情報システム技術論'08 — ＃4

久野　靖*

2008.5.9

1　Webアプリケーション

1.1　Webアプリケーションの歴史

□ Webページ→最初はHTMLによる「動かない」もの

- (アプレットによるJavaブームが来た話は前回紹介)
- 最初は「せめてどんな文書があるか検索したい」という要求に対処するためにisindex機能というのが作られた
- ブラウザの窓の隅に「検索入力蘭」が表れてそこに打ち込んだものがサーバに送られて処理できる

□ フォームを用いたWebアプリケーション(サーバ側プログラミング)

- HTML2.0でフォーム機能が標準化される→さまざまな形の入力欄が作れるようになった --- 入力欄、パスワード入力欄、チェックボックス、ラジオボタン、選択メニュー、テキスト領域、送信ボタン、リセットボタン
- スクロールする窓の中にGUI部品が入っているというのは結構カルチャーショックだった記憶が…
- 部品に記入するデータの集まり→「フォーム」としてサーバに送信→CGI(後述)で処理
- フォーム記入→送信→サーバ側で処理→結果ページ返送のサイクル(対話性はあまり高くない)

□ しかし、サーバ上でコードが実行できるので、(ほぼ)任意のクライアントサーバアプリケーションが可能。しかもブラウザだけあればいいのでインストール不要(巨大なメリット)→今日でも主流に

□ サーバ側アプリケーションの道具だて(ツール)…大きく変遷

- CGIプログラム…C言語による開発(またはシェルスクリプト)
- PerlによるCGIプログラミング→豊富なライブラリ/モジュール
- JavaによるWebアプリケーション→サーブレット、JSP。Javaによる開発ベースは多いのでそこに載せるという意味では有利
- PHPによるWebアプリケーション→より簡便な設定/記述。その後、RubyやPythonなど使用される言語は多く

□ ブラウザ上で動くスクリプト(クライアント側プログラミング)

- Netscapeブラウザ2.0からブラウザ上で動くスクリプトをサポート
- 当初は「読み込み時にHTMLを生成する」ことと「フォーム部品の内容を読み書きする」ことができる程度→入力のチェック等
- Netscape 5/IE 4くらいから、フォーム部品以外の任意のHTML要素も自由にいじれるようになってきた→より柔軟な表現が可能に(ユーザインタフェースが自由に操作可能)
- ただし、あくまでもブラウザ上で実行されるので、サーバ上のデータには手が届かない→用途に限界

□ Ajax (Asynchronous JavaScript and XML) --- サーバ側とクライアント側の連携

- ユーザインタフェース→クライアントプログラム→サーバ側プログラムを「少しだけ」呼び出す→クライアントプログラム→ユーザインタフェースに反映
- ページ全体を更新しなくても「部分的に更新」しながら動作できる→これまでのWebアプリケーションの形が大幅に変化
- 全部イチから組まなくてもよく使う機能がパッケージされたライブラリが複数存在(prototype.jsなど)

□ Webサービス API

- 多くのサイトがWebサービスAPIを公開→これらのAPIは直接ブラウザで使うのではなく、他のWebア

*筑波大学大学院経営システム科学専攻

プリから間接的に呼び出して使うことを前提としている (無償/有償)。もちろん Ajax で使えるものも。

- 各サイトでは、API を呼び出すことで高度な機能を (自分で作らなくても) 提供可能となる→「マッシュアップ」。
- 複数の API を有機的に組み合わせて役に立つサイトを作り出す可能性

□ フレームワーク

- Web アプリケーションの道具だてとして「プログラミング言語」と「データベース」を駆使して普通に作るのではなく、予め枠組みを用意しておいて必要な箇所を差し替えることで構築する
- XML による記述からアプリケーションを生成する (ジェネレータ方式)
- Convention over configuration --- Ruby on Rail (RoR)。Ruby 言語をつかってドメイン固有言語 (DSL) ふうに記述し、それがそのまま Web アプリケーションとして動作する
- OR マッピング --- リレーショナルデータベースのリレーション/タプルをオブジェクト指向言語のオブジェクトに自動的に対応づけることで、SQL を用いて記述する繁雑さを解消する (本当に解消か?)
- Ajax サポート --- 標準的なライブラリを自動的に Web ページに組み込んで「JavaScript をほとんど書かずに」Ajax を活用

□ REST (REpresentational State Transfer)

- Web アプリケーションや Web サービスの API はこれまで「こういう指令を渡したらこういう動作」という形でできていた→ RPC モデル
- RPC モデルでは、どのような指令が出せてそれがどういう風に動作するのかはそれぞれのドキュメントを見ないと分からない
- そうではなく、URI →「操作対象」、HTTP リクエスト (GET、POST、DELETE、UPDATE、…) →「操作内容」の形で表すのがよいのでは…REST 派の主張
- この場合、URI がそれなりに対象を表すものとなる。ブログなら「http://サーバ/誰/年/月/日」で特定の記事を指定するなど。
- 本当にこういうことがいいのかどうか? 宗教みたい?

1.2 CGI

□ CGI (Common Gateway Interface): Web サーバからサーバ上のプログラムを呼び出す時の「インタフェース」

- 前回やったように、サーバを全部作るのは大変だし実際的でない→ブラウザとの応答はサーバが行い、必要ならサーバが「下請けのプログラム」を呼び出すことで多様な Web アプリを柔軟に実現可能
- ブラウザ→サーバ→プログラム: 「環境変数」によって必要な情報を渡す (加えて method="post"では標準入力からデータが読み込める)。
- プログラム→サーバ→ブラウザ: 標準出力による: 先頭に「ヘッダ情報」、続いて「空行」(ヘッダの終わりを表す)、その後に任意のデータ。
- ヘッダの種類によってサーバはどういう返値を返すか判断

```
Content-type: ...    → 通常の返送
Location: ...        → リダイレクト
Status: ...          → 状態コードを指定
```

□ 例: sam41.cgi --- 環境変数をそのまま表示 (プレーンテキスト出力)

```
#!/bin/sh
echo 'Content-type: text/plain'  ←最低限のヘッダ
echo ''      ←ヘッダ終わり
env          ←環境変数表示
```

□ 例: sam42.cgi --- ヘッダで別ページへ転送

```
#!/bin/sh
echo 'Location: http://www.yahoo.com'
echo ''
```

□ 例: sam43.html (前回のフォームの例) + sam34.cgi --- さっきのフォームの結果受け取り (環境変数 QUERY_STRING で GET のパラメタが受け取れる)

```
<DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
<title>Sample</title>
<style type="text/css">
.sheet { background-color: rgb(200,220,240);
         padding: 5px 10px }
</style>
</head>
<body>
<h1>フォームのテスト</h1>
<form name="f0" method="get" action="sam43.cgi">
<div class="sheet">
名前: <input name="ident" type="text" size="12"><br>
性別:
男性<input name="sex" type="radio" value="male" checked>
```

```
女性<input name="sex" type="radio" value="female"><br>
学年: <select name="grade">
<option value="1" selected>1 年生</option>
<option value="2">2 年生</option>
<option value="3">3 年生</option></select><br>
<button name="cmd" value="x">提
出</button></div></form>
</body>
</html>
```

```
#!/bin/sh
echo 'Content-type: text/html; charset=euc-jp'
echo ''
echo '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">'
echo '<html><head><title>sample</title></head><body>'
echo "<h1>見本</h1><p>$QUERY_STRING</p></body></html>"
```

□ しかしちゃんとパラメタを受け取って内部の処理も行い、結果の HTML を出力するのはかなり大変そうですね? → PHP のようなサポートのありがた味

2 PHP による Web アプリケーション

2.1 PHP 入門

□ PHP (Hypertext Prosessor) --- どこも「PHP」になってないと思うが、初期の名前が「Personal HomePage tools」。ともかくプログラミング言語の 1 つ。

- 最大のアイデア: 「ほとんどは HTML なんだから HTML を書いて、ちょっとプログラムを使いたいとこだけプログラムを埋め込めばいい」
- 「<?php ……PHP プログラム…… ?>」の中だけプログラムコード
- 残りの (外側の) 部分→そのまま出力 (実は PHP の echo 文に翻訳されて実行されるけど一見 HTML がそのままあるように見える)
- 重要: ヘッダの出力動作は「先頭で」やらないといけない (HTML 本文に入った時はヘッダは終わっていることに注意)

□ 基本的なカタチ

```
<?php
header("Content-type: text/html; charset=euc-jp");
/* その他ヘッダ出力はここで */
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html><head><title>title...</title>
<style type="text/css">
/* CSS 記述はここに */
</style><head><body>
<?php
/* ページ本体出力を含む動作 */
?>
</body></html>
```

□ 例: sam44.php --- 数を出力する

```
<?php
  header("Content-type: text/html; charset=euc-jp");
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html><head><title>php sample</title>
<style type="text/css">
div.test { margin: 2px 20px; background: rgb(200,215,255) }
</style></head><body>
<?php
  for($i = 0; $i < 10; ++$i) {
    echo "<div class='test'>番号{$i}です。</div>";
  }
?>
</body></html>
```

□ 解説:

- for 文とかは C や Java と同じ。変数は「$」をつける (シェルと同じ)。echo は PHP の命令で文字列を出力する。「{$ i}」は文字列中に変数の値を埋め込む。

□ 演習: この例題をコピーして動かせ。動いたら出力の内容を変更して次のようなものを出力させてみよ。

- 1 から 20 までの数
- 九九の表
- fizzbuzz 問題 (1 から 100 までの数値を出力するが、ただし 3 の倍数の時は代わりに「fizz」、5 の倍数の時は代わりに「buzz」、3 と 5 の公倍数の時は代わりに「fizzbuzz」と打ち出せ)。

2.2 フォームデータの受け取り

□ PHP の利点…豊富な組み込み機能 (多数のモジュールが選択可能)

□ HTML からのパラメタの受け取りなども簡単。

- 「$_GET['　名 前']」「$_POST['　名 前']」で method="get"、method="post" で渡されたパラメタがすぐ利用可能。

□ 例: sam43.php --- 先のフォームのパラメタを表示。

```
<?php
  header("Content-type: text/html; charset=euc-jp");
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html><head><title>php sample</title>
<style type="text/css">
p { margin: 2px 20px; background: rgb(200,215,255) }
</style></head><body>
<?php
```

```
  echo '<p>こんにちは'.$_GET['ident'].' さん。</p>';
  echo '<p>性別は'.$_GET['sex'].' ですね。</p>';
  echo '<p>学年は'.$_GET['grade'].' ですね。</p>';
?>
</body></html>
```

□ 前のようにフォーム (HTML) と処理 (PHP) を分けるといまいち。

- 開発しにくい、分けてあるので分かりにくい
- ある結果を表示しつつ次の入力を入れたい。
- 次々に URI が移っていくのはやりづらい。「ある特定の処理」は 1 つの URI で。
- → 1 つの PHP ページで処理+フォーム (初回は処理しないように)。

□ 例: sam45.php --- 合計の計算

```
<?php
header("Content-type: text/html; charset=euc-jp");
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html><head><title>php sample</title>
<style type="text/css">
div { padding: 10px; background: rgb(200,215,255) }
</style></head><body>
<h1>合計を求める</h1>
<form method="get" action="#"><div>
<?php
  $total = 0; $data = 0;
  if(!isset($_GET['cmd'])) {
    echo "<p>値を入れてください</p>";
  } else if($_GET['cmd'] == 'clear') {
    echo "<p>クリアしました。</p>";
  } else if($_GET['cmd'] == 'calc') {
    if(is_numeric($_GET['total'])) $total = $_GET['total'];
    if(is_numeric($_GET['data'])) $total += $_GET['data'];
    echo "<p>現在までの合計は{$total}です。</p>";
  } else {
   echo "<p>処理内容が不明です。</p>";
  }
  echo "<input type='hidden' name='total' value='{$total}'>";
?>
<input name="data" type="text" size="6">
<button name="cmd" value="calc">計算</button>
<button name="cmd" value="clear">クリア</button>
</div></form></body></html>
```

□ 解説:

- 提出されてくるデータは total、data の 2 つ (プラス cmd…後述)。これらに対応して PHP 側でも変数を用意。初期値は 0。
- isset() はそのデータがあるかどうか。なければ初回のはず→単にメッセージを表示
- cmd は押したボタンに応じた value の値が入って来る。
- それが'clear' だったら total、data とも 0 のままでよい。
- それが'calc' だったら total に data を加える。ただし、数値の形をしているときだけ。(ブラウザから来たデータの形は絶対にそのまま信用してはいけない。) そして total を表示。
- いずれにせよ、現在の total は type="hidden"で次回に受け渡す。
- 最後にフォーム用意する。

□ 演習: この PHP プログラムを久野のページで動かせ。

```
http://w3in/~kuno/ist06/sam45.php
http://w3in/~kuno/ist06/sam45b.php
```

複数人数でバリバリ計算しても干渉しないことを確認すること。別バージョン (sam45b.php) は干渉が起きることを確認すること。

□ 演習: この PHP プログラムを自分のところに持って来て動かせ。動くことを確認したら、足し算の他に「引き算」もできるように改造してみよ。(もっと色々できるようにしてもよい。)

2.3 セッション管理

□ 先の例題は「アプリケーションに必要な全情報を毎回ブラウザとサーバで受け渡す」ことで成立していた。

□ データを一部サーバで保管しておこうとすると面倒に…

- たとえば total をファイルに格納しておくとどうなるか? (もちろん、これだけのデータならそんな必要はないが、もっと大量のデータがある場合など)

□ たとえば、先程の別バージョン…実は現在データをファイルに格納していた。そのため「総計」が 1 個しかなく干渉が起きる。

```
<?php
header("Content-type: text/html; charset=euc-jp");
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html><head><title>php sample</title>
<style type="text/css">
div { padding: 10px; background: rgb(200,215,255) }
</style></head><body>
<h1>合計を求める</h1>
<form method="get" action="#"><div>
<?php
  $total = 0; $data = 0;
  if($fp = fopen('data.txt', 'r')) {
    $total = fgets($fp); fclose($fp);
  }
```

```
  if(!isset($_GET['cmd'])) {
    echo "<p>値を入れてください</p>";
  } else if($_GET['cmd'] == 'clear') {
    echo "<p>クリアしました。</p>"; $total = 0;
  } else if($_GET['cmd'] == 'calc') {
    if(is_numeric($_GET['total'])) $total = $_GET['total'];
    if(is_numeric($_GET['data'])) $total += $_GET['data'];
    echo "<p>現在までの合計は{$total}です。</p>";
  } else {
   echo "<p>処理内容が不明です。</p>";
  }
  if($fp = fopen('data.txt', 'w')) {
    fputs($fp, $total); fclose($fp);
  }
?>
<input name="data" type="text" size="6">
<button name="cmd" value="calc">計算</button>
<button name="cmd" value="clear">クリア</button>
</div></form></body></html>
```

□ 解説:

- fopen()、fgets()、fputs()、fclose() でファイルのデータを読み書き。
- ファイルは Web サーバのユーザに読み書きできるようにしておく必要

```
echo '0' >data.txt; chmod go+rw data.txt
```

□ ファイルに書くのはいまいちか? 全部 hidden で受け渡す?

- ネット上に流せない機密データだと必ずサーバ上に保管しなければならない。→ どうすればいい?

□ 答え: それぞれのユーザに対して「セッション ID」を割り当てる。

- セッション ID は外部から推測されにくいように乱数などで生成。
- 「総計」などの個別データはセッション ID と対で保管する。
- 割り当てたセッション ID をサーバ→ブラウザに送信し、次にページデータが送信されるときにブラウザ→サーバに返送してもらう
- 返送された ID を見れば「誰の作業の続き」かが分かる

□ 具体的な送信/返送メカニズム…

- (1) type=hidden に入れて受け渡す
- (2) URL の末尾に「?」に続けて ID 情報をつける
- (3) クッキー情報に入れる

□ クッキーとは?

- コンピュータ業界では「ちょっとした情報」程度の意味
- サーバからブラウザに Set-Cookie: ヘッダで情報を送る
- ブラウザがそれを記憶しておく
- 同じサーバ/ディレクトリのリクエストでブラウザからサーバに Cookie: ヘッダで同じ情報を返送する

□ 3 つの方法の得失…

- type=hidden は全部のページに form がないといけない
- 「?」の後ろにくっつける方法はそれがブラウザの画面で見える (うるさい?)
- クッキーは表だって見えない (ただしユーザの閲覧履歴を調査したり、さまざまな用途で使われるので嫌われることがある→ OFF にする人も)

□ 重要なポイント:「ユーザ ID」などを直接セッション ID として使ってはいけない

- ユーザ ID は「固定されている」→「他人が推測可能」→自分が使用中に他人がその同じセッション ID を使ってサーバに接続可能→「自分のセッションのやりとり」を途中から他人がやってしまうこと (セッションハイジャック) が可能に。
- 乱数の ID であれば推測は困難。ネットを監視していればその ID が行き来すること自体は観測可能だが、SSL のように盗聴できないトランスポート上で使えばまず問題ない。

□ PHP のセッション機能

- 上記のように面倒なことがいろいろあるが、PHP では冒頭で session_start() という関数を呼ぶだけで自動的にセッション追跡を行ってくれる
- セッション機能が ON のときは配列「$_SESSION」に入れたデータは同じセッションの中ではずっと維持される (これに格納したデータはずっと持っておいてもらえる)。

□ sam41.php:先の「合計」を PHP のセッション機能を使って書き直したもの

```
<?php
  session_start();
  header("Content-type: text/html; charset=euc-jp");
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html><head><title>php sample</title>
```

```
<style type="text/css">
div { padding: 10px; background: rgb(200,215,255) }
</style></head><body>
<h1>合計を求める</h1>
<form method="get" action="#"><div>
<?php
  if(!isset($_GET['cmd'])) {
    echo "<p>値を入れてください</p>";
  } else if($_GET['cmd'] == 'clear') {
    echo "<p>クリアしました。</p>";
    $_SESSION['total'] = 0;
  } else if($_GET['cmd'] == 'calc') {
    if(is_numeric($_GET['data']))
      $_SESSION['total'] += $_GET['data'];
    echo "<p>現在までの合計は".
           $_SESSION['total']."です。</p>";
  } else {
   echo "<p>処理内容が不明です。</p>";
  }
?>
<input name="data" type="text" size="6">
<button name="cmd" value="calc">計算</button>
<button name="cmd" value="clear">クリア</button>
</div></form></body></html>
```

□ 解説:

- 冒頭で session_start() を呼ぶ (ヘッダと一緒の場所に入れる必要がある)
- 以後$total の代わりに$_SESSION['total'] を使うようにするだけ。この配列の内容は PHP 処理系によってずっと維持してもらえる (サーバが止まったら無くなるがそれまでは OK)。

□ 演習:

- このプログラムをコピーして来て「電卓」にせよ。
- 動いたら「メモリ機能」をつけてみよ。具体的には「メモリ記憶」ボタンを押すと現在の総計がメモリに入る。「メモリ呼び出し」ボタンを押すと入力値のところにそのメモリの値が入る。(ヒント: それには「<input name="data" type="text" value="入れたい値" size="6">」のように value で値を指定すればよい。)

3 PostgreSQL による検索

□ この後、データベースから情報を取り出して来るのをやるので、最低限の SQL select 構文と PostgreSQL(Unix 上のフリー DBMS) による演習をやります (データベースとは何か等はすべて略)

□ 覚えていただく構文はこれだけ:

```
select 欄名,… from テーブル
  where 条件 and 条件 and … ;
```

覚えましたか :-)。なお、沢山でてきて欲しくないときは「;」の直前に「limit 個数」というのを入れておくと最大その個数で止まります。

□ 次に、PostgreSQL のインタプリタを起動します。

```
psql -d kuno  ←とりあえず久野の
% \d 書籍      ←今回は「書籍」のみ
              Table "public.書籍"
  Column  |          Type          | Modifiers
----------+------------------------+-----------
 isbn     | character(10)          | not null
 題名     | character varying(50) |
 著者     | character varying(30) |
 出版社   | character varying(20) |
 発行年月 | character(5)          |
 価格     | integer                |
 種別     | character(4)           |
 順位     | integer                |
Indexes: "書籍_pkey" PRIMARY KEY, btree (isbn)
% select 題名 from 書籍 where 価格 > 2000 ;
                    題名
-----------------------------------------
 シニアマーケットに学ぶ資産運用アドバイス
 アメリカの高校生が学ぶ経済学
 Winny の技術
(3 rows)
% \q  ← psql を終わる
```

□ このように、select 文だけ分かっていれば一応なんでも検索はできるようになるわけです (ほんとか?)

□ 演習:

- マシン「smb」で上記の psql コマンドを使って書籍データベースの内容をしばらく眺めてみよ。

4 PHP と PostgreSQL の接続

□ PHP からは簡単に各種の DBMS に接続してデータベースの検索や更新が可能。

□ PostgreSQL の場合も次の 5 つの関数だけ知っておけばよい。

- $conn = pg_connect(指定文字列) --- DBMS に接続
- $res = pg_query($conn, SQL 文の文字列) --- SQL を実行する
- $num = pg_num_rows($res) --- 結果の列数を返す
- $arr = pg_fetch_row($res) --- 結果の 1 列を配列として返す

- $str = pg_escape_string(文字列) --- 文字列を安全にエスケープする

□ 最後のはなぜ必要かというと、SQL Injection（ユーザデータの中に SQL の命令を埋め込むことによってデータベースを勝手に操作するという攻撃）に対する防御のため。

□ 簡単な例題: 書籍データを書名で検索

```
<?php
header("Content-type: text/html; charset=euc-jp");
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html><head><title>php sample</title>
<style type="text/css">
div.test { padding: 4px; background: rgb(255,215,200) }
td { margin: 2px 4px; padding:
     4px; background: rgb(200,215,255) }
</style></head><body>
<form method="post" action="#"><div class="test">
<p>検索語:<input type='text' name='word'>
<button name="cmd" value="calc">検索</button>
<?php
if($_POST['word'] == '') {
  echo "<p>検索キーワードを入れてください。</p>";
} else if(!($conn = pg_connect("dbname=kuno"))) {
  echo "<p>データベース接続できません。</p>";
} else {
  $word = pg_escape_string($_POST['word']);
  $res = pg_query($conn,
    "select isbn, 題名, 著者 from 書籍"
    ." where 題名 like '%".$word."%'");
  if(pg_num_rows($res) > 0) {
    echo "<table><tbody><tr><th>ISBN</th>"
         ."<th>題名</th><th>著者</th></tr>\n";
    while($a = pg_fetch_row($res)) {
      echo "<tr><td>{$a[0]}</td><td>{$a[1]}"
           ."</td><td>{$a[2]}</td></tr>\n";
    }
    echo "</tbody></table>";
  } else {
    echo "<p>検索結果が空でした。</p>";
  }
}
?>
</div></form></body></html>
</div></form></body></html>
```

□ 解説:

- フォームは検索語と検索ボタンだけ
- 検索語があればデータベースに接続し、「select ... where 題名 like '%xxx%'」で検索。パーセントは SQL の like で任意文字列にマッチさせられる。
- 結果を HTML のテーブルに加工して表示。
- なお「.」は文字列の連結演算（長い文字列を行を分けて書くのに使用）

□ HTML のテーブルに使うタグは次のもの

- <table>...</table> --- テーブル要素
- <tbody>...</tbody> --- テーブル本体
- <tr>...</tr> --- 本体中の 1 行
- <th>...</th>、<td>...</td> --- 1 行中の 1 セル

□ 演習:

- この PHP プログラムをコピーしてきて動かせ。
- もっと「さまざまな」検索ができるように改良してみよ。