

# データベース S 演習資料

## 第 3 回 PHP による Web インターフェース (2)

九州工業大学 情報工学部 システム創成情報工学科  
講義担当：尾下真樹

### 1. フォームを使った入力インターフェースの開発

前回の演習に引き続き、データベースをウェブから利用するためのインターフェースの開発を行う。今回は、データベースに格納されているデータを表示するだけではなく、HTML のフォームの機能を利用して、利用者がウェブブラウザから入力したデータにもとづき、データの追加・削除・更新・検索が行えるようなインターフェースを開発する。

### 2. HTML フォームの書き方

HTML 中で、フォームを使うためのタグを使ってフォームを記述することで、利用者がテキストを入力したり、項目を選んだりすることのできるような、フォームを実現できる。

フォームは、<FORM> ~ </FORM>タグによってフォーム全体を指定し、<INPUT>タグによって各入力項目を指定する。

```
<FORM ACTION="url" METHOD="GET or POST"></FORM>
```

フォームを記述する。ACTION には、送信ボタンを押した時に呼び出す CGI や PHP ページの URL を記述する。フォームを使って記述された引数が、URL に対する引数として送られる。method には、引数の送り方を指定する。"GET"を指定すると、URL の後ろに引数として送信する。一方、"POST"を指定すると、プログラムが起動してからテキスト入力として送信される。基本的にはどちらの方法でも構わないが GET の方がコマンドラインからのデバッグがやりやすいので、今回の例では GET を使用している。

```
<INPUT TYPE="type" NAME="name" VALUE="value" CHECKED>
```

入力項目を記述する。type には入力エリアのタイプを指定し、テキストフィールド (text)、チェックボタン (checkbox)、ラジオボタン (radio)、などが指定できる。name には、入力されたデータを何という引数名で送信するかを指定する。value には、初期値を指定する。

```
<SUBMIT VALUE="label">
```

送信ボタンを記述する。送信ボタンを押すと、ACTION が実行される。value には、ボタンに表示するラベルを指定する。

この他に、<SELECT>タグや<OPTION>タグを使うことで、プルダウンメニューを表示してデータを選択する入力項目も実現できる。フォームの詳細な書式については、先述のウェブページなどを参照のこと。

### 3. PHP 側での引数の受け取り方

フォームから送られた引数は、PHP プログラムでは、スーパーグローバル変数という特殊な変数（連想配列）を経由して受け取ることができる。

フォームのメソッドに GET を指定した場合と POST を指定した場合で受け取り方が異なるので注意する。

<code>\$_GET[ 変数名 ];</code>
-----------------------------

<code>method="GET"</code> で送られたフォームからの変数を受け取る。
--

<code>\$_POST[ 変数名 ];</code>
------------------------------

<code>method="POST"</code> で送られたフォームからの変数を受け取る。
---

### 4. PHP によるインターフェース作成演習（2）

#### 4.1. テーブルへのデータ追加

次に、テーブルへデータを追加する処理を作成する。

追加するデータを、フォームを使って利用者に入力してもらうためのページ (`employee_add_form.html`) と、そのフォームで入力されたデータを実際にデータベースに追加するためのページ (`employee_add.php`) の 2 つを作成する。

上記の 2 つのファイルも講義のページに置いてあるので、ダウンロードして自分のディレクトリに置いてみる。上と同じく、`employee_add.php` の 18 行目を変更して、自分のデータベース名にするのを忘れないこと。

最初にアップロードしたメニューのページから (`menu.html`)、「従業員のデータ追加」を選んで、入力ページ (`employee_add_form.html`) に移動する。ここに適当なデータを入力して、送信ボタンを押してみる。すると、ここで入力された値を引数として `employee_add.php` が実行され、追加処理が成功するメッセージが表示される。再びメニューからデータ一覧を表示してみて、追加されていることを確認してみよ。

ただし、値を入力する段階で、すでに存在する従業員番号を入力しようとしたり（従業員番号には `unique` 制約が設定されているので値の重複を許さない）、許される幅を超えて入力しようとしたり、一部の値が空白だとエラーが発生してうまく入力できないので注意する。

今回の PHP プログラム (`employee_add.php`) では、フォームから送られた引数を取得して変数に格納し (11 ~ 14 行目)、その変数を使用してデータ挿入のためのクエリーを動的に生成している (28 行目)。

#### 4.2. テーブルへのデータ追加（動的生成版）

さきほど作成したページでは、最初の入力ページは `html` だけで作成されており、適切な従業員番号や部門番号を入力するのが面倒だった。

そこで、より入力しやすいように、最初の入力フォームも PHP を利用して生成してみる。

前回のページ (`employee_add_form.html`) の代わりに、`employee_add_form.php` を使用する。

`employee_add_form.php` では、まずデータベースに接続して現在の従業員番号で最も大きい値を検索し、そ

の次の値を従業員番号の初期値として使用するようになっている。また、部門番号についても、利用者が番号を入力するのは面倒なので、テーブル `department` にアクセスして各部門の名前を取得してラジオボタン（複数の項目から一つを選択できるフォームの入力項目のタイプ）として表示することで、部門名で選択できるようにしている。

#### 4.3. データの削除

データの削除処理を作成する。

ここでは、削除処理のためのページは、上記のデータ追加処理（4.1 節）と同じく、削除する従業員番号を入力する HTML のフォーム（`employee_delete_form.html`）と削除処理を実行する PHP スクリプト（`employee_delete.php`）の2つからなる。

これまでと同じようにダウンロードして試してみよ。

ただし、今回は、削除処理を行う SQL を作成する処理（`employee_delete.php` の 27 行目）を空欄にしてあるので、各自そのコードを作成してみることに。作成したら、実際に実行して削除されることを確認する。

なお、以前の講義で学習したように、SQL の DELETE 構文は以下のようになる。

```
delete from tname where condition;
```

テーブルからデータを削除。*tname* にはテーブルの名前、*condition* には削除対象のデータの条件を指定する。

#### 4.4. データの削除（動的生成版）

削除処理において、削除する従業員の番号をいちいち入力するのは面倒である。そこで、データの追加と同様に、従業員の一覧を自動的に表示して、ラジオボタンで選択可能にすることで、より使いやすいインターフェースを提供できる。

これまでと同様に、`employee_delete_form.php` をダウンロードして試してみよ。

#### 4.5. データの更新

データの更新処理を作成する。更新処理は3つのページからなる。

Step 1. 更新する従業員番号を入力する HTML のフォーム（`employee_update_form1.html`）

Step 2. 指定された従業員番号の修正情報を入力するフォーム（`employee_update_form2.php`）

Step 3. 更新処理を実行する PHP スクリプト（`employee_update.php`）

これまでと同じように、それぞれダウンロード・修正して、うまく動くかどうかテストしてみよ。

なお、今回も更新処理を行う SQL を生成する処理（`employee_update.php` の 30 行目）は空白にしてあるので、各自で加えること。

なお、データ更新のための SQL の UPDATE 構文は以下のようになる。

```
update tname set attribute1=value1, attribute2=value2, ... where condition;
```

データの更新。*tname* にはテーブルの名前、*condition* には削除対象のデータの条件を指定する。  
**set** 以降には、「属性名=代入する値」の組をコンマで区切って記述する。複数項指定することで、複数の属性を同時に更新できる。

なお、2番目のフォーム（`employee_update_form2.php`）では、データ追加処理と同様に各属性の入力のためのフィールドを生成しているが、各フィールドの初期値として、データベースから検索した現在の属性値を設定するようになっている。  
これにより、ページを表示した状態では現在の値が各フィールドに入っているため、利用者は更新したい項目のみを修正して送信ボタンを押せば良い。

#### 4.6. データの更新（動的生成版）

データの更新についても、Setp 1 で従業員番号を入力するのは不便であるため、データの削除のインターフェースと同様に、従業員一覧を表示して、更新を行う従業員を選択できるようにする。  
この修正については、サンプルのファイルは用意していないので、これまでのファイルをもとに、自分で作成すること。  
データの削除で使用したファイル（`employee_delete_form.php`）を、`employee_update_form1.php` という名前でコピーして、一部を修正し、更新する従業員を選択するページを実現する。

まずは、全体のメニュー（`menu.html`）に、データの更新の動的生成版のページへのリンクを追加する。

```
<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>
  </UL>

</BODY>
</HTML>
```

その後、データの削除で使用したファイル（`employee_delete_form.php`）を、`employee_update_form1.php` という名前でコピーして、一部を修正することで、更新する従業員を選択するページを実現する。  
どこを修正すれば良いかは、各自、自分で考えること。

ヒント：選択結果を送信したときに Setp 2 のファイル（`employee_update_form2.php`）が呼び出されるように、変更する。次のファイルに渡す変数は、削除処理と更新処理で同じであるため、変更する必要はない。

#### 4.7. データの検索

データの検索処理を作成する。今回は、部門名を指定すると、その部門に所属する従業員の一覧を表示する検索インターフェースを作成する。検索処理は3つのページからなる。

Step 1. 検索する部門名を選択するフォーム (`employee_search_form.php`)

Step 2. 検索結果の従業員の一覧を表示するページ (`employee_search.php`)

これまでと同じように、それぞれダウンロード・修正して、うまく動くかどうかテストしてみよ。

まずは、全体のメニュー (`menu.html`) に、検索ページへのリンクを追加する。

```
<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>
</HTML>
```

検索する部門名を選択するフォームのページ (`employee_search_form.php`) では、部門の一覧を検索して選択肢として表示して、選択された部門番号を次の検索結果を表示するページに引数 (`dept_no`) として渡すようになっている。このとき、全ての部門を表示するための選択肢も合わせて表示し、その選択肢が選択された場合には、引数 (`dept_no`) の値として部門番号の代わりに特別な文字列 “ALL” を渡すようになっている。

これまで同様、検索処理を行う SQL を作成する処理 (`employee_search.php` の 31,33 行目) は空白としていたので、各自で作成すること。前のページから渡された引数 (`dept_no`) の値に応じて、全従業員の一覧を表示する SQL か、指定された部門の従業員の一覧を表示する SQL のいずれかを使用するようにする。