川崎市のイベント情報のオープンデータ API利用ガイド（rev1.2）

● 概要

川崎市公式アプリ「かわさきイベントアプリ」では、川崎市内のさまざまなイベント情報を、行政、民間の隔てなく一体的に発信しており、 そのイベント情報をWebAPI機能によりオープンデータとして提供しています。 また、本APIでは、イベントの種類や開催場所、開催日時などの条件を指定することで、取得するイベント情報を絞り込むことも可能です。

ご利用に際しては、必ず、「川崎市のオープンデータ利用規約」をご一読の上、活用ください。

● HTTP リクエストの送信

利用者側にてプログラムを作成していただき、HTTP リクエストを送信してください。

> GET https://eventapp.city.kawasaki.jp/data/api/v1/events?format={レスポンス形式}&{検索対象}={条件}

検索対象にする項目を「&」で繋いで複数指定することで、AND 条件で絞り込むことができます。
例えば以下のように指定することで、開催場所とイベント種別を検索条件にすることが可能です。

> GET https://eventapp.city.kawasaki.jp/data/api/v1/events?format=JSON&**place=1&type=2**

指定できる項目については後述の「検索対象一覧」をご確認ください。

● 検索対象一覧

検索対象に指定できる項目は以下の通りです。
各項目の仕様については、API リファレンスをご参照ください。

項目名	説明
place	検索対象とする開催場所を ID で指定します
from	to と併せて指定することで、開催日時が from から to の範囲に含まれているイベントを検索します
to	from と併せて指定することで、開催日時が from から to の範囲に含まれているイベントを検索します
type	検索対象とするイベント種別を ID で指定します
target	検索対象とする対象者を ID で指定します
title	イベント名称に指定した文字列を含む、イベント情報を検索します

● HTTP レスポンスの取得

> GET https://eventapp.city.kawasaki.jp/data/api/v1/events?format=JSON&place=1&type=2

以上のようにリクエストした場合、開催場所が「川崎区」、イベント種別が「体感・体験」のイベント情報が JSON 形式で取得できます。

HTTP レスポンスの詳細は API リファレンスをご参照ください。

イベント情報のページ分割

本 API で取得するイベント情報は、100 件単位でページ分割されます。
例えば検索条件に合致したイベント情報が 150 件だった場合、イベント情報は 2 ページに分割されます。
1 回のリクエストでは 1 ページ分（100 件分）のイベント情報しか取得できません。
2 ページ目（残りの 50 件分）のイベント情報を取得する場合は、以下のようにページ番号を指定して再度リクエストを送信してください。

> GET https://eventapp.city.kawasaki.jp/data/api/v1/events?format=JSON&place=1&type=2&**page=2**

Javaサンプル

Java でのサンプルプログラムの抜粋を記載します。

- 取得データ（JSON）の変換にはGoogle製のGsonを使用しています。
- 取得データの変換対象はページ数とタイトルのみとしています。

情報格納クラス

- Events.java（イベント情報格納）

```java
package com.example.kawasakirestsample.pojo;

import java.util.List;

// イベント情報
public class Events {

    // page
    Integer page;

    public Integer getPage() {
        return page;
    }

    public void setPage(Integer page) {
        this.page = page;
    }

    // 必要に応じてフィールドを定義（total_pages、total_numbers）

    // event_data
    List<EventData> eventData;

    public List<EventData> getEventData() {
        return eventData;
    }

    public void setEventData(List<EventData> eventData) {
        this.eventData = eventData;
    }

}
```

- EventData.java（イベントデータ格納）

```java
package com.example.kawasakirestsample.pojo;

// イベントデータ
public class EventData {

    // title
    String title;

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    // 必要に応じてフィールドを定義（content、status、...）

}
```

API呼び出しクラス

- EventApi.java（イベント情報取得実行）

```
package com.example.kawasakirestsample;

import com.example.kawasakirestsample.pojo.Events;
import com.google.gson.FieldNamingPolicy;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

import java.io.BufferedInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.URL;
import java.nio.charset.Charset;

import javax.net.ssl.HttpsURLConnection;

// イベント情報取得API
public class EventApi {

    // イベント情報取得
    Events getEvents() {
        byte[] data = null;
        HttpsURLConnection connection = null;
        BufferedInputStream bufferedInputStream = null;
        ByteArrayOutputStream byteArrayOutputStream = null;
        try {
            // オープンデータAPIを実行
            URL url = new URL("https://eventapp.city.kawasaki.jp/data/api/v1/events?format=JSON&place=1&type=2");
            connection = (HttpsURLConnection) url.openConnection();
            connection.setRequestMethod("GET");
            connection.connect();
            // OK応答の場合はデータを読み込み
            if (connection.getResponseCode() == HttpsURLConnection.HTTP_OK) {
                InputStream inputStream = connection.getInputStream();
                bufferedInputStream = new BufferedInputStream(inputStream);
                byteArrayOutputStream = new ByteArrayOutputStream();
                byte[] buffer = new byte[1024];
                int length;
                while ((length = bufferedInputStream.read(buffer, 0, buffer.length)) != -1) {
                    byteArrayOutputStream.write(buffer, 0, length);
                }
                data = byteArrayOutputStream.toByteArray();
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                if (bufferedInputStream != null) {
                    bufferedInputStream.close();
                }
                if (byteArrayOutputStream != null) {
                    byteArrayOutputStream.close();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
            if (connection != null) {
                connection.disconnect();
            }
        }
        // データをJSONに変換
        String json = new String(data, Charset.forName("UTF-8"));
        // Gsonを使用してJSONを変換
        Gson gson = new GsonBuilder().setFieldNamingPolicy(FieldNamingPolicy.LOWER_CASE_WITH_UNDERSCORES).create();
        Events events = gson.fromJson(json, Events.class);
        return events;
    }

}
```