展開演習Bテキスト

椙山女学園大学現代マネジメント学部　三木　邦弘

令和 4 年 9 月 21 日版

1 QR コードの利用 1
- 1.1 PHP による QR コードの出力 1
- 1.2 QR コードによる決済 2
- 1.3 共通部分をまとめる方法 5
- 1.4 時刻の扱い方 6

2 ファイルのアップロードとデータベースへの格納 8
- 2.1 ファイルを送るための HTML のタグ . 8
- 2.2 ファイルを受け取るプログラム 9
- 2.3 ファイルをデータベースに入れる . . . 9
- 2.4 データベースから取り出したファイル内容の扱い方 10

3 メールと定期的処理 11
- 3.1 HTML でメールを送る 11
- 3.2 PHP でメールを送る方法 11
- 3.3 at を利用した予約実行 12
- 3.4 cron を利用した定期的実行 13
- 3.5 PHP で Unix のコマンドを実行する . 14

4 IoT とグラフの表示 15
- 4.1 どうやってつなぐか 15
- 4.2 電源をどうするか 16
- 4.3 センサーをどうするか 16
- 4.4 集めた情報の処理 16
- 4.5 環境センサーからの情報の取り込み . . 17
- 4.6 グラフの描画 18
- 4.7 グラフ化するデータの作成 22
- 4.8 超えたらメールを送る 23
- 4.9 web ページの自動更新 24

5 スクレイピング 25
- 5.1 Web ページの取り込み 25
- 5.2 HTML の解析 26

6 API の利用と XML・JSON 29
- 6.1 XML とは 29
- 6.2 JSON とは 29
- 6.3 API サービスの例 30
- 6.4 天気予報データの取得 32

7 物体検出プログラムの利用 34
- 7.1 YOLOv5 による物体検出 34
- 7.2 YOLOv5 用学習データの作成 36
- 7.3 YOLO による学習 38
- 7.4 学習結果の確認 39
- 7.5 PHP による画像認識 39

8 演習課題 41
- 8.1 QR コードの表示 9/21 41
- 8.2 QR 決済システム (1) 9/28 41
- 8.3 QR 決済システム (2) 10/05 42
- 8.4 QR 決済システム (3) 10/12 43
- 8.5 QR 決済システム (4) 10/19 43
- 8.6 画像データベース 10/26 44
- 8.7 メールの定期送信 11/2 45
- 8.8 利用者登録の確認 11/9 46
- 8.9 円相場のグラフ 11/16 47
- 8.10 環境センサーのデータのグラフ化 11/23 48
- 8.11 グラフの改良とメールで知らせる 11/30 48
- 8.12 スクレイピングと POST でデータを送る 12/7 50
- 8.13 天気予報の表示 12/14 50
- 8.14 YOLO による学習 12/21 51
- 8.15 YOLO による物体検出 1/11 51

索引 53

1. QR コードの利用

様々な商品に付けられているバーコードは、十数桁の数字を棒の太さや間隔で示したもので、通常会社コードと商品番号を合わせた数字を示しています。コンビニなどではこれをレーザー光などを利用して読み取り、データベースに照会して商品名や価格の情報を得て、レシートに印字します。バーコードの形と数字の内容が規格化されており、パッケージなどにバーコードを印刷するだけで安く使用できるのと、読み取り機が安価なため、広く使われています。ただバーコードで表現できる内容が数字で十数桁しかないために、個々の商品に異なる番号を振る事ができませんし、情報量が少なすぎるので商品番号以上の商品に関する情報を含めることもできません。QR コードは、1994 年 (平成 6 年) に自動車部品メーカーであるデンソーの開発部門が開発したマトリックス型二次元コードです。デンソーは QR コードを広く普及させることを考えて、特許をオープンにしました。最大約 7 千桁の数字が表現できる他、用途に合わせて誤り訂正レベルが設定でき、スマホなどで読み取り可能なため、現在 QR 決済などで様々な分野で広く使われています。

この章ではこの QR コードを扱う方法の説明や、簡単な QR 決済のシステムを作成します。

1.1 PHP による QR コードの出力

Web ページに QR コードを表示させる場合、いつも同じ内容の QR コードであれば、QR コードの画像ファイルを用意して、HTML の Img タグでその画像ファイルを指定すれば可能です。一方お客さんにお互いに異なる座席を指定する電子チケットのような QR コードは、全て異なるものでないと困ります。そしてそれを予め全て画像として用意しておくより、必要に応じて作成する方がよいでしょう。

画像を作成できるプログラムが書けるのであれば、基本的に QR コードの画像を生成するプログラムを書くことは可能です。しかしそのプログラムを書くためには QR コードの細かい仕組みを理解する必要があり大変です。幸いなことに様々なプログラミング言語用の QR コード生成プログラムが公開されているので、それを利用させていただくことにします。基礎演習のテキストの AR システムでは JavaScript によるものを使用していますが、PHP 用の QR コードを生成するプログラムの一つをここでは紹介します。

プログラムの作者は Y.Swetake 氏で https://www.swetake.com/で公開されています。このサイトからダウンロードしたものはまとめて圧縮されているので、解凍・展開をします。出てきたファイルの中で必要なものは、php ディレクトリの中の `qr_img.php` と data ディレクトリと image ディレクトリです。この 3 つを QR コードを表示する HTML や PHP のファイルがあるところにコピーし、`qr_img.php` の最初の方にある以下の部分を書き換えます。

```
$path="./../data";           /* You must set path to data files. */
$image_path="./../image";    /* You must set path to QRcode frame images. */
       ↓
$path="data";           /* You must set path to data files. */
$image_path="image";    /* You must set path to QRcode frame images. */
```

そして QR コードを表示したいところに、以下のように HTML の Img タグを記述します。

```
<Img src="qr_img.php?d=http://www.sugiyama-u.ac.jp/&e=L&s=3">
```

- d=で QR コードで表したいデータを指定します。例のように URL でも良いし、単なる数値などでも構いません。漢字は読み取り側がどのような漢字コードと解釈するのか明確でないので避けましょう。URL のファイル名の後に?が付いてパラメタがある場合は、e=以下の指定と混ざってしまうので、urlencode() を使用して変換したものを指定します。

- e=で誤り訂正のレベルを指定します。L、M、Q、H のどれかを指定します。L が一番誤り訂正レベルが低く、H が一番誤り訂正レベルが高くなります。訂正レベルが高いほど QR コードの汚れなどに強くなりますが、図 1.1 を見ても分かるように、QR コードが大きくなります。何も指定しない場合は e=M を指定したことになります。
- s=で QR コードの大きさを指定します。1 の場合が一番小さくなります。何も指定しない場合は s=4 を指定したことになります。

パラメタを変えると図 1.1 のように QR コードを表示することができます。QR コードで表しているのは全て同じ「https://www.sugiyama-u.ac.jp/」です。

図 1.1　QR コード出力の例

1.2　QR コードによる決済

図 1.2 のように我々がお店で何か買い物をする際には、代金を現金で支払うのが普通でした。支払いが高額な場合、現金を持ち歩くのは危ないので、クレジットカードによって支払うという方法が考えられました。クレジットカードを見せて、サインすると支払いの手続きは終わります。お店はクレジットカードの番号を元にクレジット会社に請求し代金を得ます。クレジット会社は銀行の口座からお店に渡した代金を引き落とします。銀行から代金を引き落とすためには、手数料が必要になります。その手数料は、通常お店から取った代金の数 % 分になるクレジットカードの利用料から支払います。お店の売上が利用料の分減りますので、クレジットカードの使用ができない店や、クレジットカードの支払いの際には割引ができない店があります。

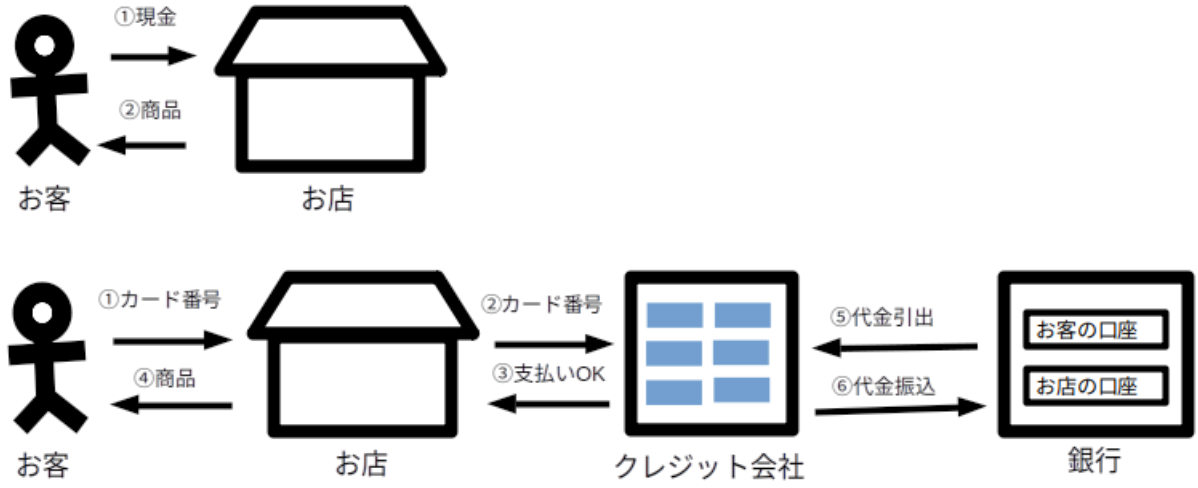

図 1.2　現金やクレジットカードによる支払い

スマホの決済や QR コードによる決済も基本的にはクレジットカードと同じです。QR コードによる決済の利点は、お店側に高価な設備が必要でないところでしょう。スマホで済ませることも可能です。お店にとっての問題はクレジットカードと同様に利用料を取られるところです。銀行の手数料がある限り、代金の大きさに関わりなくある程度の手数料を取られます。この問題を解決するのに一番簡単な方法は、お店や利用者の口座を、図 1.3 のように QR コードの決済システムの中に持つことです。同じシステムの中で数字が動くだけですので、コストはほぼゼロになります。現在の日本ではいくつかの決済システムが覇権を争っているところです。大きくなるほどシステム内でのお金のやり取りになるのでコストが減ります。赤字覚悟で利用者に還元しても将来取り返せるということでしょう。他社が無くなっていれば、取り返すために手数料を上げても逃げられません。

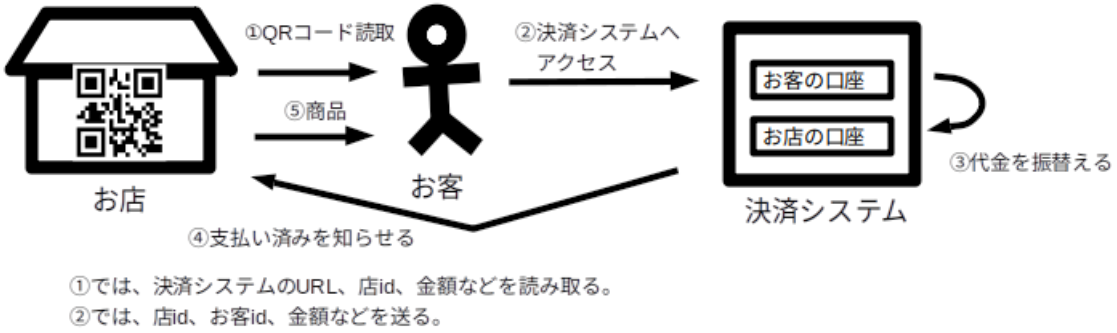

図 1.3 QR 決済による支払い

クレジットカードの利用はある程度高額な支払いでしたが、QR コードによる決済などではより少額の支払いにも使われます。誰がいつ何を購入したかがわかると、売る側は次に何が売れるか予想を立てることができます。そういう情報の収集にも役立つと考えているようです。

ここでは次のように使う QR コード決済システムを作ってみましょう。

1. お店の人が決済システムにアクセスすると、既に認証済みであれば、自分宛の支払いの QR コードと入出金リストが表示されます。もし認証がまだであれば、名前とパスワードを入力して認証を通過すると同じものが表示されます。
2. お客がお店の人の QR コードを元にアクセスすると、既に認証済みであれば、お客の残額が表示された支払額の入力画面が表示されます。もし認証がまだであれば、名前とパスワードを入力して認証を通過すると同じものが表示されます。
3. お客が支払額を入力して、支払い処理をしたら、お店の人は入出金リストを更新し、支払いがあったことを確認してから商品をお客に渡します。

これまでの話に出てこなかった認証が出てきます。大事なお金のやり取りですから、認証を使って第三者が勝手に触れないようにする必要があります。一方お買い物の度に認証をするのは面倒です。そこで前期にやった Cookie などを利用して一度認証を通過すれば、しばらくは名前とパスワードを入れなくて済むようにしています。一方そのしばらくの間に他の人に使われる恐れが生じます。心配性の方のために認証済みを解除できるようにする必要もあります。

お客が支払額を入れ間違えると面倒なことになります。足りない場合は足りない分を再度支払えば良いのですが、多すぎた場合は返金の仕組みが必要になります。これはお客の QR コードをお店の人が読み取れば可能ですが面倒です。QR コードに金額の情報も付けられるようにして、お客が支払額を入力する必要がないようにします。ただこの場合、支払金額によって QR コードが変わります。よってお店の QR コードを印刷して使うという方法は無理で、レジに QR コード表示用の画面が付いていて、そこに支払額込みの QR コードを出す

ようなシステムが必要になります。

これらを実現するために図 1.4 のような構成にします。

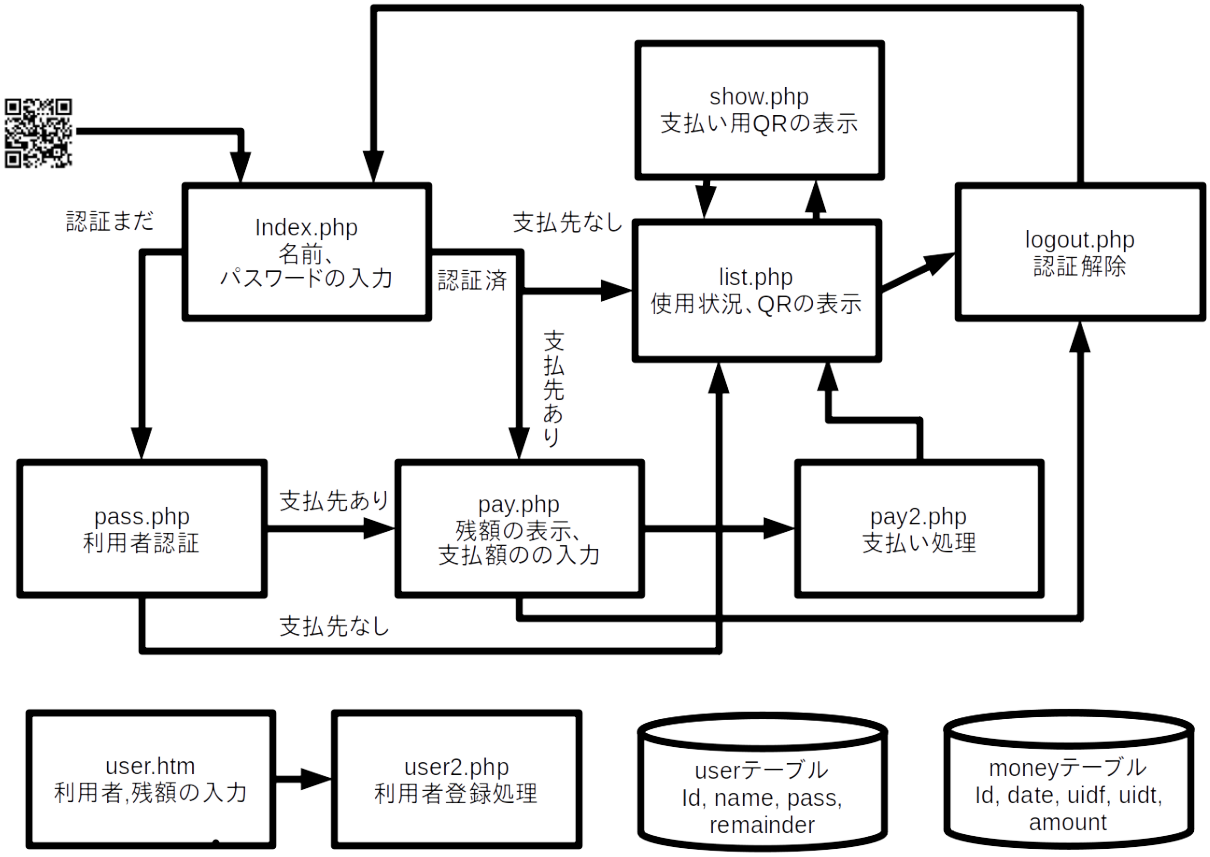

図 1.4　QR コード決済システムの構成

- user テーブルで、名前 (name)、パスワード (pass)、残額 (remainder) を管理します。
- money テーブルで、いつ (date)、誰 (uidf) が、誰 (uidt) に、いくら (amount) 送金したか記録します。
- 決済システムに入る際には利用者認証を行います。session の情報がある程度の期間残るようにして、残っている場合は利用者認証を省略できるようにします。
- user.htm で入力した内容を user2.php で user テーブルに登録することにより利用者登録をします。本来ならばきちんと管理者かどうかの認証をする必要がありますが、ここでは省略しています。
- お客やお店の人が index.php にアクセスした場合、まず認証済みかどうか調べます。認証がまだの場合はここで名前とパスワードを入力して、pass.php で user テーブルの内容と合うかどうかで判定します。認証済であれば、支払先があるかどうか調べます。あれば pay.php へ、なければ ist.php へ行きます。
- pay.php では、お客に残額を表示し、支払額を入力してもらい、支払いの指示をすると pay2.php へ行きます。
- pay2.php では、お客からお店へ支払額が動いたと言うデータを money テーブルに追加し、user テーブルの残額も修正します。そして list.php へ行きます。
- list.php では、自分宛の支払いの QR コード (index.php の URL と user テーブルの id のみで金額情報なし) と入出金リストを表示します。
- お客の支払い処理が終わったら、お店の人は list.php を更新し、支払いがあったことを確認します。
- logout.php では、認証済み情報を削除します。これによって共有の端末で使用した場合に、他の人に使

われてしまうことを防ぎます。

- 図 1.4 にはありませんが、データベースを利用する際に必要になる部分を common.php に入れて共有します。

1.3 共通部分をまとめる方法

複数のファイルで校正されたシステムでは、ファイルの内容に共通な部分がよく見られます。例えばデータベースを利用する際のデータベースを開いたり、トランザクションをスタートする部分です。このような共通する部分は、別のファイルに入れておいてそれを取り込む形にすると、各ファイルでは別ファイルを取り込む指示を入れるだけで済みます。またこの共通部分に修正が生じた場合も、別ファイルを修正するだけで済みます。共通するファイルの書き方は次のようになります。

```
<?php
共通する部分
?>
```

これが「common.php」と言う名前のファイルに入っている場合、この内容を取り込みたいところに、次のような指示を追加します。

```
include "common.php";
```

もちろんこれは PHP の機能ですので、<?php〜?>の間になければいけません。

共通する部分が、一つのファイルの中に何回か出てくる場合があります。このような場合は関数を利用します。PHP の関数は JavaScript の関数と同じ形をしています。例えば次のような形で関数の定義をします。

```
function iloveyou() {
   for ($i=0;$i<100;$i++) {
      echo "I love you. ";
   }
   echo "<Br>";
}
```

関数の定義だけでは、関数の内容が記憶されるだけで「I love you.」は出てきません。そして出したいところに、次のように記述すると「I love you.」が 100 回表示されます。

```
iloveyou();
```

いつも 100 回ではなく、違う回数の場合もあると言う場合は、関数に注文を付けることが可能です。その場合は次のように関数を定義します。

```
function iloveyou($kai) {
   for ($i=0;$i<$kai;$i++) {
      echo "I love you. ";
   }
   echo "<Br>";
}
```

最初の行の () の間に適当な変数を書きます。これが注文を受け付ける変数です。そして次のように関数を呼びます。

```
iloveyou(100);
iloveyou(10000);
```

1 つ目では 100 回、2 つ目では 10000 回の「I love you.」が表示されます。複数の注文を受ける形の関数も可能です。回数だけでなく、love する相手も変えたい場合は次のようにします。

```
function iloveyou($kai,$aite) {
   for ($i=0;$i<$kai;$i++) {
      echo "I love ",$aite,". ";
   }
   echo "<Br>";
}
```

注文を受ける変数を「,」で区切って並べます。呼び出す方も同様に「,」で区切ります。

```
iloveyou(3,"Miki sensei");
iloveyou(1000,"cats");
```

関数の中の変数は関数の中のみで有効です。呼び出す側と同じ名前の変数を関数の中で使っていても、別物として扱われます。また関数の実行が終わったら関数の中の変数は消えてしまいます。呼び出す側と同じ変数を関数の中で利用したい場合は、次のようにします。

```
function showmsg() {
   global $msg;

   echo $msg,"<Br>";
}
```

「global」の後に書いた変数は、関数の外のものと共通になります。よって showmsg() を呼ぶ前に、$msg になにか入れておく必要があります。

1.4 時刻の扱い方

コンピュータのハードウェアでは時刻は「2021 年 10 月 14 日 16 時 45 分 20 秒」と言うような形で扱っていることが多いと思います。水晶振動子による正確な周波数の信号を数える IC があるからです。一方 Windows や Unix のような OS のレベルでは、ある時点からの秒数などで管理しています。我々は現在時刻をある時点からの秒数で言われても使えませんが、期間の計算やどちらが先かなどの判定が簡単にできるからです。PHP では Unix と同じく時刻を、グリニッジ標準時の 1970 年 1 月 1 日の 0 時からの秒数で管理しています。

PHP には現在の時刻を求めるために time() と言う関数があります。この関数は呼び出された時刻を示す秒数を返します。内部ではこの数値のまま扱う方が便利です。例えばちょうど 24 時間後の時刻を求めたいときは「24*60*60」を加えます。これが通常の「2021 年 9 月 30 日 16 時 45 分 20 秒」のような形では、24 時間後に月や年が変わる場合があるので面倒です。データベースに記録するような場合も大抵秒数のまま入れます。

時刻を表示する場合は秒数ではいつなのか分からないので、PHP には秒数を指定した形式の日時の形に変える date() と言う関数があります。形式の指定方法が少し面倒ですが、曜日だけ出すようなこともできます。

```
echo time(),"<Br>";
echo date("Y/m/d H:i:s"),"<Br>";
echo date("Y/m/d H:i:s",time()+12*60*60),"<Br>";
```

これを実行すると、例えば次のようなものが表示されます。2 行目のように、date() で 2 つ目の秒数の指定がない場合は、現在の時刻が表示されます。

```
1634201509
2021/10/14 17:51:49
2021/10/15 05:51:49
```

date() で日時の形式指定は表 1.1 のような文字を使います。それ以外の空白や「/」のような記号はそのまま出てきます。

表 1.1 日時の形式指定文字 (一部)

文字	意味	文字	意味
s	秒 (00 〜 59)	D	曜日 (Mon 〜 Sun)
i	分 (00 〜 59)	w	曜日 (0 (日曜) 〜 6 (土曜))
g	時 (1 〜 12)	M	月 (Jan 〜 Dec)
h	時 (01 〜 12)	m	月 (01 〜 12)
G	時 (0 〜 23)	n	月 (1 〜 12)
H	時 (00 〜 23)	Y	年 (1970 〜)
d	日 (01 〜 31)	y	年 (00 〜 99)
j	日 (1 〜 31)		

逆に mktime() を利用すると通算秒数を求めることができます。

```
echo mktime(12,34,56,9,26,2022);
```

これで「1664163296」と言う数字が表示されますが、これは 2022 年 9 月 26 日 12 時 34 分 56 秒の通算秒数です。年月日の指定の順番がアメリカ風？に時、分、秒、月、日、年の順番になっているのでご注意ください。

2. ファイルのアップロードとデータベースへの格納

ファイルを Web サーバーに送ることができます。利用者から送られたファイルの内容をファイルの形で保存したり、データベースに入れる方法についてここでは説明します。従来はパソコンやスマホに存在するファイルを送ることしかできませんでしたが、HTML の ver. 5 では、その場で撮影した画像や動画を送ることもできるようになりました。なお Web サーバーの設定でアップロードできるファイルの大きさには制限がされていますので、巨大な動画ファイルなどはアップロードできません[*1]。

2.1 ファイルを送るための HTML のタグ

ファイルを Web サーバーに送る際には、これまでと少し形の異なる Form タグを使用します。

```
<Form method="POST" enctype="multipart/form-data" action="upload.php" name="aaa">
```

「method=」は必ず「POST」にしてください。「enctype=」の指定も必ず追加します。「name=」の指定は必須ではありませんが、サーバーに送るファイルをドラッグ＆ドロップで即座に送りたいと言う場合は、JavaScript の助けが必要となるので名前を付けておく必要があります。アップロードするファイルを指定するボタンは次のタグで出すことができます。

```
<Input name="ufile" type="file">
```

これで通常参照...と言うようなボタンが表示され、クリックするとファイルが選択できるようになります。またこのボタンの上にドラッグ＆ドロップでアップロードするファイルを指定することもできます。

HTML ver. 5 より「capture=」と「accept=」の指定をすると、スマートフォンでの使用に限定されますが、その場での録音や撮影した結果をアップロードできるようになりました。ただしパソコンでアクセスした場合は、通常のファイルのアップロードと同じ扱いになります。次のようにすると内側のカメラで撮影した画像をアップロードできるようになります。

```
<Input name="cfile" type="file" capture="user" accept="image/*">
```

「capture=」を「environment」にすると外側のカメラになります。また「capture=」の指定を省略するとスマートフォン側でカメラやファイルを選択できるようになります。「accept=」を「audio/*」にすると音声を、「video/*」にすると動画をアップロードするようになります。

ドラッグ＆ドロップで即座に送りたいと言う場合は次のようにします。

```
<Input name="dfile" type="file" accept="image/*"
       style="width: 40em; height: 5em; display: block; border: 2px solid red"
       onDrop="setTimeout('document.aaa.submit()',500)" >
```

ドロップしやすいようにスタイルの指定で、大きなボタンにしています。また「onDrop=」で 0.5 秒後に送信するようにしています。このように少し待ってから送信しないと、ファイルの内容がうまく送れないことがあります。

「onDrop=」の指定がない場合は従来と同じように送信ボタンが必要です。

```
<Input type="submit" value="アップロード">
```

[*1] mars の現在の設定では、同時に 20 ファイル送ることができ、合計のサイズは 16MB になっています。

2.2 ファイルを受け取るプログラム

「ufile」と言う名前の入力欄から送られてきたファイルに関しては次のような変数が使用可能になります。

変数	変数の内容
$_FILES['ufile']['tmp_name']	アップロードされたファイルのとりあえずの名前
$_FILES['ufile']['name']	アップロードされたファイルの元々の名前
$_FILES['ufile']['size']	アップロードされたファイルの大きさ

これらといくつかの関連する関数を使用して、PHP で送られてきたファイルの内容を保存する例を以下に示します。

```
if (is_uploaded_file($_FILES['ufile']['tmp_name']) &&
    $_FILES['ufile']['size']>0) {
   move_uploaded_file($_FILES['ufile']['tmp_name'],"aaa.jpg");
}
```

ファイルを選択せずに送信することがありますので、is_uploaded_file() を使用してファイルがアップロードされたか確認をします。さらに送られたファイルの大きさが空の場合も外していますが、ここの条件は状況に合わせて変更したり省略することも可能です。move_uploaded_file() でとりあえずの名前のファイルを「aaa.jpg」に変更しています。とりあえずの名前のファイルは PHP の実行が終わると消去されますので、必ず move_uploaded_file() をします。またこの例のように固定の名前 (aaa.jpg) では、アップロードしたファイルはどんどん上書きされるので、上書きされては困る場合は、毎回異なるファイル名を指定する必要があります。

2.3 ファイルをデータベースに入れる

かつてのデータベースは、数字、文字列などの小さなサイズのデータを大量に扱うものでしたが、現在では画像などの大きなサイズのデータも扱うことが可能です。これによって例えば写真画像のデータベースで、写真画像と共に撮影者、撮影日時、撮影対象などの情報をまとめて扱うことができます。SQLite では、データ型として整数値を入れる「INTEGER」文字列を入れる「TEXT」に加えて、バイナリーデータを入れる「BLOB」が使用可能です。ただバイナリーデータはその中に区切りの「'」などを含むこともあるため、これまで説明したのと少し異なる方法を使います。

次の例は「photo」テーブルの TEXT 型の「name」と BLOB 型の「image」にそれぞれ「sample」と「star.png」ファイルを挿入するものです。$sql に prepare() でまず SQL 文の雛形を入れて、bindParam() で「:image」の部分に入れる内容を指定しています。

```
$sql=$db->prepare("INSERT INTO photo (name, image) VALUES ('sample', :image )");
$img=file_get_contents("star.png");
$sql->bindParam(':image',$img,PDO::PARAM_LOB);
if (!($sql->execute())) { die("Insert できません "); }
```

なお、違う内容のデータを挿入したい場合は、bindParam() と execute() をやり直すだけでできます。そのような場合は、name の方も「:」で始まるパラメタにして、bindParam() も 2 個にします。

データベースの内容を修正する場合 (UPDATE) も、BOLB 型のデータは上記の例と同様に prepare() を使います。一方データを取り出す (SELECT) などは、これまでと同じやり方で変数に取り出すことができます。ただ数値や文字列と異なり画像や動画などは echo で表示することはできません。

2.4 データベースから取り出したファイル内容の扱い方

データベースから取り出したファイルをブラウザに出すには、ファイルの内容を元に、ファイルを作成するのが一つの方法です。$data['image'] と言う変数にファイルの内容が入っている際に、次の 2 行でファイル (test.png) を作成することができます。

```
file_put_contents("test.png",$data['image']);
chmod("test.png",0644);
```

ファイルの作成は file_put_contents() で終わっていますが、chmod() でファイルの読み書きの設定を変更しないと、外部からアクセスができません。「0644」で自分はファイルの読み書きできるが、それ以外のものにはファイルの読み出ししか許さないようになります。できた画像ファイルを HTML の Img タグで指定すれば表示されます。A タグで指定すればリンクをクリックすると単独で表示されます。動画ファイルであれば Video タグ、音声ファイルであれば Audio タグで指定すれば視聴できるようになります。PDF ファイルであれば A タグか iFrame タグが良いでしょう。

このファイルを作成するやり方にはいくつかの問題点があります。複数の画像を表示する場合、別のファイル名にするのが無難です。同じファイル名を使い回すと、内容が変わってもブラウザ側にそれが伝わらず、以前の画像のまま変わらないようなことが生じます。またこうして作成したファイルはいつまで置いておくのでしょうか。ほっておくとデータベースにある全てのファイルがでてきてしまいます。

もう一つのやり方は、必要に応じて画像ファイルなどの内容を送るという方法です。展開演習で QR コードを表示する方法についてやりましたが、QR コードは次のようにすれば表示されるとしていました。

```
<Img src="qr_img.php?d=http://www.sugiyama-u.ac.jp/&e=L&s=3">
```

これは「qr_img.php」が「?」の後に並ぶパラメタを元に QR コードの画像を生成し、ブラウザに送っています。これと同じことをすれば、異なるファイル名を考える必要もなく、作ったファイルの消去のタイミングも考えなくて済みます。ただ毎回画像の内容を送るために、画像の一覧のようなページの表示に毎回時間がかかることがあります。

それでは「qr_img.php」ではどのようなことを行っているのか。パラメタからデータベースを検索し、既に表示すべき内容が$data['image'] と言う変数に入っているとします。

```
header('Content-Type: image/png');
echo $data['image'];
```

header() を使用しますので、HTML のタグなどを出さないようにします。「Content-Type:」で指定するもの (MIME タイプと呼ばれる) はファイルの種類によって変更する必要があります。次の表のようにしてください。

ファイルの種類	指定するもの	ファイルの種類	指定するもの
画像 (PNG)	image/png	画像 (JPEG)	image/jpeg
音声 (MP3)	audio/mpeg	動画 (MP4)	video/mp4
文書 (PDF)	application/pdf		

echo の前に次のような header() を追加することにより、ブラウザ側で表示するのではなく、ダウンロードするように指定することもできます。

```
header('Content-Disposition: attachment; filename="star.png"');
```

3. メールと定期的処理

この章ではメールを送る方法と、定期的に処理を行う方法について説明します。メールはシステム側から何か伝えたい時に利用するのに便利なものです。web ページに大事なお知らせを掲載しても、利用者がアクセスしてくれないと伝わりません。その点メールであれば、多くの人がスマホのメールを利用しているので、即座に伝わります。LINE の方が確実という人も多いでしょうが、比較的閉じた利用者間で使う LINE を登録するのは抵抗がある人も多いと思います。

またお知らせとなると、定期的に知らせる形もあるでしょう。それほど急がない知らせをどんどんメールで送るより、決まった間隔でまとめたお知らせメールを送るという方が望ましいでしょう。使う時にしかスイッチが入らないパソコンでは、定期的な処理は難しい場合が多々あります。一方昔から 24 時間動かしているサーバーで用いられてきた Unix では、決められた間隔でプログラムを起動するような機能があります。ここではその使い方も説明します。

3.1 HTML でメールを送る

web ページでは URL がわかればアクセスできるのと同様に、メールはメールアドレスがわかればメールを送ることができます。web ページにおいて、問い合わせをメールで送ってほしい場合、メールアドレスを書いておくだけではそれを送り手は入力する必要が生じます。コピペで済ます人も居るでしょうが、操作ミスでメールアドレスが正しく貼り付けられないと届きません。このような場合、既に学んだ HTML の A タグでメールを送るリンクを作ることができます。

```
<A href="mailto:aaa@bbb.ccc">問い合わせ</A>
```

これで「問い合わせ」と言うリンクが表示されて、これをクリックするとブラウザに設定されているメーラー (メールを送るプログラム) が起動されます。そして送り先として「aaa@bbb.ccc」が自動的にセットされます。メールを送り損なう最大の原因であるメールアドレスの入れ間違いが生じないので、確実にメールをもらえる事になります。

この方法で問題になるのは、スマホでは余りないようですが、パソコンなどではブラウザにメーラーが設定されていない事が多い点です。そのために web ページの見えるところにメールアドレスも明記しておく事が考えられます。またメールアドレスがばれるので、嫌がらせなどの標的に利用される恐れもあります。

問い合わせの受付ならば、適当な入力欄を作成し、そこに問い合わせ内容を書いてもらい、「送信」ボタンをクリックしたら、PHP で入力欄の内容を取り出してファイルに入れれば良いでしょう。この場合の問題点は、問い合わせの答えをどのように返すかです。よくあるのは、問い合わせをした人のメールアドレスも入力してもらって、そこへメールで回答を送る形ですが、メールアドレスが間違っていると回答が届きません。そうならないように、メールアドレスを 2 回入れてもらうなどの対策をしますが、問い合わせる人にとっては面倒な話です。その点 maito を使ったリンクであれば、問い合わせがメールで来るので、回答を返信の形で必ず返すことができます。

3.2 PHP でメールを送る方法

PHP を利用してメールを送ることができます。そのために次のように `mb_send_mail()` を利用します。

```
mb_send_mail($to, $subject, $message, $headers, "-f ".$from);
```

$to にはメールを送る先のメールアドレスを入れておきます。$subject には件名を入れておきます。漢字を含む件名も可能です。$message には本文を入れておきます。本文に漢字を含むことは可能ですが、改行の位

置には「\n」を入れる必要があります。$from には差出人のメールアドレスを入れておきます。PHP が代わりにメールを送るために、本当のメールの差出人を設定しています。最後の$headers には次のようにして内容を入れます。

```
$headers="From: ".$from."\n"
         ."Reply-To: ".$from."\n"
         ."MIME-Version: 1.0 \n"
         ."Content-Type: text/plain; charset=ISO-2022-JP\n";
```

「From:」ではメールの差出人のメールアドレスを示します。「Reply-To:」ではメールの返信先を示します。大抵の場合は「From:」と同じメールアドレスになるので、この例では同じ変数を使っています。様々なところからメールを送るが、返信はどこか一つでまとまって受け取りたいと言う場合は、「From:」と「Reply-To:」に違うものを設定することになります。「MIME-Version:」と「Content-Type:」はメールの本文が日本語の場合、文字化けしないように入れておきます。

なお、mars の利用者のメールアドレスは次のような形です。

```
ユーザ名@mars.mgt.sugiyama-u.ac.jp
```

3.3 at を利用した予約実行

後の章に出てくる「IoT」や「スクレイピング」では、相手から定期的に情報を送ってくることもありますが、こちらから情報を取りに行かないとだめな場合も少なくありません。定期的に最新情報を収集し、そのまとめを利用者に Web ページで見てもらったり、こちらからメールでお知らせするという流れになります。ここでは、まず at コマンドを利用した単発の予約実行の方法を説明し、次に cron コマンドを利用した定期的な実行方法を説明します。

PHP で記述した内容をブラウザで呼び出すのではなく、その場で実行することができます。例えば date.php は次のような内容だったとします。

```
<?php
echo date("Y/m/d H:i:s"),"\n";
?>
```

mars に RDP で接続し、スタートメニューにある「システムツール」の中の「LXTerminal」を選択すると、コマンド入力の画面が開かれます。そこで「wget -q -o - 」の後に date.php の URL を入力して Enter を押すと、date.php が実行されて現在の時刻が表示されます。なおコマンド入力の画面での貼り付けは、Ctrl+Shift+V となるのでご注意ください。

```
miki@mars:~$ wget -q -O - http://mars.mgt.sugiyama-u.ac.jp/miki/date.php
2021/09/21 16:30:16
miki@mars:~$
```

wget は指定した URL にアクセスして返ってきた内容を保存する Unix のコマンドです。「-q」オプションで余分なメッセージが出ないようにし、「-O -」オプションで返ってきた内容を表示するようにしています。これを指定した時刻に一度だけ行いたいときには次のようにします。ただし「miki」のところは自分のユーザ名に変えてください。

```
miki@mars:~/Desktop/www$ at -t 10201345
warning: commands will be executed using /bin/sh
at> wget -q -O - http://mars.mgt.sugiyama-u.ac.jp/miki/date.php
at> <EOT>
job 5 at Wed Oct 20 13:45:00 2021
miki@mars:~/Desktop/www$
```

1 行目の「at」が実行予約をする Unix のコマンドです。「10201345」は実行する時刻で、「10 月 20 日 13 時 45 分」を意味しています。月日時分は全て 2 桁で記述します。「at>」が表示されたらその後に実行するコマンドを入力します。3 行目に先程の実行する内容を指定しています。さらに続けて実行するものを入力することもできます。4 行目のところでCtrl+Dを押すと「<EOT>」と表示されて、実行される日時を表示して at コマンドの入力が終わります。

さてこのような方法で設定しても、その時刻に端末を利用しているとは限りません。そうすると実行した際に表示された時刻はどこに行くのでしょうか。at コマンドで予約実行した結果は、設定をしたユーザにメールで送られてきます。実行した際に何も出力がなければ、メールは来ません。mars ではスタートメニューの「インターネット」のところにある「Sylpheed」で自分に届いたメールを読むことができます。予定時刻が過ぎたら「Sylpheed」で結果を確認してください。

実は PHP のプログラムを端末で実行するだけならば、wget を使用して web サーバー経由で行う必要はありません。「wget -q -O -」の代わりに「php」だけで可能です。ただこのやり方は、at コマンドでは実行する主体が違うので、うまく動かない場合があります。それを回避する方法ももちろんあるのですが、割と複雑なので詳細は省略します。peditor で実行させてちゃんと動くことが確認できれば、端末で wget での実行を予約すると言う流れになります。

3.4 cron を利用した定期的実行

サーバーでは様々な処理が定期的に行われています。毎日、毎週、毎月と言う単位での定期的実行は Unix では cron と言うシステムが起動を行います。実は at コマンドの実行する部分も cron です。設定の仕方は大昔から変わっていませんが、少しわかりにくいところもあるので、設定用のアプリがないか少し探しましたが、なぜか無いようです。よって端末を起動して、設定ファイルを書き換えると言う形で設定します。

1. 「LXTerminal」を起動します。
2. 「crontab -eEnter」と入力します。
3. 「nano」と言う編集プログラムが起動され、現在の cron の設定ファイルの内容が表示されます。「#」の記号からその行の終わりまではコメントなので cron の実行には関係ありません。
4. 定期的実行のための設定内容を入力・修正・削除します。
5. Ctrl+Oで書き込み、Ctrl+Xで終了します。
6. 定期的な実行の際に何か出力があれば、その内容がメールで届きます。

cron の設定は 1 件に付き 1 行になります。複数の設定は複数行になります。1 行の形式は次のようになります。

```
分 時 日 月 週 実行するコマンド
```

分〜週については、表 3.1 のようになっています。例えば毎週月曜日の 0 時 3 分にコマンドを実行するならば「3 0 * * 1 コマンド」、毎月 1 日の 0 時 15 分にコマンドを実行するならば「15 0 1 * * コマンド」、年中 10 分毎にコマンドを実行するならば「0,10,20,30,40,50 * * * * コマンド」となります。

表 3.1　cron の設定内容

	設定内容
分	*にすると毎分実行します。0〜59 の数値を一つか、複数を「,」で区切って並べます。
時	*にすると毎時実行します。0〜23 の数値を一つか、複数を「,」で区切って並べます。
日	*にすると毎日実行します。1〜31 の数値を一つか、複数を「,」で区切って並べます。
月	*にすると毎月実行します。1〜12 の数値を一つか、複数を「,」で区切って並べます。
週	*にすると毎曜日実行します。0〜6 の数値を一つか、複数を「,」で区切って並べます。0 が日曜日になります。

3.5　PHP で Unix のコマンドを実行する

Unix には既に出てきた at や crontab の他に多くのコマンドがあります。昔文字入力と文字出力しかできなかった時代に誕生したコマンドは、文字で制御できるので、簡単に他のプログラムから利用することができます。ここでは PHP から Unix のコマンドを利用する方法を説明します。まずコマンドを実行するだけならば次のようになります。

```
system("date");
```

Unix の date コマンドが実行されて、その出力がブラウザに送られます。echo は必要ありません。ブラウザには「Thu 21 Oct 2021 04:26:26 PM JST」と言うような実行された日時が表示されます。パラメタが必要な場合は、空白を空けてコマンドの後に入れます。date は世界標準時で表示する際は「-u」を指定することになっているので次のようにします。

```
system("date -u");
```

この場合ブラウザには「Thu 21 Oct 2021 07:26:26 AM UTC」のように表示されます。

at コマンドのように、起動してからさらに入力が必要な場合は次のようにします。

```
$p=popen("Unix のコマンド","w");
fwrite($p,"コマンドに送る内容\n");
pclose($p);
```

popen() がコマンドを起動し、”w”の指定によりコマンドに送ることができるようになります。fwrite() を必要な数だけ使い、最後に pclose() でコマンドに、これで終わりであることを伝えます。

popen() で”w”の代わりに”r”を指定すると、コマンドの出力をファイルから読み込むのと同様に fgets() で取り込むことも可能です。

4. IoT とグラフの表示

IoT (Internet of Things) は様々なものをインターネットに接続して利用しようと言う考え方で、それ自体新しい発想ではありません。ただ以前はネットワークへの接続にハードウェアやコストの面での問題が多く、それほど普及しませんでした。例えば 1990 年代後半だったと思いますが、この大学の演習室のパソコンを有線 LAN に接続するためには、各パソコンに 6 万円ぐらいする LAN ボードを別途購入する必要がありました。さらに LAN ボードを動かすためのドライバーソフトも各々インストールする必要がありました。現在ではノートパソコンならば無線 LAN、デスクトップのパソコンならば有線 LAN に接続するために何か追加で購入する必要はなく、ちょっとした設定のみでネットワークにつながります。この章で紹介する教室の温度・湿度・二酸化炭素を測定する機器 (以下環境センサー) で使われているコントローラーは、無線 LAN に接続するための機能を持ちながら 730 円 (税込み) です。

IoT については非常に多くの観点について学ぶ必要があります。既にある IoT の機器を利用するのであれば、多少楽になりますが、このような事が可能なのではないか？と思いつく土台になる知識となるとやはり大変です。ハードウェアが絡む話は女子大生向きではないかもしれませんが、ハードウェアが絡むと即座にはまねされないものにつながります。

この章では IoT に関するいくつかの観点の説明、教室の温度・湿度・二酸化炭素を測定する環境センサーを使って情報収集と処理、結果の表示方法としてグラフをどうやって出すのかについて述べたいと思います。

4.1 どうやってつなぐか

IoT ですから、当然インターネットにつながらないと困ります。ではなぜインターネットにつながる必要があるのか？それば物理的に遠いところにあっても、インターネット経由でほぼコストゼロでつながるからです。実際のところ IoT とは言うものの、近いところのものとつなげる話も少なくありません。敷地内にある工場にある機器、温室の温度計などで、歩いて行って見てくれば済むではないかと言う見方もありますが、数が多い場合や記録を残す場合などに IoT にする利点が生じます。

インターネットにつながるコンセントがすぐ側にあるとか、無線 LAN の基地局の電波の届く範囲内にあるのであれば問題ありません。しかしそうでない場合もあります。公園のゴミ箱の空きを知ることができれば、ゴミ回収の合理化ができますが、公園の真ん中に LAN のケーブルは多分来ていません。また無線 LAN の基地局も期待できません。幸いなことに携帯やスマホのおかげで、これらが使用する無線回線は山奥などでなければ利用することができます。ただお金がかかります。河川が溢れていないか動画で 24 時間監視すると言うことになると、スマホの回線の通信料金は一番高いプランになるでしょう。一方公園のゴミ箱の空きを知るためにゴミ箱の内部の動画は必要ないでしょうし、ゴミ箱が溢れてもすぐに回収には行けませんので、1 時間に 1 回程度ゴミの量が分かれば良いでしょう。幸いなことにそのような通信量が少ない用途に対する通信サービスがあります。例えば LPWA (Low Power Wide Area-network) を利用した Sigfox[*2]と言うサービスでは、年額 1,500 円程度で 1 回に付き 12 バイトのデータを 1 日最大 140 回送ることができます。これらを比較すると表 4.1 のようになります。

表 4.1 接続方法の比較

接続に利用するもの	データ量	料金	接続設定
現場にある LAN や無線 LAN	動画も可能	無料 (既存設備があれば)	現場に合わせる
携帯やスマホの回線	動画も可能	高価格 (データ量が多いと特に)	事前設定可能
Sigfox など	極めて少ない	低価格 (データ量が少ないと特に)	事前設定可能

[*2] Sigfox は世界的に展開されているサービスで、国内では京セラコミュニケーションシステムが扱っています。https://www.kccs-iot.jp/service/

データ量の多少が大きな分かれ目になります。現場にあるネットワーク環境を利用する方法は安上がりですが、そのネットワーク環境に依存する接続設定になるので、セットアップに分かる人が必要になります。他の 2 つの方法はお金がかかりますが、製造時に接続設定ができるので、特に個人向けの機器ではすぐ使えるので便利となります。

4.2　電源をどうするか

IoT の機器も電気がなければ動きません。電灯線から交流 100V の供給を受ける、電池で動かす、太陽電池パネルなどで自家発電するなどが考えられます。電灯線は、停電と電気工事が必要になると高く付くという問題があります。一方電池のように、空にならないように管理する必要がありません。電池ではスマホのように毎日充電が必要では手間がかかりすぎるので、できるだけ機器の消費電力を減らして電池が長持ちするようにします。1 年半ぐらい電池が持つのであれば、年に 1 回電池を交換するという運用で使えます。それでも Sigfox の通信料より電源の方が費用がかかる恐れがあります。太陽電池で発電し、電池を充電して使うような形であれば、電池の心配をしなくて済みます。ただお天気が悪い日が続いたり、室内など陽が当たらない場所では困ります。太陽系の果に行くような人工衛星では、太陽電池が使えないので原子力電池を使うそうです。

4.3　センサーをどうするか

インターネット経由で伝える情報をどのようにして入手するかも大きな問題です。情報は電気信号の形でないと送れません。物理的な信号を電気信号に変換するものをセンサーと呼び、現在多くの種類のセンサーがあります。例えば「Interface」と言う雑誌の 2021 年 7 月号別冊付録の「IoT センサ図鑑」には、温度、湿度、気圧、動き、力、距離、接近、匂い、ガス、色、位置、電流、生体検知、音、タッチのセンサーが紹介されています。温度センサーと言っても、測定できる温度の範囲、精度、出力形式が異なるものが色々あります。高精度なものほど高価になります。気圧センサーで高精度なものでは、5cm 程度の高さの違いによる気圧の変化が分かるものもありますが、そこまでの精度が必要な場合は限られるでしょう。重さのセンサーがありませんが、力のセンサーを用います。重さがかかると形が変わるものに力のセンサーを貼り付けて、形の変わり具合から重さを測ります。このように違うセンサーを用いて測定することがあります。例えば電気機器は電気で動きますので、電流センサーで機器に流れる電流を測れば、機器が動いているかどうか分かります。

AI で処理をすることにより、高度な判定をするセンサーになることがあります。防犯対策用にコンビニの棚を撮影するカメラ画像から、棚の空き具合を AI で推定するという話があります。棚が空いたままでは売れませんので、空いている棚をセンサーで検出し、すぐに商品を補充することによって売上が増えたそうです。このような AI の処理を IoT の方でやってしまうか、データを集めてからデータ処理でやるかが問題になります。画像データと棚の空き具合の数字となると、データ量がかなり違うのでカメラのところで推定するのが望ましいです。その代わりカメラのところに AI を実行するハードウェアが必要になります。棚の画像であれば、棚や商品が文句を言うことがないので良いのですが、人の姿を撮影して処理をするような場合は問題になる可能性があります。そのような場合カメラのところで処理をして、サーバーには個人情報にはなりにくい認識結果だけ伝わるようにする、と言うような配慮が求められることがあります。

4.4　集めた情報の処理

センサーから集めた情報は大量になる事も多く、ビッグデータと呼ばれることもあります。大量のデータを人が個々に見ると言うわけにも行かないので、統計的な処理を行うのが普通ですが、なかなかうまく行きません。処理を行って何を得たいのかを明確にし、それに合った統計的手法を選択するのが一つの方法です。漠然と様々な統計的手法を使ってみるのでは、時間や費用ばかりかかります。

IoT でゴミ箱のゴミ量を測定して、一杯になったゴミ箱だけゴミを回収をすることにより、回収費用の節減

やゴミが溢れてゴミ箱が使えなくなることを防ぐ、とします。送られてきたゴミ量から今ゴミ箱が溢れているかどうかの判定は容易でしょう。でも溢れたからと行って即座にゴミ回収には出発できないでしょうし、回収の順番によっては回収できるのはだいぶ先になることもあるでしょう。そうすると送られてきたゴミ量から、回収に行ける頃にちょうど溢れるゴミ箱が分かればよいことになります。そうなると過去のデータから予測ができるようにしなければいけません。ゴミ箱の設置場所、時間帯、曜日、季節などによって捨てられるゴミの量は変わるでしょう。予測ができるようになるまでに、それなりの量のデータが必要になります。また曜日や季節は、センサーによって得られる情報ではありませんが、より正確な予測には欠かせないと思われます。

これからゴミを回収すべきゴミ箱がきちんと予測できたら終わりではありません。できるだけ短い経路で回らないと時間や燃料の無駄が生じます。ところがこの最適な経路を求める問題は、古くから「巡回セールスマン問題」と呼ばれるもので、ゴミ箱の数が数十個のレベルでも莫大な計算量[*3]が必要となり現実的な時間で正解[*4]は得られません。現実的には最適解の 1.5 倍以上は悪くならないけど高速に求まる、と言うような解法があるのでそちらで我慢するとか、量子コンピュータであればすぐ答えが出るとか言われています。

4.5 環境センサーからの情報の取り込み

IoT 機器の多くは節電のために、適当な時間間隔でサーバーにデータを送ってきます。サーバーから要求を送ってデータをもらう形にすると、IoT 機器は常に要求待ちのために電力を消費します。送信の時だけ目覚めるようにして、ほとんどの時間を死んだようになって節電するようにして、時には数百倍も電池の寿命を延ばします。

今回利用する環境センサーは 5 秒毎に気温、湿度、二酸化炭素濃度、明るさを測定して表示します。そして 20 秒毎に mars にデータを送ります。mars は送られてきたデータに時刻情報を追加したものをデータベースに保存します。実際の IoT によるセンサー情報の収集についても同じように、センサー情報をまとめるサーバーがよく使われます。まとめるサーバーは処理能力は不要ですが、ダウンするとその間のセンサー情報が欠落するので、サーバーが停止しないように十分な対策を行います。そして別のサーバーで集めたデータの処理、その結果をまた別の Web サーバーから公開するという形になります。

手順としては次のようになります。

1. データを受け取るプログラムを作成します。データは GET の方式で送られてくるので、`$_POST['date']` で時刻、`$_POST['room']` で環境センサーのある部屋の名前、`$_POST['temp']` で室温、`$_POST['humi']` で湿度、`$_POST['CO2']` で二酸化炭素の濃度、`$_POST['bright']` で明るさを受け取ることができます。受け取ったデータはファイルへ入れるか、データベースに入れましょう。
2. プログラムを登録する際は `https://mars.mgt.sugiyama-u.ac.jp/Sensor/` へアクセスします。「送り先の設定」のリンク先の「送り先登録」のところに、作成した受信プログラムの URL の一部を入力して、登録をクリックします。登録を削除したい場合は、同じく受信プログラムの URL の一部を入力して、削除で削除できます。
3. 最初のページに最近受信したデータが表示されるので、これと比較することによって自分のプログラムが正しく受信できたかどうか確認することができます。表示内容は自動的に更新はされないので、適宜ブラウザの更新ボタンをクリックしてください。

環境センサーの時刻は特に校正されていないので、20 秒毎と言っても 1 分間に必ず 3 回データが来るとは限りません。よって 2〜4 回データが来るものとして、時刻情報をもとに 1 分間の平均を記録するようにします。割り算は難しいと言うのであれば、一番最初か最後のデータを採用すると言う手もあります。一般的には平均が望ましいとされますが、平均は極端な値に引きずられるという性質があります。つまりノイズによる小

[*3] ゴミ箱の数が n ならば n!に比例する時間がかかります。
[*4] 例えば明日の天気予報の計算に、1 日以上かかるのでは意味がないですよね。

さな変動は平均して均すことによって減らすことができますが、センサーの誤動作による極端な値が混ざった場合はうまく行きません。よく用いられるのは 5 つの測定値から、最大と最小を除いた 3 つの平均を採用すると言う方法です。また中央値を採用することもあります。

4.6 グラフの描画

web ページにグラフを描く方法は色々あります。いつも同じ内容のグラフで良ければ、Excel でグラフを作成し、それを画像ファイルにして、HTML の Img タグで表示しても良いわけです。ただこの方法は、内容を変更するには人手が必要です。IoT などを利用して刻々と変わるデータをグラフの形で示したい場合は、次のような方法が考えられます。

- PHP でグラフ画像ファイルを作成する。その画像を HTML の Img タグで表示する。
- PHP でグラフ作成に必要なデータを用意し、JavaScript でそのデータをグラフの形で表示する。

データの更新が 1 日に 1 度程度であれば、PHP でグラフ画像を作成するのが良いでしょう。グラフ画像ファイルを残しておけば、アクセスが多い場合も再処理の必要がありません。また JavaScript の実行に問題があっても、画像の表示に問題があるブラウザはないと思います。毎分のようにデータが更新されるような場合は、グラフ画像を残しても無駄になる可能性が高いので JavaScript でも良いかもしれません。グラフの画像ファイルと JavaScript のファイルのどちらが大きくなるかは場合によりますが、大きなグラフでは画像ファイルの方が大きくなるかもしれません。

ここでは JavaScript を使用する方法を紹介します。既に多くの JavaScript によるグラフ作成のためのライブラリが公開されていますが、ここでは比較的紹介する web ページの多い「Chart.js」[*5]を取り上げます。

1. <Head>と</Head>の間に以下を入力します。

```
<Script src="chart.min.js"></Script>
```

これで chart.js のコメントなどを除いたものを取り込みます。https://cdnjs.com/libraries/Chart.js によると、現在 (2022 年 11 月) の最新のものは 4.0.1 でした。より数字の大きいものの方が、新しいのでバグがなくなり、機能が増えていると思いますが、使い方が変わる事もあるので、以下の記述は ver. 3 のものです。最新のものを無理に追い求める必要はないでしょう。また検索すると使用例が数多く見つかりますが、バージョンが違う例には注意しましょう。

2. グラフを表示するところに以下を入力します。

```
<Div style="width: 400px; height: 200px">
<Canvas id="myGraph"></Canvas>
</Div>
```

グラフの大きさを Div の方で設定します。Div がない場合は、ブラウザの横幅一杯のグラフが表示されます。グラフの縦横比は折れ線グラフや棒グラフでは 1:2 になります。四角いグラフを作成したい場合は Canvas の方に「 width=”200” height=”200”」のような指定を追加します。「myGraph」の部分は適当な名前で構いません。また一つの web ページに複数のグラフを入れる場合は、必ず相異なる名前にします。

3. Canvas タグの後に以下のようなグラフの定義を入れます。

[*5] https://www.chartjs.org/、「とほほの WWW 入門」の「ライブラリ」のところに説明のページがあります。https://www.tohoho-web.com/ex/chartjs.html

```
<Script type="text/javascript">
   new Chart(document.getElementById("myGraph"), {
      type: タイプ,
      data: { データ },
      options: { オプション }
    });
</Script>
```

「タイプ」のところに表 4.2 のようなグラフの種類が入ります。この他にはレーダーチャート、散布図、ドーナツチャート、鶏頭図[*6]、バブルチャート、面グラフ、混合チャートなどが可能なようです。

表 4.2 グラフの種類

type に指定するもの	意味
'line'	折れ線グラフ
'bar'	棒グラフ
'circle'	円グラフ

「データ」のところには次のような形でグラフのデータが入ります。

```
labels: [ 見出しのデータ ],
datasets: [
     { 1 つ目の系列のデータ },
     { 2 つ目の系列のデータ }
],
```

「見出しのデータ」の部分には x 軸の下に並ぶものが入ります。例えば「'1 月', '2 月', '3 月', '4 月'」のようなものです。「1 つ目の系列のデータ」のところには、次のような形で系列のデータが入ります。もし線が一本の折れ線グラフであれば「2 つ目の系列のデータ」の部分は不要ですし、線が 3 本以上の折れ線グラフでは同じ形で「3 つ目の系列のデータ」を追加します。

```
label: [ 系列の見出し ],
data: [ 系列の数値 ],
色の指定
```

「系列の見出し」には、例えば「' 売上'」のようなものが入ります。また「系列の数値」には、例えば「100, 150, 90, 120」のように「,」で区切った数値が並びます。「色の指定」は表 4.3 のような指定を必要な数だけ「,」で区切って並べます。

表 4.3 グラフの色の指定の例

指定例	意味
borderColor: 色	線の色の指定。棒グラフでは塗りつぶす色より濃い色を指定すると綺麗。
backgroundColor: 色	塗りつぶす色の指定。半透明にすると綺麗。円グラフでは「,」で区切って色を数字の数だけ指定する
borderWidth: 1	線の太さを数字で指定する

[*6] 円グラフの一種だが、量の大きさを扇形の開く角度ではなく、半径の長さで示すもの。

「色」の部分には「'rgba(255,0,0,1)'」のように赤 (0〜255)、緑 (0〜255)、青 (0〜255)、透明度 (0〜1) を数字で指定する方法があります。透明度は 0 の時は完全に透明なので背景の色のままになります。0.5 にすると指定した色と背景の色が半々に混ざります。1 にすると背景に関係なく指定した色になります。指定した色にするのであれば、「'rgb(46,106,177)'」のようにしてもよいし、HTML と同様に 16 進数で「"#BB5179"」のように指定することもできます。
「オプション」のところにグラフの形状などの細かい設定が入ります。

```
plugins: {
   title: {
      display: true,
      text: グラフのタイトル
   }
},
scales: { 軸の設定 }
```

「グラフのタイトル」のところは、「' 売上の推移'」のような形で指定します。折れ線グラフや棒グラフでは「軸の設定」をしますが、円グラフには軸がないので、「scales: 〜」を丸ごと省略して構いません。「軸の設定」は次のような形になります。

```
y: {
   max: 目盛りの最大値,
   min: 目盛りの最小値,
   ticks: {
      stepSize: 目盛りの刻む幅,
      callback: 目盛りの数値の出し方
   }
}
```

「y」が縦軸を示しています。散布図では横軸も「x」として指定します。「目盛りの最大値」は縦軸の一番上の数字、「目盛りの最小値」は縦軸の一番下の数字、「目盛りの刻む幅」は例えば 10 を指定すれば、10、20、30 と言うような目盛りになります。これらの設定は Chart.js にお任せでよければ全て省略可能です。「目盛りの数値の出し方」は、数字に何かを付けて目盛りに出したい場合に指定します。例えば数字に「円」を付けて出したいのであれば、「`function(value,index,values){return value+'円';}`」のように指定します。

説明が重層的で分かりにくいので簡単な例を示します。リスト 1 は折れ線グラフの例です。これによって図 4.1 のようなグラフになります。

リスト 1 折れ線グラフの例

```
1  <HTML lang="ja">
2  <Head>
3  <Meta charset="UTF-8">
4  <Title>折れ線グラフのテスト</Title>
5  <Script src="chart.min.js"></Script>
6  </Head>
7  <Body>
8  <Div style="width: 600px; height: 300px; border: 1px solid black;">
9  <Canvas id="myGraph"></Canvas>
10 </Div>
11 <Script type="text/javascript">
12    new Chart(document.getElementById("myGraph"), {
```

```
13       type: 'line',
14       data: {
15          labels: ['9/1', '9/2', '9/3', '9/4', '9/5', '9/6', '9/7'],
16          datasets: [
17             {
18                label: '最高気温',
19                data: [31.7, 27.7, 24.7, 26.4, 30.7, 31.0, 27.3],
20                borderColor: 'rgba(255,0,0,1)',
21                backgroundColor: 'rgba(255,0,0,0.5)'
22             },
23             {
24                label: '最低気温',
25                data: [24.8, 24.8, 22.9, 22.2, 21.8, 21.9, 21.6],
26                borderColor: 'rgba(0,0,255,1)',
27                backgroundColor: 'rgba(0,0,255,0.5)'
28             }
29          ]
30       },
31       options: {
32          plugins: {
33             title: {
34                display: true,
35                text: '2021年　名古屋市の気温',
36             }
37          },
38          scales: {
39             y: {
40                max: 35,
41                min: 15,
42                ticks: {
43                   stepSize: 5,
44                   callback: function(value, index, values){ return value+'度'; }
45                }
46             }
47          }
48       }
49    });
50 </Script>
51 </Body>
52 </HTML>
```

リスト 1 について補足すると：

- 8 行目の Div に border の設定を付けて、グラフの周りに線が引かれるるようにしています。
- 15 行目のところは固定であればこのままで良いのですが、データによって変わるのであれば PHP で出すようにします。もし$date に「'9/1', '9/2', '9/3', '9/4', '9/5', '9/6', '9/7',」のようなものが入っていれば、次のような感じです。

```
labels: [<?= $date ?>],
```

- 20 行目で色として赤の成分しか指定していないので、線は赤になります。透明度も 1 なので背景の色の影響はありません。
- 21 行目では透明度が 0.5 なので、背景の影響を受けて薄い赤になります。折れ線グラフではこの色は凡例のところにしか使われていませんが、棒グラフではこの色が棒の色になります。
- 19 行目と 25 行目も実際の利用の際には、PHP でその内容を出すことになるでしょう。

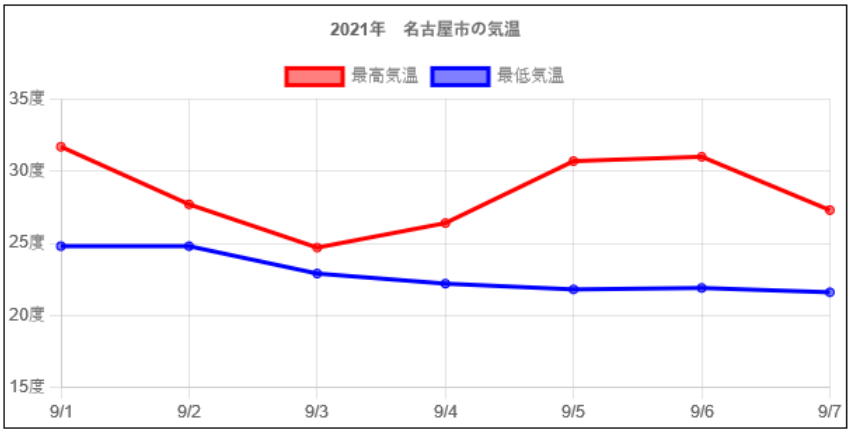

図 4.1　折れ線グラフの例

4.7　グラフ化するデータの作成

IoT を利用してデータを自動的に収集するとすぐ膨大なデータになります。環境センサのデータは 1 分間に 1 件しか来ませんが、1 時間で 60 件、1 日で 1,440 件、1 週間で 10,080 件、1 ヶ月で約 43,200 件、1 年で約 525,600 件になります。グラフでこれを示す場合、目的に合わせてデータの一部を用いるのが普通です。例えば大学では毎週同じ曜日に同じ科目が同じ教室で行われます。よって 1 週間分のグラフを見ることにより、人数が多くて二酸化炭素濃度が高くなる授業が分かるでしょう。そこでその授業の教員に換気に注意するように伝えて、その効果を確認することを考えると、前の週のグラフも見えるようにすると良いでしょう。

またグラフを表示する際のドットの数よりデータの数が多い場合は、データをまとめることを考える必要があります。1 週間のグラフの場合約 10 万件ありますので、幅が 500 ドットならば 200 件を 1 つのデータ (代表値) にする必要があります。代表値としては平均がよく使われますが、最大値、最小値、中央値、最頻値などの方が良い場合もあります。

部分的に取り出す SQL 文

主キー (id) が連続している場合を仮定すると、id の値の範囲を条件にすることによりデータの一部を取り出すことができます。例えば次のようにします。

```
SELECT * FROM sensor WHERE id BETWEEN 100 AND 199
```

これで id が 100〜199 の 100 件のデータが取り出せますが、 それが想定した時間の幅に対応するかどうかわ分かりません。機器やネットワークのトラブルなどでデータに欠損がある場合、もっと長い時間に対応する 100 件のデータになります。例えば毎分 1 件収集したデータであれば、100 件のデータは通常 100 分に対応しますが、途中トラブルのため 3 日間データが収集できなかった場合は 3 日 +100 分に対応します。

よって毎分 1 件のデータを 1 日分取り出したい場合、id で 1,440 件と言う範囲を指定するより日付のデータがあればそれを利用した方が良いでしょう。date に通算秒数が入っており、1669474800 は 2022 年 11 月 27

日 0 時 0 分 0 秒の通算秒数、24 時間は 86,400 秒なので次のようにして 2022 年 11 月 27 日の 1 日分のデータを取り出すことができます。

```
SELECT * FROM sensor WHERE date BETWEEN 1669474800 AND 1669474800+86400-1
```

ついでに co2 に二酸化炭素の濃度が入っていおり、2022 年 11 月 27 日の 1 日分の二酸化炭素濃度の平均値を求めたい場合は、次のようにして求めることができます。

```
SELECT AVG(co2) AS CO2 FROM sensor
                        WHERE date BETWEEN 1669474800 AND 1669474800+86400-1
```

平均値は「CO2」と言う別名を付けているので、この名前で取り出します。`$data['AVG(co2)']` のようなことはできません。1 日分の平均を 100 個使ってグラフを作るという場合は、この検索を 100 回繰り返すことになりますが、何度もグラフを作るのであれば、1 日分の平均を別のテーブルに入れておいた方が良いでしょう。その場合グラフを作る際でなく、受け取ったデータが次の日のデータに変わった時点で前の日の平均を別のテーブルに入れる方が良いでしょう。

余り多くの件数のデータの平均を求めるのでなければ、次のように毎回 SQL でやっても良いでしょう。これは 1 日分のデータから 30 分毎の平均を求めています。

```
SELECT date, AVG(co2) AS CO2 FROM sensor
                      WHERE date BETWEEN 1669474800 AND 1669474800+86400-1
                      GROUP BY date/1800
```

「date/1800」で通算秒数の date を 30×60 で割ることにより、date を 30 分単位の数値に直しています。date は INTEGER (整数値) なので割り算の結果も小数点以下が切り捨てられます。その結果同じ数値になるデータが同じグループになり、グループごとの二酸化炭素濃度の平均が求められます。1 日の平均二酸化炭素濃度と違って、こちらは 48 個の平均値が出てくるので、date の値も取り出しています。同じグループに属する 30 件のデータのどの date が出て来るのかですが、グループ内の一番最後のデータの date が出てくるようです。

SQL で検索条件や並び替えを行なって得られた結果の件数が多すぎて一つの表やグラフに入り切らない場合、他のページに続きを出すことになります。例えば最初のページに 100 件出したので、続きのページは 101 件目から出したい場合です。この時検索や並び替えがあるので id は当てになりません。次の例は二酸化炭素濃度の大きい順に最初の 100 件を取り出すものです。

```
SELECT * FROM sensor ORDER BY co2 DESC LIMIT 100
```

LIMIT の後の数字で取り出す件数を指定します。もしこれより少ない件数しかなかった場合は、その全てが出てくるだけでエラーにはなりません。さらに 101 件目以降を 100 件取り出したい時は次のようにします。

```
SELECT * FROM sensor ORDER BY co2 DESC LIMIT 100 OFFSET 100
```

OFFSET の後の数字で飛ばす件数を指定します。もしこれより少ない件数しかなかった場合は、何も出ないだけでエラーにはなりません。

4.8 超えたらメールを送る

センサーから送られてきた値に何か問題があった場合は、それを誰かに知らせなければなりません。既に取り上げたメールを使って知らせるのが一つの方法です。問題があるかどうかの判定は、複数の条件が複雑に絡

まっている場合もありますが、単純に数値が基準を上回ったら問題という例が多く見られます。例えば河川の水位が危険水位を超えたら避難しなければなりません。環境センサーの場合も、二酸化炭素濃度が例えば 2,000ppm を超えたら「急いで換気しましょう」と言うメールを送るなどが考えられます。もし教室にある換気扇がネットワークにつながっていてリモートコントロール可能であれば、メールを送る代わりに換気扇を動かした方が良いのでしょうが。

「2,000ppm を超えたら」と言う条件の if 文でメールを送るという方法では問題を生じることがあります。まず毎分新しいセンサーの結果が来るので、10 分間 2,000ppm を超えたら 10 通メールを送ることになります。状況が改善されるまで、ひたすらメールを送るのは多分迷惑です。そこで一度 2,000ppm を超えたらメールを送り、その後超えたままの場合はメールを送らないようにする必要があります。ところがセンサーの数字は通常微小なふらつきを伴います。2,001 → 1,999 → 2,001 → 1,999 → 2,001 のように境界をまたぐ形でフラフラするとその度にメールが行くことになります。

このような問題に対処するために基準を 2 つ用意します。例えば 2,000ppm を超えたら問題状態、1,900ppm を下回ったら問題状態解除のようにします。これならば一度 2,000ppm を超えて問題状態となれば、センサーの数字が多少ふらついても問題状態解除にはなりません。ただセンサーから 1,950ppm と言う数字が届いた場合、今は問題状態なのかはこの数字だけでは判断できません。つまりこれまで問題状態であれば問題状態ですし、これまで問題状態でなければ問題状態ではありません。ずっと 1,900ppm〜2,000ppm の間をふらふらしていた場合は、それ以前まで遡って調べないと判定できません。毎回遡って調べるのも面倒なので、センサーの値を入れるテーブルに問題状態かどうかも入れておくと良いでしょう。

4.9 web ページの自動更新

通常の web ページは一旦表示されると利用者が何かしない限り動くことができません。よって IoT から随時データが更新されても、それを伝えることができません。ここでは比較的容易に実現できる web ページの自動更新の方法を紹介します。これ以外には、web サーバーとつながったままにするとか、JavaScript でこっそり web サーバーからデータをもらう方法などがあります。

ブラウザから 10 秒毎に更新の要求を出させるには、<Head>と</Head>の間に次のような指示を入れます。

```
<META HTTP-EQUIV="Refresh" CONTENT="10">
```

これの応用として、次のようにすると 10 秒後に別のページを見せることも可能です。

```
<META HTTP-EQUIV="Refresh" CONTENT="10;URL=別のページの URL">
```

これらを PHP で行う時は、次のように header() を使用します。通常のタグ等の出力より先行して実行する必要があるので注意します。こちらはブラウザでソースの表示を行っても出てきません。

```
header("Refresh: 10");
```

```
header("Refresh: 10;URL=別のページの URL");
```

5. スクレイピング

スクレイピング (Web scraping) は既存の Web ページから情報を取り込む技術のことです。情報を取り込むには、

1. web ページの内容を取り込む
2. 取り込んだ内容から必要な部分を取り出す

ができる必要があります。その際に問題となるのは、

- URL が「https://」で始まる場合、SSL の設定ができていないと内容を取り込めない。mars の場合は既に SSL が設定されているので問題ありませんが、ちょっと面倒な話です。
- URL が決まっていない場合がある。毎日新しい情報を提供している場合、日付が URL の中に埋め込まれていることがあります。このような場合、大抵入り口となる web ページは固定の URL となっているので、そこにまずアクセスして、今日の URL を求める必要があります。
- 予告なしに URL が変わることがあります。そして従来の URL へのアクセスができなくなるとエラーが出るので気がつくのですが、従来の URL へのアクセスは可能だが内容の更新がされなくなる、と言うような場合は気づきにくいので困ります。
- ページの構造が時々変わる場合がある。コロナのワクチン接種状況のページが、医療関係者だけ、高齢者も追加、一般の人も追加と対象が広がるに従って変わった例があります。さらにその際に URL も変更になってました。
- 必要な情報が HTML のタグの中に埋もれているので、探し出さなければならない。唯一のタグに囲まれていればすぐに見つかりますが、大抵は無数にある`<Td>`〜`</Td>`や`<Div>`〜`</Div>`の一つに囲まれているのが普通です。
- 必要な情報が JavaScript の実行により動的に生成されている場合は、JavaScript のソースが得られても情報自体は得られません。このような場合は、ブラウザを動かしてその表示内容を取り出すような大変高度な技術が必要になります。

このような様々な技術的な問題もありますし、著作権の問題や多くの人がスクレイピングを行ったために、スクレイピングが禁止となったサイトや情報提供をやめてしまったサイトもあります。通常のブラウザによるアクセスと変わらないような頻度や量であれば問題にはなりにくいと思います。

5.1 Web ページの取り込み

PHP で web ページの取り込みをするのは、file 関数を用いれば簡単にできます。

```
$lines=file("https://www.mgt.sugiyama-u.ac.jp/");
foreach ($lines as $line) {
   echo str_replace("<","&lt;",$line),"<Br>\n";
}
```

オプションとして「FILE_IGNORE_NEW_LINES」を file() で指定すると一つの変数に取り込まれますが、この例のように指定しない場合は、取り込んだ結果は配列型変数になるので、foreach を利用して 1 行ずつ取り出しています。取り出したものをそのまま echo で表示すると、含まれている HTML のタグが働いてしまうので、「`<`」を全て「`&lt;`」に置き換えています。

ブラウザで URL を入力するだけで表示される web ページなのに、このやり方では「HTTP request failed!」で拒絶されるところが増えています。そのような場合は、リスト 2 のようにブラウザのふりをすると応答してくれることがあります。file() では情報が追加できないので、file_get_contents() を使用しています。

リスト 2 ブラウザのふりをする例

```
1  $url="https://sample.php";
2  $options = array(
3     'http' => array(
4        'header' => "User-Agent: Mozilla/5.0 (Windows NT 6.2) AppleWebKit/537.1 ".
5                    "(KHTML, like Gecko) Chrome/21.0.1180.75 Safari/537.1"
6     )
7  );
8  $page=file_get_contents($url,false,stream_context_create($options));
```

一方何か入力をして「送信」をクリックしたら、結果の web ページが出てくるような場合があります。入力内容を method として GET で送っている場合は、その内容が URL に反映されるので良いのですが、POST で送っている場合や cookie を利用している場合 (SESSION 変数を使用している場合も含む) は、これらの情報を付けた上で取り込まないと、エラーメッセージのみが返ってくる場でしょう。リスト 3 はそのような web ページへの対応方法の例です。

リスト 3 POST のデータや cookie を送る例

```
1  $url="https://sample.php";
2  $param=array(
3     'id' => $id,
4     'pass' => $pass);
5  $options = array(
6     'http' => array(
7        'method' => 'POST',
8        'header' => "Cookie: ".$cookie."\r\n".
9            "User-Agent: Mozilla/5.0 (Windows NT 6.2) AppleWebKit/537.1 ".
10           "(KHTML, like Gecko) Chrome/21.0.1180.75 Safari/537.1\r\n".
11           "Content-type: application/x-www-form-urlencoded",
12       'content' => http_build_query($param),
13    )
14 );
15 $page=file_get_contents($url,false,stream_context_create($options));
```

- 3〜4 行目で POST で送るデータの用意をしています。「id」と「pass」が名前で、内容は$id と$pass に入っていると仮定しています。
- 8 行目で$cookie に入っているものを cookie として設定していますが、$cookie の内容は、その前のアクセスの際の web サーバーからの応答の中から取り出しておく必要があります。cookie を使用していない場合は、「`"Cookie: ".$cookie."\r\n".`」の部分は削除します。応答から cookie を取り出す方法の詳細は省略しますが、`file_get_contents()` を使用してアクセスした後に`$http_response_header` という変数に相手から送られてきたヘッダー情報が入るので、この変数から取り出します。
- 9〜10 行目ではブラウザを偽装しています。これがないとエラーを返してくるサイトがあります。
- 15 行目で$page 変数に web ページの内容が入ります。

5.2 HTML の解析

変数の中に取り込んだ web ページの内容から必要な情報を取り出すためには、情報のある場所を示すものを目印にします。例えば「現在のポイント数」と言う文字の後に取得したい情報があるならば、mb_strpos() などで「現在のポイント数」の位置を求めます。また web ページの内容には多数の HTML のタグが含まれてい

ます。これを目印に取り出すことが考えられます。目的の情報が 3 つ目の表の 1 行目にあるならば、3 つ目の「<Table>」の後の最初の「<Tr>〜</Tr>」にあるはずです。これを単純に「<Tr>」だけを目印に探すのは無理で、「<Table>」を探し、次に「<Tr>」を探すというようなステップを踏む必要があります。

web ページの内容を HTML のタグに分解してくれるライブラリがあります。ここでは「PHP Simple HTML DOM Parser[*7]」を紹介します。他にも「phpQuery[*8]」などがあります。「PHP Simple HTML DOM Parser」を利用するには、まずこのライブラリをダウンロードする必要があります。「https://sourceforge.net/projects/simplehtmldom/files/simplehtmldom/」をブラウザで開くと、一覧が表示され様々なバージョンのものがあることが分かります。2021 年 7 月の時点では「2.0-RC2」が最新のようですが、アクセス数を見ると「1.9.1」の方が多いのでそちらにします。「1.9.1」をクリックするとこのバージョンのページが表示されるので「simplehtmldom_1_9_1.zip」をクリックしてダウンロードします。Windows の場合ダウンロードしたファイルを右クリックしてメニューで「全て展開」を選択すると解凍することができます。解凍してできたフォルダーの中にある「simple_html_dom.php」を peditor でアップロードします。

「PHP Simple HTML DOM Parser」の簡単な使用例はリスト 4 のようになります。

リスト 4 PHP Simple HTML DOM Parser の簡単な例

```
1  <HTML lang="ja">
2  <Head>
3  <Meta charset="UTF-8">
4  <Title>簡単な例</Title>
5  </Head>
6
7  <Body>
8  <?php
9  include "simple_html_dom.php";
10 $page=file_get_contents("https://www.mgt.sugiyama-u.ac.jp/");
11 $html=str_get_html($page);
12 echo $html->find("body",0)->plaintext;
13 ?>
14 </Body>
15 </HTML>
```

- 9 行目でライブラリのファイルを取り込みます。
- 10 行目は web ページの内容を$page に取り込んでいます。ここの部分は他のサイトのページを取り込む場合、前章の内容と同様に変更する必要があります。
- 11 行目で web ページの内容を HTML のタグで分解したものを$html に入れています。
- 12 行目では 1 番目の「Body」タグの内容を取り出し、その中の文字の部分のみを取り出して、echo で表示しています。find() の中のゼロが 1 番目を示しています。

「Body」タグの中の「H1」タグの中身を取り出したい場合は次のようにします。

```
echo $html->find("body",0)->find("h1",0)->plaintext;
```

このように find() を重ねることにより入れ子になった HTML のタグの中の方にあるものを取り出すことができます。また HTML のタグで挟まれたものではなく、タグの中で指定したものを取り出すこともできます。

[*7] https://simplehtmldom.sourceforge.io/

[*8] https://code.google.com/archive/p/phpquery/downloads

```
echo $html->find("body",0)->find("a",0)->href;
```

これで A タグで指定した href の値、つまりリンクの URL を取り出すことができます。途中の HTML のタグは、次のように省略することができます。ただしその場合何番目かの数字が変わる可能性があります。

```
echo $html->find("a",0)->href;
```

Div タグはよく使われていますが、その中身に合わせて id や class の指定がされていることがよくあります。例えば「<Div id="menulist">」の中の文字を表示するならば、

```
echo $html->find("div[id=menulist]",0)->plaintext;
```

のようにします。class の場合も同様です。

次のように foreach と組み合わせて、同じタグの内容を全て取り出すことも可能です。foreach の中の find() の中の指定がタグだけになっているところがこれまでの使い方と違います。タグだけ指定すると、指定されたタグを全て取り出します。そして foreach で全てから 1 つずつ取り出しているのです。

```
foreach($html->find("a") as $e) {
   echo $e->href,"<Br>\n";
}
```

これで web ページ中の A タグに指定された URL が全て表示されます。

同じタグの内容を全て取り出すのではなく、出てきたタグを順番に扱いたい場合はリスト 5 のようにします。

リスト 5 タグを出てきた順に扱う例

```
1 $parent=$html->find('h3',0)->parent();
2 foreach ($parent->children() as $tag) {
3    if ($tag->tag=='h3') {
4       H3 のタグに対する処理
5    }
6    if ($tag->tag=='table') {
7       Table のタグに対する処理
8    }
9 }
```

1. 1 行目では 1 つ目の H3 タグの親となるタグを$parent に入れています。<X>〜<Y>〜</Y>〜</X>のようにタグ X の中にタグ Y が含まれる時、タグ Y の親はタグ X になります。またタグ X の子はタグ Y になります。表示されているタグを全てという場合は、$parent に body タグを入れます。
2. 2 行目の繰り返しでは、1 つ目の H3 タグの親タグの子のタグを順番に取り出して$tag に入れます。
3. 3 行目と 6 行目で取り出したタグが H3 や Table かどうか調べています。

6. API の利用と XML・JSON

スクレイピングでは web ページからデータを取り込むことをやりましたが、むしろ積極的にデータを提供してくれるサイトも少なくありません。もちろん無料とは限りませんが。ここではデータだけでなくサービスの提供をしてくれる場合もある API (Application Programming Interface) と、データの提供の際によく使われる XML (Extensible Markup Language) と JSON (JavaScript Object Notation) を取り上げます。API は web に限らず様々な場面で使われている技術で、余り決まった形がないと思いますが、XML や JSON は扱いに慣れておけばあちこちで役に立ちます。

6.1 XML とは

データの受け渡しに、XML と言うものもあります。XML はなんとなく HTML と似たような名前ですが、その内容も自分で勝手にタグを決められる HTML と考えてほぼ間違いありません。例えば次のような形です。

```
<?xml version="1.0" encoding="utf-8"?>
<student>
  <item>
    <id>1</id>
    <name>Maki</name>
  </item>
  <item>
    <id>2</id>
    <name>Mika</name>
  </item>
</student>
```

タグの名前は中身に合わせて適当に決めてよいのですが、必ず閉じるタグが必要です。PHP にはスクレイピングで紹介した「PHP Simple HTML DOM Parser」のような機能が組み込まれていて、XML 形式のデータから必要な部分を取り出したり、逆に XML 形式のデータを作ることもできます。例えば上記の XML の例が`$xdata` に入っている時、次のようにして各要素を取り出すことができます。

```
$xml = simplexml_load_string($xdata);
echo $xml->item[0]->id;   // 1 が出る
echo $xml->item[0]->name; // Maki が出る
```

6.2 JSON とは

JSON は元々 JavaScript 用のデータ交換用の形式なので、JavaScript であればそのまま扱える形をしています。PHP などで扱う場合も XML より字数が少なくて済むなどの特徴があります。文字コードとしては UTF-8 を使うことが基本なので XML のように指定は不要です。先程の XML の例と同じ内容を JSON で表現すると次のようになります。

```
[
   {"id": 1, "name": "Maki"},
   {"id": 2, "name": "Mika"}
]
```

JavaScript ではこの形をそのまま変数の内容の定義に使うことができますので、特にライブラリーなどを用いなくても利用可能です。PHP で JSON を利用する場合は、そのまま変数には入らないので、次のように `json_decode()` を利用して変換します。

```
$json = '[ {"id": 1, "name": "Maki"},  {"id": 2, "name": "Mika"} ]';
$data = json_decode($json,true);
foreach ($data as $item) {
   echo $item['name'],"<Br>";   // Maki と Mika が表示される
}
```

この場合、JSON の記述は [ ] で囲まれているので配列になりますので、foreach を利用して$data から順番に$item に取り出しています。取り出した$item も配列型変数なので [ ] で取り出すものを指定します。

6.3 API サービスの例

以下の一覧は「個人でも使える！おすすめ API 一覧」(https://qiita.com/mikan3rd/items/ba4737023f08bb2ca161) の一部を元にしたものです。「個人でも使える！おすすめ API 一覧」には、もう少し詳しい説明と、これらを使ってみた例のページへのリンクがあります。

世の中の多くの API を利用するにはユーザー登録が必要です。ユーザー登録により API サービスを提供する側は、使用料を請求したり、API の仕様が変わった際に連絡を取ったりします。API サービスは、有料でも大抵使用回数に制限があります。ユーザー登録によりユーザーごとの制限管理が行えるので、各ユーザーは認められた回数の使用が確実に行えます。

サービスの一部を無料で提供していても、ユーザー登録の際にクレジットカードの登録を求められる場合もあります。例えば Google Cloud API などです。サービスの無料提供は有料サービスのお試し、とほとんどの API 提供側は考えているので仕方ないのかもしれませんが、クレジットカードの情報は直接お金に絡むので、本当に API サービスの利用を考えているのでなければ、登録は避けたいところです。

- Google YouTube Data API
  YouTube を検索して動画・再生リスト・チャンネルなどの一覧などの取得や更新ができる。
  https://developers.google.com/youtube/v3/docs/
- Google Maps JavaScript API
  WEB 上で地図を表示してピンを立てたり経路案内を表示できる。
  https://developers.google.com/maps/documentation/javascript/tutorial
- Google Cloud API
  コンピューティング API、ストレージとデータベースの API、ネットワーキング API、データ分析 API、機械学習 API、管理ツール API、オペレーション API、セキュリティと ID の API、マネージド インフラストラクチャ API などがある。
  https://cloud.google.com/apis?hl=ja
- Microsoft Computer Vision API
  Microsoft の画像認識 AI を使える。
  https://azure.microsoft.com/ja-jp/services/cognitive-services/computer-vision/
- Microsoft Face API
  顔認識 AI を使い、顔の識別や特徴・感情の分析などができる。
  https://azure.microsoft.com/ja-jp/services/cognitive-services/face/
- docomo API
  言語解析や画像認識などができる。
  https://dev.smt.docomo.ne.jp/?p=docs.api.index
- Facebook Graph API
  Facebook のユーザや Facebook ページの情報の読み取り・更新などができる。
  https://developers.facebook.com/docs/graph-api

- Instagram Graph API
  Instagram の情報を取得・更新ができる。
  https://developers.facebook.com/docs/instagram-api
- Twitter API
  ツイートの検索・取得・投稿などができる。
  https://developer.twitter.com/en/docs
- LINE Messaging API
  LINE で自動返信する bot や一方的にメッセージを送ることなどができる。
  https://developers.line.me/ja/docs/messaging-api/overview/
- DMM Web サービス
  商品情報 API や女優検索 API など DMM のサービスの検索ができる。
  https://affiliate.dmm.com/api/
- ホットペッパー API
  位置情報や様々な条件を元に、ホットペッパーで飲食店を検索できる。
  https://webservice.recruit.co.jp/hotpepper/reference.html
- ぐるなび API
  位置情報や様々な条件を元に、ぐるなびで飲食店を検索できる。
  https://api.gnavi.co.jp/api/
- Amazon Product Advertising API
  Amazon の商品情報や関連コンテンツを得ることができ、これによって Web サイトで Amazon の商品を紹介することによる紹介料の獲得が可能になる。
  https://affiliate.amazon.co.jp/gp/advertising/api/detail/main.html
- OpenWeatherMap API
  世界各地の天気や転居法を得ることができる。
  https://openweathermap.org/
- e-Stat API
  政府統計の総合窓口 (e-Stat) で提供している統計データを機械判読可能な形式で取得できる。
  https://www.e-stat.go.jp/api/
- 駅すぱあと Web サービス
  「駅すぱあと」が持つ経路検索・運賃計算などの機能や鉄道・バスなどの公共交通機関データを、 Web サイトやアプリに自由に組み込むことができる。
  https://ekiworld.net/service/sier/webservice/free_provision.html
- Stripe
  135 種類以上の通貨と支払い方法に対応しており、海外相手でもオンライン決済を実現できる。
  https://stripe.com/jp
- LINE Pay
  LINE Pay による決済を実現できる。
  https://pay.line.me/jp/developers/documentation/download/tech?locale=ja_JP
- Yahoo API
  ショッピング、 YOLP (地図)、テキスト解析、求人、ニュース、Yahoo! ID 連携、メールなどを扱うことができる。
  https://developer.yahoo.co.jp/sitemap/
- openBD
  ISBN などで書籍情報の検索ができる。

https://openbd.jp/

- Rakuten Webservice
  楽天市場系、楽天ブックス系、楽天トラベル系、楽天ブックマーク系、楽天レシピ系、楽天 Kobo 系、楽天 GORA 系などがある。
  https://webservice.rakuten.co.jp/
- NHK の番組表
  NHK の番組の検索や番組の詳細などを得ることができる。
  https://api-portal.nhk.or.jp/
- Microsoft Text Analytics
  テキストの分析ができる。
  https://api.rakuten.net/microsoft-azure/api/microsoft-text-analytics
- Rakuten Rapid API
  楽天だけでなく様々なサイトから提供されている API を検索して、それが使えるかどうかを簡単にテストし、利用することができる。
  https://api.rakuten.co.jp/docs/ja/docs/what-is-rapidapi/

6.4 天気予報データの取得

ここでは「Open Weather Map」(https://openweathermap.org/) より API を利用して、天気予報データを入手する方法について説明します。「Open Weather Map」では世界中の気象情報を得ることができます。その一部は無料で利用可能です。

「Open Weather Map」にユーザー登録するには、https://home.openweathermap.org/users/sign_in にアクセスします。すると図 6.1 のようなダイアログが表示されますので、「Create an Account」をクリックします。すると登録に関するダイアログが表示されますので、「Username」、「email」、「Password」などを入力してから、「Create Account」をクリックします。次に会社名や使用目的を聞かれます。会社名は入力しなくても良いようです。登録確認のメールが来ますので、本文中程にある「Verify Your email」をクリックします。これで登録は完了しますが、実際に使えるようになるまでは 10 分程度待つ必要があるようです。

無事登録ができたならば、先程の URL で図 6.1 を出し、「email」と「Password」を入力して「Submit」をクリックします。「API keys」のタブをクリックすると key の値が表示されるので、それを次の「キー」のところに入れます。

```
https://api.openweathermap.org/data/2.5/onecall?lat=緯度&lon=経度&lang=ja&appid=キー
```

「緯度」や「経度」は天気を調べたい場所の値を入れます。緯度や経度の値は Wikipedia などに出ています。例えば名古屋市は、北緯 35 度 10 分 53 秒、東経 136 度 54 分 23 秒にあるので、緯度としては 35.1814=(35+10/60+53/3600)、経度としては 136.9064=(136+54/60+23/3600) を指定します。「lang=ja」で天気が日本語表記になります。firefox でこの URL にアクセスすると図 6.3 のようになります。(もし多くの行が表示された場合は、「すべて折りたたむ」をクリックしてください。) 最初に緯度と経度、次に時間帯、その時間帯が世界標準時と何秒ずれているか (32,400 秒=9 時間) が表示されます。

その下に「current」とあるのが現在の情報、「minutery」が 60 分先までの分ごとの情報、「hourly」が 48 時間後までの 1 時間毎の情報、「daily」が一週間先までの日毎の情報です。ブラウザの「▶」をクリックすると畳み込まれた情報が展開されて見えるようになります。ブラウザで構造はだいたい分かると思いますが、次のように var_dump() 関数を使用すると `json_decode()` で取り出した内容がわかりやすくなります。

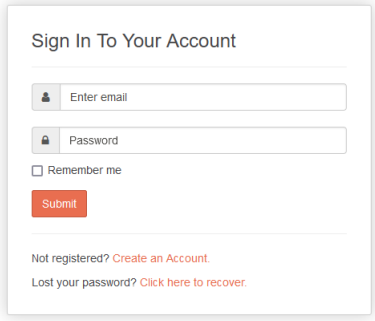

図 6.1　Open Weather Map のログイン画面

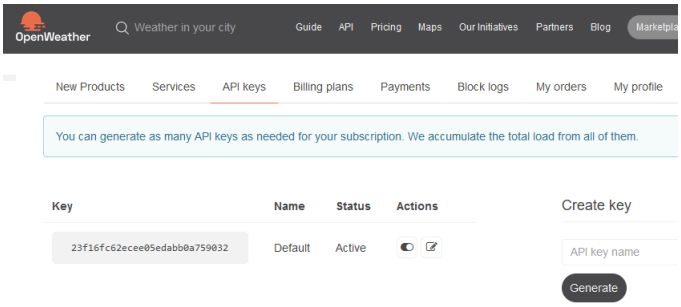

図 6.2　API キーの取得

```
$json=file_get_contents("http://api.openweathermap.org/... 中略...");
$data=json_decode($json,true);
echo $data['lat'],"<Br>\n";
$cur=$data['current'];
echo date("Y/m/d H:i:s",$cur['dt']),"<Br>\n";
echo "<Pre>\n";
var_dump($cur);
echo "</Pre>\n";
```

またこの例では`$data['current']` →`$cur` をしてから、`$cur['dt']` を取り出すというような多段階の取り出しをやっていますが、多次元配列の表記を利用すると`$data['current']['dt']` のように [ ] を複数使用して一気に取り出すことも可能です。

図 6.3　天気情報の取り込み

項目	内容
dt	通算秒数による時刻 (date() で変換可能)
temp	気温 (単位は絶対温度、273.15 を引けば摂氏になる)
humidity	湿度　(%)
pressure	気圧　(hPa)
wind_speed	風速 (m/秒)
wind_gust	最大瞬間風速 (m/秒)
main	天気詳細 (weather 内)
description	天気詳細 (weather 内)

表 6.1　天気情報の内容

7. 物体検出プログラムの利用

自動車の世界では現在 EV (Electric Vehicle) と自動運転が大きな課題になっています。後者の自動運転で欠かせない技術として高速の物体検出があります。高精度で正確な地図があれば目的地に行くことはできますが、途中の道路上に何がいるのかは分かりません。車の運転手は走行中に数秒よそ見をするだけで大事故の危険性が高まります。自動運転車は常に周囲の状況を把握しなければなりません。レーダーを使えば周囲にある物体までの距離が分かります。さらにドップラー効果を利用すれば、その物体が近づいているのか、遠ざかっているのかも分かります。これで衝突は防げるかもしれませんが、レーダーでは信号機が分かりません。赤信号の交差点に突っ込んだら大惨事になります。

画像の中のどこに何があるのかを検出する物体検出は、1970 年代から研究されていましたが、なかなかうまく行きませんでした。黒い目と赤い口から顔を検出するぐらいはなんとかなりましたが、様々な姿をした人の検出は無理でした。ところが 2012 年に Alex Krizhevsky らが画像認識 (画像分類) のコンテストである ILSVRC2012 で提出した、畳み込みニューラルネットワーク (CNN) を用いたシステムが優勝したことにより、深層学習を利用した物体検出が主流になりました。そして 3 年後の ILSVRC2015 で優勝した ResNet は人間よりも少ないエラー率を達成しました。

ここで紹介する YOLO は物体検出のシステムで、最初のバージョンが 2016 年に発表され現在も改良が続けられています。YOLO は “You only look once” から来ており、候補領域をまず抽出し、その中で検出や分類を行うのではなく、一気に抽出から分類まで行う方式である事を示しています。一気にやることにより精度の点で劣るものの、動画をそのまま処理できる速さを実現しました。自動運転での利用を考えると、人が飛び出したのを数秒後に検出したのでは間に合いません。自動運転で使用される物体検出システムは、YOLO を元に更に改良が進められています。

ここでは 2020 年 6 月に Glenn Jocher が発表した YOLO ver.5 (`https://github.com/ultralytics/yolov5`) を実際に使ってみます。

7.1 YOLOv5 による物体検出

YOLO ver.5 は python3 で書かれた PyTorch と言う名前のフレームワークで動きます。そのため YOLOv5 を動かすためには PyTorch を初め多数の関連ライブラリをインストールする必要がありますが、全て mars にインストール済み[*9]なので、YOLOv5 本体のみインストールすれば使うことができます。

1. 「教材フォルダ」から「yolov5」フォルダを丸ごと mars のデスクトップにある「www」フォルダーの中にコピーします。
2. コピーした「yolov5」フォルダをダブルクリックして開き、メニューまたはF4で「LXTerminal」を起動します。
3. 次のコマンドを入力するとサンプルデータの処理を行います。

```
python3 detect.py --source data/images/bus.jpg
```

　するとリスト 6 のようなメッセージが表示されます。14 行に「4 persons, 1 bus, 7.9ms」とありますが、画像の中に 4 人とバスを 1 台を 7.9 ミリ秒で見つけたと言う意味です。

リスト 6　YOLOv5 による画像認識

```
1 miki@mars:~/Desktop/yolov5$ python3 detect.py --source data/images/bus.jpg
2 detect: weights=yolov5s.pt, source=data/images/bus.jpg, data=data/coco128.yaml,
```

[*9] 大抵の YOLOv5 の解説では、関連ライブラリのインストールから説明されています。PyTorch などはかなり大きなライブラリなので全て入れると 4GB 程度のディスク容量を必要とします。

```
imgsz=[640, 640], conf_thres=0.25, iou_thres=0.45, max_det=1000, device=,
view_img=False, save_txt=False, save_conf=False, save_crop=False, nosave=False,
classes=None, agnostic_nms=False, augment=False, visualize=False, update=False,
project=runs/detect, name=exp, exist_ok=False, line_thickness=3,
hide_labels=False, hide_conf=False, half=False, dnn=False, vid_stride=1
YOLOv5 □ v6.2-266-g72cad39 Python-3.8.10 torch-1.13.0+cu117 CUDA:0
(NVIDIA GeForce GTX 1080 Ti, 11178MiB)

Fusing layers...
YOLOv5s summary: 213 layers, 7225885 parameters, 0 gradients
image 1/1 /home/miki/Desktop/yolov5/data/images/bus.jpg: 640x480
 4 persons, 1 bus, 7.9ms
Speed: 0.3ms pre-process, 7.9ms inference, 0.9ms NMS per image at shape
 (1, 3, 640, 640)
Results saved to runs/detect/exp
```

4. リスト 6 の最後の行に「runs/detect/exp」に結果を保存したとあるので、「runs」フォルダ、「detect」フォルダ、「exp」フォルダの順にダブルクリックをして開いていくと、bus.jpg があります。これをダブルクリックすると図 7.1 のような、検出した物体が枠線で囲われているものが表示されます。枠線に「bus 0.85」のように表示されていますが、これは物体が「bus」で確からしさが 0.85 と言う意味です。

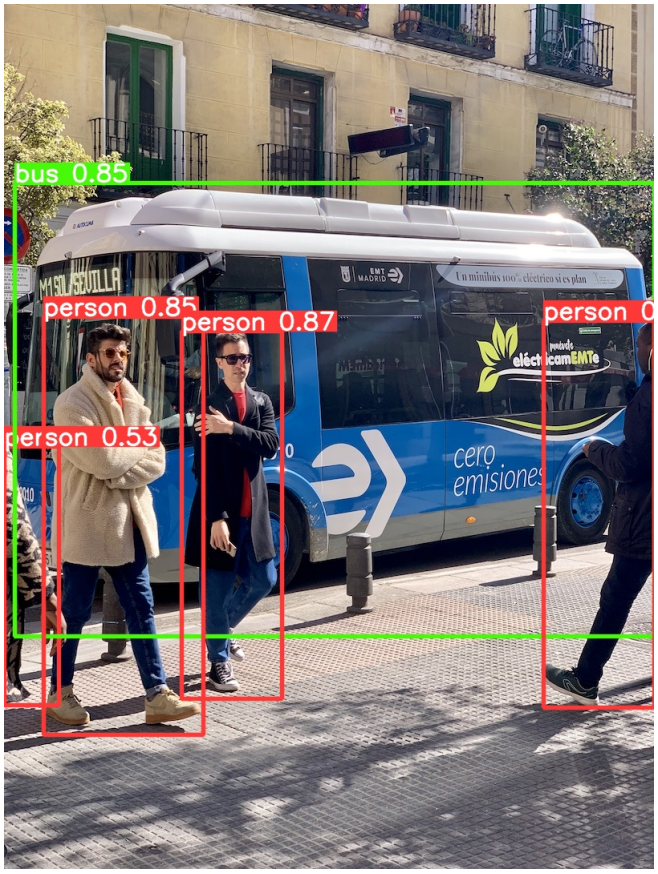

図 7.1　YOLOv5 による画像認識結果

7.2 YOLOv5 用学習データの作成

YOLO に自分が検出して欲しい物体を学習させることができます。そのためにはまず学習データを作成する必要があります。4 つの文具を認識させる場合は、画像ごとに何の画像かが分かるようにしていました。YOLO は物体の位置も答えてくれるので、学習データも「何がどこにあるか」が分かるようにする必要があります。このような学習するための情報をアノテーションと呼びます。

YOLO の場合、画像 (〜.jpg) ごとにアノテーションの入ったテキストファイル (〜.txt) を用意します。アノテーションの形式は 1 行が 1 つの物体に対応しているので、一つの画像に 5 つの物体があれば、5 行になります。1 行には、物体のクラス名、物体の中心の X 座標、物体の中心の Y 座標、幅、高さの 5 つが空白で区切られて並びます。座標や幅、高さは、画素数ではなく、画像全体の幅や高さを 1 としたときの相対的な数値になっています。このような情報を作成するためのツールプログラムがいくつかあり、mars には labelImg[*10] をインストールしたので、これの使い方を説明します。

1. 「yolov5」を開いたところで「LXTerminal」を起動します。
2. 次のコマンドを入力して LabelImg を起動します。ただしこれは既にクラス名の入ったファイル (classes.txt) がある場合のコマンドで、そのようなものがない場合は「labelImg」のみで起動します。

```
labelImg train/images train/labels/classes.txt train/labels
```

3. 起動すると図 7.2 のような画面になるので、このように左側のところに「PascalVOC」が出ていたら、「ファイル」メニューの中の「PascalVOC」をクリックして「YOLO」に切り替えます。また「表示」メニューの中の「自動で保存する」にチェックを付けておくと、別の画像に移動する際に保存をしてくれるようになります。中央の細い長方形のところに画像が表示されるので、ウィンドウを最大化しておくと良いでしょう。
4. 学習用の画像は「yolov5」の中の「train」の中の「images」に入れておきます。5 つのサンプル画像が入っているはずです。学習用の画像を追加するならば同じところに入れます。この入れる場所は後に出てくる設定ファイルに合っていれば、他の場所でも構いません。「train」の中にはもう一つ「labels」があります。ここにはサンプル画像の一つのアノテーション情報の入ったファイルと、クラス名の一覧が入った「classes.txt」というファイルがあります。分担してアノテーション追加の作業を行う場合は、まず全てのクラス名の入った「classes.txt」を作成して、それをコピーして使うのが良いでしょう。
5. labelImg コマンド単独で起動した場合は、左側にある「ディレクトリを開く」をクリックして、画像があるディレクトリ (images) を開いて行って、画像のファイル名が並んでいるところまで行ったら「Open」をクリックします。また、左側にある「保存先を変更する」をクリックして、アノテーションを入れるディレクトリ (labels) を指定します。YOLO の場合、画像が入っているディレクトリの隣にこのディレクトリを置きます。
6. 中央に最初の画像が表示されます。そこでアノテーションを付ける物体に対して次の操作を繰り返します。
   (1) Ⓦを押します。
   (2) マウスポインターの形が変わって、右と下へ黒い細線が伸びます。物体の上と左がちょうど入るところまでマウスポインターを移動します。
   (3) ドラッグしてマウスポインターを物体をちょうど囲う右下へ移動します。
   (4) マウスボタンを離すと、クラスを尋ねてくるので新しいクラスの名前を入力するか、既に存在するクラスをリストの中から選択します。

   全ての物体に操作を繰り返すと図 7.3 のようになります。

[*10] https://github.com/heartexlabs/labelImg

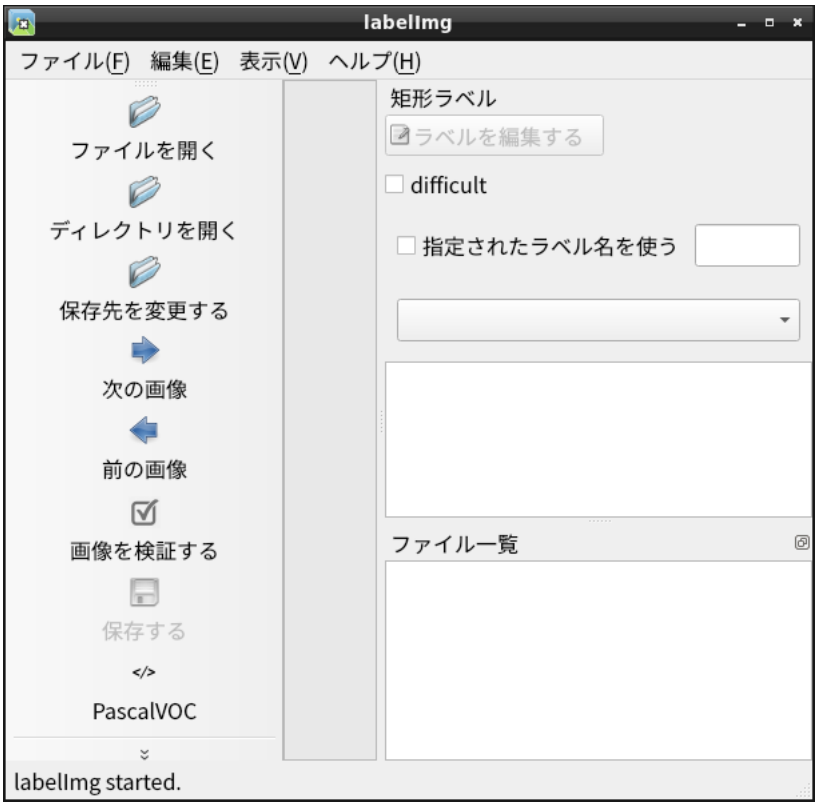

図 7.2　LabelImg の最初の画面 (labelImg 単独で起動した場合)

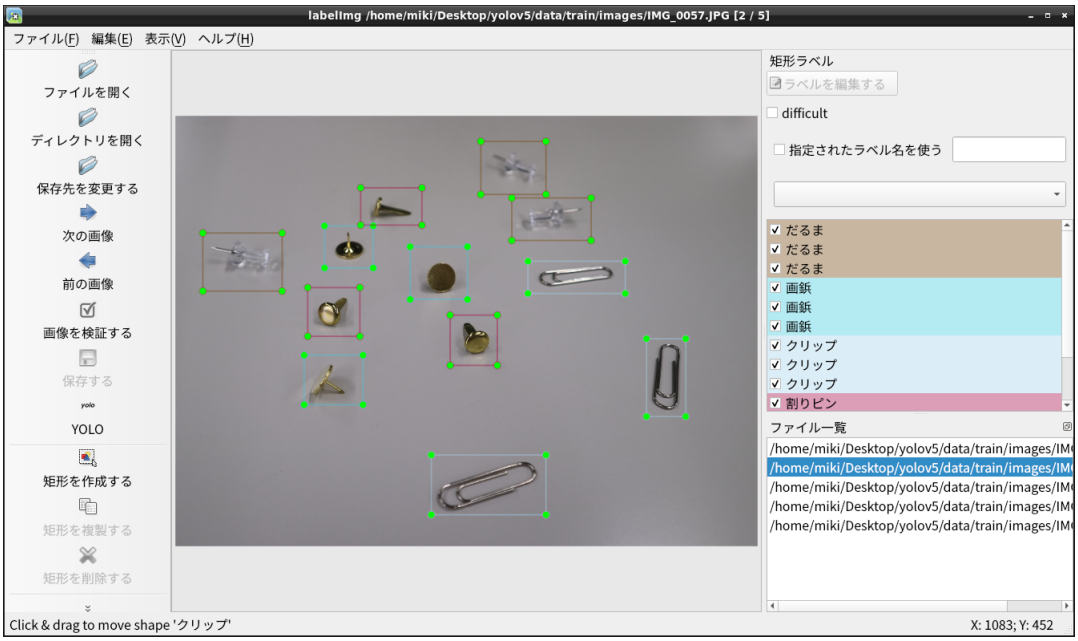

図 7.3　アノテーションの設定が終わったところ

7. まだアノテーションが付いていない画像がある場合は、左側にある「次の画像」をクリックします。

画像の置き場所などが変わると、これまでのアノテーションの結果が表示されなくなることがあります。そのような場合は「ファイル」メニューの「全て初期化する」を選択して、再度設定をやり直すと治るようです。また設定途中で「Attention」のダイアログが出ることがあります。何か設定を変えた場合は「Yes」、そうでない場合は「No」と答えておけば良いでしょう。

7.3 YOLO による学習

YOLOv5 で学習させる場合は、まずモデルを選択します。「YOLOv5n」、「YOLOv5s」、「YOLOv5m」、「YOLOv5l」、「YOLOv5x」と 5 つのモデルがあり、「YOLOv5n」が一番小サイズで、短い時間で認識しますが、精度がその分悪くなります。「YOLOv5x」は一番大きなサイズで、「YOLOv5n」と比べて認識に 2 倍弱の時間がかかりますが、精度はその分良くなります。他の 3 つのモデルはその中間の性能になります。特に指定をしない場合は、「YOLOv5s」になります。違うモデルを使用する場合は、後に出てくるコマンドの最後に「`--weights yolov5x.pt`」のような指定を追加します。

画像がどのディレクトリにあるか、クラスの名前などの設定ファイルを用意します。既に用意したもの (dgkw.yaml) が「data」の中にあるのでこれを利用します。その内容はリスト 7 のようになっています。

リスト 7 学習のための設定ファイルの内容

```
1  # sample 2022/12/02 by K.Miki
2
3  path: ./  # dataset root dir
4  train: train/images  # train images
5  val:   train/images  # val images
6  test:  # test images (optional)
7
8  # Classes
9  names:
10   0: 画鋲
11   1: 割りピン
12   2: だるま
13   3: クリップ
```

4 行目で学習用データがあるディレクトリを、5 行目で検証用データがあるディレクトリを指定しています。今回の設定では学習用データをそのまま検証用に使用しています。10 行目以降は、「labels」ディレクトリにあった classes.txt の内容にゼロからの番号を付けたものになっています。各業の先頭は同じ数の空白が必要なようです。この例のように日本語の名前を付けると、各種グラフで文字化けします。

次のコマンドで学習を実行します。

```
python3 train.py --batch 10 --epochs 100 --data dgkw.yaml
```

大量のメッセージが表示されます。特に指定をしないと GPU を使用します。「`--batch 10`」は学習データをまとめる単位で、数字が大きくなるほど認識率が高まりますが、処理時間と GPU のメモリーの使用量が増えます。メッセージの中の「CPU_mem」の下に「1.05G」と出ていますが、これがグラフィックボードのメモリーを越えると動きません。mars のグラフィックボードには 12GB のメモリーが付いているので、まだかなり余裕があります。「`--epochs 100`」は何回学習を行うかを示しています。この数字が大きいほど学習は進みますが学習時間がかかり、ある程度の数を越えるとあまり結果が変わらなくなってきます。

学習の進み具合は「runs」の中の「train」の中の「exp」(番号付き) の中の画像などを見ることにより分かります。「results.png」がグラフをまとめた画像になっていますが、グラフがまだまだ上がりそうや下がりそうな場合は epochs の数値を大きくすればよいでしょう。

7.4 学習結果の確認

次のコマンドで学習結果を用いてテスト画像を認識させることができます。

```
python3 detect.py --source train/tests --weights runs/train/exp/weights/best.pt
```

「`--source`」で認識する画像ファイルそのものか、画像ファイルが入っているディレクトリを指定します。「`--weights`」で認識に使用する学習結果を指定します。ここの「exp」の部分は学習を実行した際の最後に表示される「Results saved to runs/train/exp3」の最後の exp に合わせます。学習を繰り返すとどんどん exp の後の数字が大きくなるので注意します。同じディレクトリに best.pt と last.pt の 2 つのファイルがあります。best.pt は学習中で一番良かった学習結果で、last.pt は最後の学習結果です。内容が同じものになることもあります。

様々なメッセージが表示されますが、最後の行に「Results saved to runs/detect/exp3」と出たならば、「runs」の中の「detect」の中の「exp3」の中に認識結果が入っています。信頼度の低い結果まで描き込まれるので、一つの物体に検索結果が 3 つも重なっているようなものも出てきます。前にやった一つの画像に一つしか出てこないものの認識であれば、一番確からしいものを認識結果として採用すればよいのですが、こちらは位置が微妙にずれていますので、そう言うわけには行きません。代わりに「`--conf-thres 0.3`」のような設定を行の最後に追加して実行すると、信頼度が 0.3 より低い結果は表示されなくなり見やすくなります。

7.5 PHP による画像認識

ここでは PHP で python のプログラムを実行することにより、アップロードした画像にある物体を YOLOv5 で認識して表示するものを考えます。これまで他の章で取り上げた内容を組み合わせて実現します。

7.5.1 アップロードしたファイルの名前

画像ファイルのアップロードについては、既に 2 章 (p.8〜) でやっています。ここで問題になるのはアップロードしたファイルの名前です。いつも同じ名前のファイルにすると複数の利用者が違う画像ファイルをアップロードした場合、先にアップロードしたものが消えてしまいます。また通常 YOLOv5 で認識した結果のファイル名も同じものになりますが、同じ人が異なるファイルを何度も送った場合、結果のファイル名が同じ場合、ブラウザが違う内容のファイルだと認識してくれないので、再読込をしても最初と同じものが表示されてしまいます。

アップロードしたファイルの名前としては、ランダムなものを使用するのが良いのですが、次善の策としてアップロードした日時によるファイル名にすることが考えられます。time() を使って秒単位の名前にすればまず重なって消えることはないでしょう。異なる名前を使用した場合の問題点はどんどんファイルが貯まることです。定期的に古いファイルを削除するようにするか、アップロードした際に古いファイルがあれば一つ削除すれば良いでしょう。

元のファイルの拡張子は維持する必要があります。YOLO は PNG 形式でも JPEG 形式でも対応してくれるようですが、拡張子が違っているとだめなようです。文字列関数の strpos() で「.」を探して substr() で切り出すのが良いでしょう。ファイル名の途中に「.」があるような困ったファイル名が心配であれば、使い方は strpos() と同じですが、後ろから探してくれる strrpos() を使いましょう。

アップロードしたファイルは yolov5 ディレクトリに入り、そのファイル名は`$fname` に、拡張子の部分 (例えば「.png」) は`$ext` に入っているものとします。

7.5.2 python のプログラムの実行

3.5 節で Unix のコマンドを system() で実行する方法を説明しましたが、同様のやり方で python のプログラムを実行することができます。この時に少し問題になりそうなのは、実行するディレクトリです。これまでの YOLO の実行は、「yolov5」ディレクトリで開いた端末ソフトで行っていました。PHP のファイルも同じディレクトリにあれば、端末ソフトと同じ内容で実行が可能です。そうでなければ、ディレクトリの移動のコマンドも合わせて実行する必要があります。後述の例では「yolov5」ディレクトリと同じところに PHP のファイルがあると仮定しています。

YOLO の物体認識を行う detect.py は、特に指定がなければ毎回新しいディレクトリに結果を保存します。一番新しいディレクトリを探す方法では、複数の実行が続いた際に先に実行した結果でなく、後からの実行結果が入ったディレクトリを開いてしまう可能性があります。`--exist-ok` を付けて実行すると同じディレクトリを再使用するようになるので、いつも同じディレクトリに結果が入るようになります。

これまでの YOLO の実行例では物体認識結果を上書きした画像のみが得られていましたが、認識した結果を元にさらに処理を行う場合は、どこに何があったかが必要になります。`--save-txt` オプションを付けると別のテキストファイルに認識結果が入るようになります。さらに信頼度も必要な場合は、`--save-conf` オプションも追加します。

以上をまとめると次のようになります。

```
system("cd yolov5; python3 detect.py --source ".$fname.$ext.
       " --exist-ok --save-txt --save-conf  2>&1");
```

一番最後に「 `2>&1`」と言うおまじないのようなものが付いていますが、これを省略すると detect.py 実行中のメッセージがブラウザに送られないようになります。`--exist-ok` オプションのため認識結果の画像は、「yolov5」ディレクトリの中の「runs」の中の「detect」の中の「exp」の中にあります。この画像をブラウザに表示させるならば、次のような Img タグを出力します。

```
<Img src="yolov5/runs/detect/exp/<?= $fname.$ext ?>">
```

また画像のあるところにある「labels」ディレクトリの中に認識結果の数字の入ったテキストファイル (拡張子は.txt) があります。その内容は認識した物体の数と同じ行数になっており、各行は 5 つの数字が空白で区切られています。左から認識した物体のクラス番号、中心 X 座標、中心 Y 座標、幅、高さ、信頼度になっており、クラス番号は 0 以上の整数値、次の 4 つの数字は画像全体を 1 とした相対的な数値、最後の信頼度は 0〜1 の数値になっています。`--exist-ok` を付けて実行すると、認識結果がどんどん追加されるので、実行前にファイルを削除する方が良いでしょう。

クラスの番号が何のことかは、学習の際に使用した拡張子.yaml のファイルの内容で分かります。YOLOv5 に付いている学習済みデータについては、「yolov5」ディレクトリの中の「data」の中にある coco128.yaml の内容を見れば分かります。

8. 演習課題

8.1 QR コードの表示 9/21

図 8.1 のような QR コードを表示する k0921.php を QR ディレクトリの中に作ってください。

1. URL を入力し、誤り訂正レベルを選択し、大きさ (1〜) を入力して、「QR コード表示」をクリックすると、QR コードが表示されるようにする。
2. URL は urlencode() を使ってパラメタ付きの URL でも問題 (例えば「`http://example.com?x=x&y=999`」を指定した時に「`&y=999`」の部分が無い QR コードが表示される) が生じないようにする。
3. 最初は何も入力されていないので QR コードは表示されないようにする。
4. QR コードを表示した時、URL と大きさの入力内容が消えないようにする。
5. QR コードを表示した時、誤り訂正レベルも QR コードを表示する前に選択したものになるようにする。

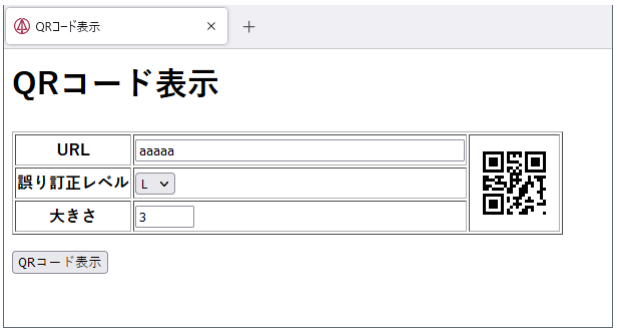

図 8.1 作成例

8.2 QR 決済システム (1) 9/28

これから 4 回かけて QR 決済システム作成をします。今回はまずデータベースを作成します。

1. 「QR」ディレクトリの中に data.db という名前のデータベースファイルを DB Browser for Sqlite で開いて、以下のような内容の user テーブルを作成する。id と remainder は Integer 型、name と pass は Text 型です。id には PK と AI も設定してください。

id	name	pass	remainder
1	miki	12345	1000
2	maki	abcde	1000
3	mika	abc123	1000

2. 同じく data.db に以下のような内容の money テーブルを作成する。全て Integer 型で、id には PK と AI も設定してください。

id	uidf	uidt	date	amount
1	1	2	1664164000	500
2	2	3	1664165000	500
3	3	1	1664166000	500

3. このデータベースファイルはこれから作成する PHP のファイルで利用することになります。そこでデータベースとの接続の部分を common.php に入れて、pass.php と list.php で取り込むようにします。

4. 次にログイン→残高表示→ログアウトの部分を作成する。既にログインを行っていた場合は、図 8.2 の左側のようなログインの入力画面はスキップしますが、今回はスキップなしです。スキップは PHP を使って実現するので、ファイル名は index.php にします。そして「送信」をクリックしたら、pass.php に行くようにします。
5. pass.php はデータベースの user テーブルを調べて name と pass に一致すれば`$_SESSION['id']` に id の値を入れて、list.php へ行くようにします。もしどの name と pass にも一致しない場合は、index.php へ戻します。
6. 図 8.2 の右側の list.php では、`$\_SESSION['id']` の値を元にデータベースから検索して、名前、残額、入出金状況を表示します。入出金状況は難しいのでその上の部分ができたら、ログアウトのリンクの先である logout.php を先にやった方が良いでしょう。
7. logout.php では`$_SESSION['id']` を削除してから、index.php へ行くようにします。

図 8.2 作成例

8.3 QR 決済システム (2) 10/05

今回は QR 決済システムで支払いができるようにします。

1. common.php に`$_SESSION['id']` と`$_POST['namae']` が存在しない場合、index.php へ行くようにします。これによってログインしないで直接 list.php へ言った場合に、エラーが出たりしなくなります。
2. 支払金額や支払先は URL で指定する事にします。これは支払金額や支払先を含む QR コードが出せるようにするためです。「`http://mars..../index.php?to=2&pay=300`」で支払先の利用者の ID は 2 で、支払金額は 300 円であることを示します。なおこのように指定した値は、`$_GET['to']` などで取り出すことができます。
3. index.php の最初に支払先があれば支払先を`$_SESSION['to']` を入れます。さらに金額があれば、`$_SESSION['pay']` に入れますが、無ければゼロを`$_SESSION['pay']` に入れます。
   次に`$_SESSION['id']` があり、支払先もあれば pay.php に行くようにします。`$_SESSION['id']` があるが、支払先が無い場合は list.php へ行くようにします。pay.php や list.php に行かない場合は、これまでと同じように、名前とパスワードの入力画面を出します。
4. pay.php では、まず支払先と金額を`$_SESSION` から取り出して図 8.3 のように表示します。`$_SESSION['to']` には数字しか入っていないので、user テーブルを検索して名前に直してください。「O.K.」ボタンをクリックしたら pay2.php へ行くようにします。「支払わない」のリンクをクリックしたら list.php へ行くようにします。
5. pay2.php では`$_SESSION` や`$_POST` に入っている情報を元に money テーブルにレコードを追加します。日時については time() の数字を入れます。また user テーブルの残額の修正を 2 人分忘れないよう

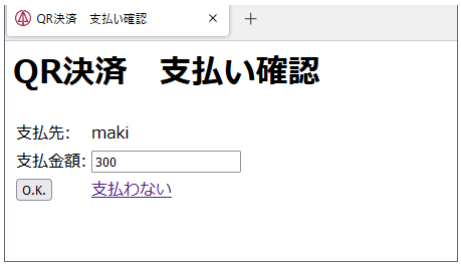

図 8.3 作成例

にします。データベースの更新ができたら list.php へ行くようにします。

残額の修正には SELECT で現在の値を取り出して UPDATE で新しい値を入れる必要がありますが、実はテキストには書いてないのですが UPDATE だけでやることが可能です。次の UPDATE 文で id が 1 の人の remainder の値を 300 減らすことができます。

```
UPDATE user SET remainder=remainder-300 WHERE id=1
```

6. list.php の最初に、もし支払先や金額が`$_SESSION` にあれば消去する部分を追加します。

8.4 QR 決済システム (3) 10/12

QR 決済システムと言いながら、まだ QR コードを表示するところがありませんので、既にある list.php を修正し、show.php を追加します。

1. passs.php の最後のところに、データベースを検索した結果 id の値が得られたら、list.php へ行くところがあります。ここを`$_SESSION['to']` があれば pay.php に、なければ list.php に行くように修正します。
2. list.php を修正して、図 8.4 の左側のように請求金額を入れるところと、「QR コード表示」のボタンを追加します。このボタンをクリックしたら show.php を呼ぶようにします。前回入力した請求金額を`$_SESSION['req']` に入れておくことにし、`$_SESSION['req']` が無ければ 0 が請求金額の欄に入るようにします。
3. show.php では図 8.4 の右側のように、「`http://..../index.php?to=`**ログインした人の** `id&pay=`**請求金額**」に相当する QR コードと、「利用状況表示にもどる」のリンクを表示する。このリンクをクリックしたら list.php へ行くようにする。urlencode() を使用しないと正しい QR コードができないので注意しましょう。
4. 前期のテキストの「5.15 送信前のチェック」を参考にして、pay.php に JavaScript の記述を追加して、「O.K.」ボタンをクリックすると支払金額が残額を超えていたら window.alert() を使って「残額が足りません。」と表示されるようにします。支払金額が残額以下ならば、これまで通り pay2.php に行くようにします。

8.5 QR 決済システム (4) 10/19

session 情報がブラウザを閉じると消えてしまうので、.htaccess を設置します。また利用者を登録する部分も作成します。

1. 前期のテキストの「5.14 Web ページ間の情報のやり取り」の「session を使う方法」を参考にして、QR ディレクトリに ses ディレクトリと .htaccess ファイルを作成します。有効期間は 10 日間とす

図 8.4　作成例

る。.htaccess ファイルを置く場所を間違えると peditor が動かなくなることがあります。そのような場合は、mars に RDP で接続してファイルの移動や削除をすること。

2. 図 8.5 のような追加する利用者の情報を入力する user.htm を作成します。誰でも勝手に利用者を追加しては困るので、本来はパスワードによる認証を通過しないとこのページにたどり着けないようにするべきですが、本筋とは関係ないので今回は認証はしません。

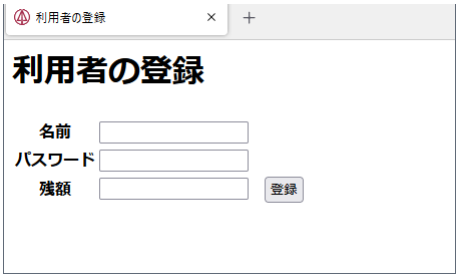

図 8.5　作成例

3. 図 8.6 の左側のような利用者の追加結果を示す user2.php を作成する。ただし、既に同じ名前の利用者が居た場合は、図 8.6 の右側のように、二重登録せずに登録できなかったことを示します。なお「もどる」のリンクの行き先は user.htm です。

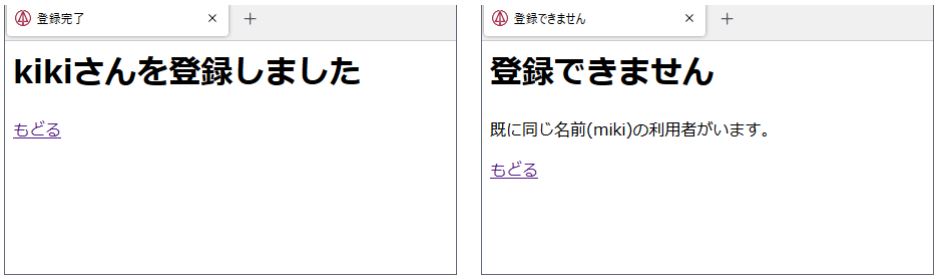

図 8.6　作成例

8.6　画像データベース 10/26

簡単な画像データベースを作成します。全て「Image」ディレクトリの中に作成してください。

1. 画像データを入れるデータベースファイル (data.db) を作成します。次のようなテーブル (photo) を作成します。

名前	データ型	備考
id	INTEGER	PK、AI
name	TEXT	
photo	BLOB	

2. QR 決裁システムで使用した common.php をコピーします。ただし`$_SESSION` が出て来る部分は削除します。
3. 図 8.7 のような index.php を作成します。画像は最大幅が 200px になるように設定します。なお画像ファイルをドラッグ＆ドロップしても送信されないようにします。画像ファイルは PNG 形式のみアップロードすることにします。

図 8.7　作成例

4. index.php から送信された画像をデータベースに追加して、index.php に戻る upload.php を作成します。
5.「`<Img src="display.php?id=1">`」のようにして呼び出すと photo テーブルの中の id=1 の画像を表示する display.php を作成します。
6. index.php の削除のリンクをクリックした際に画像データを削除する delete.php を作成します。

8.7　メールの定期送信 11/2

次の課題もメール関連でファイルが多数関連するので「Mail」と言うディレクトリを作成して、その中にこの課題のファイルも入れてください。

1. mars の自分のメールアドレスに一日に 3 回メールを送るものを作成します。ファイル名は aisatu.php にしてください。aisatu.php は実行する時間帯によって次のように異なる内容のメールを送るようにします。

時間帯	件名	本文
7:00 から 12:00 まで	おはよう！	今日も良い一日だとよいですね。
12:00 以降 17:00 まで	こんにちは！	明るいうちにがんばりましょう。
上記以外	こんばんは！	夜は早く寝ましょう。

2. cron を設定して aisatu.php を毎日 8:00、13:00、22:00 に呼び出すようにします。設定した内容を aisatu.php の`<?php` の次の行に、「//」を入力し、その後に入力しておいてください。なお、設定通りメールが届いたら、cron の設定は消しておきましょう。そうしないとどんどんメールが溜まってしまいます。

8.8　利用者登録の確認 11/9

利用者に自分の好きな名前とパスワードで使ってもらうシステムはあちこちで見られます。登録の際には、名前とパスワードの他に連絡用にメールアドレスの入力を求める事も多いと思います。その場合問題になるのは、メールアドレスが間違っていると連絡が取れないことです。対策の一つとして、登録の際に入力されたメールアドレスへメールを送り、それに対する応答を元に登録を行う事が考えられます。これによって登録された利用者のメールアドレスは、登録時には届いた事が確認できます。

1. 以下のファイルは「Mail」ディレクトリーの中に入れるようにします。
2. Image ディレクトリにある common.php と同じ内容の同じ名前のファイルを作成します。
3. 「DB Browser for Sqlite」を利用して data.db を作成します。テーブルは「user」の一つだけで、その内容は id INTEGER PK AI、name TEXT、password TEXT、address TEXT、status INTEGER とします。データを入れる必要はありません。
4. index.htm では図 8.8 のように名前、パスワード、メールアドレスを入力するところと、「登録」ボタンを設けます。「登録」ボタンをクリックしたら sendmail.php へ行くようにします。

図 8.8　作成例

5. sendmail.php ではまず data.db に入力内容を入れるようにします。名前→ name、パスワード→ password、メールアドレス→ address、0 → status。「DB Browser for Sqlite」を利用してちゃんと data.db に入ったかどうか確認しましょう。それから図 8.9 のように出るようにしましょう。
6. さらに sendmail.php で、前期のテキストの「6.5 INSERT 文」を参考に今データベースに入れたレコードの id の値を取り出して、echo で表示するようにしてみましょう。ちゃんと表示されたらこの部分は削除します。
7. そして sendmail.php でさらに入力されたメールアドレス先に、「登録を完了するために以下のリンクをクリックしてください。」と URL からなる本文のメールを送るようにします。URL は次に作成する touroku.php の URL の後に、6. で取り出した id の値を touroku.php で取り出せるような形にしたものを付けます。

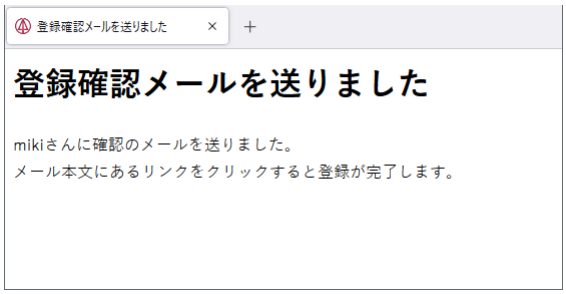

図 8.9 作成例

8. touroku.php では、id の値をもとに、status の値を 1 に変更します。よってデータベースの中の status の値を見れば、確認済みの利用者かどうか分かるようになります。変更後に図 8.10 のように出るようにしてください。

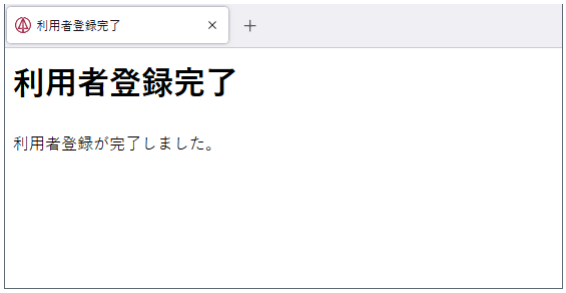

図 8.10 作成例

8.9 円相場のグラフ 11/16

次の課題で環境センサーの測定値をデータベースに入れて、データベースの内容をグラフにして表示するものを作ります。今回はその後半の部分の練習になります。

1. 「IoT」と言うディレクトリを作成し、以下のファイルはその中に入れてください。
2. 「教材フォルダ」にある yen.db をコピーしてください。このデータベースファイルには souba と言うテーブルがあり、そこには id、date、yen と言う列があり、今年になってからの毎日の円相場の数字が入っています。
3. yengraph.php を作成して、yen.db の最初の 15 件のデータを元に図 8.11 のようなグラフが出るようにしてください。id が 1 以上かつ 15 以下で検索すると、最初の 15 件のデータが得られます。データベースの検索結果は日付と円がセットで順番に出て来るので、データベースから foreach を使って検索結果をまず変数に入れてグラフにすると良いでしょう。
4. 5 秒ごとに更新されるようにします。
5. 更新されるごとに、表示されるデータが後にずれるようにします。つまり、一番最初は 01-03 からですが、更新されると 01-04 からになり、さらに更新されると 01-05 からになるようにします。souba テーブルの id の値が 1 から順番になっていることを利用します。これを実現するには、更新後のページにどこから表示するのかを、URL を使って伝えましょう。そのままずっと放置すると yen.db に入っていないところまで行ってしまいますが、それに対応する必要はありません。
6. 「最初から」リンクの URL は「yengraph.php?id=1」になっています。これをクリックすると、グラフが 01-03 から表示されるようにしてください。

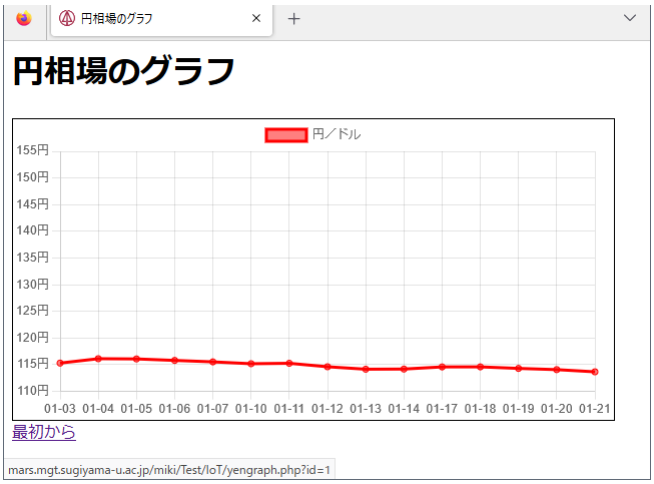

図 8.11　作成例

8.10　環境センサーのデータのグラフ化 11/23

環境センサーからのデータを受け取りグラフ化する課題です。今回の課題のファイルは「IoT」ディレクトリに入れてください。

1. データベース (data.db) を新規に作成します。テーブル名は「sensor」にして、フィールドとして「id」、「date」、「temp」、「humi」、「co2」、「bright」を追加します。データ型は「date」、「co2」、「bright」は「INTEGER」、「temp」と「humi」は「REAL」です。「id」のみ AU と PK を設定します。
2. 同じデータベースを複数のファイルで扱うので common.php を作成します。
3. テキストの「4.5 環境センサーからの情報の取り込み」を参考にして、receive.php を作成します。何も echo する必要はありませんし、HTML のタグも不要です。`$_POST[ ]` で受け取ったデータを data.db に入れてください。
4. receive.php ができたら、`http://mars.mgt.sugiyama-u.ac.jp/Sensor/` で登録してください。
5. data.db のデータを元に図 8.12 のようなグラフを表示する tempgraph.php を作成してください。一番新しいデータが最大 60 個表示されるようにしてください。また 60 秒ごとに自動更新されるようにします。両側に縦軸のあるグラフの作り方は「とほほの WWW 入門」の例を参考にしてください。なお時刻のところは、自動的に間引かれて表示されます。
6. さらに「30 分ごと」のリンクをクリックすると図 8.13 のように 30 分ごとのグラフが出るようにしてください。「1 分ごと」のリンクをクリックすると最初のグラフに戻ります。

8.11　グラフの改良とメールで知らせる 11/30

環境センサーからのデータを受け取り二酸化炭素濃度が基準値を超えたらメールを送るのと、平均濃度のグラフ化の課題です。今回の課題のファイルも「IoT」ディレクトリに入れてください。

1. data.db の「sensor」テーブル名に「alert」フィールドを追加します。データ型は「INTEGER」です。ここに二酸化炭素が基準値を超えたら 1 を、下回ったら 0 を入れます。
2. receive.php を修正してセンサーの二酸化炭素濃度が 1000 を超えたらメールを送って知らせるようにします。ただし一度 1000 を超えたら 800 未満になるまでメールは送りません。800 未満になったらメールで知らせます。メールの本文には、co2graph.php の URL も入れてください。メールの送り先は mars でも自分のスマホでも構いません。

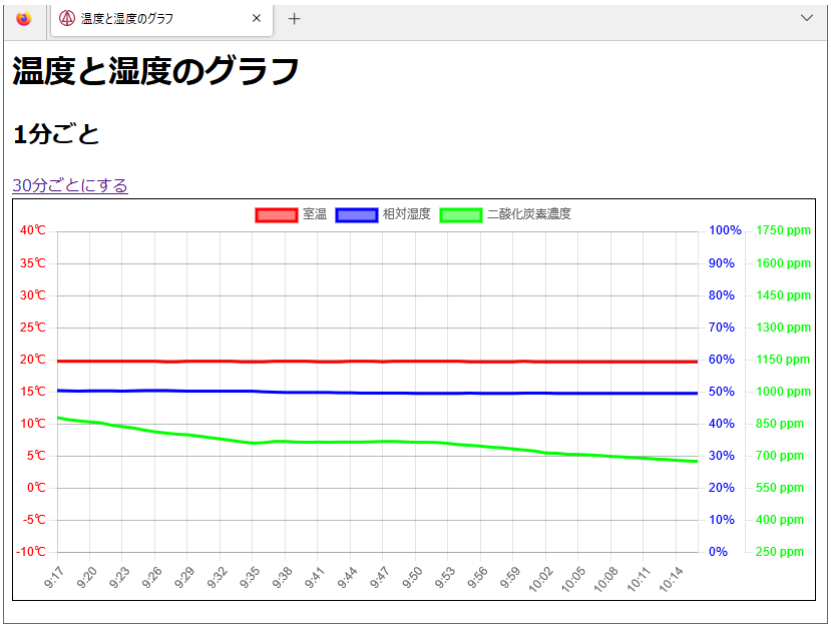

図 8.12　作成例

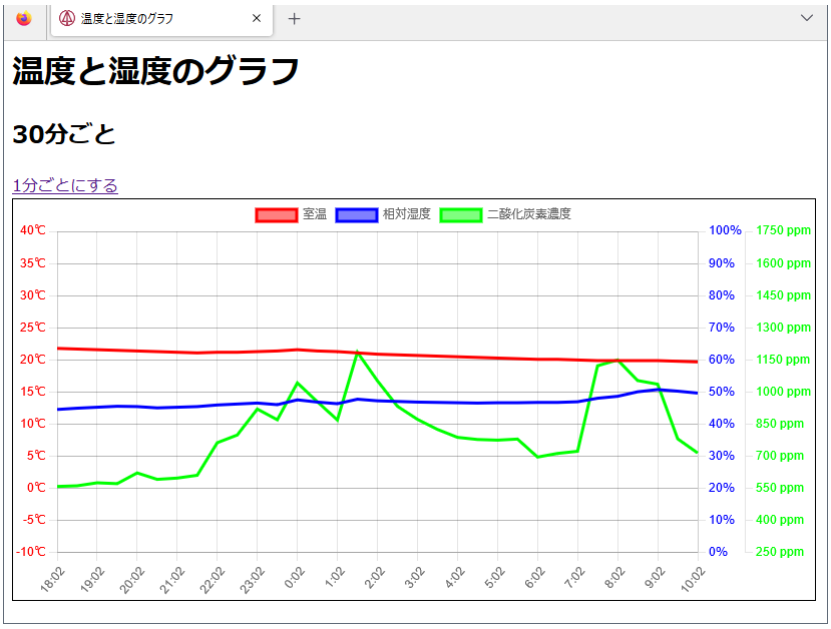

図 8.13　作成例

3. 図 8.14 のようなグラフ 20 分間の二酸化炭素の平均濃度を表示する co2graph.php を作成してください。いつも 2 日分が表示されるようにします。60 秒ごとに自動更新されるようにします。図の下方にあるボタンの画像は教材フォルダからアップロードしてください。ボタンの意味は一番左から「一番最初から」、「1 日分前に」、「1 日分後に」、「一番最後から」です。最初は最後から表示されるので右の 2 つのボタンは表示されないようにします。逆に一番最初を表示している場合は、左の 2 つのボタンは表示されないようにします。

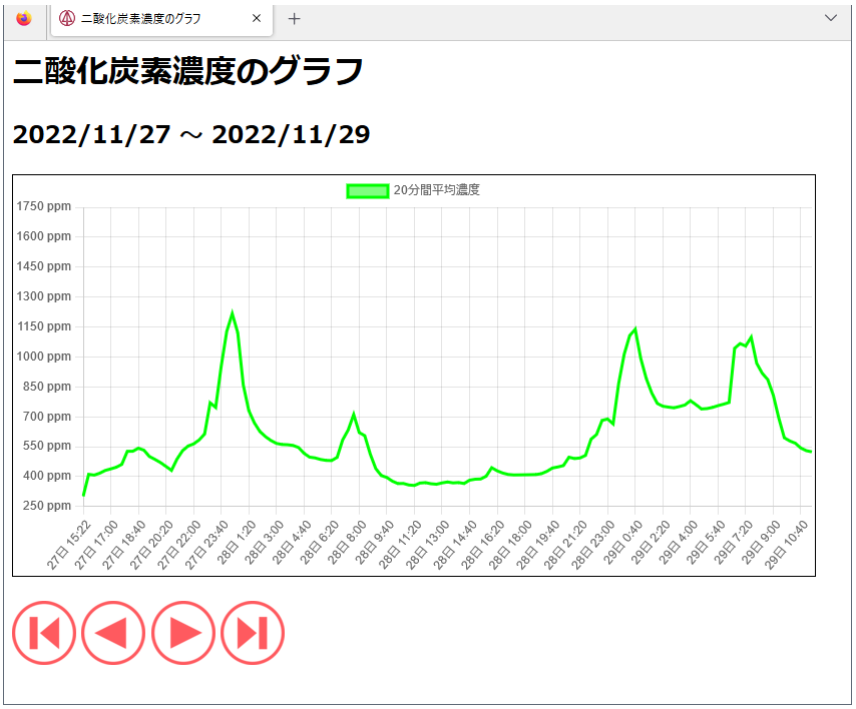

図 8.14　作成例

8.12　スクレイピングと POST でデータを送る 12/7

クレイピングを使って株価のデータを取り込むものと、POST 形式のデータを送る課題です。

1. 「株マップ.com」(`https://jp.kabumap.com/`) にアクセスして現在の日経平均株価が図 8.15 の左側ように表示されるような kabmap.php を作成せよ。
2. 「QR/user2.php」へ直接アクセスして名前が「miko」、パスワードが「99999」、残額が「10000」の利用者を登録する adduser.php を作成せよ。なお `file_get_contents()` で取ってきた内容はそのまま echo で出力すると図 8.15 の右側ようになる。

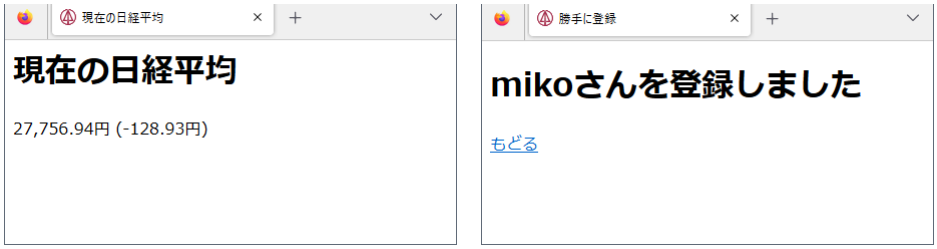

図 8.15　作成例

8.13　天気予報の表示 12/14

今回の課題は「Open Weather Map」からデータをもらって表の形で表示するものです。

1. 「Open Weather Map」に登録して API キーを取得してください。
2. 「Open Weather Map」から 1 週間の天気予報を受け取り、図 8.16 のような表の形で表示する weather.php を作成してください。

名古屋市の天気予報

名古屋市の天気予報

年月日	最高温度	最低温度	湿度 (%)	気圧 (hPa)	風速 (m/s)	風向	天気
2022/12/13	12.56	8.51	81	1015	5.62	322	適度な雨
2022/12/14	9.23	3.9	48	1019	9.16	298	小雨
2022/12/15	8.68	2.88	46	1021	7.02	301	小雪
2022/12/16	9.24	3.77	51	1022	4.74	321	厚い雲
2022/12/17	7.91	6.92	68	1020	2.18	333	小雨
2022/12/18	6.23	2.05	34	1014	9.28	293	雨と雪
2022/12/19	6.61	1.47	56	1022	3.96	292	小雪
2022/12/20	8.51	2.52	52	1022	2.23	4	雲

図 8.16 作成例

8.14 YOLO による学習 12/21

今回の課題は YOLOv5 に学習させるものです。

1. labelImg を利用して 5 つの画像の全てにアノテーションを付けてください。
2. batch や epochs の数字を色々試して、一番良い結果の画像を提出してください。図 8.17 は基礎演習で良かった人の例です。

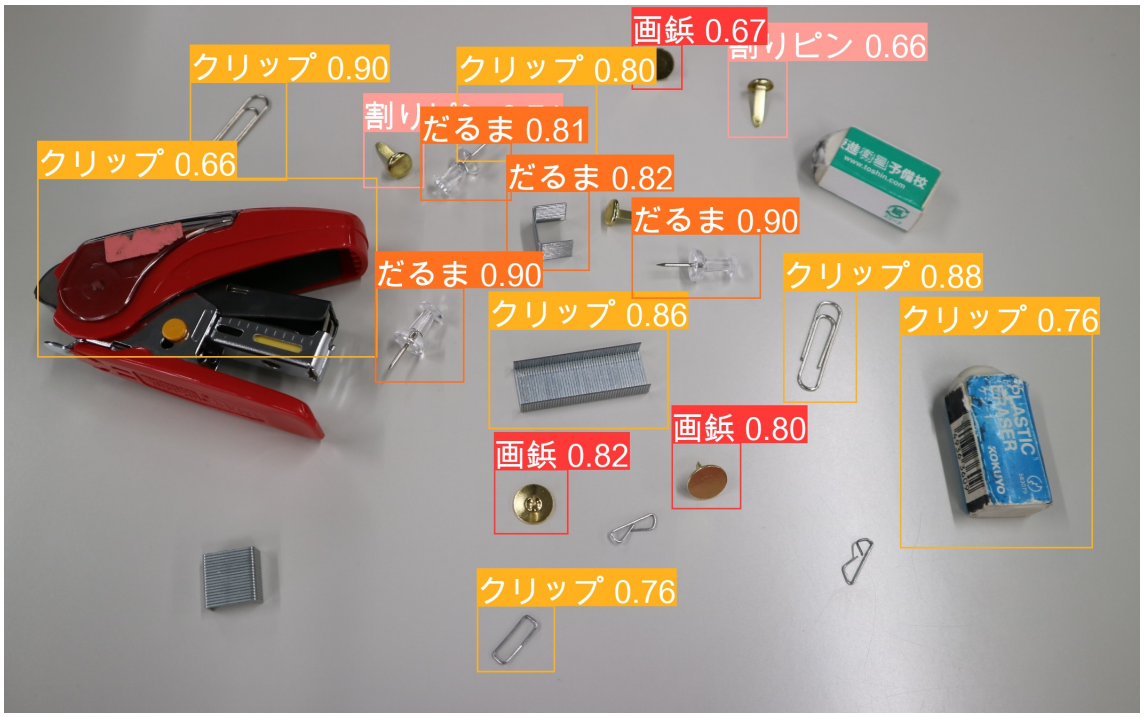

図 8.17 基礎演習でかなりうまく学習できた人のもの

8.15 YOLO による物体検出 1/11

今回の課題はアップロードした画像を YOLOv5 で物体検出させるものです。

1. 図 8.18 の左側のような yolo.htm を作成します。「認識」をクリックしたら yolo.php へ行くようにし

ます。

2. yolo.php で図 8.18 の右側のようなものが表示されるようにします。
3. アップロードの際には「yolo5」の中にファイルを入れる必要があるので、

```
move_uploaded_file($_FILES['ufile']['tmp_name'],"yolov5/".$fname.$ext);
```

のようにアップロードしたファイルを保存する時に付ける名前の前に「yolov5/」を付けます。

4. 表のところで数値 (例えば 0) を名称 (例えば person) に直す必要がありますが、こちらから提供する class.php を取り込んで活用してください。
5. 写真画像は高解像度のものは大変大きく表示されることがあるので、ブラウザの幅で制限をかけてください。

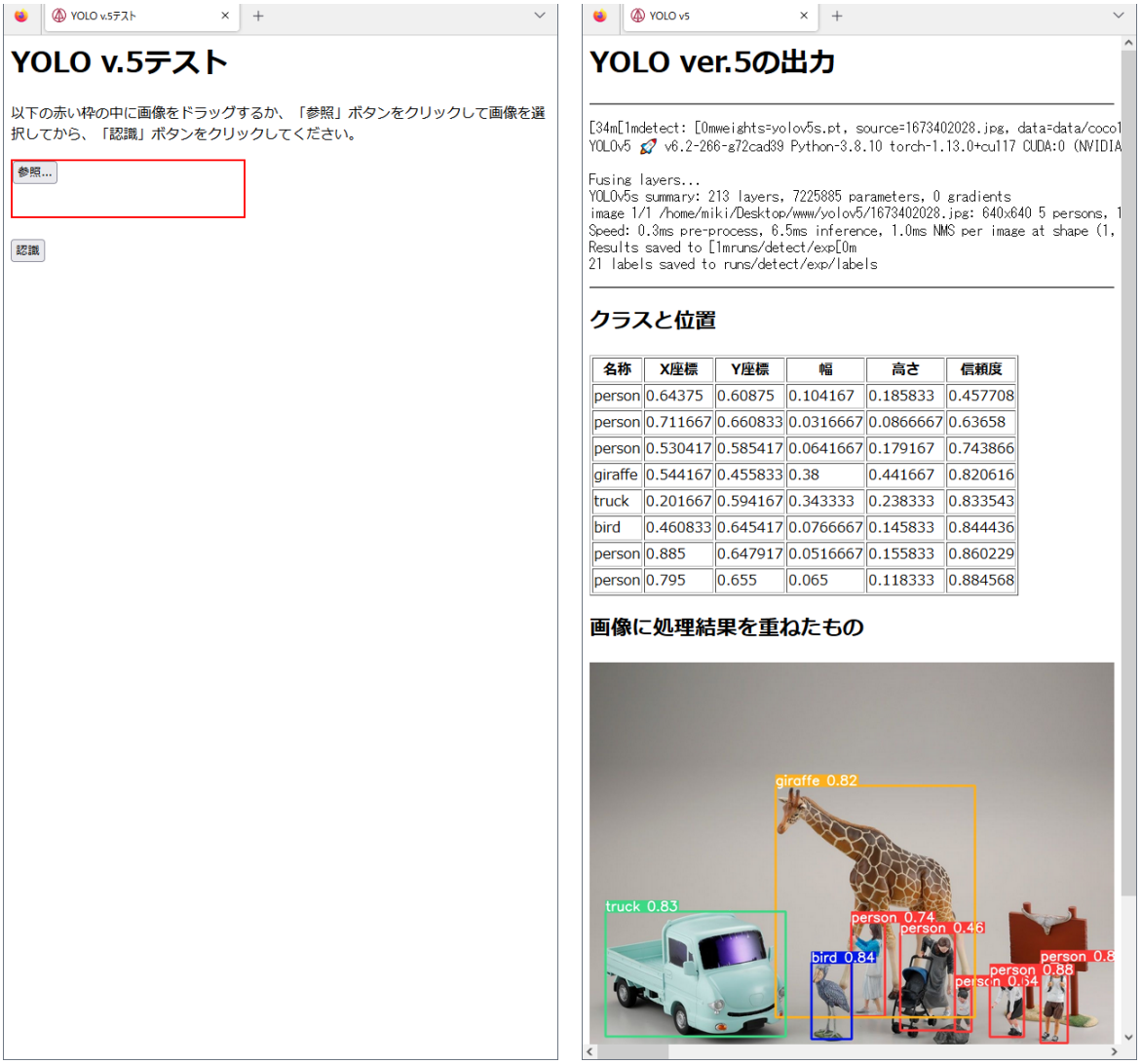

図 8.18　作成例

索引

A

API (Application Programming Interface) ... 29
at: 実行予約をする (Unix) ... 13

B

bindParam(): SQL のパラメタを設定する (PHP) . 9

C

Chart.js: グラフ描画ライブラリ ... 18
cron: 定期的実行予約をする (Unix) ... 13

D

date(): 秒数を指定した形に変換する (PHP) 6

E

execute(): パラメタ付きの SQL を実行する (PHP) 9

F

file(): ファイルや web ページを取り込む (PHP) 25
file_get_contents(): ファイルや web ページを取り込む (PHP) ... 25
file_put_contents(): 変数の内容をファイルにする (PHP) ... 10
function: 関数の定義 (PHP) ... 5

H

HTML 解析ライブラリー ... 27

I

include: プログラムの取り込み (PHP) ... 5
IoT (Internet of Things) ... 15
is_uploaded_file(): アップロードファイルが指定されたか調べる (PHP) ... 9

J

JSON (JavaScript Object Notation) ... 29
json_decode(): JSON 形式を PHP の変数に変換する ... 29

L

LabelImg: アノテーション設定プログラム (Unix) . 36
LPWA (Low Power Wide Area-network) ... 15

M

mb_send_mail(): メールを送る (PHP) ... 11
MIME タイプ ... 10
mktime(): 指定した日時から通算秒数を求める (PHP) ... 7
move_uploaded_file(): アップロードしたファイルに名前を付ける (PHP) ... 9

O

Open Weather Map 天気予報情報のサイト 32

P

pclose(): Unix のコマンドとのやり取りを終える (PHP) ... 14
PHP Simple HTML DOM Parser (PHP) ... 27
popen(): Unix のコマンドを実行する (PHP) .. 14

Q

QR コード生成ライブラリ ... 1

S

SQL のパラメタを設定する (PHP) ... 9
system(): Unix のコマンドを実行する (PHP) . 14

T

time(): 時刻を求める ... 6

U

Unix のコマンドを実行する (PHP) ... 14

W

web ページの自動更新 (HTML) ... 24
web ページの自動更新 (PHP) ... 24
wget: web ページを取り込む (Unix) ... 12

X

XML (Extensible Markup Language) ... 29

Y

YOLO: 物体検出システム ... 34

あ

アップロードしたファイルに名前を付ける ... 9
アップロードファイルが指定されたか調べる 9
アノテーション設定プログラム ... 36

か

関数の定義 (PHP) ... 5
グラフ描画ライブラリ (JavaScript) ... 18

さ

時刻を求める (PHP) ... 6
実行予約をする (Unix) ... 13
指定した日時から通算秒数を求める (PHP) ... 7
スクレイピング (Web scraping) ... 25

た

ダウンロードするように指定する (PHP) ... 10
定期的実行予約をする ... 13
データベースにバイナリーデータを入れる ... 9
天気予報情報のサイト ... 32

は

パラメタ付きの SQL を実行する (PHP) ... 9
秒数を指定した形に変換する (PHP) ... 6
ファイルや web ページを取り込む (PHP) ... 25
ファイルを送るときの Form タグ ... 8

ファイルを送るときの Input タグ 8
物体検出システム 34
プログラムの取り込み (PHP) 5
変数の内容をファイルにする 10

ま

メールを送る (HTML) 11
メールを送る (PHP) 11