

データベース

第11回 PHPによるWebインターフェース 開発演習(2)

九州工業大学 情報工学部 尾下真樹

今日の内容

- 前回の復習
- PHPによるインターフェース開発(2)
 - データの追加
 - データの削除・更新・検索
 - 注意点
 - 演習課題
- レポート課題(後半)

演習資料

- 演習資料(データベース演習資料(3))
 - この資料に従って、今回の演習を進める
- プログラムリスト(印刷用)
 - 本演習のためのサンプルプログラムをまとめて印刷用のPDFにしたもの(行番号付き)
 - 授業中のプログラムの解説では、本資料の行番号を参照しながら説明する

前回の復習

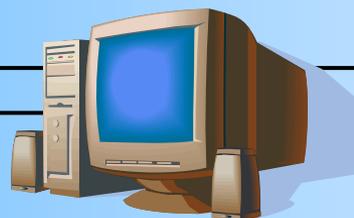
Webインターフェース

- Webページを経由してデータベースを操作

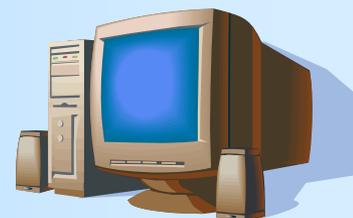
利用者



Webサーバ



データベースサーバ



操作



結果



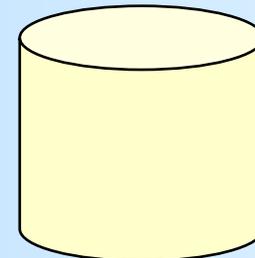
SQLを使ったコマンド
ライン環境での操作



Webブラウザによる
GUI環境での操作
(データベースを意識
しなくても使える)

HTML
(+スクリプト)

HTML中にスクリプトを
記述することで、データ
ベースにアクセス



データを管理
コマンドラインインター
フェース

HTMLファイルの例

- メニュー(menu.html)

```
<HTML>
<HEAD>
  <TITLE>データ操作メニュー</TITLE>
</HEAD>
<BODY>
  操作メニュー<BR>
  <UL>
    <LI><A HREF="employee_list.php">従業員の一覧表示</A>
    <LI><A HREF="employee_add_form.html">従業員のデータ追加</A>
    <LI><A HREF="employee_add_form.php">従業員のデータ追加(動的生成版)</A>
    <LI><A HREF="employee_delete_form.html">従業員のデータ削除</A>
    <LI><A HREF="employee_delete_form.php">従業員のデータ削除(動的生成版)</A>
    <LI><A HREF="employee_update_form1.html">従業員のデータ更新</A>
  </UL>
</BODY>
</HTML>
```

HTMLファイルの表示結果の例

- ウェブブラウザでの表示結果
 - フォントの種類や大きさ等は、ブラウザの設定により異なる

操作メニュー

- [従業員の一覧表示](#)
- [従業員データの追加](#)
- [従業員データの追加\(動的生成版\)](#)
- [従業員データの削除](#)
- [従業員データの削除\(動的生成版\)](#)
- [従業員データの更新](#)

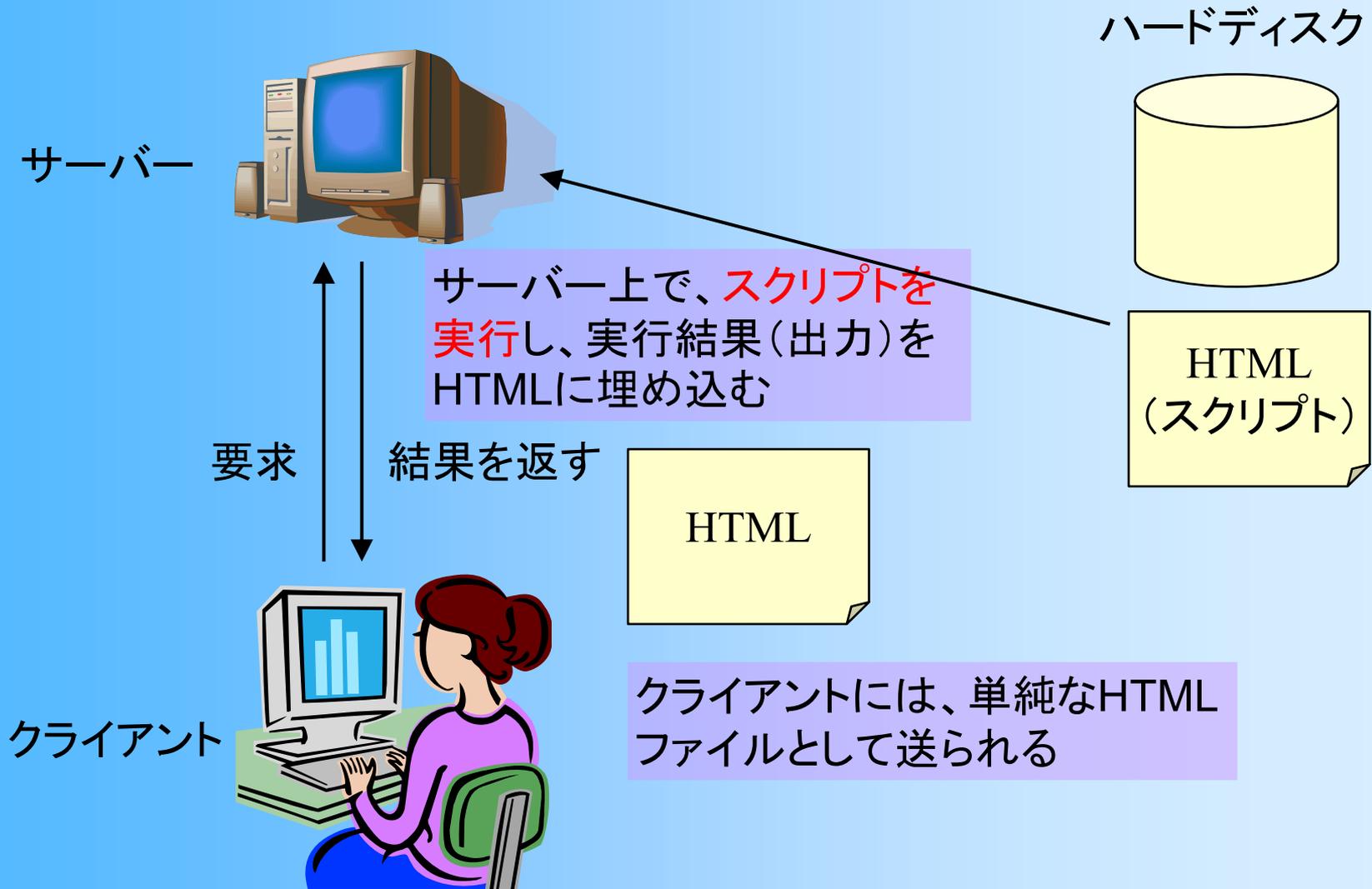
PHPの利用

- PHP
 - サーバーサイド・スクリプトとしての利用に適したプログラミング言語
 - JavaやC++と同様のオブジェクト指向言語
 - Webシステムの開発に便利な標準関数を備える
 - 多くのWebシステムで使われている（WordPress等）
- 本演習では、PHPを使って、Webインターフェースを作成する

PHPの利用形態

- サーバーサイド・スクリプトとしての利用
 - HTML と PHPスクリプト が混在したソースファイルを記述
 - ウェブサーバ側で PHPスクリプトを実行
 - PHPスクリプトから出力されたテキストが、元のHTMLに埋め込まれる
 - クライアント(ウェブブラウザ)には、最終的に生成された HTML が送られる

サーバサイド・スクリプト



PHPの記述(1)

- HTML内へのPHPスクリプトの記述
 - `<?php ~ ?>`
- PHPの文法
 - if や while などの制御構文は、C や Java と同じ
- 変数
 - \$で始まる文字列を変数とみなす
 - 宣言せずに使って良い
 - 型は指定しなくても良い(値により自動的に決まる)
 - C や Java とは、上記の点が大きく異なるので注意

PHPからPostgreSQLの操作

- 専用の関数が用意されている

- pg_ で始まる関数

- pg_connect(option);

- データベースに接続

- pg_query(connection, query);

- クエリーを実行

- pg_num_rows(result);

- クエリーの結果の行数を取得

- pg_fetch_result(result, i, j);

- クエリーの結果のテーブルから i行j列の値を取得

- i,j は 0 から始まることに注意(例:2行3列目→ i=1, j=2)

- pg_close();

従業員番号	部門番号	氏名	年齢
0001	01	織田 信長	48
0002	02	豊臣 秀吉	45
0003	03	徳川 家康	39
0004	01	柴田 勝家	60
0005	01	伊達 政宗	15
0006	02	上杉 景勝	26
0007	03	島津 家久	35

SQL文の作成

- SQL文は文字列として扱える

- `$sql = "select * from employee where id='001'";`

- 注意: " (ダブルクォート) はPHPの文字列の区切り、
' (シングルクォート) はSQLの文字列の区切り

- 文字列を埋め込むことで動的にSQL文を作成できる (以下の3つは、どれも同じ結果になる)

- `$sql = "select * from employee where id=" . $id . "'";`

- 文字列の連結

- `$sql = "select * from employee where id='$id'";`

- \$id の値が文字列に埋め込まれる

- `$sql = sprintf("select * from employee where id='%s'", $id);`

- 文字列中の %s の箇所が、\$id の値で置き換えられる

ウェブページの準備

- ウェブサーバ

- <http://popuradb.ces.kyutech.ac.jp>

- 今回はデータベースサーバと同じコンピュータ

- ※ 大学外からはアクセスできないので注意

- ウェブサーバにファイルを置く

- ホームディレクトリの下に public_html

- クライアント端末からファイルを転送

- 以下のURLでアクセスできる

- <http://popuradb.ces.kyutech.ac.jp/~ユーザ名/ファイル名>

全体の演習手順

- データベースの準備
 1. テーブルの作成、データの追加(前回終了)
 2. テーブルの利用権限の設定
- HTML+PHP ファイルの作成
 3. 講義のページからダウンロードした `menu.html` を適切な場所に置き、表示されることを確認
 4. 同じく `employee_list.php` を置き(一部修正が必要)、従業員一覧が表示されることを確認
 5. 他のファイル(追加、更新、削除)についても、動作を確認(次回の演習)

演習課題

- 前回までの演習は完了しているものとする
- メニュー・一覧表示 (menu.html, employee_list.php) のファイルをアップロード (+一部変更) して、動作確認をする
- 一覧表示を行なうPHPプログラムを一部変更して、従業員の一覧が 部門ごとに表示 されるようにする (exemployee_list.php を変更)

PHPによるインターフェース開発(1)

(前回の復習)

インターフェースの作成

- 作成する機能

- 従業員データの一覧表示
- 従業員データの追加
- 従業員データの追加(動的生成)
- 従業員データの削除
- 従業員データの削除(動的生成)
- 従業員データの更新
- 従業員データの更新(動的生成)
- 従業員データの検索

サンプルページの構成

黒字は html、
赤字は php を
表す

- メニュー(menu.html)
 - 従業員の一覧表示(employee_list.php)
 - 従業員のデータ追加 入力フォーム(exmployee_add.html)
 - 追加処理(employee_add.php)
 - 従業員のデータ追加 入力フォーム(動的生成版)(exmployee_add_form.php)
 - 追加処理(employee_add.php)
 - 従業員のデータ削除 選択フォーム(employee_delete.html)
 - 削除処理(employee_delete.php)
 - 従業員のデータ削除 選択フォーム(動的生成版)(employee_delete_form.php)
 - 削除処理(employee_delete.php)
 - 従業員のデータ更新 選択フォーム(employee_update_form1.html)
 - 更新フォーム(employee_update_form2.php)
 - 更新処理(employee_update.php)
 - 従業員のデータ更新 選択フォーム(動的生成版)(employee_update_form1.php)
 - 更新フォーム(employee_update_form2.php)
 - 更新処理(employee_update.php)
 - 検索フォーム(動的生成)(employee_search_form.php)
 - 検索処理(employee_search.php)

サンプルページの構成

- **メニュー(menu.html)**

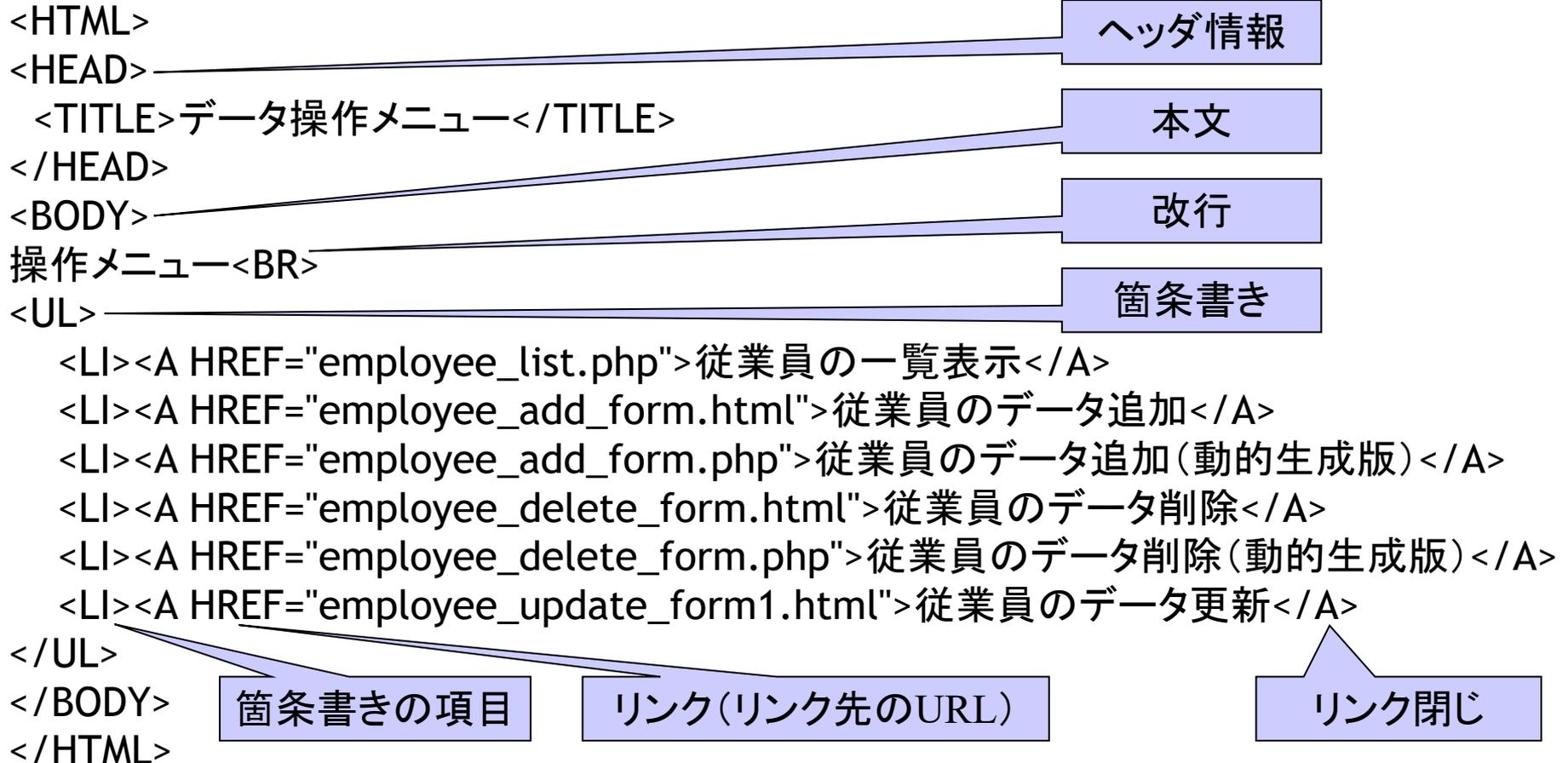
- 従業員の一覧表示(employee_list.php)
- 従業員のデータ追加 入力フォーム(exemployee_add.html)
 - 追加処理(employee_add.php)
- 従業員のデータ追加 入力フォーム(動的生成版)(exemployee_add_form.php)
 - 追加処理(employee_add.php)
- 従業員のデータ削除 選択フォーム(employee_delete.html)
 - 削除処理(employee_delete.php)
- 従業員のデータ削除 選択フォーム(動的生成版)(employee_delete_form.php)
 - 削除処理(employee_delete.php)
- 従業員のデータ更新 選択フォーム(employee_update_form1.html)
 - 更新フォーム(employee_update_form2.php)
 - 更新処理(employee_update.php)
- 従業員のデータ更新 選択フォーム(動的生成版)(employee_update_form1.php)
 - 更新フォーム(employee_update_form2.php)
 - 更新処理(employee_update.php)
- 検索フォーム(動的生成)(employee_search_form.php)
 - 検索処理(employee_search.php)

メニュー

- メニュー (menu.html)
 - `<HTML> <HEAD> <TITLE> <BYDY>`
 - ` ~ ` によるリスト
 - 各機能のページへのリンク
` ~ `

メニュー

• メニュー(menu.html)



表示結果

- ウェブブラウザでの表示結果
 - フォントの種類や大きさ等は、ブラウザの設定により異なる

操作メニュー

- [従業員の一覧表示](#)
- [従業員データの追加](#)
- [従業員データの追加\(動的生成版\)](#)
- [従業員データの削除](#)
- [従業員データの削除\(動的生成版\)](#)
- [従業員データの更新](#)

サンプルページの構成

- メニュー(menu.html)
 - 従業員の一覧表示(employee_list.php)
 - 従業員のデータ追加 入力フォーム(exemployee_add.html)
 - 追加処理(employee_add.php)
 - 従業員のデータ追加 入力フォーム(動的生成版)(exemployee_add_form.php)
 - 追加処理(employee_add.php)
 - 従業員のデータ削除 選択フォーム(employee_delete.html)
 - 削除処理(employee_delete.php)
 - 従業員のデータ削除 選択フォーム(動的生成版)(employee_delete_form.php)
 - 削除処理(employee_delete.php)
 - 従業員のデータ更新 選択フォーム(employee_update_form1.html)
 - 更新フォーム(employee_update_form2.php)
 - 更新処理(employee_update.php)
 - 従業員のデータ更新 選択フォーム(動的生成版)(employee_update_form1.php)
 - 更新フォーム(employee_update_form2.php)
 - 更新処理(employee_update.php)
 - 検索フォーム(動的生成)(employee_search_form.php)
 - 検索処理(employee_search.php)

一覧表示(1)

- 一覧表示(exmployee_list.php)
 - PHPプログラムの開始(13行目)
 - データベースへの接続(17行目)
 - データベース名を、各自の名前に変更する必要がある(前回の資料の通りに作業していれば、自分のアカウント名でデータベースを作成しているはず)
 - 接続情報を \$conn に記録

一覧表示(2)

```
<HTML>
<HEAD>
  <TITLE>従業員リスト</TITLE>
</HEAD>
<BODY>
<CENTER>
検索結果を表示します。<BR><BR>
```

```
<!-- ここからPHPのSCRIPT始まり -->
```

```
<?php
```

```
// データベースに接続
```

```
// ※ your_db_name のところは自分のデータベース名に書き換える
```

```
$conn = pg_connect( "dbname=your_db_name" );
```

```
// 接続が成功したかどうか確認
```

```
if ( $conn == null )
```

接続情報が返される(失敗時はnull)

```
    print( "データベース接続処理でエラーが発生しました。<BR>" );
    exit;
}
```

PHPスクリプトの開始

PostgreSQLデータベースへの接続を行う、PHPの関数

データベース名を指定
(自分のデータベース名に
書き換える)

一覧表示(3)

- 一覧表示(exmployee_list.php)
 - SQL文を作成・実行(27, 30行目)
 - 全従業員データの取得するSQL文(\$sqlに格納)
 - 検索結果が \$result に格納される

```
// SQLを作成
$sql = "select id, department.name, employee.name, age from employee,
department where employee.dept_no = department.dept_no order by id";

// Queryを実行して検索結果をresultに格納
$result = pg_query( $conn, $sql );
if ( $result == null )
{
    print( "クエリー実行処理でエラーが発生しました。<BR>" );
    exit;
}
```

一覧表示(4)

- 一覧表示(exmployee_list.php)
 - 検索結果の行数・列数を取得(38, 39行目)
 - SQL文で4つの出力属性を指定しているため、列数は必ず4になる(わざわざ列数を取得しなくても分かっているが、今回は例のために、あえて取得している)

```
// 検索結果の行数・列数を取得  
$rows = pg_num_rows( $result );  
$cols = pg_num_fields( $result );
```

引数には、さきほどのSQLの実行結果を格納した変数を指定

SQLの実行結果から、行数(データ数)と列数(属性数)を取得するPHPの関数

一覧表示(5)

- 一覧表示(exmployee_list.php)
 - テーブルを使って結果を表示(43～70行目)
 - <TABLE> <TR> <TD>
 - 各データ(検索結果の各行)の情報を順番に表示(54～66行目)
 - for 文を使って、各行・列ごとに繰り返し
 - 検索結果から属性値を取得して表示(60行目)
 - pg_fetch_result(結果, 行番号, 列番号)

一覧表示(6)

```
// 検索結果をテーブルとして表示  
print( "<TABLE BORDER=1>¥n" );
```

テーブルの開始

```
// 各列の名前を表示  
print( "<TR>" );  
print( "<TH>従業員番号</TH>" );  
print( "<TH>部門</TH>" );  
print( "<TH>氏名</TH>" );  
print( "<TH>年齢</TH>" );  
print( "</TR>¥n" );
```

1行目の見出しの表示

.....

検索結果を表示します。

従業員番号	部門	氏名	年齢
0001	開発	織田 信長	48
0002	営業	豊臣 秀吉	45
0003	総務	徳川 家康	39
0004	開発	柴田 勝家	60
0005	営業	伊達 政宗	15
0006	総務	上杉 景勝	26
0007	開発	島津 家久	35

以上、7件のデータを表示しました。

[操作メニューに戻る](#)

表示されるテーブル

一覧表示(7)

```
// 各行のデータを表示
for ( $j=0; $j<$rows; $j++ )
{
    print( "<TR>" );
    for ( $i=0; $i<$cols; $i++ )
    {
        // j行i列のデータを取得
        $data = pg_fetch_result( $result, $j, $i );

        // テーブルのj行i列に属性値を表示
        print( "<TD> $data </TD>" );
    }
    print( "</TR>\n" );
}
}
```

テーブルの各行ごとに繰り返し

各行全体を <TR> タグで囲む

各列ごとに繰り返し

// j行i列のデータを取得
\$data = pg_fetch_result(\$result, \$j, \$i);

SQLの実行結果からj行i列の属性値を取得するPHPの関数

// テーブルのj行i列に属性値を表示
print("<TD> \$data </TD>");

各セルを <TD> タグで囲む

```
// ここまででテーブル終了
print( "</TABLE>" );
print( "<BR>\n" );
```

変数 \$data の値を表示

一覧表示(8)

- 一覧表示(exmployee_list.php)
 - データ数を表示(74行目)
 - 終了処理(78, 81行目)
 - 検索結果の開放
 - データベースへの接続を解除

```
// 検索性数を表示  
print( "以上、$rows 件のデータを表示しました。<BR>¥n" );
```

```
// 検索結果の開放  
pg_free_result( $result );
```

```
// データベースへの接続を解除  
pg_close( $conn );
```

文字列の中に、変数 \$rows の値
が埋め込まれて出力される

実行結果の例

```
<HTML>
<HEAD>
  <TITLE>従業員リスト</TITLE>
</HEAD>
<BODY>
<CENTER>
検索結果を表示します。<BR><BR>
<!-- ここからPHPのSCRIPT始まり -->
<TABLE BORDER=1>
<TR><TH>従業員番号</TH><TH>部門</TH><TH>氏名</TH><TH>年齢</TH></TR>
<TR><TD> 0001 </TD><TD> 開発 </TD><TD> 織田 信長 </TD><TD> 48 </TD></TR>
<TR><TD> 0002 </TD><TD> 営業 </TD><TD> 豊臣 秀吉 </TD><TD> 45 </TD></TR>
<TR><TD> 0003 </TD><TD> 総務 </TD><TD> 徳川 家康 </TD><TD> 39 </TD></TR>
.....
</TABLE><BR>
以上、7 件のデータを表示しました。<BR>
<!-- ここまででPHPのSCRIPT終わり -->
<BR>
<A HREF="menu.html">操作メニューに戻る</A>
</CENTER>
```

テーブル(表)の開始

表の一行(<TR>タグ)

表の要素(<TD>or<TH>タグ)

テーブル(表)の終了

データ数の出力

従業員番号	部門	氏名	年齢
0001	開発	織田 信長	48
0002	営業	豊臣 秀吉	45
0003	総務	徳川 家康	39

表示結果の例

- ウェブブラウザでの表示結果

検索結果を表示します。

従業員番号	部門	氏名	年齢
0001	開発	織田 信長	48
0002	営業	豊臣 秀吉	45
0003	総務	徳川 家康	39
0004	開発	柴田 勝家	60
0005	営業	伊達 政宗	15
0006	総務	上杉 景勝	26
0007	開発	島津 家久	35

テーブル(表)として
表示される

以上、7件のデータを表示しました。

[操作メニューに戻る](#)

PHPによるインターフェース開発(2)

インターフェースの作成

- 作成する機能

- 従業員データの一覧表示
- 従業員データの追加
- 従業員データの追加(動的生成)
- 従業員データの削除
- 従業員データの削除(動的生成)
- 従業員データの更新
- 従業員データの更新(動的生成)
- 従業員データの検索

操作メニュー

- 従業員の一覧表示
- 従業員データの追加
- 従業員データの追加 (動的生成版)
- 従業員データの削除
- 従業員データの削除 (動的生成版)
- 従業員データの更新
- 従業員データの更新 (動的生成版)
- 従業員の検索 (部門名での検索)

PHPによるインターフェース開発
(デモ動画)

サンプルページの構成

- メニュー(menu.html)
 - 従業員の一覧表示(employee_list.php)
 - 従業員のデータ追加 入力フォーム(exmployee_add.html)
 - 追加処理(employee_add.php)
 - 従業員のデータ追加 入力フォーム(動的生成版)(exmployee_add_form.php)
 - 追加処理(employee_add.php)
 - 従業員のデータ削除 選択フォーム(employee_delete.html)
 - 削除処理(employee_delete.php)
 - 従業員のデータ削除 選択フォーム(動的生成版)(employee_delete_form.php)
 - 削除処理(employee_delete.php)
 - 従業員のデータ更新 選択フォーム(employee_update_form1.html)
 - 更新フォーム(employee_update_form2.php)
 - 更新処理(employee_update.php)
 - 従業員のデータ更新 選択フォーム(動的生成版)(employee_update_form1.php)
 - 更新フォーム(employee_update_form2.php)
 - 更新処理(employee_update.php)
 - 検索フォーム(動的生成)(employee_search_form.php)
 - 検索処理(employee_search.php)

追加

- 2つのページにより実現される

1. 追加フォーム (exemployee_add.html)

- HTMLのフォームを使ってデータを入力できるようにする
- 各データの変数名を指定(次のページでデータを受け取るために必要)

2. 追加処理 (exemployee_add.php)

- 前のページで入力されたデータをもとに、データ追加のためのSQL文を作成し、実行

フォーム

- ウェブページに入力できる仕組み
 - <FORM> ~ </FORM>
 - 送信ボタンを押すと、指定したURLを呼び出し
 - フォーム記入したデータをURLで指定したプログラムに引数として送信できる
 - データの受け渡し方に、GET と POST の2種類がある

従業員データ 追加フォーム

従業員番号: 部門番号: 氏名: 年齢: 性別: 男 女

送信

引数の受け渡し方の違い

- GET

- URLに付与する形で受け渡し(利用者に引数が見える)
- サーバー側のプログラムは、環境変数という仕組みを使って受け取る

- POST

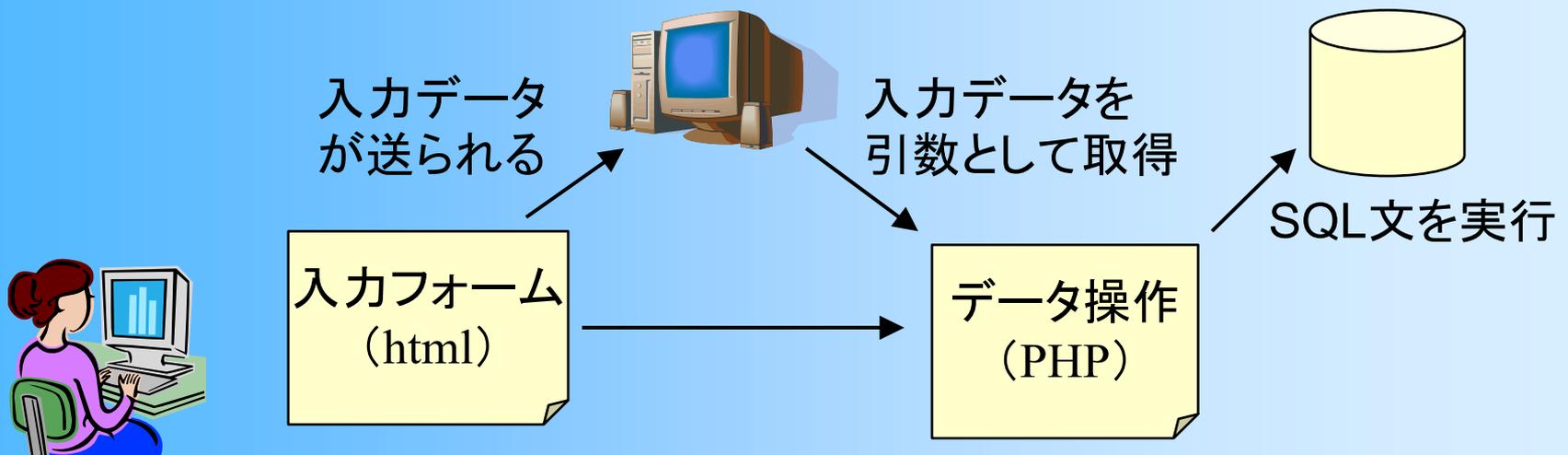
- ウェブブラウザとサーバーが通信をして受け渡し(利用者には引数が見えない)

- 両者の使い分け

- 一般に、POSTの方が処理が面倒な代わりに、高機能
- PHPで使う場合は、PHPが細かい処理をやってくれるので、どちらも簡単に使える
- 本演習では、引数が分かりやすいように、GETを使用

PHPでの引数の受け取り

- スーパーグローバル変数経由で取得
 - フォーム側が GET で出力した場合
 - `$_GET[引数名]`
 - フォーム側が POST で出力した場合
 - `$_POST[引数名]`



追加フォーム

- 追加フォーム (exmployee_add.html)
 - フォームの開始(9行目)
 - `<FORM ACTION="～" METHOD="～" >`
 - 各入力フィールド(12～28行目)
 - `<INPUT TYPE ="～" NAME="変数名" >`

従業員データ 追加フォーム

従業員番号: 部門番号: 氏名: 年齢: 性別: 男 女

追加フォーム

フォームの開始

データ送信後に実行されるページ

.....
従業員データ追加フォーム

データの受け渡し方法に GET を使用

<FORM ACTION="employee_add.php" METHOD="GET">

従業員番号:

<INPUT TYPE="text" SIZE="4" NAME="id">

テキスト入力エリアを表示
(4文字分のスペースを用意)

部門番号:

<INPUT TYPE="text" SIZE="4" NAME="dept_no">

入力されたデータは、id という
名前で、実行ページに渡される

氏名:

<INPUT TYPE="text" SIZE="24" NAME="name">

年齢:

<INPUT TYPE="text" SIZE="4">

選択項目(ラジオボタン)。この項目が選択されると、
sex という名前のデータに、文字列 MALE が入る。

性別:

<INPUT TYPE="radio" NAME="sex" VALUE="MALE" CHECKED>男</INPUT>

<INPUT TYPE="radio" NAME="sex" VALUE="FEMALE">女</INPUT>

<INPUT TYPE="submit" VALUE="送信">

</FORM>

.....

送信ボタンを表示

追加処理(1)

- 追加処理(exemployee_add.php)
 - データベースへの接続などは、一覧表示と同じ(説明は省略)
 - フォームから渡された引数を取得(11~14行目)
 - \$_GET[変数名]
 - 取得データを変数に格納 \$id, \$dept_no, \$name, \$age

```
// フォームから渡された引数を取得
$id = $_GET[ id ];
$dept_no = $_GET[ dept_no ];
$name = $_GET[ name ];
$age = $_GET[ age ];
```

\$_GET は、PHPが持つグローバル変数(配列) 前のページのフォームから GET を使って渡された 値を受け取ることができる

追加処理(2)

- 追加処理(exemployee_add.php)
 - データ追加のためのSQL文を作成(28行目)
 - ここでは、sprintfを使う方法を使用(前回の講義で説明したように、別の方法を使っても構わない)
 - SQLの実行(31~33行目)

```
// データ挿入のSQLを作成
$sql = sprintf( "insert into employee( id, dept_no, name, age ) values( '%s',
'%s', '%s', '%s' );", $id, $dept_no, $name, $age );
```

```
// 確認用のメッセージ表示
```

```
print( "クエリー「" ); print( $sql ); print( "」を実行します。<BR>" );
```

```
// Queryを実行して検索結果をresultに格納
```

```
$result = pg_query( $conn, $sql );
```

4つの %s は、次の4つの引数の値に置き換えられる

サンプルページの構成

- メニュー(menu.html)
 - 従業員の一覧表示(employee_list.php)
 - 従業員のデータ追加 入力フォーム(exmployee_add.html)
 - 追加処理(employee_add.php)
 - 従業員のデータ追加 入力フォーム(動的生成版)(exmployee_add_form.php)
 - 追加処理(employee_add.php)
 - 従業員のデータ削除 選択フォーム(employee_delete.html)
 - 削除処理(employee_delete.php)
 - 従業員のデータ削除 選択フォーム(動的生成版)(employee_delete_form.php)
 - 削除処理(employee_delete.php)
 - 従業員のデータ更新 選択フォーム(employee_update_form1.html)
 - 更新フォーム(employee_update_form2.php)
 - 更新処理(employee_update.php)
 - 従業員のデータ更新 選択フォーム(動的生成版)(employee_update_form1.php)
 - 更新フォーム(employee_update_form2.php)
 - 更新処理(employee_update.php)
 - 検索フォーム(動的生成)(employee_search_form.php)
 - 検索処理(employee_search.php)

追加フォームの動的生成

- 全てのデータを入力するのは大変、また、一部のデータは入力可能なデータに限られる
 - 例えば、部門番号には、外部参照整合性制約があるので、存在しない部門番号は入力不可能

従業員データ 追加フォーム

従業員番号: 部門番号: 氏名: 年齢: 性別: 男 女

送信

- 適切な初期値や選択肢を表示することで、入力を簡便化したり、不適切なデータが入力されることを防止したりできる

表示結果の例

- ウェブブラウザでの表示結果

従業員データ 追加フォーム

従業員番号:

部門: 開発 営業 総務

氏名: 年齢: 性別: 男 女

次の空いている従業員番号を自動的に初期値とする (4桁の数字)

部門を選択肢から選べるようにする (選択肢は部門テーブルに応じて動的に生成される)

他の項目は、変更なし

追加フォームの動的生成(1)

- 追加フォーム2 (exemployee_add_from.php)
 - html ではなく php である点に注目
 - phpスクリプトを使って動的にフォームを生成する
 - 従業員番号の初期値を取得(26~48行目)
 - 最大の従業員番号 +1 を変数 \$max_id に格納

```
// 最も大きな従業員番号を取り出すSQLの作成
```

```
$sql = "select max(id) from employee";
```

```
// Queryを実行して検索結果をresultに格納
```

```
$result = pg_query( $conn, $sql );
```

```
// 最大の従業員番号を取得
```

```
if ( pg_num_rows( $result ) > 0 )
```

```
    $max_id = pg_fetch_result( $result, 0, 0 );
```

```
$max_id ++;
```

SQLの出力の属性値を取得
(出力は必ず1行1列のテーブルになる)

+1 加算

追加フォームの動的生成(2)

- 追加フォーム2 (exemployee_add_from.php)
 - 従業員番号の初期値を設定する(44行目)
(フォームを表示したときに、自動的に初期値が表示される)

```
// 従業員番号の初期値を指定して入力エリアを作成  
print( "従業員番号:¥n" );  
printf( "<INPUT TYPE=text SIZE=4 NAME=id VALUE=%04s>", $max_id );  
print( "<BR>¥n" );
```

%04s が、次の引数の \$max_id で置き換えられる
(%04s の 04 は、変数の値が3桁以下であれば、左に
0 を加えて 4 桁の数字として表示することを表す)

追加フォームの動的生成(3)

- 追加フォーム2 (exmployee_add_from.php)
 - 部門の一覧を取得し、選択肢として表示(62～75行目)
 - 残りの項目には変更はないので、php スクリプトが終わった後に html として記述(85～96行目)

追加フォームの動的生成(4)

```
// 部門一覧を取得するSQLの作成
$sql = "select dept_no, name from department";

// Queryを実行して検索結果をresultに格納
$result = pg_query( $conn, $sql );

// 検索結果の行数を取得
$rows = pg_num_rows( $result );

// 部門の数だけ選択肢を出力
print( "部門:¥n" );
for ( $i=0; $i<$rows; $i++ )
{
    $dept_no = pg_fetch_result( $result, $i, 0 );
    $dept_name = pg_fetch_result( $result, $i, 1 );
    printf( "<INPUT TYPE=¥"radio¥" NAME=¥"dept_no¥"
            VALUE=¥"%s¥"> %s </INPUT>¥n", $dept_no, $dept_name );
}

```

部門番号(\$dept_no)、部門名(\$dept_name)を取得

"(ダブルクォート)を出力するためには、前に ¥ をつける必要がある

選択肢が選択されたときに、部門番号(\$dept_no)を次のページに渡す値とする

選択肢として部門名(\$dept_name)を表示

実行結果の例

.....

従業員データ 追加フォーム


```
<FORM ACTION="employee_add.php" METHOD="GET">
```

従業員番号の初期値を設定

```
<!-- ここからPHPのスクリプト始まり -->
```

従業員番号:

```
<INPUT TYPE="text" SIZE=4 NAME=id VALUE=0008><BR>
```

部門の選択肢を表示

部門:

```
<INPUT TYPE="radio" NAME="dept_no" VALUE="01"> 開発 </INPUT>
```

```
<INPUT TYPE="radio" NAME="dept_no" VALUE="02"> 営業 </INPUT>
```

```
<INPUT TYPE="radio" NAME="dept_no" VALUE="03"> 総務 </INPUT>
```

表示される部門名

```
<!-- ここまででPHPのスクリプト終わり -->
```

氏名:

以降は、html をそのまま使用

```
<INPUT TYPE="text" SIZE="24" NAME="name">
```

年齢:

```
<INPUT TYPE="text" SIZE="4" NAME="age">
```

部門を選択したときに、
dept_no の値として次に
渡される部門番号

.....

表示結果の例

- ウェブブラウザでの表示結果

従業員データ 追加フォーム

従業員番号:

部門: 開発 営業 総務

氏名: 年齢: 性別: 男 女

サンプルページの構成

- メニュー(menu.html)
 - 従業員の一覧表示(employee_list.php)
 - 従業員のデータ追加 入力フォーム(exmployee_add.html)
 - 追加処理(employee_add.php)
 - 従業員のデータ追加 入力フォーム(動的生成版)(exmployee_add_form.php)
 - 追加処理(employee_add.php)
 - 従業員のデータ削除 選択フォーム(employee_delete.html)
 - 削除処理(employee_delete.php)
 - 従業員のデータ削除 選択フォーム(動的生成版)(employee_delete_form.php)
 - 削除処理(employee_delete.php)
 - 従業員のデータ更新 選択フォーム(employee_update_form1.html)
 - 更新フォーム(employee_update_form2.php)
 - 更新処理(employee_update.php)
 - 従業員のデータ更新 選択フォーム(動的生成版)(employee_update_form1.php)
 - 更新フォーム(employee_update_form2.php)
 - 更新処理(employee_update.php)
 - 検索フォーム(動的生成)(employee_search_form.php)
 - 検索処理(employee_search.php)

削除

- 選択フォーム (employee_delete_form.html)
 - 削除する従業員の従業員番号を入力
- 削除処理 (employee_delete.php)
 - 指定された従業員番号のデータを削除
(DELETE構文 → 各自記述)
- プログラムの中身は、これまでと同じなので、説明は省略

サンプルページの構成

- メニュー(menu.html)
 - 従業員の一覧表示(employee_list.php)
 - 従業員のデータ追加 入力フォーム(exmployee_add.html)
 - 追加処理(employee_add.php)
 - 従業員のデータ追加 入力フォーム(動的生成版)(exmployee_add_form.php)
 - 追加処理(employee_add.php)
 - 従業員のデータ削除 選択フォーム(employee_delete.html)
 - 削除処理(employee_delete.php)
 - 従業員のデータ削除 選択フォーム(動的生成版)(employee_delete_form.php)
 - 削除処理(employee_delete.php)
 - 従業員のデータ更新 選択フォーム(employee_update_form1.html)
 - 更新フォーム(employee_update_form2.php)
 - 更新処理(employee_update.php)
 - 従業員のデータ更新 選択フォーム(動的生成版)(employee_update_form1.php)
 - 更新フォーム(employee_update_form2.php)
 - 更新処理(employee_update.php)
 - 検索フォーム(動的生成)(employee_search_form.php)
 - 検索処理(employee_search.php)

削除フォームの動的生成

- 選択フォーム(動的生成版)
(`exmployee_delete_form.php`)
 - 従業員の一覧表示 + 削除する従業員の選択ボタンの表示(64行目)
- 削除する対象を一覧から選べるようにすることで、使いやすくなる
- プログラムの実現方法は、これまでの方法の組み合わせなので、説明は省略

表示結果の例

- ウェブブラウザでの表示結果

従業員データ 削除フォーム

削除したい従業員を選択して送信ボタンを押してください。

従業員番号	部門	氏名	年齢
<input type="radio"/> 0001	開発	織田 信長	48
<input type="radio"/> 0002	営業	豊臣 秀吉	45
<input type="radio"/> 0003	総務	徳川 家康	39
<input type="radio"/> 0004	開発	柴田 勝家	60
<input type="radio"/> 0005	営業	伊達 政宗	15
<input checked="" type="radio"/> 0006	開発	上杉 景勝	26

以上、6人の従業員が登録されています。

[操作メニューに戻る](#)

削除する従業員を、ラジオボタンで選択可能

サンプルページの構成

- メニュー(menu.html)
 - 従業員の一覧表示(employee_list.php)
 - 従業員のデータ追加 入力フォーム(exmployee_add.html)
 - 追加処理(employee_add.php)
 - 従業員のデータ追加 入力フォーム(動的生成版)(exmployee_add_form.php)
 - 追加処理(employee_add.php)
 - 従業員のデータ削除 選択フォーム(employee_delete.html)
 - 削除処理(employee_delete.php)
 - 従業員のデータ削除 選択フォーム(動的生成版)(employee_delete_form.php)
 - 削除処理(employee_delete.php)
 - 従業員のデータ更新 選択フォーム(employee_update_form1.html)
 - 更新フォーム(employee_update_form2.php)
 - 更新処理(employee_update.php)
 - 従業員のデータ更新 選択フォーム(動的生成版)(employee_update_form1.php)
 - 更新フォーム(employee_update_form2.php)
 - 更新処理(employee_update.php)
 - 検索フォーム(動的生成)(employee_search_form.php)
 - 検索処理(employee_search.php)

更新

- 3段階の処理になる
 1. 選択フォーム (employee_update_form1.html)
 - どの従業員のデータを更新するかを指定
 2. 更新フォーム (employee_update_form2.php)
 - 指定された従業員の現在の属性値を表示し、修正するためのフォームを表示
 - 追加とは異なり、現在の属性値を初期値として表示する
 3. 更新処理 (employee_update.php)
 - データを受け取って更新処理 (UPDATE構文 → 各自記述)

以降の機能の作成は演習課題

1. 更新機能の拡張

- 対象の従業員を選択するフォームの動的生成
- 削除処理での従業員選択を参考に作成

2. 検索機能の追加

- 選択した部門に所属する従業員の一覧を表示
(「全ての部門」を選択すると、全従業員を表示)
- 実現方法はこれまでに学習したものと同様
- 演習資料を参考に各自作成すること

メニューの拡張

- メニュー(menu.html)に項目を追加する

```
<HTML>
<HEAD>
  <TITLE>データ操作メニュー</TITLE>
</HEAD>
<BODY>
  操作メニュー<BR>
  <UL>
    <LI><A HREF="employee_list.php">従業員の一覧表示</A>
    <LI><A HREF="employee_add_form.html">従業員のデータ追加</A>
    <LI><A HREF="employee_add_form.php">従業員のデータ追加(動的生成版)</A>
    <LI><A HREF="employee_delete_form.html">従業員のデータ削除</A>
    <LI><A HREF="employee_delete_form.php">従業員のデータ削除(動的生成版)</A>
    <LI><A HREF="employee_update_form1.html">従業員のデータ更新</A>
    <LI><A HREF="employee_update_form1.php">従業員のデータ更新(動的生成版)</A>
    <LI><A HREF="employee_search_form.php">従業員の検索(部門名での検索)</A>
  </UL>
</BODY>
```

サンプルページの構成

- メニュー(menu.html)
 - 従業員の一覧表示(employee_list.php)
 - 従業員のデータ追加 入力フォーム(exmployee_add.html)
 - 追加処理(employee_add.php)
 - 従業員のデータ追加 入力フォーム(動的生成版)(exmployee_add_form.php)
 - 追加処理(employee_add.php)
 - 従業員のデータ削除 選択フォーム(employee_delete.html)
 - 削除処理(employee_delete.php)
 - 従業員のデータ削除 選択フォーム(動的生成版)(employee_delete_form.php)
 - 削除処理(employee_delete.php)
 - 従業員のデータ更新 選択フォーム(employee_update_form1.html)
 - 更新フォーム(employee_update_form2.php)
 - 更新処理(employee_update.php)
 - 従業員のデータ更新 選択フォーム(動的生成版)(employee_update_form1.php)
 - 更新フォーム(employee_update_form2.php)
 - 更新処理(employee_update.php)
 - 検索フォーム(動的生成)(employee_search_form.php)
 - 検索処理(employee_search.php)

更新フォームの動的生成

- 削除と同様に、動的生成版を作成する
 1. 選択フォーム (exemployee_update_form1.php)
 - 従業員の一覧表示 + 更新する従業員の選択ボタンの表示
 - 以降の2つのページは、前と同じものを使用する
 2. 更新フォーム (exemployee_update_form2.php)
 3. 更新処理 (exemployee_update.php)

更新の実行例(1)

1. 選択フォームから更新する従業員を選択
 - この例では、従業員番号 0005 の「伊達 政宗」を選択

従業員データ更新フォーム

更新したい従業員を選択して送信ボタンを押してください。

従業員番号	部門	氏名	年齢
<input type="radio"/> 0001	開発	織田 信長	48
<input type="radio"/> 0002	営業	豊臣 秀吉	45
<input type="radio"/> 0003	総務	徳川 家康	39
<input type="radio"/> 0004	開発	柴田 勝家	60
<input checked="" type="radio"/> 0005	総務	伊達 政宗	15

以上、5人の従業員が登録されています。

[操作メニューに戻る](#)

更新の実行例(2)

2. 更新フォームへの入力

- 選択した従業員の現在の情報が表示される
- この例では、部門を「総務」から「営業」に変更して、送信する

従業員データ更新フォーム

部門： 開発 営業 総務

氏名： 年齢：

性別： 男 女



従業員データ更新フォーム

部門： 開発 営業 総務

氏名： 年齢：

性別： 男 女

更新の実行例(3)

3. 更新処理の実行

- 実行結果の表示
- 一覧表示で更新されていることを確認

クエリー「update employee set dept_no='02', name='伊達 政宗', age=15 where id='0005';」を実行します。
データの更新処理が完了しました。

[操作メニューに戻る](#)



検索結果を表示します。

従業員番号	部門	氏名	年齢
0001	開発	織田 信長	48
0002	営業	豊臣 秀吉	45
0003	総務	徳川 家康	39
0004	開発	柴田 勝家	60
0005	営業	伊達 政宗	15

以上、5件のデータを表示しました。

[操作メニューに戻る](#)

サンプルページの構成

- メニュー(menu.html)
 - 従業員の一覧表示(employee_list.php)
 - 従業員のデータ追加 入力フォーム(exemployee_add.html)
 - 追加処理(employee_add.php)
 - 従業員のデータ追加 入力フォーム(動的生成版)(exemployee_add_form.php)
 - 追加処理(employee_add.php)
 - 従業員のデータ削除 選択フォーム(employee_delete.html)
 - 削除処理(employee_delete.php)
 - 従業員のデータ削除 選択フォーム(動的生成版)(employee_delete_form.php)
 - 削除処理(employee_delete.php)
 - 従業員のデータ更新 選択フォーム(employee_update_form1.html)
 - 更新フォーム(employee_update_form2.php)
 - 更新処理(employee_update.php)
 - 従業員のデータ更新 選択フォーム(動的生成版)(employee_update_form1.php)
 - 更新フォーム(employee_update_form2.php)
 - 更新処理(employee_update.php)
 - 検索フォーム(動的生成)(employee_search_form.php)
 - 検索処理(employee_search.php)

検索

- 2段階の処理になる
 1. 検索フォーム (employee_search_form.php)
 - どの部門(部門名)の従業員を検索するかを指定
 - 部門名の選択肢を動的に生成
 2. 検索結果表示 (employee_search.php)
 - 選択された部門番号を受け取って、検索結果の従業員の一覧を表示
 - 全従業員の一覧表示のプログラムを拡張することで実現

検索フォームの動的生成

- 検索フォームの生成結果
 - 部門名の選択肢を動的に生成

従業員データ 検索フォーム

検索したい従業員の部門を選択して送信ボタンを押してください。

開発

営業

総務

全ての部門

[操作メニューに戻る](#)

参考：より高度な検索

- 複数の条件を組み合わせた検索
- 任意の複数の条件を組み合わせた検索
 - 例：部門と年齢の任意の条件の組み合わせ

従業員データ 検索フォーム

検索したい従業員の部門と年齢の範囲を選択して、送信ボタンを押してください。

部門：

- 開発
- 営業
- 総務
- 全ての部門

年齢：

下限： ～ 上限：

[操作メニューに戻る](#)

参考: 選択項目が多い場合の対処

- プルダウンメニュー

- 選択対象の項目数が多い場合に適している

- <SELECT>, <OPTION> タグを使用(詳細は省略)

従業員データ 検索フォーム

検索したい従業員の部門と年齢の範囲を選択して、送信ボタンを押してください。

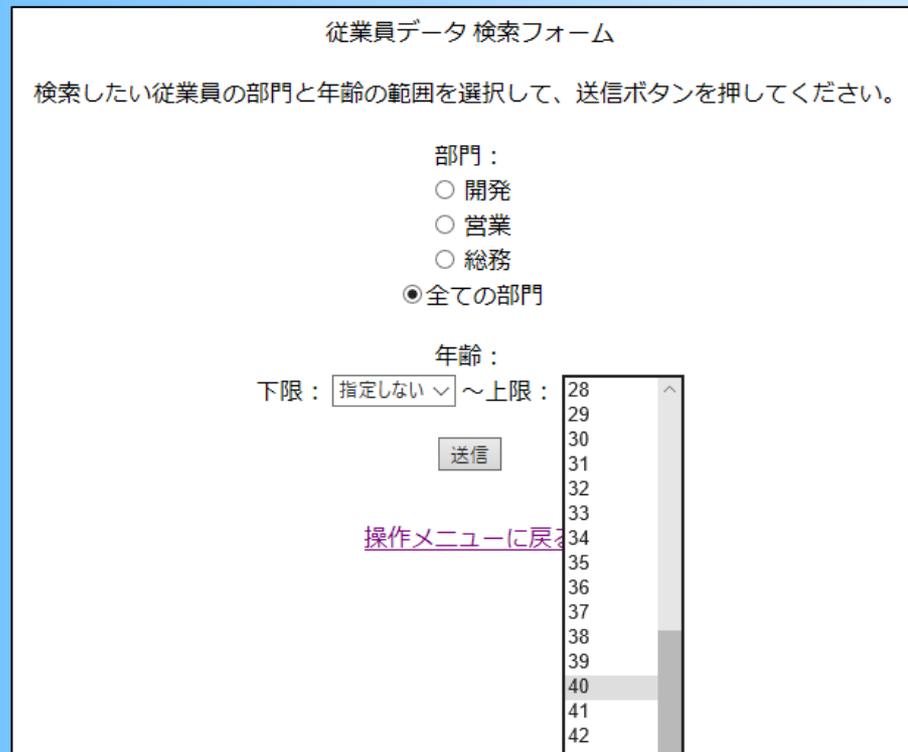
部門 :

- 開発
- 営業
- 総務
- 全ての部門

年齢 :

下限 : ~ 上限 :

[操作メニューに戻る](#)



参考：一度に複数の値を渡す方法

- 主キーが複数の属性になる場合の選択など
 - 例：販売（従業員番号，企業番号，価格）
 - フォームの一つの選択項目では、一つの値しか渡すことができない

従業員番号	部門	氏名	年齢
<input type="radio"/> 0001	開発	織田 信長	48
<input type="radio"/> 0002	営業	豊臣 秀吉	45
<input type="radio"/> 0003	総務	徳川 家康	39
<input type="radio"/> 0004	開発	柴田 勝家	60
<input type="radio"/> 0005	営業	伊達 政宗	
<input checked="" type="radio"/> 0006	開発	上杉 景勝	26

一つのラジオボタンで、主キーの属性の値を指定することで、対象を選択できる。主キーが複数の属性になるリレーションでは、対象を選択するために、複数の属性の値を渡す必要がある。

参考：一度に複数の値を渡す方法

- 主キーが複数の属性になる場合の選択など
 - 例：販売（従業員番号, 企業番号, 価格）
 - フォームの一つの選択項目では、一つの値しか渡すことができない
 - 主キーは一つの属性にすることが望ましい
 - スキーマ設計の時点で、重複のない通し番号の属性を追加
 - 例：販売（販売番号, 従業員番号, 企業番号, 価格）
 - 複数の値を一つの値に結合・分解する方法もある
 - 例：従業員番号（8桁の数字）＋企業番号（4桁の数字）を結合した文字列（12桁の数字）を値として渡し、受け取った文字列を従業員番号と企業番号に分解する

注意点

- サニタイズ
- 文字コード

サニタイズ

- 引数として受け取った文字列をSQL文に埋め込むときは、本来はサニタイズが必要
 - 悪意のある利用者が入力にSQL文を記述して実行すると、意図しない操作が実行されてしまう
 - 例: ... where \$id= " 001; delete from employee ";
(全データが削除される)
 - SQLインジェクションと呼ばれる、セキュリティホールになりうる
 - 本来は、入力をそのまま用いず、数値以外は取り除くなどの無害化処理(サニタイズ)が必要
 - 本講義の演習では、ここまでは扱わない

文字コード(1)

- 日本語の文字コードは、Shift-JIS、EUC、UTF8/16(ユニコード)など複数ある
 - 状況に応じて適切な文字コードの使用が必要
 - 場合によっては、文字コードの変換も必要
 - ソフトウェアによっては、複数の文字コードを用いることが可能
 - 自動的に判別・変換、もしくは、手動で切り替え可能
 - 例えば、ほとんどのウェブブラウザは、文字コードを自動判別して正しく表示
 - ただし、ひとつのファイルの中に、複数の文字コードを混在させてしまうと、確実に問題(文字化け)が発生する

文字コード(2)

- 本演習で使用する PostgreSQLサーバ、paql クライアントの文字コードは UTF8
 - 演習で作成する html や php ファイルも、UTF8 で記述する必要がある
 - データベースから取得するデータの文字コードは UTF8 なので、UTF8 で統一することが必要
 - データベースから取得したデータの文字コードを、PHPの関数を使って変換する方法もある(本講義では扱わない)
- 改行コードの違いにも注意
 - CR+LF (Windows)、CR (Mac)、LF (Unix)

演習課題(1)

前回までの演習は完了しているものとする

- 1.削除処理を行なうPHPプログラムに削除処理のためのSQLを追加し、削除が正しく動作するようにせよ(exmployee_delete.php)
- 2.更新処理を行なうPHPプログラムに更新処理のためのSQLを追加し、更新が正しく動作するようにせよ(exmployee_update.php)

演習課題(2)

3. 更新機能で、更新する従業員をリストから選択できるように拡張したものを作成し、正しく動作するようにせよ
(exemployee_update_form1.php)
4. 検索機能として、選択された部門の従業員の一覧を表示するSQLを追加し、検索が正しく動作するようにせよ
(exemployee_search_form.php,
exemployee_search.php)

演習課題の提出

- 演習課題のテキストファイルに回答を記述して、Moodleから提出
- 提出締め切りは、Moodleを参照（厳守）

レポート課題(後半)

レポート課題

- 自分で決めた何らかのテーマを題材にして、データベースとWebインターフェースを作成

1. データベースのスキーマの設計

- 全ての正規形を満たすように正規化を行う

2. データベースの作成

- テーブルの作成、データの追加

説明済み

3. データベースへの問い合わせ

- 問い合わせの具体例を考えてSQLを作成・実行

4. Webインターフェースの作成

- 一覧表示、追加・削除・更新、実用的な検索

今回説明

レポート課題の提出

- 2回に分けて提出する
- 第1回提出
 - 授業期間中の提出締切(詳細はMoodleを参照)
 - 課題1・2の内容のみを提出(配点:35点)
- 第2回提出
 - 授業期間後の提出締切(詳細はMoodleを参照)
 - 課題3・4の内容を追加して提出(配点:65点)
 - 課題1・2の変更も受け付ける(第1回分の点数に考慮)
 - 第1回の未提出者は、提出を受け付けない

レポート課題資料

- 演習レポート課題説明
- レポート LaTeX テンプレート + サンプル(PDF)
 - テンプレートに従って作成する
 - LaTeX が使用できない場合は、指定されている通りの項目や形式が正しく守られていれば、別のソフトウェアを使って作成しても構わない
- 課題説明資料やテンプレートに明記されていない項目は、各自で判断して作成すること
 - 資料で説明されている以上の個別の作成方法や評価基準についての質問には、回答しない

LaTeX 使用方法

- LaTeX の基本的な使用方法は、1年生の「プログラミング (リテラシー)」で学習した通り
- その他の使用方法は、レポート課題説明の付録の解説や、テンプレートの例を参照
- 図の挿入方法も、テンプレートを参照する
 - スクリーンショットは、GIMPなどのソフトウェアを使って、トリミング等の編集を行うことができる
 - 画像は、EPS形式で保存したものを、LaTeX ソースファイルから参照して、挿入することができる
 - 同じく、テキストやソースファイルも、ソースファイルに挿入できる
- LaTeX は、今後もレポートや卒業論文作成等で使う機会があるので、慣れておくと良い

レポート課題

- 自分で決めた何らかのテーマを題材にして、データベースとWebインターフェースを作成
 1. データベースのスキーマの設計
 - 全ての正規形を満たすように正規化を行う
 2. データベースの作成
 - テーブルの作成、データの追加
 3. データベースへの問い合わせ
 - 問い合わせの具体例を考えてSQLを作成・実行
 4. Webインターフェースの作成
 - 一覧表示、追加・削除・更新、実用的な検索

4. Webインターフェースの作成

- Webインターフェースを作成する
 - 一覧表示
 - 主要なテーブルに対する追加・削除・更新
 - フォームの動的生成を用いて使いやすい機能にすること
 - 検索
 - なるべく実用的な検索機能にすること
- レポート
 - 全体のページの構成
 - 各ファイルの全体を引用した上で、主要な点を説明
 - 前のページから渡される入力や、そのページで使用するSQLとその作成方法、結果の表示方法等
 - どのようにして各機能を実現しているかが分るレポートを作成する(説明が不十分であれば、減点となる)

4. Webインターフェースの作成

- これまでの演習で使ったファイルを元に作成して構わない
 - ただし、「従業員」「部門」等の文言は、コメントの部分も含めて、自分のデータベースに合わせて書き替えること
- PHPプログラムの作成
 - 意図通りの結果が正しく表示されない場合は、プログラムの間違いを見つけて修正する(デバッグ)
 - ファイルが正しくアップロードされているかを確認する
 - エラーメッセージが表示される場合
 - エラーメッセージを参考に、プログラムの間違いを探す
 - 何も表示されない or 意図通りの結果が表示されない場合
 - まずは最小限のプログラムで正しく表示されることを確認してから、徐々に処理を追加していく
 - 前ページからの引数が正しく渡されているか、意図した通りのSQL文になっているか、SQLの実行結果が正しく取得できているかなど、変数の値を print で表示して確認する

評価基準

- レポート課題の説明に書かれている各課題ごとの評価項目を参照
 - 説明に書かれている以上の、各項目の詳細な判断基準や点数に関する質問は受け付けないので、各自で判断すること

レポート課題(後半)の提出

- 第2回提出

- Moodleの課題提出ページから提出

- PDF形式のファイルを指定されたファイル名で提出

- 他の提出方法は受け付けないので注意する

- サーバ上に作成されたインターフェースも評価対象

- 授業期間後の提出締切(詳細はMoodleを参照)

- 課題3・4の内容を追加して提出

- 課題1・2の変更も受け付ける(第1回分の点数に考慮)

- 変更がない場合も、必ず課題1・2の内容を含めること

- 第1回の未提出者は、提出を受け付けない

レポート課題に関する注意(1)

- くれぐれも十分早くから準備を始めること
 - 締め切りの直前に始めても、間に合わない
 - きちんと計画的に課題に取り組むこと
 - 少なくとも締め切りの1週間前までには課題の内容は終らせて、残りの時間はレポート作成に使った方が良い
 - LaTeXの使用方法もあらかじめ確認しておく
 - あらかじめ、自分の演習環境で、テンプレートが問題なくコンパイルできることを確認しておくこと
 - 締切直前になって問題に気付いても間に合わない
 - 質問や相談等があれば、早めに申し出ること
 - 締切直前になって質問して来ても間に合わない

レポート課題に関する注意(2)

- BYOD端末のトラブルに備えること
 - BYOD端末のトラブルを理由とするレポート提出締切の延長は認めない
 - BYOD端末の管理は学生の責任
 - 生協の代替機貸出サービスなどの利用
 - BYOD端末外部に作業データや作成途中のレポートのバックアップを取る
 - トラブルによる遅延に備えて、早めにレポートを作成する

レポート課題に関する注意(3)

- できるだけ工夫をすること
 - 選択肢の動的生成、高度な検索インターフェース、など
 - 実用的なデータベースであれば、高評価
 - こういった自由度の高い課題で、どれだけ工夫できるかが、非常に重要
 - 就職活動等でも、自己PRのポイントになる

レポート課題に関する注意(4)

- レポートをきちんと書くこと
 - どのようにして処理を実現しているのかが、きちんと分かるように書く
 - プログラムを書けるだけでなく、きちんと文章で説明できることも重要
 - 見出しや段落分け、適切な余白・行間、引用は枠で囲むなど、レポートの見やすさも重要
 - LaTeX を使用しない場合は、特に要注意

レポート課題に関する注意(5)

- 不正行為は絶対にしないこと
 - たとえ一部でも、他人のプログラム・レポートを丸写しした場合は、不正行為となり、厳重に処罰・減点される

よくある間違い(1)

- 課題1 .スキーマの設計
 - キー属性・関数従属性がきちんと書かれていない、もしくは、間違っている
 - 正規化の間違い(正規形の判定・分解)
- 課題4. Webインターフェースの作成
 - 選択肢の動的生成に対応していない
 - 検索機能がない、もしくは、ほとんど意味のない検索機能しかない
 - サンプルプログラムの一部がそのまま残っている(タイトルなど)

よくある間違い(2)

- レポートの書き方

- 説明不足のレポート

- 処理の実現方法の説明、使用した変数やSQLの説明、ファイル構成の図、など
- 作成結果だけではなく、何故そのようなプログラムを作成したのか、理由の説明が必要

- レポートの見易さ・読みやすさ、レポートの体裁

- 適切な余白・行間、章分け・段落分け、インデント、本文と引用プログラムの区別、誤字脱字、など

- 詳細は、レポート課題の説明を参照すること

エラーへの対処

- よくあるエラーへの対処方法は、演習資料の末尾の付録を参照する
 - php作成時のエラー
 - ファイルを正しく編集・アップロードしていない
 - phpプログラムやSQLの間違い
 - SQLのエラーについては、まずは psql 上で SQL を実行して、SQL文が正しく実行できることを確認してから、php プログラムに組み込む
- 演習資料の説明通りに対処しても解決しない場合は、相談する

演習で問題が生じた場合

- 演習に関する質問や問題には、個別に対応する
 - Q&Aフォーラム、電子メール、Zoomオンライン講義中の質疑応答
 - 締切直前(平日24時間以内)の質問には対応できないので、余裕を持って質問する
- 相談する場合は、具体的な情報を記述すること
 - 学生番号、九工大ID、どのような操作(入力)を行った結果、どのような問題(出力)が生じたのか、など
 - 曖昧な内容の質問をして来ても、問題が確認できない
- 正しい環境や手順で演習を行っていることを確認すること
 - 説明を無視して誤った手順で演習を行って、問題が生じる人が多い
- BYOD端末の仮想環境の問題は情報基盤センタに相談する
 - 仮想環境が起動しなくなった、極端に動作が遅い、など

まとめ

- 前回の復習
- PHPによるインターフェース開発(2)
 - データの追加
 - データの削除・更新・検索
 - 注意点
 - 演習課題
- レポート課題(後半)

次回予告

- データベースシステム内部で用いられる技術
 - 物理的データ格納方式
 - 同時実行制御
 - 問い合わせ処理の最適化
 - 障害回復
- 物理的データ格納方式
 - データ格納方法
 - データアクセス方法

教科書・参考書

- 「リレーショナルデータベース入門 第3版」
増永良文 著、サイエンス社（3,200円）
– 8章
- 「データベースシステム 改訂2版」
北川 博之 著、オーム社（3,200円）
– 8章

