
展開演習 B テキスト

椋山女学園大学現代マネジメント学部 三木 邦弘
令和3年10月6日版

1	SQL 複数のテーブル	1	5	スクレイピング	24
1.1	ER 図	1	5.1	Web ページの取り込み	24
1.2	テーブルの結合	2	5.2	HTML の解析	25
1.3	レコードの並び替え	3			
1.4	レコードのグループ化	3	6	API の利用と JSON	28
1.5	インデックスの設定	4	6.1	JSON とは	28
			6.2	API サービスの例	29
2	QR コードの利用	5	6.3	天気予報データの取得	31
2.1	PHP による QR コードの出力	5			
2.2	QR コードによる決済	6	7	演習課題	33
2.3	共通部分をまとめる方法	9	7.1	QRsystem (1) 10/6	33
2.4	時刻の扱い方	10	7.2	QR コード表示 10/13	33
			7.3	QRsystem (2) 10/20	33
3	メールと定期的処理	12	7.4	QRsystem (3) 10/27	35
3.1	HTML でメールを送る	12	7.5	QRsystem (4) 11/03	36
3.2	PHP でメールを送る方法	12	7.6	QRsystem (5) 11/10a	36
3.3	at を利用した予約実行	13	7.7	メールの定期送信 11/10b	37
3.4	cron を利用した定期的実行	14	7.8	利用者の確認 11/17a	37
3.5	PHP で Unix のコマンドを実行する	15	7.9	グラフの表示 11/27b	39
			7.10	データベースからグラフ作成 11/24	39
4	IoT とグラフの表示	16	7.11	グラフの改良とスクレイピング 12/1	40
4.1	どうやってつなぐか	16	7.12	環境センサーのデータ受信とグラフ 作成 12/8	41
4.2	電源をどうするか	17	7.13	天気予報の表示 12/15	42
4.3	センサーをどうするか	17	7.14	NHK 朝ドラ予告 12/22	43
4.4	集めた情報の処理	17			
4.5	環境センサーからの情報の取り込み	18			
4.6	グラフの描画	19			
4.7	web ページの自動更新	23			
				索引	45

1. SQL 複数のテーブル

1.1 ER 図

昔々、私は学生時代に初めて関係データベースの話を知った時に、二次元のテーブル(表)の形で様々なデータを扱うのは無理ではないか?と思いました。例えば学生の名前、身長、体重などのように1対1に対応しているものであればテーブルの形で済みます。これが1対多の関係になると難しくなります。例えば学生の好きな芸人を入れるとすると、最大で3人までと決まっていればテーブルの形になりますが、乃木坂全員が好きな学生も居るかもしれません。とりえず最大を3人にして、これを越える学生が出たら、最大を増やすと言うようなデータを入れる過程でどんどんテーブルの形が変わる、というのも色々困ります。

この問題に対して、学生と好きな芸人を1名で1行と言うテーブルを用意して、好きな芸人が100人居る学生については、100行使うというのが関係データベースのやり方です。このやり方であればどんなに多くの芸人が好きな学生が出現しても問題は生じません。また誰も好きな芸人が居ない学生への対応も簡明です。行数が好きな芸人の数を示すので、芸人はみんな嫌いな学生の行は存在しないだけです。

この方法で問題になるのは、学生の身長や体重のデータを、好きな芸人が100人居る学生については、100回記述するのか?です。このような場合、関係データベースでは1対1の関係のものと、1対多の関係のものは別のテーブルに入れて、体重を100回記述するような無駄を省きます。

関係データベースの設計は、データベースに入れるデータの関係が、1対1か1対多か多対多なのかを調べて、それを元にテーブルを分けます。そのためにデータの関係を表現するために図1.1のようなER図(Entity-Relationship Diagram)が使われます。

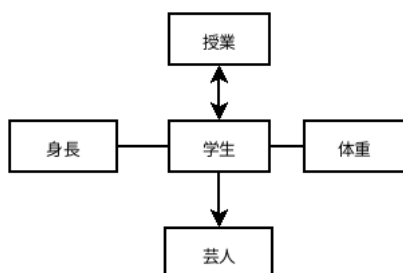


図 1.1 ER 図の例

学生テーブル

id	氏名	住所	身長	体重
----	----	----	----	----

授業テーブル

id	授業名	曜日	時限	担当
----	-----	----	----	----

受講テーブル

id	授業 id	学生 id
----	-------	-------

芸人テーブル

id	芸名	学生 id
----	----	-------

図 1.2 ER 図に対応するテーブルの例

図 1.1 の四角い箱は実体 (entity) を示します。直線や矢印が実体の関係 (relationship) を示します。まず「体重」と「学生」は1対1の関係なので直線で結びます。「身長」と「学生」の関係も同様です。「学生」と「芸人」は1対多の関係なので「学生」から「芸人」への矢印になります。「学生」と「授業」の関係は多対多なので両方向の矢印になります。一人の学生は複数の授業を受けますが、一つの授業も通常複数の学生が受講するからです。それを言うならば、同じ芸人が好きな学生は複数居る場合が多いので、こちらも多対多ではないか?となります。それは正しいのですが、とりえずみんな違う芸人が好きということにしてください。

図 1.1 の実体と関係を関係データベースで扱うとなると、テーブルが4つになります。まず「学生テーブル」に学生の氏名などの個人情報に加えて直線で結ばれた「身長」や「体重」が入ります。「授業テーブル」には、授業名その他、何曜日の何限目にあるとか、担当教員などが入ります。そして「受講テーブル」には、「学生テーブル」と「授業テーブル」をつなぐものが入ります。「芸人テーブル」には芸名などに加えて、「学生テーブル」とつなぐものが入ります。

「つなぐもの」は、ちゃんと対象となるテーブルの1行を指定できるものでないと困ります。関係データベースではテーブル同士をつなぐために、各テーブルにテーブルの1行を指定できる「主キー」と呼ばれる項目を設けることになっています。1行だけを指定できるように、主キーの内容は各行全て異なる必要があります。

ただ異なっていれば文字でも数値でも構いません。また「姓」だけでは区別できないので「名」の方と合わせて主キーとするようなことも可能です。各テーブルの主キーの名前を「id」とすると、各テーブルの項目は図 1.2 のようになります。

1.2 テーブルの結合

ここでは2つのテーブルからなるデータベースを題材に複数のテーブルの扱い方を説明します。前期に取り上げた表 1.2 のような student テーブルに加えて、表 1.1 のような money テーブルを設けて、学生のお小遣いの管理をすることにします。一人の学生が複数回お小遣いをもらったり使ったりするために、student テーブルだけではお小遣いには対応できません。

表 1.1 money テーブルの例

id	sid	date	amount
1	1	10/1	1000
2	1	10/5	-500
3	1	10/8	-350
4	1	10/15	800
5	2	10/17	-350
6	2	10/24	-400

表 1.2 student テーブルの例

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

student テーブルと money テーブルの id は主キーです。money テーブルの sid は student テーブルの id の値が入ります。最初の4行の sid は1なので、student テーブルの内容が表 1.2 だとすると、この4行は miki さんの記録ということになります。student テーブルの name の値である miki を使わず、student テーブルの id の値を使うのは、student テーブルの id は主キーなので複数のレコードが対応する恐れがないからです。

student テーブルの name と money テーブルの date と amount を元に、だれがいつ、いくら使用したかの一覧を求める SQL は次のようになります。

```
SELECT name, date, amount FROM student, money WHERE student.id=money.sid
```

2つのテーブルを使用するために FROM の後にテーブルの名前が2つ必要になります。そして WHERE のところに「student.id=money.sid」と指定することによって2つのテーブルが結合されます。この SELECT 文によって student テーブルの id と money テーブルの sid が同じレコードが結合して残ります。表 1.1 には sid の値が3のレコードがありません。そのため mika のレコードは出てきません。対応するレコードが無い場合も残したい場合は、外部結合という方法で対応することができますがここでは省略します。

student.id は student テーブルの主キーでしたが、money.sid は結合のために使われるものとして「外部キー」と呼ばれます。外部キーはテーブルの結合のために使われる、と言う条件だけなので主キーのように同じ値があってはいけないなどの制約はありません。

テーブルのどの列が他のテーブルのどの列に対応しているか、と言う条件を複数指定すれば3つ以上のテーブルを結合することもできます。ところが複数のテーブルに同じ名前の列名が使われていると、曖昧な条件になってしまいます。それを避けるために「student.id」のようにテーブルの名前を付けます。先程の例では name、date、amount は片方のテーブルにしか出てこないためにテーブル名を省略していますが、両方のテーブルにある id を表示したい場合は、次のようにテーブル名も付ける必要があります。

```
SELECT student.id, money.id FROM student, money WHERE student.id=money.sid
```

これらを PHP で実行する際に一つ問題になるのは、SELECT 文の結果を取り出そうとする際に、student.id のようなテーブル名が付いた列は取り出せない点です。そのために AS を使用して別名を付けます。次の例では money.id に mid という別名を付けて取り出しています。

```
$sql="SELECT money.id AS mid FROM student, money WHERE student.id=money.sid";
$result=$db->query($sql);
if (!$result) { die($sql."の実行ができません。"); }
foreach ($result as $data) {
    $data['mid'] に money.id の値が入っている
}
```

なおここで説明したテーブルの結合の書き方は古い書き方です。新しい書き方では INNER JOIN を使用して次のように書きます。テーブルの結合の条件が WHERE の検索条件と混ざらないため、わかりやすいと言われますがいかがでしょうか。3 つ以上のテーブルを結合する際には INNER JOIN 以降を繰り返すことになります。

```
SELECT name, date, amount FROM student INNER JOIN money ON student.id=money.sid
```

1.3 レコードの並び替え

SELECT 文でテーブルの内容を取り出す際に並び替えの指示を追加することができます。例えば name の昇順であれば次のようにします。

```
SELECT * FROM student ORDER BY name
```

もし WHERE があるならばその条件の後に「ORDER BY」の指示を付けます。降順にする場合は次のように DESC を追加します。

```
SELECT * FROM student ORDER BY name DESC
```

name に同じ値を持つレコードがあり、その場合は pass の値で並び替えたいような場合は次のように「,」で区切って複数の列名を指定します。

```
SELECT * FROM student ORDER BY name DESC, pass
```

1.4 レコードのグループ化

テーブルのある項目について、同じ値が複数入っている時に、同じ値ごとに集計したくなることがあります。例えば表 1.1 の場合、sid が同じ値のものについて amount の合計を出せば、各自が使用した金額が得られます。同じ値でまとめるということはグループにすることとして、次のように「GROUP BY」でまとめて、さらに SQL の SUM 関数を使ってグループごとの合計の計算をすることができます。

```
SELECT sid, SUM(amount) AS goukei FROM money GROUP BY sid
```

関数の結果を PHP で取り出すために AS を利用して goukei という別名を付けています。SUM 関数以外に表 1.3 のような関数があります。

表 1.3 SQL の関数

関数の形	関数の働き
AVG(列名)	指定した列の値の平均値を求めます
COUNT(列名)	指定した列の値が NULL 以外の行数を求めます
COUNT(*)	行数を求めます
MIN(列名)	指定した列の値の最小値を求めます
MAX(列名)	指定した列の値の最大値を求めます
SUM(列名)	指定した列の値の合計を求めます

1.5 インデックスの設定

テーブルの列ごとにインデックス (索引) を設定することができます。検索対象として使われる列にインデックスを設定すると、検索の高速化ができます。これはインデックスが設定されていない場合、列に指定した値が入っているレコードを探すためには、テーブルの全てのレコードを順番に見る必要があるためです。テーブルとテーブルを結合する場合も、対応するレコードを全て探す必要があるため、インデックスの効果はレコード数が多い場合絶大的ものになります。ただテーブル内のデータの変更に合わせて、インデックスの内容も更新しなければならないので、データの変更が頻繁に大量に行われる場合は、システムへの負担が問題になるかもしれません。

インデックスの設定は SQL でも行うことができますが、ここでは DB Browser で設定する方法を紹介します。まず DB Browser でインデックスの設定をしたいデータベースファイルを開き、図 1.3 のようにまず①「データベース構造」のタブを選択し、②「インデックスの作成」をクリックします。すると図 1.4 のような設定のウィンドウが出てきますので、①適当な名前を入力し、②のところでインデックスを設定したいテーブルを選択し、③のところでインデックスを設定したい列名をクリックして、④のボタンで右側に列名を送ります。すると⑤のところの「OK」ボタンをクリックできるようになるので、クリックします。なお、各テーブルの主キーに対しては、最初からインデックスの設定がされているので、このような設定をする必要はありません。

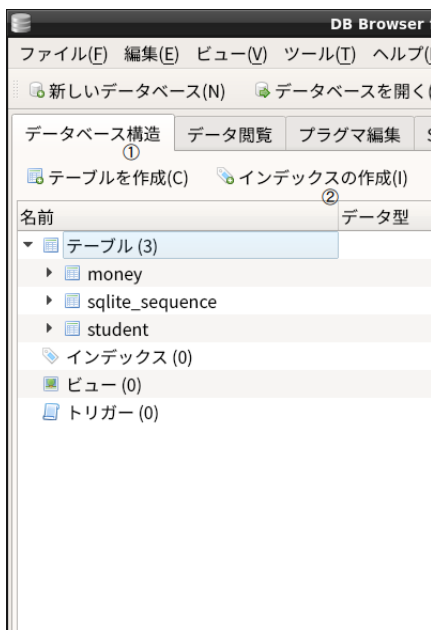


図 1.3 インデックスの設定の呼び出し



図 1.4 インデックスの設定画面

2. 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。

2.1 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、それを利用し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。
- `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 2.1 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 2.1 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「<http://www.sugiyama-u.ac.jp/>」です。

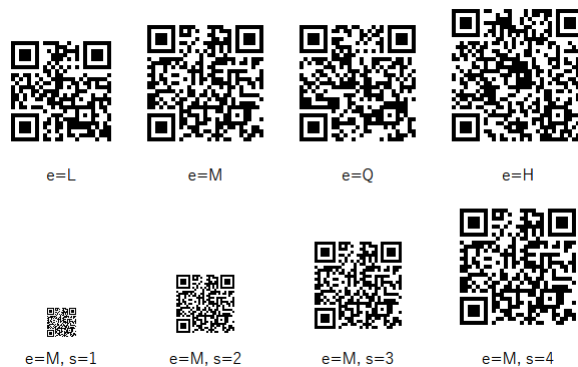


図 2.1 QRコード出力の例

2.2 QRコードによる決済

図 2.2 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 % 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。

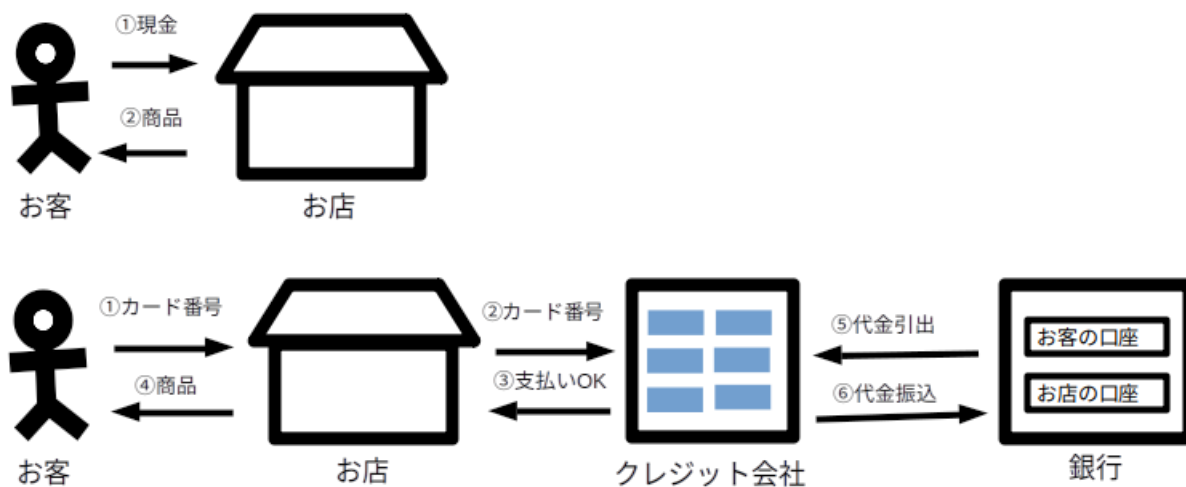


図 2.2 現金やクレジットカードによる支払い

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 2.3 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、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けば、取り返すために手数料を上げて逃げられません。

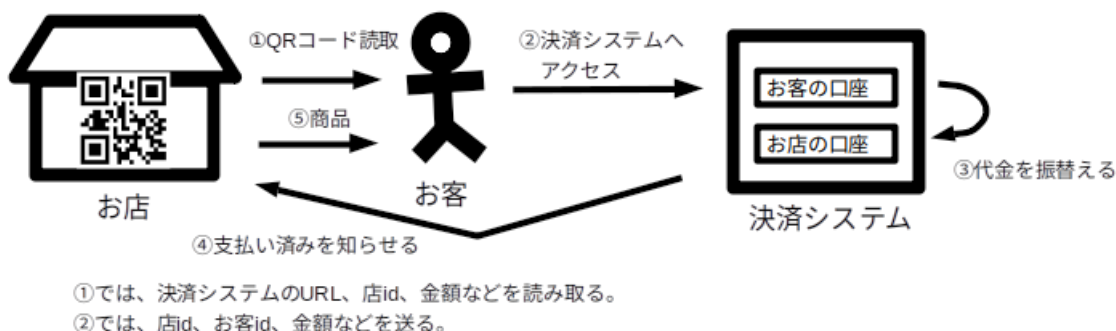


図 2.3 QR 決済による支払い

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、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なっています。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ね。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 2.4 のような構成にします。

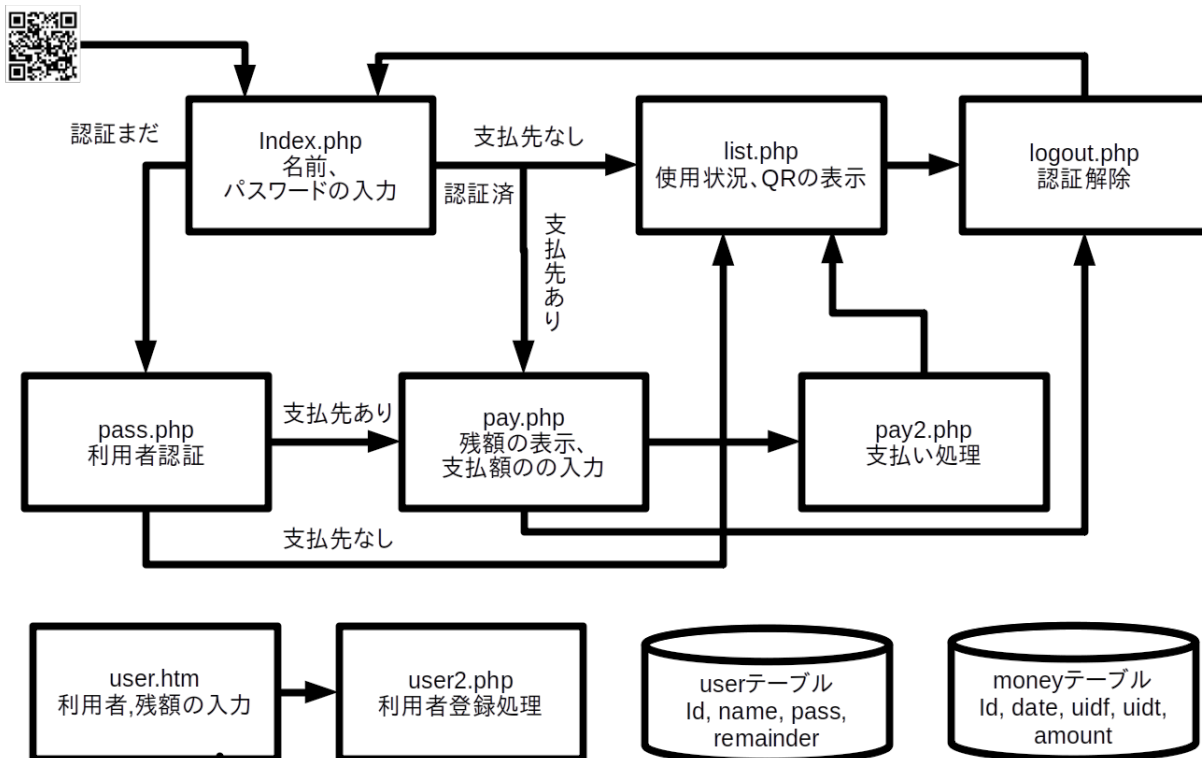


図 2.4 QRコード決済システムの構成

- 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 2.4 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。

2.3 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」とい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 目では 100 回、2 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、関数の外のものと同じになります。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。

2.4 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() という関数があります。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() という関数があります。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 2.1 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 2.1 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)		

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。コピー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」とい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 `mailto` 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);
```

`$to` にはメールを送る先のメールアドレスを入れておきます。`$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。最後の\$headers には次のようにして内容を入れます。

```
$headers="From: ".$from."\n";  
$headers.="Reply-To: ".$from."\n";  
$headers.="MIME-Version: 1.0 \n" ;  
$headers.="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。よって\$from には差出人のメールアドレスを入れておきます。「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め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。そのような場合は変数に入れずに、直接メールアドレスを入れても構いません。「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。同じ変数に何度も入れても大丈夫なのか？についてはよく例を見ると2行目から「.=」になっているので、現在の変数内容への追加になっています。

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。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、そのまともを利用者に 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。ただこのやり方では、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、割と複雑なので避けました。`peditor` で実行させてちゃんと動くことが確認できれば、端末で実行を予約するという流れになります。

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 とい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 -e`」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件につき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「`30 * * 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	*にすると何分でも実行します。0~59 の数値を一つか、複数を「,」で区切って並べます。
時	*にすると何時でも実行します。0~23 の数値を一つか、複数を「,」で区切って並べます。
日	*にすると毎日でも実行します。1~31 の数値を一つか、複数を「,」で区切って並べます。
月	*にすると毎月でも実行します。1~12 の数値を一つか、複数を「,」で区切って並べます。
週	*にすると毎日でも実行します。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、それ自体古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？それは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、近いところのものを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い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^{*1} というサービスでは、年額 1,500 円程度で 1 回につき 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

^{*1} 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わるので、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。その代わりにカメラのところに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。処理を行って何をし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。そうす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ります。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^{*2}が必要となり現実的な時間で正解^{*3}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、というような解法があるのでそちらで我慢するか、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ろう形にすると、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、二酸化炭素濃度を測定して表示します。そして 20 秒毎にサーバーにデータを送ります。送られてきたデータに時刻情報を追加したものを登録された URL に転送します。よって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。POST 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. 本来ならば環境センサーにデータの送り先を設定するところですが、リモートでそのような設定を受け付けるように作るのは面倒なので、環境センサーからのデータは mars で一旦受け取って、それを必要なところに転送するようにしています。<https://mars.mgt.sugiyama-u.ac.jp/...> へアクセスします。「受信プログラムの登録」のところに、作成した受信プログラムの URL、mars での自分の id とパスワードを入力して、**登録**をクリックします。登録を削除したい場合は、同じページにある「受信プログラムの登録の削除」で削除できます。
3. 同じ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、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しょう。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

^{*2} ゴミ箱の数が n ならば $n!$ に比例する時間がかかります。

^{*3} 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ね。

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」*4を取り上げます。

1. `<Head>`と`</Head>`の間に以下を入力します。

```
<Script
  src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/3.5.1/chart.min.js">
</Script>
```

これでバージョン 3.5.1 の chart.js が取り込まれます。このテキストを作成する際に配布元の `cdnjs.cloudflare.com` へアクセスしたところ、その時点では 2.8.0 へのアクセスが多いと出ていました。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、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も構いません。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。

*4 <https://www.chartjs.org/>

```
<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。そ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*⁵、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

*⁵ 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,value){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
6   src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/3.5.1/chart.min.js">
7 </Script>
8 </Head>
9 <Body>
10 <Div style="width: 600px; height: 300px; border: 1px solid black;">
11 <Canvas id="myGraph"></Canvas>
12 </Div>
```

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

リスト 1 について補足すると：

- 10 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。
- 17 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。例えば次のような感じです。

```

labels: [<?php
echo "'9/1'";
for ($i=2;$i<=7;$i++) {
    echo ", '9/", $i, "'";
}
?>],

```

- 22 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。
- 23 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。
- 21 行目と 27 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。

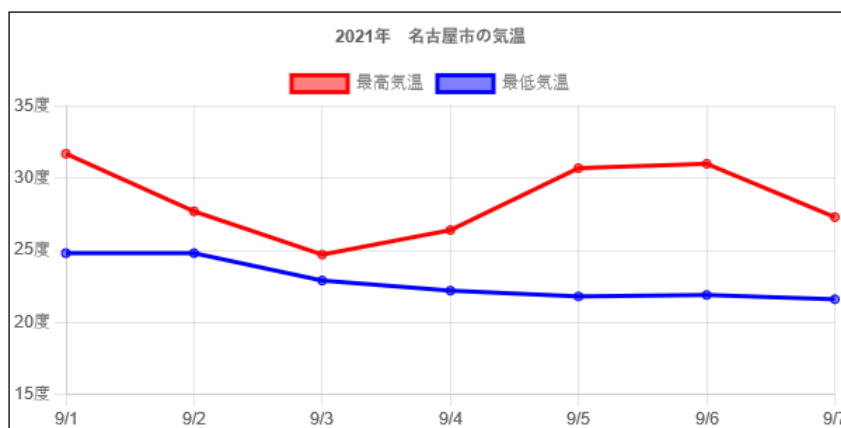


図 4.1 折れ線グラフの例

4.7 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りますので注意します。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ると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「<」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きます。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 (KHTML
                    , like Gecko) Chrome/21.0.1180.75 Safari/537.1"
5      )
6  );
7  $page=file_get_contents($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' => $cookie,
9          'header' => "User-Agent: Mozilla/5.0 (Windows NT 6.2) AppleWebKit/537.1 (KHTML
                    , like Gecko) Chrome/21.0.1180.75 Safari/537.1",
10         'header' => 'Content-type: application/x-www-form-urlencoded',
11         'content' => http_build_query($param),
12     )
13 );
14 $page=file_get_contents($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、この行は削除します。
- 9 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。
- 14 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」とい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>」にあるはずですが、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^{*6}」を紹介します。他にも「phpQuery^{*7}」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分かることが分かります。2021年7月の時点では「2.0-RC2」が最新のようにですが、アクセス数を見ると「1.9.1」の方が多いためそちらにします。「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」をペディタ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("aaa.htm");
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なります。

```
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。

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

^{*6} <https://simplehtmldom.sourceforge.io/>

^{*7} <https://code.google.com/archive/p/phpquery/downloads>

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 と組み合わせると、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 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 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 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 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、JSON は扱いに慣れておけばあちこちで役に立ちます。

6.1 JSON とは

データの受け渡しには、JSON 以外に古くから使われている形式として XML (Extensible Markup Language) 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できます。

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 など扱う場合も XML より字数が少なく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。

6.2 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があります。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あれば、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/>
 - 駅すばあと Web サービス
「駅すばあと」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/>

6.3 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。

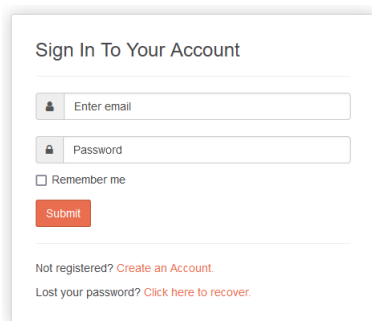


図 6.1 Open Weather Map のログイン画面

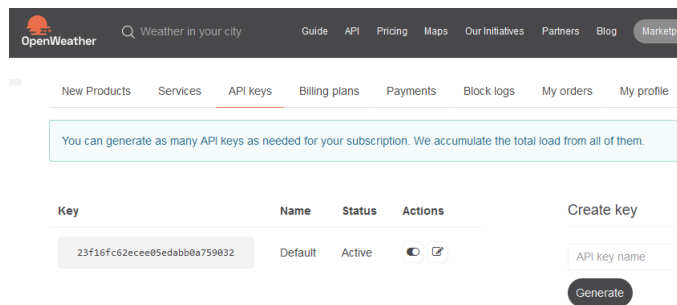


図 6.2 API キーの取得

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。

```
$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、`$cur['dt']` を取り出すというような多段階の取り出しをしていますが、多次元配列の表記を利用すると `$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。

```
JSON 生データ ヘッダー
保存 コピー すべて折りたたむ すべて展開 文字列を引用
lat: 35.1814
lon: 136.8964
timezone: "Asia/Tokyo"
timezone_offset: 32400
current: {}
  minutely: [-]
  hourly: [-]
  daily: [-]
```

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 (<code>date()</code> 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. 演習課題

7.1 QRsystem (1) 10/6

これから3週間ぐらいかけてQR決済システムを作成します。

1. peditor で「QRsystem」という名前のディレクトリを作成する。以下のファイルは全てこの中に作成する。
2. テキストの p.2 の表 1.1 と表 1.2 の内容のデータベースを作成する。ファイル名は「data.db」とする。授業中に作成したファイルをコピーしても良い。
3. 図 7.1 の左側のような入り口となる「index.php」を作成する。「送信」をクリックすると「pass.php」へ行くようにする。
4. 「pass.php」ではデータベースを参照して、Name と Password が一致したら「list.php」へ行くようにする。その際に\$_SESSION[' id '] に student テーブルの id、\$_SESSION[' name '] に student テーブルの name を入れておく。一致しなければ「index.php」へ戻る。
5. 「list.php」では\$_SESSION[' id '] の内容を元に、図 7.1 の右側のように使用状況や残高を表示する。

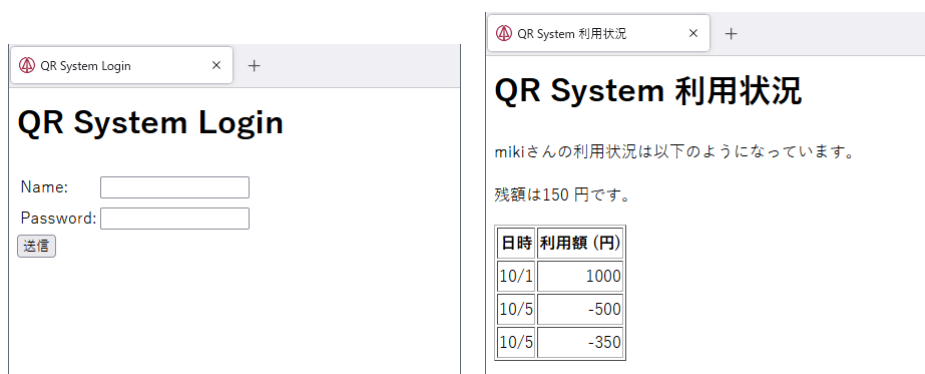


図 7.1 作成例

7.2 QRコード表示 10/13

図 7.2 のような QR コードを表示する k1013.php を作ってください。

1. URL を入力し、誤り訂正レベルを選択し、大きさを入力して、「QRコード表示」をクリックすると、QRコードが出るようにしてください。最初は何も入力されていないので QR コードは出しません。
2. QRコードを表示した後、URL と大きさの入力内容が消えないようにしてください。
3. 可能であれば、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が出るようにしてください。

7.3 QRsystem (2) 10/20

QRコード決済システム作成の続きをします。テーブルの名前がこちらは user になっているのを忘れていて10月6日の課題は student でやってしまいました。どこから user でやったら良かったのかなあ。今回はその修正からスタートして、利用者の登録の部分を作成します。

1. 「QRsystem」ディレクトリの中にある data.db を DB Browser for Sqlite で開いて、student テーブルの名前を user に変更し、ついでに Integer の入る remainder という項を追加します。値としては以下

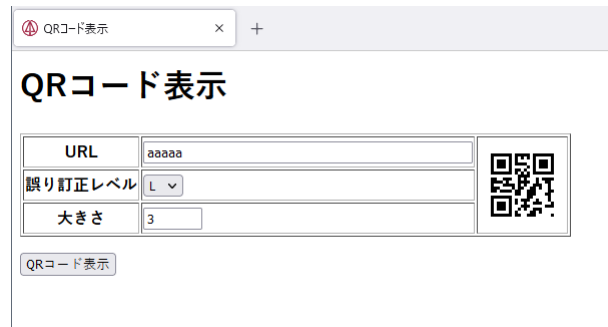


図 7.2 作成例

のように全員 1000 を入れておいてください。

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

2. 同じく data.db の money テーブルは、sid の項を削除し、代わりに Integer の入る uidf と uidt という項を追加します。date の項は Text が入るようになっていますが、Integer が入るように変更してください。そして money テーブルの中身は以下のように修正してください。

id	uidf	uidt	date	amount
1	1	2	1634630000	500
2	2	3	1634631000	500
3	3	1	1634632000	500

3. 図 7.3 のような追加する利用者の情報を入力する user.htm を作成する。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べきだが、本筋とは関係ないので今回は認証はしない。

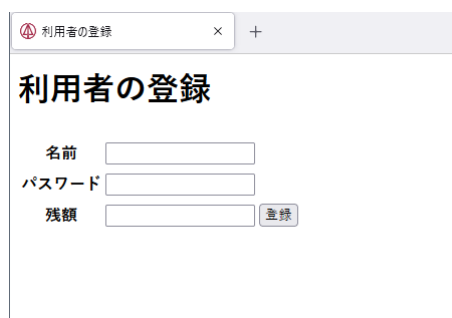


図 7.3 作成例

4. 図 7.4 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、
- 既に作成した pass.php や list.php の最初のデータベースなどに係る共通した部分をコピーして common.php を作成し、user2.php では最初これを取り込むようにする。
 - 既に同じ名前の利用者が居た場合は、図 7.4 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す。user.htm に戻る必要はない。

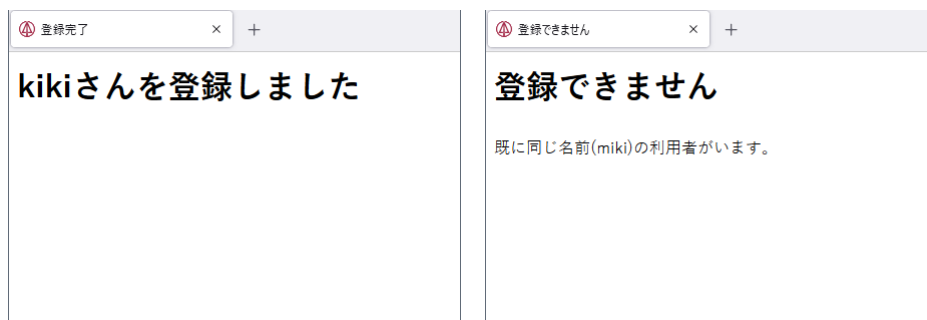


図 7.4 作成例

7.4 QRsystem (3) 10/27

10月6日に作成した index.php、pass.php と list.php を修正し、機能を追加し、新たに logout.php を作成します。

1. 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=1&pay=300」で支払先の利用者の ID は 1 で、支払金額は 300 円であることを示します。これらは \$_GET['to'] や \$_GET['pay'] で取り出せます。
2. index.php のまず最初に支払先があれば支払先と金額を \$_SESSION[] 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。
3. \$_SESSION['id'] がない場合は、図 7.5 の左側のように Name と Password を入れる事ができるようにします。「送信」をクリックすると pass.php へ行くようにします。

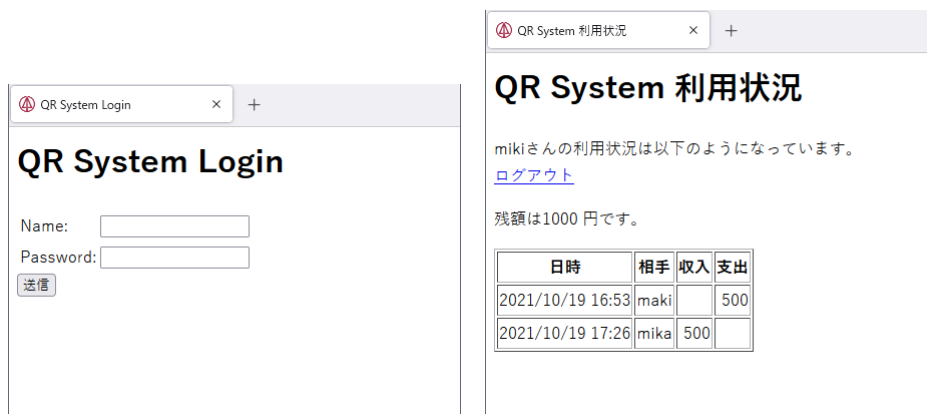


図 7.5 作成例

4. pass.php ではデータベースを参照して、Name と Password が一致しなければ index.php へ戻る。この部分はテーブル名が変わったのでちょっと直さないといけません。一致していれば list.php へ行くようにする。その際に \$_SESSION['id'] に user テーブルの id を入れておく。(name は \$_SESSION[] には入れないようにする。)
5. list.php では \$_SESSION['id'] の内容を元に、図 7.5 の右側のように使用状況や残高を表示する。利用者の名前と残高は user テーブルから取り出します。ログアウトのリンクをクリックすると logout.php へ行くようにします。
6. logout.php では \$_SESSION['id'] を消去して index.php へ行くようにします。

7. pay.php まで作成する元気がないと思うので、これは来週作成する事にします。

7.5 QRsystem (4) 11/03

今回は支払いの実行部分である pay.php と pass2.php を作成します。その前に前回の課題で扱った pass.php や list.php が common.php を利用していないようであれば、利用した形に修正しましょう。

1. index.php による「QR System Login」でデータベースに登録されている Name (miki) と Passowrd (12345) を入力して「送信」をクリックすると、list.php による「QR System 利用状況」が表示されるはずですが、ここでブラウザで表示されている URL を「http://mars.../index.php?to=2&pay=300」のように書き換えて Enter キーを押すと、支払先の利用者の ID は 2 で、支払金額は 300 円であることになり pay.php が呼ばれますが、まだ作っていないのでエラーが表示されます。
2. pay.php では、まず最初に支払先と金額を\$_SESSION[] から取り出して図 7.6 の左側のように表示してみましょう。index.php をコピーして始めると良いでしょう。この場合、金額は良いのですが、支払先が数字で表示されて誰だか分からないものになります。「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。

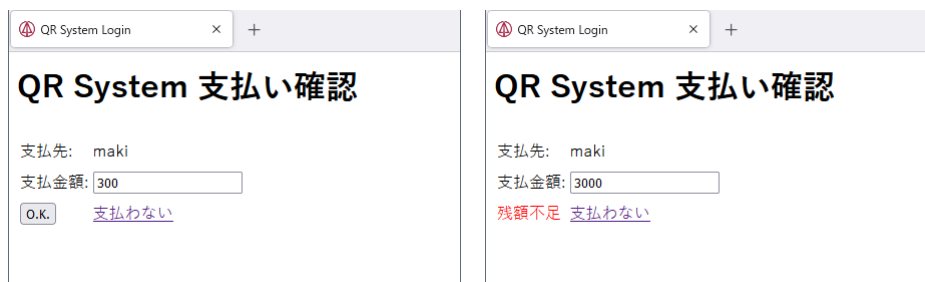


図 7.6 作成例

3. pay.php で\$_SESSION[] に入っていた支払先を元にデータベースを検索し、支払先が名前で表示されるようにしましょう。
4. pay.php でさらに\$_SESSION[] に入っていた支払元を元にデータベースを検索し、支払金額が残額を超える場合は、図 7.6 の右側のように「O.K.」ボタンの代わりに「残額不足」と表示するようにします。
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。日時については先週やった time() の数字を入れます。また user テーブルの残額の修正を 2 人分忘れないように、データベースの更新ができれば 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 だけでやるのが可能です。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. 先週の課題で修正した 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。

7.6 QRsystem (5) 11/10a

本日は 2 コマ授業があるので、課題も 2 つあります。QR 決済システムは前回でほぼ完成しましたが、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 を改良し、list2.php を追加します。また

session 情報がブラウザを閉じると消えてしまうので、設定を少し変更します。

1. 前期のテキストの 54 ページあたりを参考にして、QRsystem のところに .htaccess ファイルを作成せよ。有効期間は 10 日間とする。 .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る。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. list.php を修正して、図 7.7 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」のボタンを追加せよ。このボタンをクリックしたら list2.php を呼ぶようにする。前回入力した請求金額が予め入るようにしておく。前回が無ければ 0 が入るようにする。



図 7.7 作成例

3. list2.php では図 7.7 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、「利用状況の表示」のボタンを表示する。このボタンをクリックしたら list.php を呼ぶようにする。urlencode() を使用しないと正しい QR コードができないので注意する。

7.7 メールの定期送信 11/10b

本日は 2 コマ授業があるので、課題も 2 つあります。

1. mars の自分自身に一日に 3 回メールを送るものを作成します。ファイル名は mail1110.php にしてください。呼び出されると 7:00 から 12:00 までは、件名として「おはようございます」、本文としては「今日も良い一日だとよいですね。」というメールを送ります。12:00 以降 17:00 までは件名として「こんにちは!」、本文としては「明るいうちががんばりましょう。」というメールを送ります。それ以外の時間では件名として「こんばんは!」、本文としては「夜は早く寝ましょう。」というメールを送ります。
2. cron を設定して mail1110.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 mail1110.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。

7.8 利用者の確認 11/17a

本日は 2 コマ授業があるので、課題も 2 つあります。これはそのうちの 1 つ目です。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. いくつかのファイルから構成されるので「Kakunin」と言うディレクトリーを作成し、そこに以下のファイルを入れるようにします。
2. QRsystem で使用した common.php とほぼ同じ内容の同じ名前のファイルを作成します。ほぼ同じ内容と言うのは、\$_SESSION[] を使わないので、session_start(); の行が不要だからです。
3. 「DB Browser for Sqlite」を利用して data.db を作成します。テーブルは「newuser」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。データは入れる必要はありません。
4. index.htm では図 7.8 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、「送信」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 7.8 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 には行ったかどうか確認しましょう。それから図 7.9 のように出るようにしましょう。



図 7.9 作成例

6. 前期のテキストの「INSERT 文」の説明を参考に、今データベースに入れたものの 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。
7. 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。
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??のように出るようにしてください。

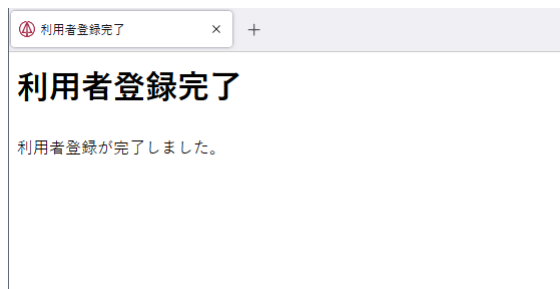


図 7.10 作成例

7.9 グラフの表示 11/27b

本日は2コマ授業があるので、課題も2つあります。これはそのうちの2つ目です。グラフのデータが配列型変数に入っている場合をやってみましょう。ファイル名は graph1117.php とします。まず、リスト1の折れ線グラフの<Body>タグの次に次のような内容を入力してください。

```
<?php
$hi=array(31.7, 27.7, 24.7, 26.4, 30.7, 31.0, 27.3);
$lo=array(24.8, 24.8, 22.9, 22.2, 21.8, 21.9, 21.6);
?>
```

これで\$hi[0]~\$hi[6]にそれぞれ31.7~27.3が入ります。同様に\$lo[0]~\$lo[6]にも同様に数値が入ります。これを利用して図7.11のようなグラフが出るようにしてください。

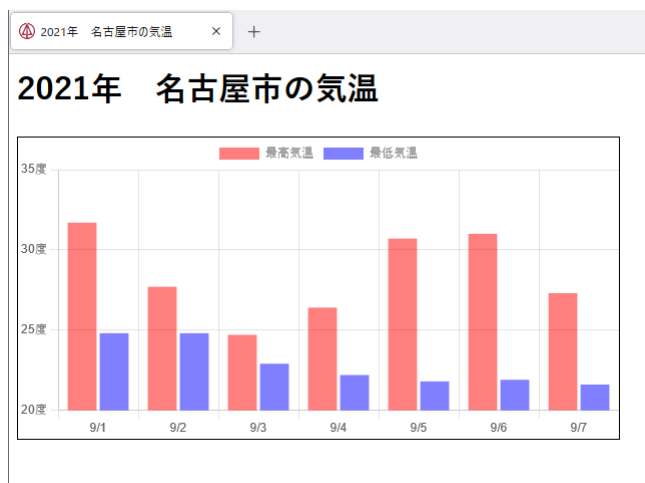


図 7.11 作成例

7.10 データベースからグラフ作成 11/24

環境センサーの準備がちっとも進んでいないので、今回はデータベースのデータをもとに円相場のグラフを表示するページを作成することにしました。

1. 「IoT」とい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 というテーブルがあり、そこには id、date、yen という列があり、今年になってから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 7.12 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。データベースから 15 件取り出すところは、foreach ではなく for を使ってデータベースの検索結果をまず配列型の変数に入れてグラフにすると、前回の課題とほぼ同様になります。



図 7.12 作成例

4. 10 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 1/4 からですが、更新されると 1/5 からになり、さらに更新されると 1/6 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伝える必要があります。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 1/4 から表示されるようにしてください。その後更新されると 1/5 からにならない事がありますので、注意してください。その原因は、グラフが表示されている画面をよく見ると分かります。

7.11 グラフの改良とスクレイピング 12/1

今回の課題は、グラフの改良とスクレイピングです。作成する 3 つのファイルは全て「IoT」ディレクトリに入れてください。

1. yengraph.php をコピーして yengraph2.php を作成して、図 7.13 のようにボタンでグラフの範囲を変えられるようにしてください。自動更新はやめます。「|<」のリンクでデータの最初の 15 件を表示します。「<」のリンクで 15 件前を表示します。最初より前に行かないようにしてください。「>」のリンクで 15 件後を表示します。最後より後に行かないようにしてください。「>|」のリンクでデータの最後の 15 件を表示します。データベースにデータは 15 件以上入っているものとしませんが、何件入っているかは毎回調べてください。(ヒント：今回は自動更新ではないので、\$_SESSION は必要ありません。)
2. 「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 7.14 の左のように表示されるようにせよ。ファイル名は kabmap.php にする。
3. 「QRsystem/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() でとってきた内容はそのまま



図 7.13 作成例

echo で出力すると図 7.14 の右のようになる。(ヒント：user2.php がどのような形でデータを user.htm から受け取っているかを見て考えましょう。)



図 7.14 作成例

7.12 環境センサーのデータ受信とグラフ作成 12/8

環境センサーが不完全ならがなんとか動き出したので、そちらに関する課題です。今回の課題のファイルも必ず「IoT」の中に入れてください。

1. data.db を新規に作成します。テーブル名は「data」にして、フィールドとして「id」、「date」、「temp」、「humi」、「co2」、「bright」を追加します。データ型は全て「INTEGER」です。「id」のみ AI と PK を設定します。
2. テキストの p.18 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。
3. receive.php ができたら、で登録してください。
4. data.db のデータを元に図 7.15 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 120 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。

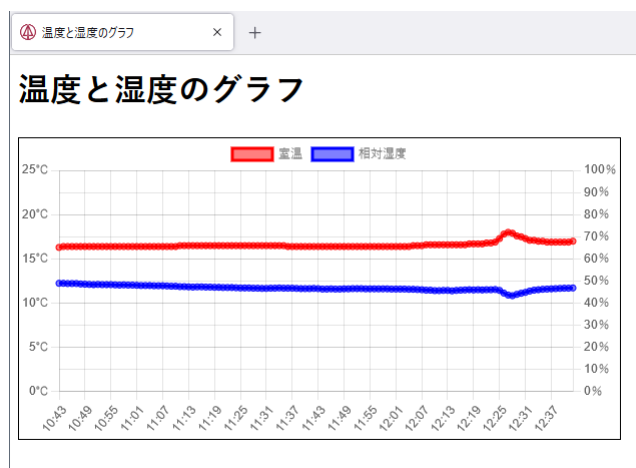


図 7.15 作成例

7.13 天気予報の表示 12/15

前回の課題ができた人が少なかったので再度やってもらうのと、「Open Weather Map」からデータをもらって表示するものにしました。

1. 先回の課題で作成した tempgraph.php を完成させてください。関連するファイルや設定ができていない場合はそちらも直してください。データの最後は sqlite_sequence テーブルから取り出してください。グラフ用のデータの取り出しは foreach を使う方法にしてください。グラフの設定の中の「borderColor:」の前に「radius: 0,」を追加すると図??のようにグラフの線が細くなります。

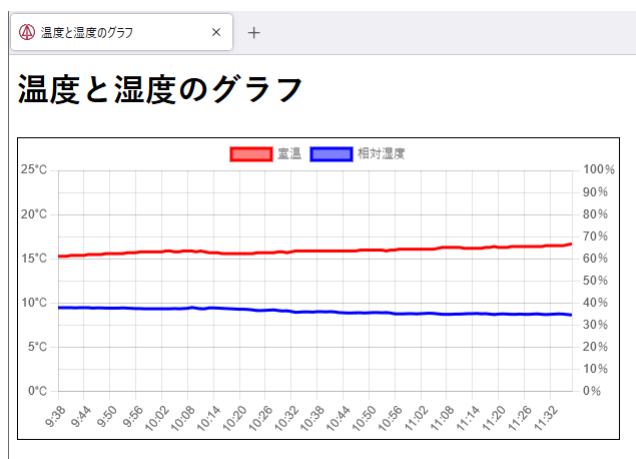



図 7.16 作成例

2. 「Open Weather Map」に登録して API キーを取得してください。
3. 「Open Weather Map」から 1 週間の天気予報を受け取り、図 7.17 のように表の形で表示する weather.php を IoT の中に作成してください。



年月日	最高温度	最低温度	湿度 (%)	気圧 (hPa)	風速 (m/s)	風向	天気
2021/12/15	12	6.06	48	1021	3.78	319	clear sky
2021/12/16	13.05	7.37	53	1023	3.15	13	light rain
2021/12/17	13.41	3.38	77	1009	9.53	301	rain and snow
2021/12/18	6.04	2.34	36	1020	7.63	295	rain and snow
2021/12/19	6.820000000000001	2.16	46	1018	4.24	326	light rain
2021/12/20	10.39	4.34	55	1018	1.97	323	clear sky
2021/12/21	11.44	5.05	50	1015	6.08	332	light rain
2021/12/22	9.820000000000001	4.3	40	1020	5.83	330	clear sky

図 7.17 作成例

7.14 NHK 朝ドラ予告 12/22

図??のような NHK の朝ドラの内容を、今日を含めて 6 日分、「NHK 番組表 API」を利用して表示するものを作ってください。ファイル名は IoT の中の nhk.php とします。

1. 「NHK 番組表 API」を利用するには、API Key が必要で、その取得には利用者登録が必要ですが、今回は私の取得した API Key を使ってください。1 日に 300 回と言う利用制限がありますので、使いすぎにご注意ください。
2. 番組の検索の際には、service として「NHK 総合 1」、area として「名古屋」を選択してください。
3. title の最初が「【連続テレビ小説】」なのが朝ドラです。ただし再放送や昔の朝ドラの放送もあるので、最初の一つだけ出力して、その後 break; で foreach から抜け出すようにした方が良いでしょう。
4. 今年から NHK の朝ドラは土曜日の放送の内容が、その週のまとめになりました。途中でまとめが入ると分かりにくいので、date() で曜日を調べて土曜日は出ないようにします。なお日曜日は朝ドラの放送はありません。

放送日時	内容
2021-12-21	雫真家を出るという決意を固めた安子（上白石萌音）でしたが、千吉（段田安則）は安子が出ることは受け入れたものの、るいを連れていくことは許しませんでした。「るいの額の傷を治すには莫（ばく）大な金がかかり、それは雫真繊維の力が無ければ不可能。雫真の子として育てられるのがるいにとって一番幸せなことなのだ」という千吉の言葉に反論できなかった安子。そんなある日、算太（濱田岳）が失踪したという知らせが入り…
2021-12-22	安子（上白石萌音）は、失踪した算太（濱田岳）を探して大阪の街を何日も歩き回り続けましたが、消息はつかめませんでした。疲労困憊（ばい）し、雨の中に倒れこんでしまった安子。目を覚ますと、そばにいたのはロバート（村雨辰剛）でした。その日がるいの入学式だと気が付いた安子は、岡山の家に電話をかけてるいの様子を確かめようとしてますが、るいがいなくなると聞き…
2021-12-23	成長したるい（深津絵里）は、雫真家を出て一人で生活を始めたいと岡山を離れることに。向かったのは大都会・大阪。見たこともないほど華やかな建物や街の人々の様子を見て、胸をときめかせるいでしたが、道でぶつかりそうになった自転車をよけたところ、仕事の面接のためにせっかく新調したワンピースを汚してしまいます。責任を感じた自転車の持ち主・竹村平助（村田雄浩）は、経営するクリーニング店にるいを連れていき…
2021-12-24	仕事の面接がうまくいかなかったるい（深津絵里）は、荷物を預かってくれたクリーニング店に戻り、事情を話しました。話を聞いてくれた店主の竹村平助（村田雄浩）と妻の和子（濱田マリ）の提案によって、るいは店に住み込みで働くこととなります。和子からクリーニングの仕事を一つ一つ丁寧に教えてもらいます。ある日、店番を任せられたのですが、そこにちょっと変わったお客さん（オダギリジョー）が訪れ…
2021-12-27	クリーニング屋での仕事が性に合ったるい（深津絵里）。時々やってくるちょっと変わったお客さん（オダギリジョー）の正体はつかめぬままでしたが、仕事にもだんだんと慣れていきました。そんなある日、るいが一人で店番をしていると、田中（徳井優）という強面の男が店にやってきて、預けた服に穴が空いていたとクレームをつけてきました。弁償だけでなく慰謝料を払えと迫られ、困ったるいでしたが…
2021-12-28	るい（深津絵里）は、時々クリーニング店にやってくる片桐という男に恋心を抱いていました。ある日、弁護士のお卯だという片桐から映画に誘われたるい。生まれて初めてのデートに心が沸き立ちます。一緒に暮らす和子（濱田マリ）と平助（村田雄浩）に温かく送り出され、片桐とのデートを楽しむいでしたが…。その後るいは、店を訪れていたちょっと変わったお客さん（オダギリジョー）の正体を知ることになります。

図 7.18 作成例

索引

A

API (Application Programming Interface) ... 28
 at: 実行予約をする (Unix) 14

C

Chart.js: グラフ描画ライブラリ 19
 cron: 定期的実行予約をする (Unix) 14

D

date(): 秒数を指定した形に変換する (PHP) ... 10

E

ER 図 1

F

file(): ファイルや web ページを取り込む (PHP) 24
 file_get_contents(): ファイルや web ページを取り込む (PHP) 24
 function: 関数の定義 (PHP) 9

G

GROUP BY: レコードのグループ化 (SQL) 3

H

HTML 解析ライブラリ 26

I

include: プログラムの取り込み (PHP) 9
 IoT (Internet of Things) 16

J

JSON (JavaScript Object Notation) 28
 json_decode(): JSON 形式を PHP の変数に変換する 28

L

LPWA (Low Power Wide Area-network) 16

M

mb_send_mail(): メールを送る (PHP) 12

O

Open Weather Map 天気予報情報のサイト 31
 ORDER BY: レコードの並び替え (SQL) 3

P

pclose(): Unix のコマンドとのやり取りを終える (PHP) 15
 PHP Simple HTML DOM Parser (PHP) 26
 popen(): Unix のコマンドを実行する (PHP) .. 15

Q

QR コード生成ライブラリ 5

S

SQL の関数 3
 system(): Unix のコマンドを実行する (PHP) . 15

T

time(): 時刻を求める 10

U

Unix のコマンドを実行する (PHP) 15

W

web ページの自動更新 (HTML) 23
 web ページの自動更新 (PHP) 23
 wget: web ページを取り込む (Unix) 13

X

XML (Extensible Markup Language) 28

あ

インデックスの設定方法 4

か

外部キー 2
 関数の定義 (PHP) 9
 グラフ描画ライブラリ (JavaScript) 19

さ

時刻を求める (PHP) 10
 実行予約をする (Unix) 14
 主キー 1
 スクレイピング (Web scraping) 24

た

定期的実行予約をする 14
 テーブルの結合 (INNER JOIN) 3
 テーブルの結合 (PHP) 3
 テーブルの結合 (SQL) 2
 天気予報情報のサイト 31

は

秒数を指定した形に変換する (PHP) 10
 ファイルや web ページを取り込む (PHP) 24
 プログラムの取り込み (PHP) 9

ま

メールを送る (HTML) 12
 メールを送る (PHP) 12

ら

レコードのグループ化 (SQL) 3
 レコードの並び替え (SQL) 3