# 情報システム技術論'00 — ＃1

久野　靖*

2000.5.25

## はじめに

こんにちは、久野です。今週から「情報システム技術論」を開始します。この科目は「現在の情報システムにおいてどのような技術が使われているのか」をテーマとして、前半3週を講義形式、後半を論文紹介形式で実施する予定です。前半3回ぶんの内容は次のように予定しています。

- WWWとウェブアプリケーション
- オブジェクト指向とコンポーネント技術
- 並列処理と分散システム技術

この科目は「特定関連科目」として修士の人も取っていただけることになっていますが、あくまで博士課程の科目なので初歩的はことはやらないつもりです。また、できるだけ各回とも、お話だけでなく何か手を動かして実習してもらえるようにしたいと考えています。

また、後半2回で紹介する論文も早めに決めてもらって用意するようにしたいです。論文は各自の興味で選択していただいて結構ですが、

- 上記のいずれかの内容に関係がある
- 英文の論文

ということでお願いします。詳しくは後で相談しましょう。

## 1　今回のテーマ→WWWとWeb application

□ WWW --- 現在の情報システムにおいて避けられない存在

□ 一見「カンタン」だが非常に多様な技術の集合体

□ 単なる「技術」だけでなくその「背景」「思想」が重要

□ 役に立つ、楽しい

## 2　WWWの歴史

### 2.1　はじまり

□ 1990 --- CERNでTim Berners-Leeが

□ 「ネットワーク上でドキュメントを共有する仕組み」

□ ネット上のハイパーテキスト → 現在のWWWと同じ(当然だが)

□ ブラウザ → NextStep上(テキストのみ)

□ サーバ → CERN HTTPD

### 2.2　躍進

□ 1993 --- NCSA(Univ.　Illinois)で学生たちが

□ WWWを知って「面白い」と思った

□ グラフィクスを入れたら絶対ウケる → Mosaic（最初のグラフィカルブラウザ)

□ Internetな人(当時はUnixユーザばかり)の間で爆発的に普及

□ Mosaicも当初はUnix版 → のちMac、Windows

### 2.3　日本では…

□ 最初は日本語なんか表示できなかった

□ NTT高田ほか…Mosaicを「国際化」Mosaic-L10N → 日本語表示

- I18N --- Internationalization --- 複数言語を同時に
- L10N --- Localization --- 英語+当地の言語

### 2.4　Mosaicのその後

□ Mosaicは爆発的に普及→イリノイ大はこれを管理しようとした→作者の学生たちは反発

*筑波大学大学院経営システム科学専攻

□ SGI（Sillicon Graphics）の会長 Jim Cleark →「これからはコレだ」→学生たちを誘って Netscape Communications 社を創立

□ 打倒 Mosaic →Mozilla(Netscape Navigator) の開発 → たちまち No.1 ブラウザに→ Netscape 社はサクセスストーリの代表に

□ さまざまな「新機能」を搭載→ HTML も独自拡張→問題を残した

### 2.5 インターネットブーム

□ Mozilla は Unix、Mac、Windows 版をほぼ同時にリリース → PC ユーザが「手軽に」WWW を使えるようになった

□ Mozilla はほぼ最初から日本語対応 (しかも文字コード自動判別) →日本での普及も問題なし

□ WWW の「Point and click」→難しいことを知らなくてもネットから情報が取れる→インターネットの爆発的な普及の原動力

### 2.6 ブラウザ戦争

□ Microsoft は当初、インターネットなんて駄目で MSN(Microsoft Network) がよい、という戦略

□ WWW の爆発的な普及を目にして急拠戦略を転換

□ Netscape 社を追い落すには→ Spygrass 社が Mosaic を元に開発していたブラウザを買収して Internet Explorer と名付け、Windows に無償で添付

□ さらに「Netscape の添付をやめないと不利になるぞ」と圧力を掛ける

□ Netscape 社はブラウザの無償化、オープンソフト化などで対抗したがジリ貧となり、AOL に買収される

- まもなく出る Netscape 6 はそれなりに期待されているが…

### 2.7 そして現在は…

□ ブラウザはもはや「メシの種」ではなく、WWW は「当り前のもの」に→「その上で何をやるか」が重要に

□ ポータルサイト囲い込み競争

□ 赤字が出てもとにかく WWW ビジネス参入競争

□ これからは…技術的には…

- ビジネスで何が起こるかはまったく分からない
- 技術的には…まだ技術の話をしてないがな!

## 3 HTML — WWW の基盤技術

### 3.1 マークアップと WYSIWYG

□ 計算機の用途の 1 つ→整形された文書の作成

□ 「古くからある」やり方→マークアップ

- 文書に「印」「コマンド」(マークアップ) を埋め込む
- runoff → roff → troff、nroff、groff

□ 「もう 1 つの」やり方→ WYSIWYG

- 整形ずみ画面 (に近いもの) を表示しつつ編集
- Alto のエディタ→ MacWrite、各種 DTP

□ それぞれの得失は…

### 3.2 SGML

□ マークアップのついたドキュメントを処理→計算機で多く扱われるように→そのたびに個別に処理プログラムを作るのでは大変

□ 「マークアップ言語」の「構文」は標準化、どのような「命令」があるかは用途ごとに指定→SGML(Standard Generalized Markup Language)

□ SGML → ISO 規格/JIS 規格になっている→特定の言語というより「さまざまなマークアップ言語を作り出すための枠組み」

□ DTD(Document Type Definition) → SGML の枠組で特定の言語を定義した記述。ないしその書き方。例: HTML DTD

### 3.3 SGML の書き方

□ 基本的に「<…>」(タグ) でマークアップを行う

□ 「<a>〜</a>」のように対になったタグで範囲 (要素/element) を表す

- 範囲は「完全な入れ子」つまり互い違いは駄目

□ 「&名前;」→文字エントリ…キーボードで直接入れられない文字をドキュメントに入れるための記法

- さらに、「<」→<、「>」→>「&」→&

### 3.4 HTMLの歴史

□ WWWでは「ブラウザが環境に合わせて整形」だから当然マークアップ

□ Timが作ったときに「マークアップだからSGMLふうに」

□ しかし最初(0.9、1.0)はSGMLふうなだけでSGMLに準拠ではなかった

□ HTML 2.0 → IETFによる標準化、SGMLに準拠した版

□ せっかく標準ができたのに「ブラウザ戦争」で新たな「拡張」が…

- 機能的に重要なものも。例: 表、フレーム、クライアント側イメージマップ

### 3.5 W3C(WWW Consortium)

□ CERNはWWWの発現の地だったが、もともと物理の研究所→ WWWを背負っていくのは無理

□ 任意団体としてWWWコンソーシアムを設立し、標準の取りまとめや研究を行って行くことに

- W3Cをホストしているのは MIT(米国)、INRIA(フランス)、慶応大学 (日本)。興味があれば http://www.w3.org/ を参照のこと。

### 3.6 HTML 3.2

□ HTML 2.0の後の改良を取り込む動き→W3CはHTML 3.0を提案したが、各社の意見が統一されずうまく行かなかった

□ 当面「最低限の共通部分」という形で仕切り直しまとめたものがHTML 3.2 → 当面の妥協という側面が大きい

- 表機能、さまざまな飾り機能、アプレット、スクリプト、クライアント側イメージマップなど。

### 3.7 HTML 4.0

□ W3CがHTML 3.0で目指した包括的なHTMLの整理→HTML 4.0として成立

□ 論理構造と表現指定の分割

□ 表現指定はスタイルシートに移す

□ 3種類のDTD --- Strict、Transitional、Frameset

□ 正式な4.0はStrict DTD →表現に関する機能は「非推奨」に

□ アクセシビリティに配慮した設計

□ その他機能増強→フレームセット、オブジェクト埋め込み、等

## 4 HTML入門

### 4.1 簡単なHTMLドキュメント

```
<!DOCTYPE HTML PUBLIC
 "-//W3C//DTD HTML 4.0//EN">
<html lang="ja">
<head>
<title>A Sample</title>
</head>
<body>
<h1>This is a sample</h1>
<p>こんにちは…</p>
</body>
</html>
```

□ DOCTYPE宣言 --- SGMLの記述で、この文書がHTML 4.0であることを宣言している(省略するとHTML 2.0と仮定されるが省略しない方がよい)

□ <html>…</html> --- HTML記述の範囲全体を囲む

□ lang="ja" --- 言語が日本語であることを示す属性(タグ内の指定)

□ <head>…</head> --- ヘッダ要素(文書全体に関する記述)の範囲

□ <title>…</title> --- 表題

□ <body>…</body> --- 本体

□ <h1>…</h1> --- 見出し

□ … --- 段落

### 4.2 文書本体の書き方

□ とにかくすべてのものは要素に入れる

□ 段落等は適宜詰め合わせされる(空白は無視)

□ 見出し --- h1、h2、…、h6まで(レベルはHTML 4.0では飛んでもよいがISO HTMLでは連続)

□ 段落 --- …

□ 箇条書き --- <ul>…</ul>、<ol>…</ol>、<dl>…</dl> --- 番号なし箇条書、番号つき箇条書、キーワード型箇条書

- ulとolの1項目は<li>…<li>
- dlの1項目は<dt>…<dt><dd>…<dd>

```
<!DOCTYPE HTML PUBLIC
 "-//W3C//DTD HTML 4.0//EN">
<html lang="ja">
<head>
<title>A Sample</title>
</head>
<body>
<h1>Lists</h1>
<ul>
 <li>これはペンです。</li>
 <li>これは本です。</li>
 <li>ごきげんいかがですか?</li>
</ul>
</body>
</html>
```

### 4.3 表

□ <table border="2">…</table> --- 表 (枠つき)

□ <tr>…</tr> --- 1 つの列

□ <th>…</th>、<td>…</td> --- 1 つのセル (見出し、通常)

□ colspan="数"、rowspan="数" --- セルを「ぶち抜き」にする

```
<!DOCTYPE HTML PUBLIC
 "-//W3C//DTD HTML 4.0//EN">
<html lang="ja">
<head>
<title>A Sample</title>
</head>
<body>
<h1>Table</h1>
<table border="2">
 <tr>
  <th></th><th>筑波</th><th>大塚</th></tr>
 <tr>
  <th>娯楽</th><td>少</td><td>多</td></tr>
 <tr>
  <th>緑</th><td colspan="2">多</td></tr>
</table>
</body>
</html>
```

### 4.4 実習 — HTML を書いてみる

□ emacs を立ち上げファイル名「なんとか.html」を指定

□ 例題の内容を打ち込む

□ Netscape Navigator でファイルを開いてみる

□ 内容を追加変更したら reload

□ 「htmllint ファイル名」でチェックを受けてみる

## 5 スタイルシート機能

### 5.1 スタイルシートとは?

□ 一般的に「論理構造に対して表現を指定する」(例: 大見出しは 24 ポのゴシックで中央揃え、等々) 機能

- Microsoft Word にもあるが使ってます? (すごく使いにくい…)

□ 論理構造と物理表現を分離→それぞれの論理構造に対して物理表現を指定する機能がスタイルシート

- いちいち「ここで大きく」「ここは何色」と細かく言わないで済む

### 5.2 CSS(直列スタイルシート)

□ スタイルを指定するのは誰?

- 文書の作成者→作者だから当然
- ブラウザ→表示するんだから当然
- 読み手→好みに応じてスタイルを指定したいから当然

□ スタイル指定は複数あるのが当然→それらを「混ぜる」→実際には「順にならべて優先度の高いものから適用」→「直列順」→直列スタイルシート

□ W3C が推奨規格としてまとめている

- 他のライバルもあったが敗れている (なくなったわけではないが…)

### 5.3 HTML から CSS を指定する方法

□ 要素タグの中で「style="スタイル指定本体…"」という属性を使う

- それぞれの箇所に指定しなければならない

□ ヘッダ (<head>…</head>の内側) に次の形で入れる

```
<style type="text/css"><!--
スタイル指定
…
--></style>
```

- <!-- … --> は HTML(SGML) のコメント←念のため
- そのファイルについてスタイルをまとめて指定できる

□ スタイル指定を別のファイルに書き、そのありかをヘッダ内で次の形で指定する

```
<link rel="stylesheet" href="ファイルの URI">
```

- 複数のファイルに対してスタイル指定を 1 つにまとめられる

### 5.4 CSS によるスタイル指定

□ この要素にはこのような表現→「この要素」:セレクタで指定、「この表現」:スタイル記述本体で指定

```
h1 { color: blue; font-size: xx-large }
```

□ すべての要素に均等でなくでなく「こういうもの」「これ」と言いたい場合はクラス指定と ID 指定がある。

```
h1.red { color: red } ← <h1 class="red">...</h1>
h1#001 { color: red } ← <h1 id="001">...</h1>
```

□ どのようなスタイル記述があるか…別資料参照

### 5.5 簡単な例

```
<!DOCTYPE HTML PUBLIC
 "-//W3C//DTD HTML 4.0//EN">
<html lang="ja">
<head>
<title>A Sample</title>
<style type="text/css"><!--
h1 { color: blue;
     border-style: double;
     border-width: 8px;
     border-color: green;
     text-align: center }
p { text-decoration: underline;
    text-indent: 30%; }
--></style>
</head>
<body>
<h1>This is a sample</h1>
<p>こんにちは…</p>
</body>
</html>
```

### 5.6 特定の意味を持たない HTML 要素

□ スタイルを指定するときは論理的な要素と関係ない範囲を指定したいこともある→単なる「範囲」が欲しい

- … --- division (複数の段落から成る範囲)
- … --- 段落内部の範囲

```
<!DOCTYPE HTML PUBLIC
 "-//W3C//DTD HTML 4.0//EN">
<html lang="ja">
<head>
<title>A Sample</title>
<style type="text/css"><!--
div.fr { border-style: solid;
         border-width: 4px;
         border-color: blue;
         width: 100%;
         text-align: center }
span.ll { color: red }
--></style>
</head>
<body>
<div class="fr">
<h1>This is a sample</h1>
<p>このように<span class="ll">
一部</span>だけ協調も…</p>
</div>
</body>
</html>
```

### 5.7 実習 — CSS による指定をつけてみる

□ これまでに書いた HTML ファイルに CSS 指定を追加してみよう

□ とくに「position: absolute」は面白いのでやってみよう (?)

## 6 HTTP と Web サーバ

### 6.1 WWW は何からできているか?

□ HTML+CSS+なんとか→コンテンツの記述言語

□ ブラウザ→見るためのツール

□ サーバ→コンテンツをネットワーク経由で提供する

□ HTTP(HyperText Transfer Protocol) →ブラウザとサーバがやりとりするプロトコル

□ URI(Uniform Resource Identifier) → WWW コンテンツの「番地」

- 形式: 「スキーム:アドレス」
- HTTP URI: 「http://サーバ/所在」
- 「サーバ」は IP アドレスでもホスト名でもドメインアドレスでもよい
- 「所在」はディレクトリ+ファイル (ディレクトリだけでもよい)

### 6.2 具体的なしくみ

□ ブラウザで URI を打ち込む (または HTML 中のリンクを選択)

- リンク→「<a href="$uri$">テキスト</a>」

- いずれにせよ URI に指定されているサーバに接続
- そのサーバにディレクトリ+ファイルを渡して内容を返送してもらう

## 6.3 HTTP

□ わりと単純なプロトコル。0.9 → 1.0 → 1.1 と変化

□ サーバに接続→コマンドを送る (+付加情報を表すヘッダ --- 要求ヘッダ --- を送る) →サーバからヘッダ群 --- 応答ヘッダ --- が返される→ (あれば) データが送られてくる→接続継続なら次のコマンドへ進む

- 前はいちいち接続を切断していた→保持したままにした方がオーバヘッドが小さいので改良された

```
% telnet smm 80              ← telnet でサーバに接続
Trying 192.168.128.1...
Connected to smm.
Escape character is '^]'.
GET /index.html HTTP/1.1    ←要求 (コマンド行)
Host: smm                   ←ヘッダ開始
Connection: close
                            ←ヘッダ終り (要求終り)
HTTP/1.1 200 OK             ←応答 (状態行)
Date: Wed, 05 Apr 2000 14:59:40 GMT  ←ヘッダ開始
Server: Apache/1.3.6 (Unix)
Last-Modified: Sat, 25 Mar 2000 06:32:48 GMT
ETag: "945c4-6ae-38dc5d90"
Accept-Ranges: bytes
Content-Length: 1710
Connection: close
Content-Type: text/html
                            ←ヘッダ終り
<!DOCTYPE ...               ←データ開始
(途中略)
</html>                     ←データ終り
Connection closed by foreign host.
%
```

□ コマンドには GET、POST、HEAD などがある

- GET --- 内容を取り寄せる
- HEAD --- GET と同じだがヘッダだけ返す (更新日時を調べる等に使う)
- POST --- フォームの提出

□ 演習 --- telnet で HTTP を喋って smm から色々取り寄せてみよう

## 6.4 Web サーバ

□ 要するに、HTTP の要求に従ってコンテンツを返送すればよい

□ さまざまな Web サーバが作られて来た

- フリーのもの: CERN http → NCSA httpd → Apache (現在主流)
- 商売もの: Netscape Enterprise Server、MS IIS / Personal Web Server
- 最近のネット機器: Web サーバが内蔵されていてその画面で設定ができる

## 6.5 Apache

□ オープンソースプロジェクトの代表格、高いシェア

□ もともとは NCSA 互換で改良されたサーバを、というプロジェクトだった

□ さまざまな機能を搭載している

- それらの機能を「モジュール」という単位で取捨選択可能

□ もとは Unix 用だったが現在は Windows 他の多くの OS でも稼働する

## 6.6 コンテンツの置き場所

□ URI の「ディレクトリ+パス」→実際には「どこかのディレクトリ以下」に対応させるのが普通→「/」で表されるディレクトリを「ドキュメントルート」という

□ 全部ドキュメントルートの下に入れなくても「つぎはぎ」してもよい。たとえば/X というディレクトリは実はこちら、ということもできる (ソフトで対応をつけてるからどうにでもなる)。

□ とくに「~ユーザ名」という書き方で各ユーザに自分のホームディレクトリ以下の特定ディレクトリ (多くのサイトでは public_html、我々のところでは WWW) 以下にコンテンツを自由に置かせることができる。

## 6.7 コンテンツ種別と MIME タイプ

□ 普通の環境だとファイルの種別は名前 (拡張子) で分かる

□ WWW では URI の末尾に拡張子があるとは限らない→その方法は使えない

□ Web サーバは HTTP 応答ヘッダの一種 Content-type ヘッダでデータの種別を返送してくる→ブラウザはそれに応じてコンテンツを扱う

□ 種別の表記方法→ MIME タイプ

□ MIME → multipurpose internet mail extension → 電子メールでテキストに限らずさまざまな情報を送れるようにする規格→さまざまなものを取り扱う規格にも当然なっている→種別を表す方法も含まれている

□ MIME タイプの形式→「種別/副種別」例: text/plain、text/html、image/gif、image/jpeg、application/postscript、等

□ 実際にはサーバは返送するものが単なるファイルのときは「.html」なら text/html 等「常識的な」やり方で種別を判断している

### 6.8 ディレクトリインデクス

□ URI が「ディレクトリ部だけで終る」場合は? つまり最後が「/」

□ もともと、そのディレクトリにあるファイルの一覧を返すというような意味があった

□ しかし、それをカスタマイズしたいという要求があったので、そこに index.html というファイルがあればそれが返されるようにした

□ 今では index.html は「ファイルの一覧」ではなく「そのディレクトリを代表するページ (ホームページ)」を表すことになっている

- ホームページって分かります?

### 6.9 実習 — 自分のディレクトリ以下にファイルを置こう

□ 「mkdir WWW」「chmod a+rx WWW」「chmod a+x .」を実行

□ 「cd WWW」でドキュメントディレクトリに行く

□ そこにテキストファイルや HTML ファイルを置いてアクセスしてみよう

□ ファイルは「chmod a+r *」などで「だれでも読める」ように設定しないとうまく行かない

□ うまく行ったらサブディレクトリを作ってそっちもアクセスしてみる

□ また index.html なども作ってみる

## 7 CGI と動的コンテンツ

### 7.1 CGI とは

□ CGI(Common Gateway Interface) とは→ Web サーバがクライアント (ブラウザ) の要求に応じてサーバ上にあるプログラムを呼び出して実行させ、その出力をクライアントに返送するような枠組み

□ なぜ必要か…「計算機内部での処理に応じて変化するコンテンツ」(動的コンテンツ) のためにはこれが必要

- 現在ではアプレットとか色々あるけど最初はこれしかなかった

□ CGI プログラム (以下単に「CGI」) の必須の条件 --- HTTP 応答ヘッダの一部 (通常は Content-type:ヘッダ) を返すこと

- ヘッダの終りは空行で表す (メール等と同じ)

### 7.2 ごく簡単な CGI

```
#!/bin/sh
echo 'Content-type: text/plain'
echo ''
date
```

□ これを date.cgi という名前で用意して「chmod a+rx date.cgi」で実行可能に→サーバからアクセスしてみる

□ text/plain じゃなく text/html という嘘を書いたらどうなるか?

□ ちゃんと HTML を返すようにしてみよう

□ HTML ファイルからこの CGI へのリンクを貼って動かしてみよう

```
#!/bin/sh
PATH=$PATH:/usr/local/X11R6.3/bin; export PATH
echo 'Content-type: image/gif'
echo ''
ppmpat -anticamo 200 200 | ppmtogif
```

□ こっちだとどうか? man ppmpat でマニュアルを見て ppmpat のオプションを変更してみたらどうなるか? 再読み込みするとどうなるか?

□ たとえば上記が image.cgi という名前だとして、スタイル指定「body background-image: url(image.cgi)」とするとどうなるか?

# 8 フォームと CGI

## 8.1 フォームとは

□ HTML の機能で、「記入欄」等の集まりを生成する機能

□ 記入した内容は CGI に送られ、CGI で好きなように処理

□ form 要素→開始タグの action 属性で CGI を指定する

```
<form method="post" action="…">…</form>
```

## 8.2 フォームの入力部品

□ <input type="text" name="名前" size="文字数"> --- 入力欄

□ <input type="checkbox" name="名前" [checked]> --- チェックボックス

□ <inptu type="radio" name="名　前" value="値" [checked]> --- ラジオボタン

□ <input type="submit" value="ラベル"> --- 提出ボタン

□ <input type="reset" value="ラベル"> --- リセットボタン

□ <select name="名　前">…</select> --- 選　択　メニュー (中に option 要素)

□ <option [value="値"] [selected]>…</option> --- 選択メニューの項目

□ <textarea name="名前" rows="行数" cols="文字数">…</textarea> --- テキスト入力領域

## 8.3 簡単なフォームの例

□ たとえば「2 つの数値を入力して計算する」

```
<!DOCTYPE HTML PUBLIC
 "-//W3C//DTD HTML 4.0//EN">
<html lang="ja">
<head>
<title>A Sample</title>
</head>
<body>
<h1>Sample Form</h1>
<form method="post" action="sample8.cgi">
<p>
 <input type="text" name="n1" size="5">
 <select name="op">
  <option>+</option><option>-</option>
 </select>
 <input type="text" name="n2" size="5"><br>
 <input type="submit" value="計算">
 <input type="reset" value="リセット"></p>
</form>
</body>
</html>
```

## 8.4 Perl — CGI「にも」適したスクリプト言語

□ Perl --- Larry Wall が開発したスクリプト言語

- 「スクリプト言語」とは「ちょっとした処理をさっと書く」ための言語→Perl の場合、できが良いので結構大きなものまでこれで書かれている
- 言語の構文や概念はちょっと変わっているので慣れが必要
- version 5 でオブジェクト指向機能が加わった

□ Perl は CGI で多く必要な文字列処理を得意とする

□ CGI のためのモジュールが提供されている

## 8.5 Perl の CGI モジュール

□ フォームデータの受け取り等各種機能を提供

□ 使い方は次の通り簡単

- 冒頭部分での初期設定:

  ```
  use CGI; $in = new CGI;
  ```

- フォームデータの参照 (「名前」はフォーム側で指定したもの):

  ```
  $var = $in->param('名前');
  ```

## 8.6 簡単な Perl による CGI

□ 先のフォームを処理

```
#!/usr/local/bin/perl
use CGI; $in = new CGI;
print "Content-type: text/html\n\n";
$n1 = $in->param('n1');
$n2 = $in->param('n2');
$op = $in->param('op');
if($op eq '+') { $result = $n1 + $n2; }
else           { $result = $n1 - $n2; }
print '<!DOCTYPE HTML PUBLIC', "\n";
print '  "-//W3C//DTD HTML 4.0//EN">', "\n";
print "<head><title>sample</title></head>\n";
print "<body><h1>sample</h1>\n";
print "<p>$n1 $op $n2 = $result</p>\n";
print "</body></html>\n";
```

## 8.7 ファイルへの書き込み

□ 単純に計算して返すだけなら簡単

□ 実際には入力データを保存したい

□ ファイルの所有者は誰に? いくつかの選択肢

- Web サーバプロセス (nobody) が所有
- 自分が所有して「誰にでも書ける」に

□ 1 つの CGI の複数のインスタンスが同時実行される→排他制御が必要→ファイルにロックを掛ける機構を利用する

□ 簡単な BBS プログラム

```
#!/usr/local/bin/perl
require 'jcode.pl';
use CGI;
$in = new CGI;
umask 022;
print "Content-type: text/html\n\n";
print '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN>';
print "\n", '<html lang="ja">, "\n";
print "<head><title>BBS</title></head><body>', "\n";
print '<h1>かんたん BBS</h1><hr>', "\n";
if($in->request_method() eq 'POST') {
  if(!open(DATA, '>>bbs.data')) {
    print "<p>作業ファイルが書けない。</p>\n";
    print "</body></html>\n"; exit(); }
  while(!flock(DATA, 2+4)) { # LOCK_EX+LOCK_NB
    if(++$count > 5) {
      print "<p>ロックが獲得できない。\n";
      print "</p></body></html>\n"; exit(); }
    sleep(3); }
  if(open(SEQ, 'bbs.seq')) {
    $seqno = <SEQ>; close(SEQ);
  }
  ++$seqno;
  if(open(SEQ, '>bbs.seq')) {
    print SEQ "$seqno"; close(SEQ);
  }
  seek(DATA, 0, 2); select(DATA);
  $name = $in->param('name');
  jcode::convert(*name, 'euc');
  $face = $in->param('face');
  $text = $in->param('text');
  jcode::convert(*text, 'euc');
  $text =~ s/&/&amp;/g;
  $text =~ s/</&lt;/g;
  $text =~ s/>/&gt;/g;
  $time = gmtime();
  print "<h2 class=\"name\">No.$seqno\n";
  print "$face [ $name ] $time</h2>\n";
  print "<pre class=\"text\">\n$text</pre><hr>\n";
  flock(DATA, 8); close(DATA); select(STDOUT);
}
if(open(DATA, 'bbs.data')) {
  while(<DATA>) { print; }
}
print '<h2 class="write">書き込み</h2>', "\n";
print "<form method='post' action='sample9.cgi'>\n";
print '<p><select name="face">', "\n";
print "<option selected>^_^</option>\n";
print "<option>-_-</option>\n";
print "<option>;_;</option>\n";
print "<option>?_?</option></select>\n";
print "名前:\n";
print "<input type='text' name='name' size='12'>\n";
print "<input type='submit' value=' 書き込み'><br>\n";
print '<textarea name="text" rows="10" cols="30">', "\n";
print ' めっせえじ</textarea></p>', "\n";
print '</form></body></html>', "\n";
```

## 8.8 状態の保存

□ CGI では「ある実行」と「次の実行」は全く別のもの→何かを「記憶」しておきたければかなり工夫が必要

- たとえば「顧客番号」を割り当てて買物をするたびに何を買ったかという情報を蓄積し、最後にまとめて支払処理をしたい→それらの情報はどうする? CGI は個別に「これを買う」というごとにばらばらに起動される
- 情報をファイルに書けばよい? →複数人が同時に買物をしても全く不思議ではない→「誰が」という情報は別に用意しないと混乱する

□ 隠し入力欄を利用する方法

- <input type="hidden" name="名　前" value="値"> --- フォームの画面には何も現れないが値はデータとして送信されてくる→データを記憶しておくことに相当
- 画面がフォーム提出の連続でできていないといけない→不自由

□ クッキー機能 --- Netscape 独自拡張だったが現在では広く普及

- CGI からブラウザに「クッキー」と呼ばれる情報を送出→その CGI(ないし近辺) が呼ばれる時、同じクッキー情報が提出されてくる→それによってデータを記憶

## 8.9 クッキーを使う例

□ クッキー→「ちょっとした情報」という意味。WWW でのクッキーと言うのは、Netscape 社が独自拡張の 1 つとして始めたもので、便利なので多くのブラウザで実装されている。

- サーバはブラウザへの応答中に Set-cookie:応答ヘッダを含めることで、ブラウザにクッキー情報を記憶させることができる
- ブラウザはクッキーをそれが「どのドメインのどのパスから来たか」の情報を保存しておく
- ブラウザはURIにアクセスするとき、そのURIにマッチするクッキーの情報をCookie:ヘッダに入れてサーバに送る

□ これらの機構を使えばCGIが使う情報を「ブラウザに覚えておいてもらう」ようにできるので前述の問題が簡単になる

## 8.10 クッキーの制御

□ クッキーの形:

Set-cookie: 名 前=値; domain=ド メ イ ン; path=パス; expires=日時

- 「名前=値」がクッキーの本体
- ドメインはそのクッキーがどの範囲内のサーバで有効かを示す
- パスはクッキーがそのサーバ内のどの範囲で有効かを示す
- 日時はそのクッキーがいつまで「生きている」かを示す。指定しないとそのブラウザを終了させたときになくなる。

□ クッキーの制御方法:

- 「名前」と「パス」が違うクッキーは別のものとして扱われる
- expiresが過去の日時だとクッキーは削除される
- ドメインには「.」が3個以上(.edu等では2個以上)含まれる必要←クッキーを使って「顧客の嗜好調査」(悪くいえばプライバシーをあばく行為)を行う会社がある

## 8.11 クッキーを使った例題

□ よくあるパターン: 買物サイト

- 最初に顧客登録→クッキーを発行
- 商品を選ぶ→クッキーに基づいて商品情報を追加していく
- 最後に支払い→クッキーに基づいて支払、発送

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<html lang="ja">
<head>
<title>Form Examples</title>
</head>
<body>
<h1>おかいもの</h1>
<form method="post" action="sample10a.cgi">
 <p>名前: <input type="text" name="name" size="20">
    <input type="submit" value="登録"></p>
</form>
<hr>
<form method="post" action="sample10b.cgi">
 <p>
 <input type="submit" name="goods" value="鉛筆"></p>
</form>
<form method="post" action="sample10b.cgi">
 <p>
 <input type="submit" name="goods" value="定規"></p>
</form>
<form method="post" action="sample10b.cgi">
 <p>
 <input type="submit" name="goods" value="手帳"></p>
</form>
<hr>
<form method="post" action="sample10c.cgi">
 <p><input type="submit" value="会計"></p>
</form>
</body>
</html>
```

```
#!/usr/local/bin/perl
use CGI;
$in = new CGI;
$name = $in->param('name');
if(!$name) {
  print "Content-type: text/plain\n\n";
  print "名前を入れてください\n"; exit;
}
$cookie = $in->cookie(-name=>"goods",
            -value=>"$name", -expires=>"+1h");
print "Content-type: text/html\n";
print "Set-cookie: $cookie\n\n";
print '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">';
print "\n", '<html lang="ja">', "\n";
print "<head><title>Registerd</title></head><body>\n";
print "<h1>登録完了</h1>\n";
print "<p>名前「$name」で登録しました。</p>\n";
print "<p><a href='sample10.html'>戻る</a></p>\n";
print "</body></html>\n";
```

---

```
#!/usr/local/bin/perl
use CGI;
$in = new CGI;
$goods = $in->cookie('goods');
$added = $in->param('goods');
$cookie = $ENV{'HTTP_COOKIE'};
if(!$goods) {
    print "Content-type: text/plain\n\n";
    print "最初に登録してください\n"; exit;
}
$goods = "$goods:$added";
```

```
$cookie = $in->cookie(-name=>"goods",
          -value=>"$goods", -expires=>"+1h");
@list = split(/:/, $goods);
$name = shift(@list);
$list = join(' ', @list);
print "Content-type: text/html\n";
print "Set-cookie: $cookie\n\n";
print '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
print "\n", '<html lang="ja">', "\n";
print "<head><title>Registerd</title></head><body>\n";
print "<h1>買物かごに追加</h1>\n";
print "<p>$name さんの選定品: $list</p>\n";
print "<p><a href='sample10.html'>戻る</a></p>\n";
print "<p>(生クッキー: $cookie)</p>\n";
print "</body></html>\n";
```

# 9 JavaScript

## 9.1 JavaScript とは

□ HTML の中に埋め込むことができるスクリプト言語

- もともとは Netscape 社が開発し LiveScript という名前だったが Java ブームのとき Sun から名前を買って JavaScript になった

□ できること→ HTML の一部をスクリプトで生成、フォームなどの内容をスクリプトで読み書きして処理、ステータスバーにメッセージを表示する (定期的に実行も可能) など

□ ブラウザ内で実行するので、サーバ内の情報は参照できないが、その代わり「ユーザと短い周期でのやりとり」が可能

## 9.2 JavaScript の特徴

□ プロトタイプ方式のオブジェクト指向言語

□ データは「値」か「配列」かその他の「オブジェクト」

□ オブジェクトに対してはメソッドが呼べる

- 「このページにあるすべてのフォームオブジェクトの配列」「このフォームにあるすべての部品の配列」などが利用可能→これらのメソッドを呼びながら動作できる

□ どうやって起動? →ロード時、提出時、…、「ボタン」部品が押された時

## 9.3 JavaScript による簡単な例題

□ 例: 最小公倍数の計算

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<html lang="ja">
<head>
<title>JavaScript Examples</title>
<script type="text/javascript">
function keisan() {
  fieldx = document.forms[0].elements[0];
  fieldy = document.forms[0].elements[1];
  fieldz = document.forms[0].elements[2];
  x = parseInt(fieldx.value);
  y = parseInt(fieldy.value);
  while(x != y) {
    if(x > y) {
      x = x - y;
    } else {
      y = y - x;
    }
    fieldz.value = x;
  }
}
</script>
</head>
<body>
<h1>最大公約数</h1>
<form>
 <p>
  X: <input type="text" value="0" size="8"><br>
  Y: <input type="text" value="0" size="8"><br>
  最大公約数: <input type="text" value="0" size="8"><br>
  <input type="button"
         value="計算!" onclick="keisan()">
 </p>
</form>
</body>
</html>
```

# 10 まとめ

□ WWW の由来と歴史

□ WWW の基本技術 --- HTML、CSS、サーバ、ブラウザ、HTTP

□ 動的コンテンツ

- CGI --- サーバ内で動作するプログラム→多様な処理、結果を HTML 等の形で生成して表示→ユーザとやりとり
- JavaScript --- ブラウザ内で動作する→サーバ内の情報には頼ることができないが対話性は高い

□ 基本的には、これらの技術を組み合わせて任意の Web アプリケーションを構築して行くことができる

□ もちろん裸からやらなくてもよいようにツールサポートもいろいろある→久野としては「原理から理解しておいた方が結局早道」だと思うので基本的なところを説明させて頂いた

## 11 紹介論文の選定

□ 図書室で …「ACM Transaction on Office Information Systems」「IEEE Software」その他こういう話題が載っている雑誌がいろいろあるので、まずは眺めてみて欲しい

□ 次回までに読んでみたそうな論文の候補を 2〜3 くらい選択して来ること。4 回目と 5 回目が論文紹介なので次回に半分くらいの人については決定する必要がある。