.xからバイナリになり

Internet Explore 8

human-Unreadableに！！

\\Users\[ユーザー]\\AppData\\Roaming\\Microsoft\\Windows\\Cookies

Chrome

Windows Vista/7

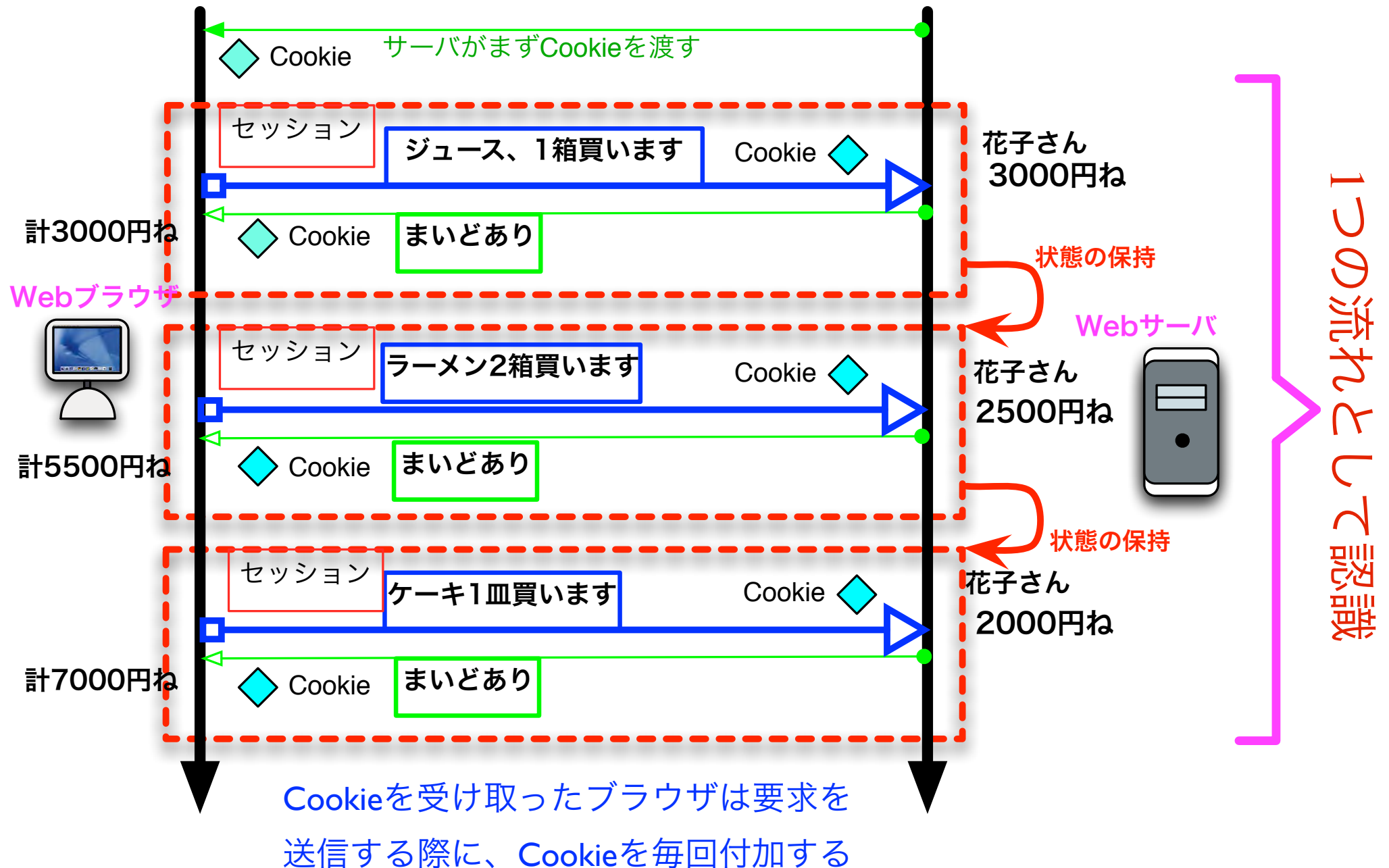
C:\\Users\[ユーザー]\\AppData\\Local\\Google\\Chrome\\User Data

MacOS (ver.10.8.x)

~/Library/Application\\ Support/Google/Chrome/Default/Cookies

バイナリhuman-Unreadableに！！

Cookieを使ったHTTP通信



Cookieを使う

1) Webサーバは応答ヘッダで指定したCookieをブラウザに渡す (Set-Cookie)

```
Set-Cookie: Customer="Taro_Jirou"; domain=happy-shopping-town.com;  
path=/shopping/; expires="25 Nov 2015 08:36:20 GMT"; secure
```

2) ブラウザが同じWebサーバと通信する時に、Cookieをサーバに送信 (Cookie)

```
Cookie: Customer="Taro_Jirou"; PartItem="Lemon_0987",Shipping="Post";
```

3) サーバがCookieを受け取ると、サーバはDBに問い合わせでユーザを特定、ブラウザに渡す情報をカスタマイズして送信

以降は、2,3（または1,2,3）を繰り返して通信する

Set-Cookie 構文 http://www.studyinghttp.net/rfc_ja/rfc2965

NAME=値; expires=値; domain=値; path=値; secure

パラメータ	意味
NAME=値	必須 。好きな名前に好きな値を指定。セミコロン(;)、カンマ(,)、空白文字()や日本語を使うときは何らかの形式に符号化 (UTF-8など) する。
expires=値	クライアント側に記録される Cookie の有効期限を Thu, 1-Jan-2030 00:00:00 GMT (Wdy, DD-Month-YYYY HH:MM:SS GMT) のような形式で指定。タイムゾーンは必ず GMT で指定。省略するとブラウザを終了させるまでが有効期限となる。 過去の値を指定すると Cookie が削除される。
domain=値	Cookie を発行する Webサーバーの名前を指定。省略すると、現在の閲覧しているWebサーバー名になる。サーバーのドメインと無関係な文字列は指定できない
path=値	指定したパス名にマッチするページを参照したときに、ブラウザは保存しているCookie情報をサーバーに送信。path=/taro と指定すると、/taro にマッチするすべてのページに対してCookie情報が送られます。省略時はCookieを設定したページのパス名部になる。
secure	これを記述した時、サーバーとの接続がセキュアである時のみ、Cookie情報が送信。

Cookieの条件

- 有効期限（expires属性）内のCookieは保存
- Cookieの数は最大300個まで
- 1つのCookieは最大4KBまで
- 1つのサーバにつき、Cookieは最大20個まで

HTML を用いて Cookie の値を記録させることもできる

```
<meta http-equiv="Set-Cookie" content="～">
```

～の部分に `NAME=値; expires=値; domain=値; path=値; secure`

Cookieを表示する (1)

Internet Explorer

テキストファイルなのでエディタで開くだけ

Safari

(a) [環境設定]/[詳細]/メニューバーに”開発”メニューを表示



[開発]Webインスペクタを表示

[ストレージボタン]

該当ページのクッキーを表示

(b) Safari Cookieを使う(MacOS) <http://sweetpproductions.com/safaricookies/>



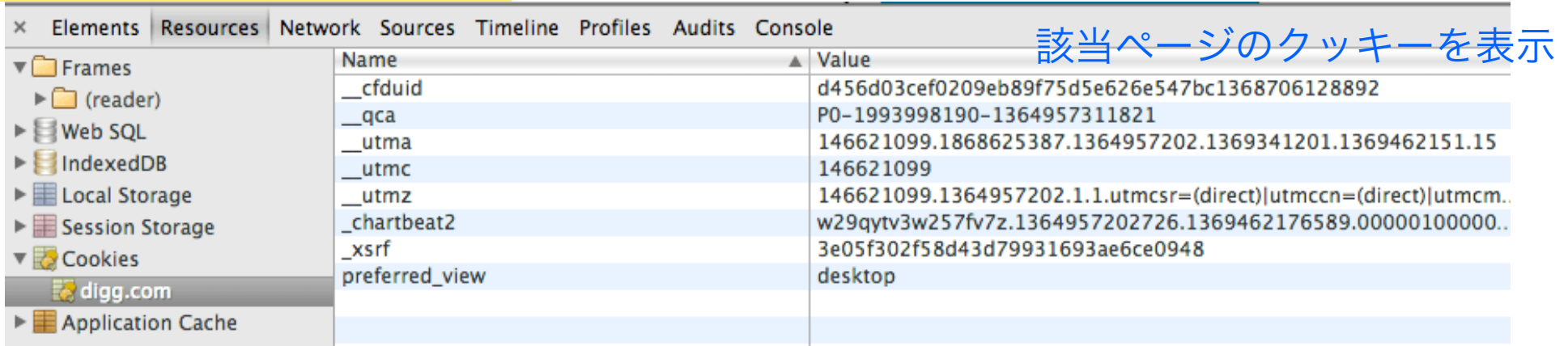
クッキーを管理できて便利

Cookieを表示する (2)

Google Chrome

[表示]/[開発/管理]/デベロッパーツール

該当ページのクッキーを表示

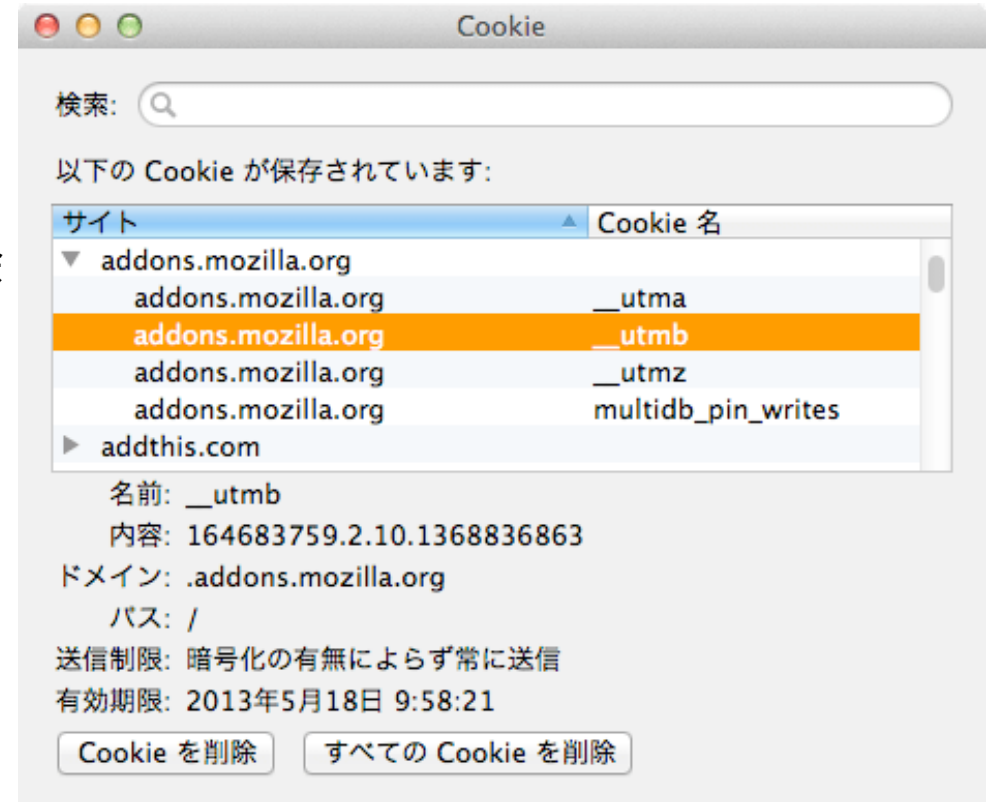


Name	Value
__cfduid	d456d03cef0209eb89f75d5e626e547bc1368706128892
__qca	P0-1993998190-1364957311821
__utma	146621099.1868625387.1364957202.1369341201.1369462151.15
__utmc	146621099
__utmz	146621099.1364957202.1.1.utmcsr=(direct) utmccn=(direct) utmcm..
_chartbeat2	w29qyvtv3w257fv7z.1364957202726.1369462176589.00000100000..
_xsrf	3e05f302f58d43d79931693ae6ce0948
preferred_view	desktop

Firefox

(a) [環境設定]/[プライバシー]Cookieを個別に削除

(b) アドオン Firebug を使って編集



検索:

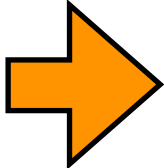
以下の Cookie が保存されています:

サイト	Cookie 名
▼ addons.mozilla.org	
addons.mozilla.org	__utma
addons.mozilla.org	__utmb
addons.mozilla.org	__utmz
addons.mozilla.org	multidb_pin_writes
▶ addthis.com	

名前: __utmb
内容: 164683759.2.10.1368836863
ドメイン: .addons.mozilla.org
パス: /
送信制限: 暗号化の有無によらず常に送信
有効期限: 2013年5月18日 9:58:21

Cookieの課題

- セッション・ハイジャック
 - 第三者にセッションIDを読み取られる
- トラッキング・クッキー
 - ユーザのアクセス履歴の追跡
 - Web広告業者、マーケティング
 - 第三者への情報開示

 **Cookieの信頼性=Cookieの管理者の信頼**

サイト内容が信頼できないとき
はCookieも信頼すべきでない