Monaca

WebAPIとデータベース

アシアル株式会社
アシアル情報教育研究所
岡本 雄樹

■ 気象庁のWebAPI

- 気象庁のWebサイトが2021年度からWebAPI化されました
- 気象庁のWebサイトで使われているWebAPIを外部から参照可能です
- つまり、アプリの中から呼び出せます
- ただしサポートが行われているわけではないため気象庁には問い合わせないで下さい
- 詳しくはあんこエデュケーションの記事を参照
  - https://anko.education/apps/weather_api

■ WebAPI機能付きの簡易データーベース

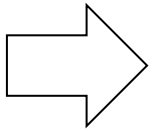
- Monaca Education簡易データベース機能・2021年度より提供開始
- データを表計算ソフトのように管理できます
- WebAPI機能による登録や参照もサポート
- つまり、Monaca以外の環境からも利用可能

気象庁のWebAPIを試す

■ サンプルアプリの紹介①

- 気象庁のWebAPIから情報を取得します
- 取得した情報の値を元に画面を生成しています
- 例えば東京の値は以下のURLから取得できます
  - https://www.jma.go.jp/bosai/forecast/data/overview_forecast/130000.json


```
{
  "publishingOffice":"気象庁",
  "reportDatetime":"2021-07-
12T10:39:00+09:00",
  "targetArea":"東京都",
  "headlineText":"東京地方、伊豆諸島
では、**********,
  "text":"　梅雨前線が、********* "
}
```

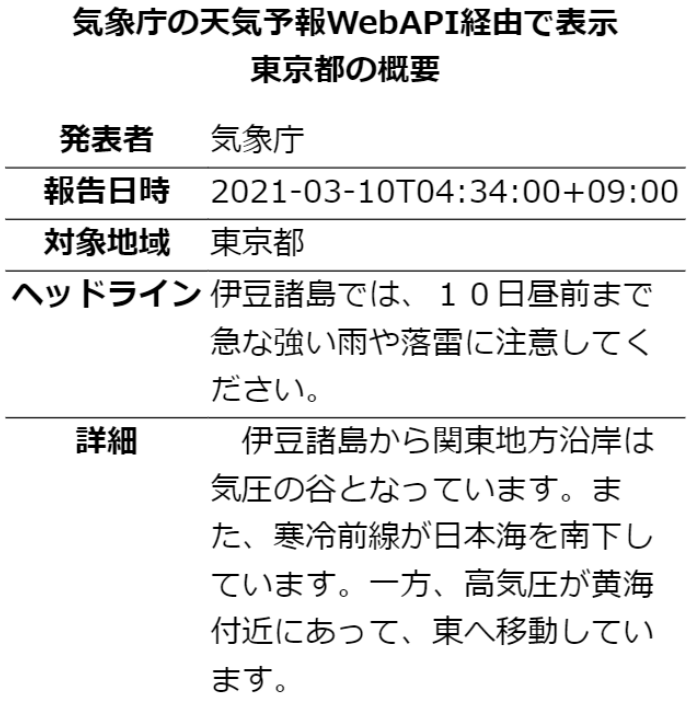
気象庁の天気予報WebAPI経由で表示
東京都の概要

発表者	気象庁
報告日時	2021-03-10T04:34:00+09:00
対象地域	東京都
ヘッドライン	伊豆諸島では、１０日昼前まで急な強い雨や落雷に注意してください。
詳細	伊豆諸島から関東地方沿岸は気圧の谷となっています。また、寒冷前線が日本海を南下しています。一方、高気圧が黄海付近にあって、東へ移動しています。

■ JSONとは？

- WebAPIでデータをやり取りするときに使われる形式の一つ
- JSONのJSは『JavaScript』のJS、ONは『オブジェクト表記』
- JavaScriptの連想配列との相互変換が容易
- 現在では、JavaScript以外の言語でもJSONがサポートされている

【実習】サンプルアプリを動かしてみよう



気象庁の天気予報WebAPI経由で表示
東京都の概要

発表者	気象庁
報告日時	2021-03-10T04:34:00+09:00
対象地域	東京都
ヘッドライン	伊豆諸島では、１０日昼前まで急な強い雨や落雷に注意してください。
詳細	伊豆諸島から関東地方沿岸は気圧の谷となっています。また、寒冷前線が日本海を南下しています。一方、高気圧が黄海付近にあって、東へ移動しています。

■ お天気アプリ（サンプルアプリ）の解説記事

└ https://anko.education/apps/weather_api

■ HTML側

```
<body>
    <h1>気象庁の天気予報WebAPI経由で表示<br>東京都の概要</h1>
    <table>
        <tr id="publishingOffice">
            <th>発表者</th>
            <td></td>
        </tr>
        <tr id="reportDatetime">
            <th>報告日時</th>
            <td></td>
        </tr>
        <tr id="targetArea">
            <th>対象地域</th>
            <td></td>
        </tr>
        <tr id="headlineText">
            <th>ヘッドライン</th>
            <td></td>
        </tr>
        <tr id="text">
            <th>詳細</th>
            <td></td>
        </tr>
    </table>
</body>
```

■ JavaScript側

```
// 東京(130000)の予報を取得
let url = "https://www.jma.go.jp/bosai/forecast/data/overview_forecast/130000.json";

fetch(url)
   .then(function(response) {
       return response.json();
   })
   .then(function(weather) {
       console.log(weather);
       // 画面に書き出す
       document.getElementById("publishingOffice").lastElementChild.textContent = weather.publishingOffice;
       document.getElementById("reportDatetime").lastElementChild.textContent = weather.reportDatetime;
       document.getElementById("targetArea").lastElementChild.textContent = weather.targetArea;
       document.getElementById("headlineText").lastElementChild.textContent = weather.headlineText;
       document.getElementById("text").lastElementChild.innerHTML = weather.text.replace(/¥n¥n/g, '<br>');
   });
```

■ 地域を変更してみよう

- 東京都の都道府県コードは13番
- 今取得しているJSONのファイル名は130000.json
- 13の部分を別の都道府県コードに変えたら？？？

■ 明日・明後日の予報も確認するためには？

└ 記事の中では気象庁の別のAPIを使ったサンプルも紹介されているようです

気象庁の天気予報WebAPI経由で表示
東京の詳細

発表者	気象庁
報告日時	2021-03-10T05:00:00+09:00
対象地域	東京地方
今日の天気	晴れ
明日の天気	晴れ　夜　くもり
明後日の天気	

データベース

■ データベースとは

- データの基地

■ データベースの種類

- リレーショナルデータベース
  - 昔から使われており、現在でも主力のデータベース。RDBを操作するSQL言語は国家試験にも出る
  - データを表計算ソフトのシートのような、『表』の形で管理する
- NoSQL型データベース
  - RDBやSQLとは違うタイプのデータベースの総称
- キーバリュー型データベース
  - NoSQL型でも特に、『キーと値のペア』でデータを管理するタイプのデータベース
  - 連想配列やJSONのようなキーバリュー型でデータを保持する
  - ブラウザのWebStorageはキーバリュー型データベースの一種

■ リレーショナルデータベース

リレーショナルデータベース(関係データベース)はIBMのエドガー・F・コッド氏が発明した関係モデルを元にしたデータベースで、商用のリレーショナルデータベースが最初に登場したのは1979年と比較的古いですが、現在でも主要なデータベースの一つです。

■ 関連技術

リレーショナルデータベースにはデータベースの表を設計するための正規化理論やER図、データを取得するためのSQL(Structured Query Language) という強力な問い合わせ言語が備わっています。

■ 【参考】SQLを活用したサンプルアプリ

JavaScript製のデータベースエンジンAlaSQLを活用したRDBの解説記事とサンプルアプリが、あんこエデュケーションに掲載されています。

https://anko.education/joho/network/relational_database

■ 配列のキーの特徴

└ 配列のキー(添え字)は0から始まる連番です。連番は自動的に採番されるので宣言も楽ですし、for文での繰り返し処理も容易です。

■ 連想配列のキーの特徴

└ キーは個別に定義します。連番である必要はなく、また、文字列を使うこともできます。

```
sweetsLabel = {
    japanese:"和菓子",
    western:"洋菓子"
}
```

■ 配列と連想配列の応用

- 配列の値に配列を入れ子にすることが可能です。また、配列と連想配列を入れ子にすることも可能です。
- 入れ子を活用すれば様々なデータを格納することが出来ます
- 入れ子を応用すればデータベースのように活用することも可能となります。

■ 応用例

```
sweets = {
    japanese:["もなか", "だんご", "すあま", "ようかん"]
    western:["ビスケット", "クッキー", "パンケーキ"]
}
```

仮にこの配列からビスケットを参照したい場合プログラミング上では
sweets.western[0]　あるいは　sweets["western"][0]
と参照します。

■ Monaca Education簡易データベース

└ 表計算ソフト風の管理画面を備えたデータベース

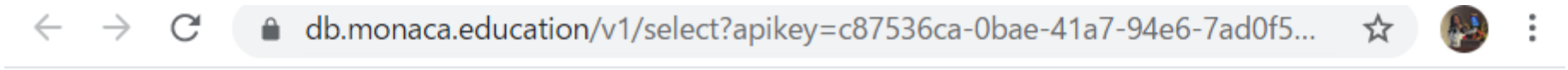
ID ↑	整数1	整数2	浮動小数...	浮動小数...	文字列1	文字列2
1	1000	1			コーヒー豆の選別のお手伝い	
2	150	-1			最中	
3	200	-1			桃	

■ Monaca Education簡易データベース

└ WebAPIキーの発行が可能

■ Monaca Education簡易データベース

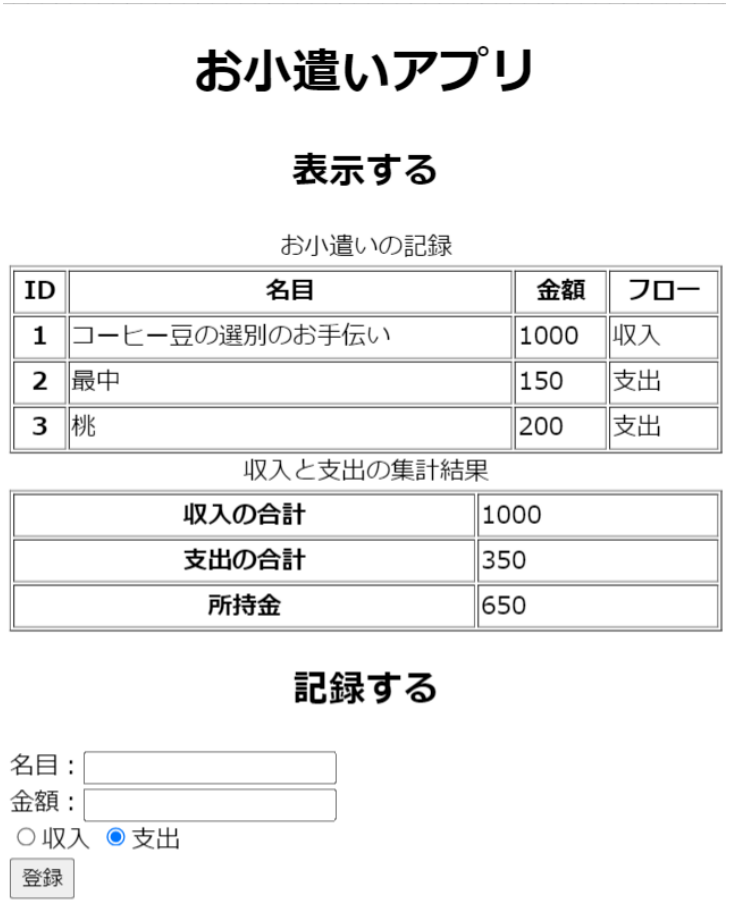
- WebAPIキーがあればデータをJSONで参照できる
- もちろん、登録や削除も可能




```
["status":200,"records":
[{"OID":"60b8e015726518154d71e546","created":1622728725,"updated":1622728725,"serial":1,"int1":1000,"int2":1,"int3":null,"int4":null,"float1":null,"float2":null,"float3":null,"float4":null,"text1":"¥u30b3¥u30fc¥u30d2¥u30fc¥u8c46¥u306e¥u9078¥u5225¥u306e¥u304a¥u624b¥u4f1d¥u3044","text2":null,"text3":null,"text4":null},
{"OID":"60b8e1c7c0a21207d972de7a","created":1622729159,"updated":1622729159,"serial":2,"int1":150,"int2":-1,"int3":null,"int4":null,"float1":null,"float2":null,"float3":null,"float4":null,"text1":"¥u6700¥u4e2d","text2":null,"text3":null,"text4":null},
```

■ Monaca Education簡易データベース

└ ちょっとしたアプリならこれで開発可能



【実習】Monaca Education簡易データベースの操作

お小遣い帳のサンプルアプリを動かしてみよう。

なお、最初に設定されているキーは『所長の読み取り専用キー』なので追記できません。『自分のマスターキー』に変更して下さい。

■ お小遣い帳(サンプルアプリ)の解説記事

└ https://anko.education/apps/money

■ 簡易DBリファレンス記事

└ https://anko.education/tool/db

■ HTML側：データ表示部分

- table要素でデータ表示部分を用意
- データベースの値をJavaScriptで取得し、tr要素として後から追記

```
<table border="1" id="moneyRecord">
    <caption>お小遣いの記録</caption>
    <tr>
        <th>ID</th>
        <th>名目</th>
        <th>金額</th>
        <th>フロー</th>
    </tr>
</table>
```

お小遣いの記録

ID	名目	金額	フロー
1	コーヒー豆の選別のお手伝い	1000	収入
2	最中	150	支出
3	桃	200	支出

■ JavaScript側：データ取得部分①

```
let apikey = "8c6d68a4-7bb8-4d62-84ff-348b47a1fbda";
select();

function select () {
    let url = "https://db.monaca.education/v1/select?apikey=" + apikey;
    console.log(url);
    fetch(url)
    .then(function(response) {
        return response.json();
    })
    .then(function(db) {
            console.log(db);
```

■ JavaScript側：データ取得部分②

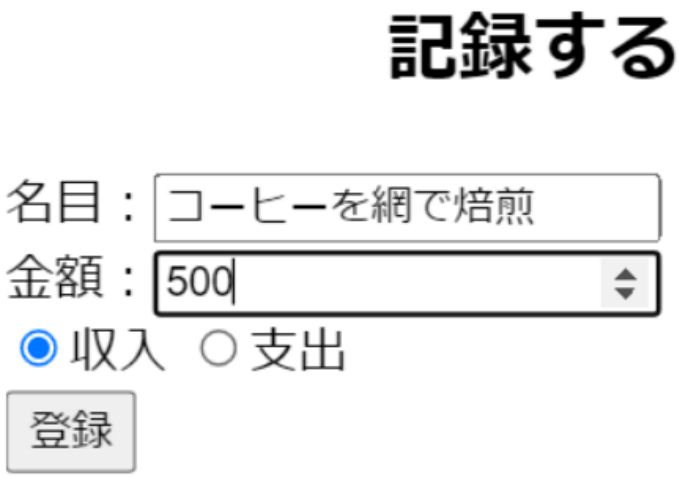
```
let moneyRecord = document.getElementById("moneyRecord");

for (let i = 0; i < db.records.length; i++) {
    // DOM準備
    let tr = document.createElement("tr");
    let id = document.createElement("th");
    let title = document.createElement("td");
    let price = document.createElement("td");
    let flow = document.createElement("td");
    // DOMに値を格納
    id.innerHTML = db.records[i].serial;
    title.innerHTML = db.records[i].text1;
    price.innerHTML = db.records[i].int1;
    if(db.records[i].int2 == 1) {
        flow.innerHTML = "収入";
    } else if (db.records[i].int2 == -1) {
        flow.innerHTML = "支出";
    } else {
        flow.innerHTML = "□□□";
    }
    // 構築したDOM要素を表に追加
    tr.appendChild(id);
    tr.appendChild(title);
    tr.appendChild(price);
    tr.appendChild(flow);
    moneyRecord.appendChild(tr);
}
```

■ HTML側：データ表示部分

- form要素でデータ入力部分を用意
- 登録ボタンが押されたら関数insert()を実行

```
<form id="insertForm" autocomplete="off">
    名目：<input type="text" name="title"><br>
    金額：<input type="number" name="price" ><br>
    <input type="radio" name="flow" value="1" >収入
    <input type="radio" name="flow" value="-1"
checked>支出<br>
    <button type="button" onclick="insert()">登録
</button>
</form>
```



■ JavaScript側：データ登録部分

```
function insert (){
    // フォ□ムの値を取得
    let form = document.getElementById("insertForm");
    let title = form.title.value;
    let price = form.price.value;
    let flow = form.flow.value;

    // DBに登録
    let url = "https://db.monaca.education/v1/insert?apikey=" + apikey;
    url += "&text1=" + title + "&int1=" + price + "&int2=" + flow;
    fetch(url)
        .then(function(response) {
            return response.json();
        })
        .then(function(result) {
            console.log(result);
            location.reload();
    });
}
```