情報システム技術論'08 — ＃5

久野　靖*

2008.5.15

1　PHP とデータベースの連携

1.1　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。

1.2　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
```

*筑波大学大学院経営システム科学専攻

```
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>
```

□ 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。

1.3 PHPセッションとデータベースの連携

□ データベースのいいところ→並行処理などの機能が最初から含まれている

- このため、複数ユーザによる更新(Webアプリの面倒なところ)も問題なし
- PHP→ユーザインタフェース側、DB→データ保管側で組み合わせると便利
- 「このユーザのユーザID」などをセッションオブジェクトに保管して、このキーを併用してデータの検索/更新を行うだけ
- 直接クッキーなどに入れると危険だがセッションオブジェクトはPHPがそれなりに処理してくれる

□ 例題:パスワードを用いたログイン処理のあるページ群(ほとんどそれしかないとも言う)。ページは3つある。

- ページA --- ログインページ。IDを持っている人はIDとパスワードを打ち込んでログインするとページCへ行く。IDを持っていない人は新しいログインIDを選び、ページBへ行く。
- ページB --- ユーザ情報変更ページ。新規ユーザも既存ユーザも、このページでユーザ情報(パスワード、名前、電話番号)を変更することができる。変更は何回でも繰り返し行え、変更完了したらページCへ行く。
- ページC --- 「本体」ページ。本来はここから先で色々な処理をするわけだが、この例題では単に「ようこそ○○さん」と表示するだけ。ここから「更新」を選択するとページB、「ログアウト」を選択するとセッションを破棄した上でページAへ行く。

□ ユーザ情報を格納するテーブルの形:

```
           Table "public.ユーザ"
 Column |          Type          | Modifiers
--------+------------------------+-----------
 id     | character varying(20) |
 pass   | character varying(20) |
 name   | character varying(60) |
 tel    | character varying(20) |
 other  | character varying(60) |
```

- 要するにID、パスワード、名前、電話番号、その他、だけ。もっと多数あっても何ら問題ないが。

□ 3つのページの基本的な構成: 先頭部分で(HTMLを1行も出さないうちに)チェック等の処理をすべておこなう。

- 理由: 別ページにジャンプするにはLocation:ヘッダを出力する必要があるから。
- 同じページにとどまっていてよい場合だけ、そのまま下へ進んでHTML出力をおこなう。

□ ページA:ログインページ

```
<?php
session_start();
if(!($conn = pg_connect("dbname=kuno"))) {
  $mesg = "データベース接続不能。";
} else if($_GET['cmd'] == 'newuser' &&
          $_GET['newid'] != '') {
  $newid = pg_escape_string($_GET['newid']);
  $res = pg_query($conn, "begin");
  $res = pg_query($conn,
    "select id from ユーザ where id='{$newid}'");
  if(pg_num_rows($res) == 0) {
    $res = pg_query($conn,
      "insert into ユーザ values('{$newid}',NULL,NULL)");
    if(pg_query($conn, "commit")) {
      $_SESSION['userid'] = $newid;
    }
    header("Location: sam51b.php"); return;
  } else {
    $res = pg_query($conn, "commit");
    $mesg = "その ID は取得されています。";
  }
} else if($_GET['cmd'] == 'login' &&
          $_GET['userid'] != '') {
  $userid = pg_escape_string($_GET['userid']);
  $res = pg_query($conn, "select id,pass from "
    ."ユーザ where id='{$userid}'");
  $a = pg_fetch_row($res);
  if(pg_num_rows($res) == 0) {
    $mesg = "ユーザ ID かパスワードが違います。";
  } else if($a[0] == $userid &&
            $a[1] == $_GET['pass']) {
    $_SESSION['userid'] = $userid;
    header("Location: sam51c.php"); return;
  } else {
    $mesg = "ユーザ ID かパスワードが違います。";
  }
} else {
  $mesg = "ユーザ ID とパスワードを入れてください。";
}
header("Content-type: text/html; charset=euc-jp");
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html><head><title>php sample</title>
<style type="text/css">
div.test { margin: 2px 20px; padding: 4px;
           background: rgb(200,215,255) }
</style></head><body>
<form method="get" action="#"><div class="test">
<p>ユーザ ID とパスワードを入力してログインを。</p>
 ユーザ ID:<input type="text" name="userid">
 パスワード:<input type="password" name="pass">
 <button name="cmd" value="login">
 ログイン</button><br>
<p>はじめての方はユーザ ID を選んでください。</p>
 新規ユーザ ID:<input type="text" name="newid">
 <button name="cmd" value="newuser">
 新規 ID</button><br>
<?php if($mesg) echo "<p>{$mesg}</p>"; ?>
</div></form></body></html>
```

□ 概要:

- 最初に session_start() する。また、この例題では「よそへ飛ぶ」という処理が何回もあるが、よそへ飛ぶためには「Locaiton:　行き先」というヘッダを出力する必要があるため、ほとんどの処理を冒頭 (HTML を出力するより前) で行う。もちろん、HTML を出力しない場合は Content-type:も出力してはいけない。
- 次の枝分かれは、(1) データベース接続不調、(2) 新規ユーザボタン、(3) ログインボタン、(4) それ以外 (初回) の処理、の 4 つの枝分かれになっている。
- メッセージはページ本体の中で出す必要があるので、変数 mesg に入れておいて後でこれを参照する。たとえばデータベース接続不調の場合は「接続できない」というメッセージを入れるだけ。
- 新規 ID の場合は、まずそのユーザ ID が表に登録されているか検索し、結果数が 0 なら新たなデータを挿入し (このときユーザ ID 以外の欄はとりあえず NULL にしてある)、ページ B へ移動する。0 でなければ、「もうある」というメッセージを出力する。なお、チェックしてから挿入までの間に誰かが同じ ID を挿入すると困るので、これらの処理は begin〜commit で囲んである。
- 通常ログインの場合は、そのユーザ ID のデータを検索して、データベース内のパスワードと入力されたパスワードの一致を確認する。一致していればページ C へ移動する。一致していなければ違うというメッセージを出す。
- いずれでもない場合は最初にこのページへ来たので、単に「ID とパスワードを入れてください」というメッセージを出す。
- 以上で PHP 部分はほとんど終わりで、その先は HTML によりユーザ ID 欄、パスワード欄、新規 ID 欄、そしてログインボタン、新規登録ボタンを用意している。ただし、一番最後に変数 mesg の内容を表示しておく。
- このページからページ B/C に移動するときは、その前に$_SESSION['userid'] にこのユーザのユーザ ID を格納。

□ ページ B:ユーザ情報更新ページ

```
<?php
session_start();
$userid = $_SESSION['userid'];
if(!($conn = pg_connect("dbname=kuno"))) {
  $mesg = "データベース接続不能。";
} else {
  $res = pg_query($conn,
```

```
    "select * from ユーザ where id = '{$userid}'");
  if(pg_num_rows($res) == 0) {
    header("Location: sam51a.php"); return;
  }
  $a = pg_fetch_row($res);
  $name = $a[2];
  $tel = $a[3];
  if($_GET['cmd'] == 'ok') {
    $mesg = "";
    $f = array('ユーザ ID','パスワード',
               '名前','電話番号','追加情報');
    for($i = 0; $i < 3; ++$i) {
      if($a[$i] == NULL) {
        $mesg .= "「{$f[$i]}」が空です。<br>";
      }
    }
    if($mesg == "") {
      header("Location: sam51c.php"); return;
    }
  } else if($_GET['cmd'] == 'update') {
    $res = pg_query($conn, "begin");
    $mesg = "";
    if($_GET['pass1'] != '') {
      if($_GET['pass1'] != $_GET['pass2']) {
        $mesg .= "2 つのパスワード不一致。<br>";
      } else {
        $pass = pg_escape_string($_GET['pass1']);
        $res = pg_query($conn,
          "update ユーザ set pass='{$pass}'".
            " where id='{$userid}'");
      }
    }
    if($_GET['name'] != '') {
      $name = pg_escape_string($_GET['name']);
      $res = pg_query($conn,
        "update ユーザ set name='{$name}'".
          " where id='{$userid}'");
    }
    if($_GET['tel'] != '') {
      $tel = pg_escape_string($_GET['tel']);
      $res = pg_query($conn,
        "update ユーザ set tel='{$tel}'".
          " where id='{$userid}'");
    }
    if(pg_query($conn, "commit")) {
      $mesg .= "更新完了。";
    } else {
      $mesg .= "更新エラー; 再試行してください。";
    }
  } else { // 最初に入力ページを表示した時
    $mesg = "フィールドに記入して「更新」。".
            "変更不要なら「完了」";
  }
}
header("Content-type: text/html; charset=euc-jp");
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html><head><title>php sample</title>
<style type="text/css">
div.test { margin: 2px 20px; padding: 4px;
           background: rgb(200,215,255) }
</style></head><body>
<form method="get" action="#"><div class="test">
<?php
echo "<p>ユーザ「{$userid}」のデータ登録/更新</p>".
  "パスワード:<input type='password' name='pass1'><br>".
  "確認用入力:<input type='password' name='pass2'><br>".
  "名前:<input type='text' name='name' value='{$name}'>".
  "<br>電話:<input type='text' name='tel' value='{$tel}'>".
  "<br><p>{$mesg}</p>";
?>
<button name="cmd" value="update">更新</button>
<button name="cmd" value="ok">完了</button>
</div></form></body></html>
```

□ 解説:

- データベースに接続できない場合は先の例と同様。
- 接続できる場合は、セッションで記録しているユーザ ID に対応する組を取得し、そこから name と tel を変数に取り出しておく (パスワードは画面表示しないので取り出さないでよい)。
- 続いてコマンドの種類で (1) 更新完了、(2) 更新、(3) 最初にこのページへ来た状態、のどれかに分岐する。
- (1) の場合は、まだ NULL の項目があったら完了ではないので、mesg に空文字列を入れ、ループで全フィールドについて NULL かどうかチェックし NULL なら警告を追加する。ループが終って mesg が空だったら NULL はなかったのでページ C へ飛ぶ。
- (2) の場合は、各フィールドごとに SQL の update を使って値を設定してゆく。一連の動作がまとめて完了したら OK なので、全体は begin 〜 commit で囲んでいる。
- (3) の場合は、単に説明メッセージを設定する。
- あとはほぼ通常の HTML だが、名前と電話の入力欄の初期値はデータベースから取得した値を入れておく (確認用)。

□ ページ C:「本体」ページ

```
<?php
session_start();
$userid = $_SESSION['userid'];
if($_GET['cmd'] == 'logout') {
  session_destroy();
  header("Location: sam51a.php"); return;
} else if($_GET['cmd'] == 'update') {
  header("Location: sam51b.php"); exit;
} else if(!($conn = pg_connect("dbname=kuno"))) {
  $mesg = "データベース接続不能; 管理者に連絡を。";
} else {
  $res = pg_query($conn,
    "select * from ユーザ where id = '{$userid}'");
  if(pg_num_rows($res) == 0) {
    header("Location: sam51a.php"); return;
```

```
  }
  $a = pg_fetch_row($res);
  $name = $a[2];
  $tel = $a[3];
}
header("Content-type: text/html; charset=euc-jp");
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html><head><title>php sample</title>
<style type="text/css">
div.test { margin: 2px 20px; padding: 4px;
           background: rgb(200,215,255) }
</style></head><body>
<form method="get" action="#"><div class="test">
<?php
echo "<p>こんにちは、{$name}さん (ID:{$userid}、".
       "電話: {$tel})。</p>";
if($mesg != "") echo "<p>{$mesg}</p>";
?>
<button name="cmd" value="update">情報更新</button>
<button name="cmd" value="logout">ログアウト</button>
</div></form></body></html>
```

□ 解説:

- コマンドが logout なら、セッションを破棄してページ A へ移動。
- コマンドが update なら、ページ B へ移動。
- データベースに接続できなければその旨表示。
- 接続できれば、セッションで覚えているユーザ ID でデータを検索して名前と電話を変数に取り出す。
- 以下は普通の HTML だが、メッセージの一部に名前と電話を挿入。

□ 演習: この例題を久野のページから動かしてみよ。

```
http://w3in/~kuno/ist08/sam51a.php
```

まず自分の好きな ID を登録し、情報を登録、完了、また情報を変更。しばらく遊んでみる。

1.4 Web アプリの不正アクセス対策

□ 先の演習つづき:

- 「名前」欄に次の文字列を登録して完了してみるとどうなるか?

```
<script>alert(document.cookie)</script>
```

- 新しい窓を開き、次のページを表示してみる。

```
http://w3in/~kuno/ist08/sam51x.html
```

そして、そこにある test というリンクを選択してみる。

□ 上の演習で何が起こっていたか?

- (a) クッキーの中身が他人に知られそうになる。セッションを乗っ取られる。
- (b) 自分のアカウント内で自分が意図していない情報更新が行われる

□ (a) が起これば当然 (b) も起こってしまう。(b) が起こったら「勝手に買物」とかいくらでも起こり得る。

- 代表的な手法について説明しておく:

□ XSS(クロスサイト・スクリプティング)

- 悪意ある人がページに「セッション情報などを盗む」スクリプトを挿入する→そのページを見た人がセッション情報を盗まれる
- 掲示板のような「ある人が書き込んだものを多くの人が見る」場面に起こりやすい。
- さきの「alert(...)」は自分の情報を自分で見るだけだったが、たとえば「ユーザの名前一覧」機能がついていたら他人の書いた「alert(...)」が動いてしまう。
- 「alert(...)」は画面で見えるだけだが、そのかわりに「悪意ある人のサイトにクッキー情報をこっそり送信」なども可能

□ 対策: ユーザが書き込んだものをページに貼るときは「<」「>」などをすべて文字エントリにする。エスケープ処理 (SQL インジェクション対策などと同様)。

- htmlspecialchars(...) でエスケープできる。

```
echo "<p>こんにちは、".htmlspecialchars($name)
      ."さん (ID:".htmlspecialchars($userid)
      ."、電話:".htmlspeccialchars($tel)
      .")。</p>";
```

□ CSRF(クロスサイト・リクエストフォージ)

- ログイン中の人に別のページに仕込んだリンクや JavaScript を踏ませる→こっそり「偽のコマンド」を送信→ログイン中なので正しく処理されてしまう
- リンクを踏まなくても JS なら表示しただけでも起こる

□ 対策: クッキーだけにたよらず、type=hidden にもセッション情報を入れておいて提出時に合っているかどうかチェック (どの対策がよいかは論争あり)

□ セッション固定攻撃

- ログインする「前に」そのページへ行くときに「?PHPSESSID=123456」のようにセッション ID を指定したリンクから飛ぶように仕向ける

- PHP では「?」の後ろのセッション ID を受け取ってしまうため、セッション ID が他人に知られてしまう (というか、他人が指定したセッション ID をそのまま使ってしまう)

□ 対策: セッション ID は「?」の後ろにつける方法を使わせないように PHP 側を設定する。ログイン前にセッション ID をつけ直す。(ログインページで start_session() の直後に session_regenerate_id() を呼ぶ)

□ 結局…

- さまざまな「新たな攻撃」が生み出されている
- どういうことに注意すべきかは先人の知恵を十分活用するべき→本などで勉強してください。
- 参考: 萩原佳明, 不正アクセス対策入門の入門, 翔永社, 2006

1.5 PHP のまとめ

□ PHP:サーバ上で動くコード (サーバ側プログラミング)

- HTML コード中への埋め込み、HTML フォーム値の受け取り
- セッション管理
- データベース連携
- →サーバ上で共有データを操作する上で威力
- →一方、ブラウザ上での細かい操作はできない (当然)

□ 不正アクセスには十分用心を!

2 JavaScript とクライアント側コード

2.1 クライアント側プログラミング

□ クライアント側コード→サーバからブラウザに転送されてきて、そこで動作するコード

- Java Applet →いちばん最初
- Macromedia Flash →グラフィクス系ではいちばんメジャー
- JavaScript →ブラウザ上で直接動作 (他のものはプラグイン)

□ 利点:ブラウザ上で動作→ネットワーク通信をはさまない、反応が速い

- アニメーション、ユーザインタフェースなどにおいてメリット

□ 弱点:サーバに置かれていたコードが手元のマシンで動作→セキュリティ対策必要

- サンドボックス:ファイルアクセス禁止、ネットワークアクセス禁止 (もともとのサーバのみ)、個人情報アクセス禁止
- それでもしばしば多様なセキュリティホールや攻撃の原因になる

2.2 JavaScript

□ 1994 年ころ、ブラウザシェア No.1 だった Netscape 社が開発した言語

- もともと LiveScript という名前だったが Java ブームにあやかり改名
- No.1 ブラウザの Netscape ブラウザに組み込み→MSIE も追いかけて JScript という類似品を搭載→事実上の標準となった
- 目的→ブラウザに組み込み、ブラウザの動作を調整する (アプレット等プラグインで動作する言語と違ってブラウザと親密にくっついて制御可能)

□ 当初は JavaScript で制御可能なのは「フォーム部品の表示内容」だけだった (Netscape 4.x、IE 4.x)

□ 現在ではブラウザ内の多くのオブジェクトが直接 JavaScript から制御可能になっている (DOM --- Document Object Model)。

- DHTML (Dynamic HTML) --- HTML 記述の内容を JS から操作してページ内容を変更すること全般を指す

□ さらに現在ではサーバとの非同期通信が可能に→ Ajax (Asynchronous JavaScript + XML)

□ JavaScript コードの入れ方…

- イベントハンドラ: 「onclick="コード"」みたいに HTML タグの属性で指定。なお、このコードから true/false を返すこともある (たとえばクリックだとその結果の動作に進むか何もしないかを制御)。
- JavaScript URI: 「href="javascript: コード"」で A タグに書く。ただしコードの結果が undefined という値でない限り、コードの生成した結果を文字列に変換してページに表示してしまうので、最後に「;void(0)」をつけるのがよい。
- 「<script type="text/javascript">...</script>」にはさんで何行でもコードを書く。長いものはこ

の方式で。関数定義なども。なお、この中で「document.write(...)」を実行すると出力した文字列がこのタグの位置に挿入される (あまり使わない機能)。

2.3 JavaScript によるフォーム部品操作

□ フォーム部品の操作→いちばん初期から提供されている機能 (DOM0)

- 操作する部品等は「var o = document.forms. フォーム名.elements. 部品名;」でオブジェクト取得
- 入力欄等なら「o.value」で内容テキストを参照/更新できる
- 選択メニューなら「o.selectedIndex」で選択番号を参照/更新できる。またその各 option は o.options[i] で参照できる。
- ラジオボタン/チェックボックスなら「o.checked」で ON/OFF を参照/更新できる

□ 動作のきっかけ→イベントハンドラ

- ボタンであれば HTML 属性「onclick="コード"」でクリック時に動作
- 一般の部品であれば HTML 属性「onchange="コード"」で値変更時に動作

□ 例題: sam52.html --- ボタンを押すと入力欄の値が加算される

```
<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>
<script type="text/javascript">
function calc() {
  var count = document.forms.f0.elements.count;
  var step = document.forms.f0.elements.step;
  var i = parseInt(count.value) +
    parseInt(step.options[step.selectedIndex].value);
  count.value = i;
}
</script>
</head>
<body>
<h1>JavaScript のテスト</h1>
<form name="f0" method="get" action="sam34.cgi">
<div class="sheet">
数値:<input name="count" type="text" value="1"><br>
加算値:<select name="step">
<option value="1" selected>1</option>
<option value="10">10</option>
<option value="100">100</option></select><br>
<button type="button" onclick="calc()">
加算</button></div></form>
</body>
</html>
```

□ 演習:

- 上記の例題をコピーして動かせ
- ボタンを押した時別の計算をさせてみよ
- ボタンを押した時選択メニューの選択が変わるようにしてみよ。なお、0〜2 の一様乱数 (整数) は「Math.floor(2*Math.random())」で計算できる。

2.4 JavaScript による DOM 操作

□ DOM(Document Object Model) --- HTML 文書などの構造を操作するための API(プログラミングインタフェース) ←フォーム部品に限定されない

- HTML 文書は Node オブジェクトのツリー構造でできている
- 特定要素を参照するには「id="ID 名"」で名前をつけ、「document.getElementById('ID名')」で参照。
- ツリー要素を取り除いたり挿入したりするメソッド
- 各要素 (HTML の要素に対応) のプロパティを設定/参照可能

□ CSS プロパティの設定。たとえば x が要素だとして…

- x.backgroundColor = 'red' --- CSS の background-color: red に対応
- このようにプロパティの設定が分かりやすく書ける

□ この後試してみたいプロパティ

- top、left、width、height --- 位置と大きさ
- color、backgroundColor --- 文字色、背景色

□ 例題: sam53.html --- ボタンによる画面の変更

```
<DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
<title>Sample</title>
<style type="text/css">
#x0 { position:absolute;
      top:150px;
      background-color: green;
      width:20px; height:20px; }
```

```
</style>
<script type="text/javascript">
var top = 150, left = 0;
function t1() {
  document.getElementById('x0')
   .style.top = (top+=10) + 'px';
}
function t2() {
  document.getElementById('x0')
   .style.backgroundColor = 'red';
}
</script>
</head>
<body id="b0">
<h1>JavaScript test</h1>
<p><button onclick="t1()">t1</button>
   <button onclick="t2()">t2</button></p>
<div id="x0">X</div>
</body>
</html>
```

□ 解説:

- 移動させる要素は position:absolute 指定必要
- ボタンクリックで関数を呼び出す
- 関数内部でオブジェクトのプロパティを設定

□ 演習:

- 上記のプログラムをコピーして動かせ
- 動いたらボタンを増やして別の機能をつけてみよ
- 「setInterval(関数, ミリ秒数);」で一定時間間隔で関数を実行させられる。これを利用して要素を動かしてみよ。

2.5 innerHTML

□ DOM で HTML 要素の追加/削除、テキストの追加/削除もできるが、けっこう面倒

□ 別の方法として、要素オブジェクトの innerHTML プロパティに HTML 文字列を設定すると「その要素の内側にその HTML が書かれていた」のと同じ効果 (標準でないが主要なブラウザがサポート)

□ 例題: sam54.html --- ランダムな色の段落生成

```
<DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
<title>Sample</title>
<script type="text/javascript">
function chg() {
  var d = document.getElementById('d0');
  var s = '';
  for(var i = 0; i < 100; ++i) {
    var r = Math.floor(256*Math.random());
    s += '<p style="background: rgb(' +
      r + ',' + (255-r) + ',255)">x</p>';
  }
  d.innerHTML = s;
}
</script>
</head>
<body id="b0">
<h1>JavaScript test</h1>
<p><button onclick="chg()">chg</button></p>
<div id="d0"></div>
</body>
</html>
```

□ 解説:

- ボタンを押すと関数 chg() を起動
- 文字列 s に次のような段落を 100 個追加

  ```
  <p style="background: rgb(x,y,z)">x</p>
  ```

- 最後に s を div の innerHTML に入れて表示

□ 演習:

- 上記の例題をコピーして動かせ
- ボタンを押すと 1 秒ごとに色が変わるようにしてみよ
- 別の HTML を生成するように直してみよ

3 Ajax

3.1 Ajax とは

□ Ajax (Asynchronous JavaScript + XML) とは…

- JavaScript による非同期通信を用いた Web アプリ構成
- 技術的には新しくないとされる。Google Maps で注目
- Flash や Java Applet でも同様のことはできる (が、そんなにメジャーでもないような…)
- AjaxWrite など見るとなかなかすごいと思う

□ 従来の Web アプリ:

- 複数の「ページ群」として動作
- 「送信」したとき始めてサーバにデータが渡って処理、それ以外はローカルで動作
- サーバのプログラムは基本的に HTML を返送

□ Ajax:

- 1 つの「ページ」内だけで何でもできる。

- 必要があったらそのつどサーバと通信。通信は「裏側」でやるのでユーザにはそれと認識されないことが多い
- サーバのプログラムは何を返送してもよい (JavaScript でそれを読んで自由に処理可能)

3.2 XMLHttpRequest

□ JS とサーバの通信手段→ XMLHttpRequest

- HTTP プロトコルによる通信をすべて JS 上から制御可能
- その名前にもかかわらず、普通のテキストでも取り扱える
- IE6 と Mozilla 系、IE7 とで多少違うので処理の枝分かれ必要
- 受信完了時 (その他状態変化時) に関数が呼ばれる

3.3 XMLHttpRequest

□ とりあえず作った「指定 URI をサーバから取り寄せ、それを文字列パラメタとして関数 f を呼び出す」関数

```
function load(uri, f) {
  var r = null;
  if(window.XMLHttpRequest)
    r = new XMLHttpRequest();
  else if(window.ActiveXObject)
    r = new ActiveXObject('Microsoft.XMLHTTP');
  if(!r) return false;
  r.onreadystatechange = function() {
    if(r.readyState==4) f(r.responseText); };
  r.open('GET', uri, true); r.send(null);
  return true;
}
```

□ 解説:

- new XMLHttpRequest() → 通信オブジェクト作成 (IE6 だと ActiveX で作成)
- r.onreadstatechange → ここに関数をセットすると状態変化時に呼ばれる
- r.readyState → 状態。4 が「完了」
- r.responseText → サーバから取り寄せたデータを文字列として返す
- r.open(メソッド, URI, 非同期フラグ) → 通信内容の設定
- r.send(データ) → 通信開始

3.4 例題: Ajax によるキーワード検索

□ sam54.php: 書籍タイトルを文字列検索するサーバ側コード

- Ajax から呼び出すので次のような仕様とする
- 検索語は「?word=文字列」を URI の後ろにくっつける (GET メソッド)
- 結果は 1 つの<table>...</table>として返す (HTML の*断片*)
- エラーがあった場合はそのメッセージを ... として返す

```
<?php
header("Content-type: text/plain; charset=euc-jp");
if(!($conn = pg_connect("dbname=kuno"))) {
  echo "<p>データベース接続できません。</p>"; return;
} else {
  $word = iconv('UTF-8','EUC-JP',$_GET['word']);
  $word = pg_escape_string($word);
  $res = pg_query($conn,
    "select 題名 from 書籍".
      " where 題名 like '%{$word}%' limit 10");
  if(pg_num_rows($res) > 0) {
    echo "<table><tbody>\n";
    while($a = pg_fetch_row($res)) {
      echo "<tr><td>{$a[0]}</td></tr>\n";
    }
    echo "</tbody></table>";
  } else {
    echo "<p>検索結果が空でした。</p>";
  }
}
?>
```

□ 解説:

- 普通にデータベース接続
- word を取り出すが、UTF8 で渡ってくるので EUC に変換
- 書籍の題名を検索 (最大 10 件)
- テーブル要素の文字列を返すか、「空でした」を返す

□ 呼び方 (テスト用):

http://w3in/~kuno/ist06/sam55.php?word=単語

- 日本語コードがうまく渡らないので英数字だけ OK

□ 演習: 上記 URI をブラウザで打ち込み試してみよ

□ sam55.html: 書籍タイトルを文字列検索する Ajax アプリ

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html><head><title>Sample</title>
<script type="text/javascript">
function load(uri, f) {
  var r = null;
  if(window.XMLHttpRequest)
    r = new XMLHttpRequest();
  else if(window.ActiveXObject)
    r = new ActiveXObject('Microsoft.XMLHTTP');
  if(!r) return false;
  r.onreadystatechange = function() {
    if(r.readyState==4) f(r.responseText); }
  r.open('GET', uri, true); r.send(null);
  return true;
}
var word = '';
function check() {
  var w = document.getElementById('t0').value;
  if(w == '' || w == word) return; else word = w;
  load('sam55.php?word=' + encodeURIComponent(word), show);
}
function show(s) {
  document.getElementById('d0').innerHTML = s;
}
setInterval(check, 100);
</script>
</head><body>
<h1>Ajax の例題</h1>
<p><input name="t0" id="t0" type="text"></textarea></p>
<div id="d0"></div>
</body></html>
```

□ 解説:

- load() は先に解説した通り
- word が現在の検索語
- check は 100msec に 1 回呼び出されて入力欄の内容をチェック
- 入力欄が前と変化していてなおかつ'' でない→それを word パラメタとして sam55.php を呼ぶ
- 表示関数は show()。これは渡された文字列を div の innerHTML に入れるだけ。

□ 演習:

- sam55.php、sam55.hmlt をコピーして動かせ
- 動いたら別の情報で検索するように直してみよ
- さらに複数の情報の組み合わせ検索もできるとなおよい

4 科目全体のまとめ

□ 分散アプリケーションとその位置付け、構造

□ ネットワーク接続、C/S モデル

□ ステートフル/ステートレス、マルチスレッド

□ RPC、ORB(CORBA)、RMI、オブジェクトモビリティ

□ 他の分散モデル、Linda

□ World Wide Web、HTML+CSS、HTML フォーム、CGI

□ PHP、フォームデータ受け取り、セッション管理

□ データベース、select 文による検索、PHP との接続

□ PHP セッションとデータベース、Web アプリと不正アクセス

□ クライアント側コード、JavaScript、DHMTML、Ajax