オープンソース協議会 IBM i

Ver. 2.0 | 2019/01/18

【ハンズオン】

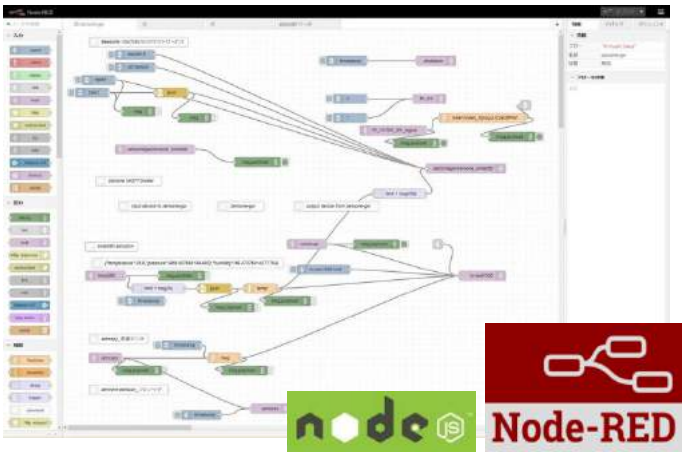
IBM i と連携する Node.js と Node-RED で アプリ構築を体験しよう

ティアンドトラスト株式会社 北原征夫
http://www.tat.co.jp

Node-RED 概説

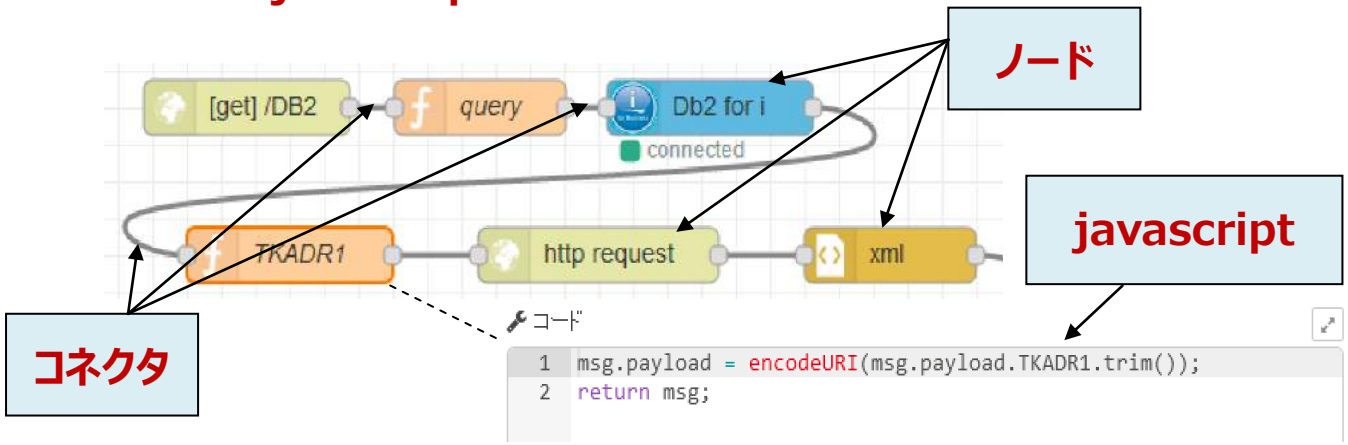
❒Node-RED ?

- ハードウェアデバイス、API、オンラインサービスを**つなげる**ためのプログラミングツール
- IBM 英国ハーズレイ研究所のエマージング・テクノロジー・チームが開発したソフトウェア
- オープンソースで提供されている
- Node.js で作られている
- 開発は「ブラウザ」で行う
- 基本はマウスによる操作
- JavaScript でコーディングも可能



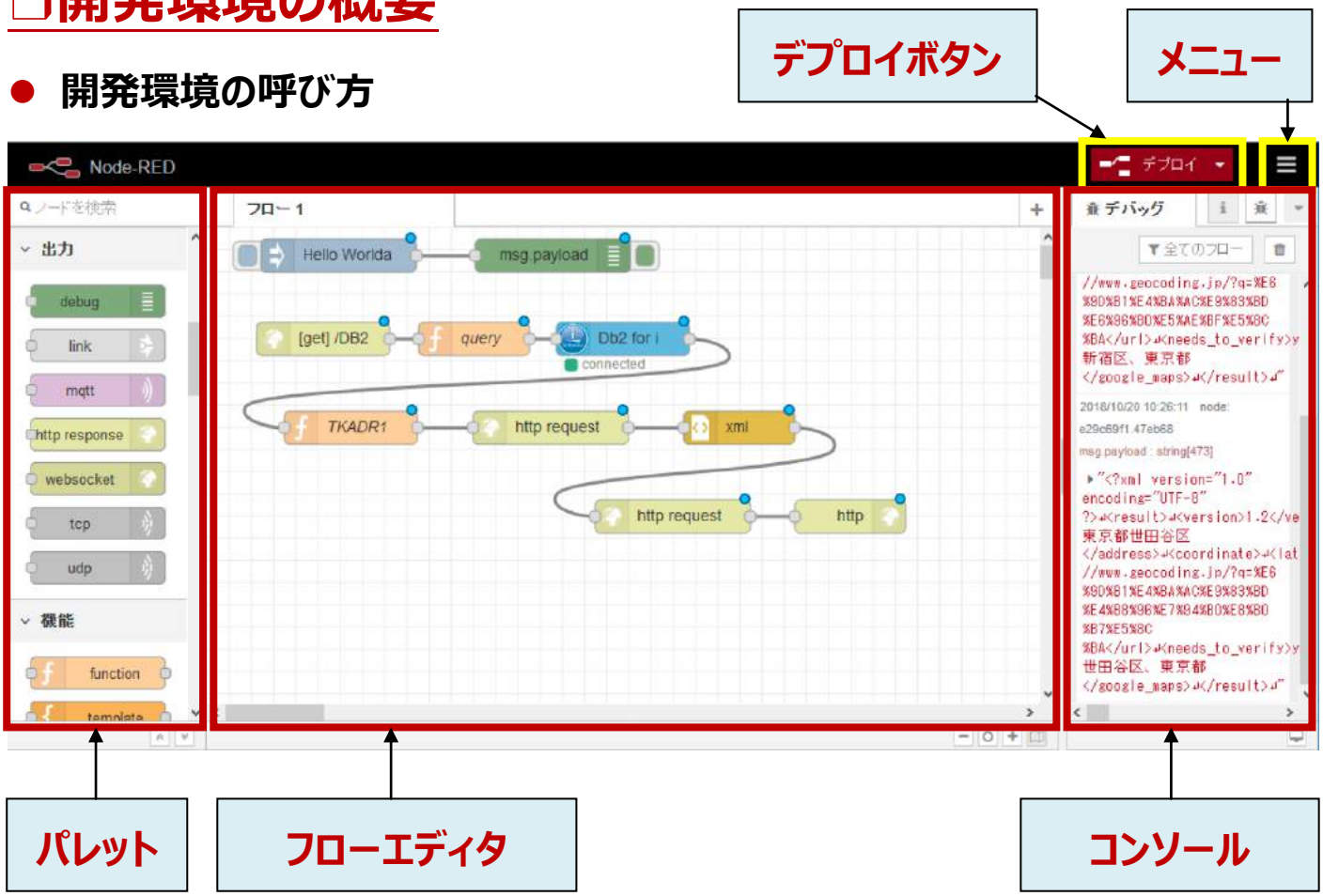
❒開発の概要

- **ノード**と呼ばれる部品をマウス操作でエディタに配置する
- ノードとノードをマウス操作により**コネクタ**で接続する
- 接続したノードからノードへ**メッセージ**を渡し処理を行う
- メッセージには **Json** が利用される
- 細かな処理は **javascript** でコーディングできる



❒開発環境の概要

- 開発環境の呼び方



❒参考：ハンズオンの環境

- IBM i
  - Power7 8202-E4C (EPC5) 4Core
  - CPU: 1 コア割り当て （CAP あり）
  - メモリー：21GB
  - IBM i V7R3
  - PTF : CUMULATIVE Level 17061
- 5733-OPS： オープンソース for IBM i
  - OPT1 から OPT11 まですべて導入済み
  - 今回は OPT10：Node.js V6 の環境を利用

ハンズオンの解説と準備

※このハンズオンで利用する推奨ブラウザは Firefox です。（それ以外のブラウザも利用可能です）

※ IE の場合、以下の設定を事前に行ってください。（設定後 IE の再起動が必要です）

- インターネットオプション
  - 「詳細設定」タブの「インターナショナル」にある以下の 2 項目にチェックを付け適用
    - ✧ イントラネットの URL の UTF-8 クエリ文字を送信する
    - ✧ イントラネット以外の URL の UTF-8 クエリ文字を送信すす

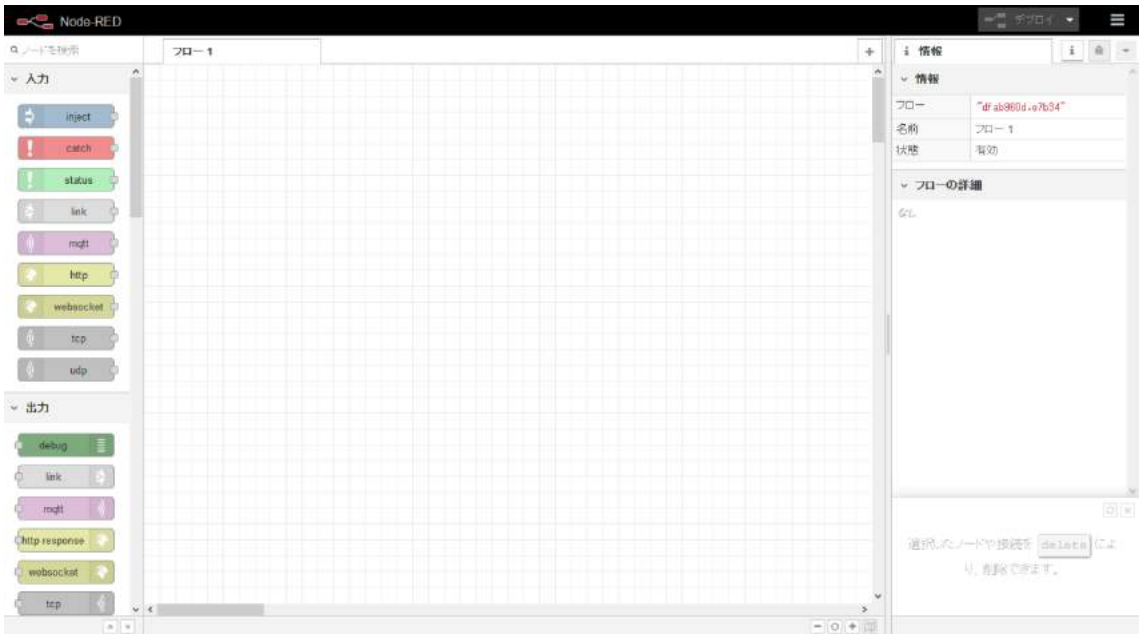
ハンズオンの解説

このハンズオンを通して以下を体験します。

1. Hello World
2. IBM i からのデータ取り出し
3. 外部 API サービスの利用

ハンズオンの準備

1. ブラウザを起動します
2. 以下の URL へアクセスします
   http://60.32.64.174:XXXX/
   ※ XXXX は講師よりお伝えいたします
3. 以下の画面が表示される事を確認します



4. 準備は完了です。「ハンズオン 1 」へ進んでください

ハンズオン１：Hello World

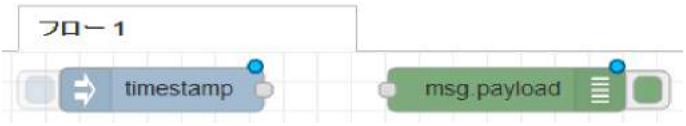
単純な “Hello World” を用い、Node-RED を体験します。

ハンズオンの手順

1. ノードの配置

「パレット」 から以下の 「ノード」 を 「フローエディタ」 にドラッグアンドドロップします

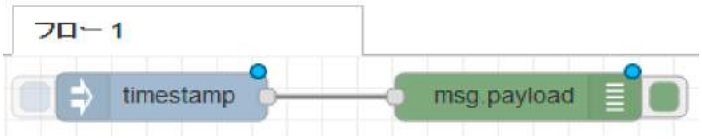
- 「入力」にある「Inject」
- 「出力」にある「debug」



※ドラッグアンドドロップ後、「Inject」 は 「timestamp」 に、「debug」 は 「msg.payload」 に表示が変わります

2. ノードの接続

「timestamp」(Inject) の右端にあると  「msg.payload」(debug) の左端にあるをドラッグアンドドロップし接続し  ます



3. ノードの属性設定画面を開く

「timestamp」(Inject) をダブルクリックし 「Inject ノードを編集」 を表示します

4. ノードの属性を設定する

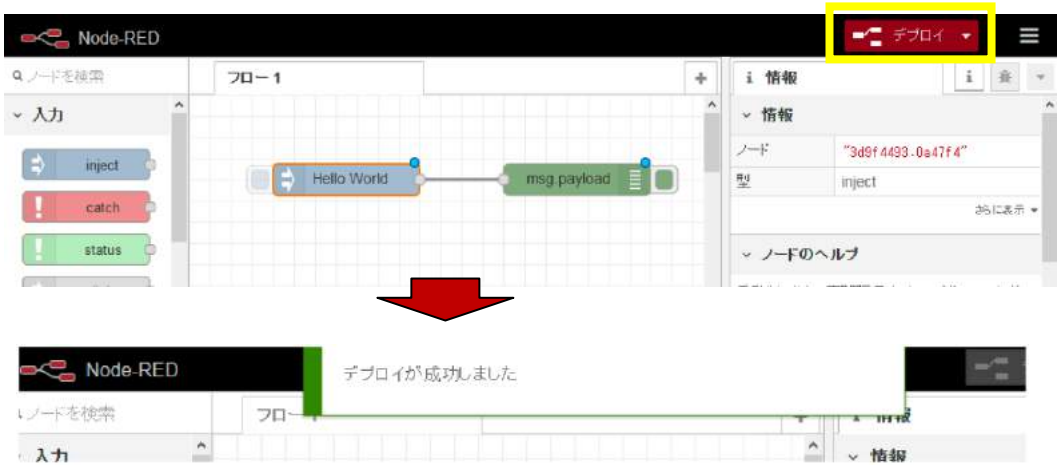
「プロパティ」 にある 「ペイロード」 から 「文字列」 を選択し、”Hello World” と入力し、「完了」 をクリックします

※ Inject の表示が「Hello World」 に変わります

5. デプロイ

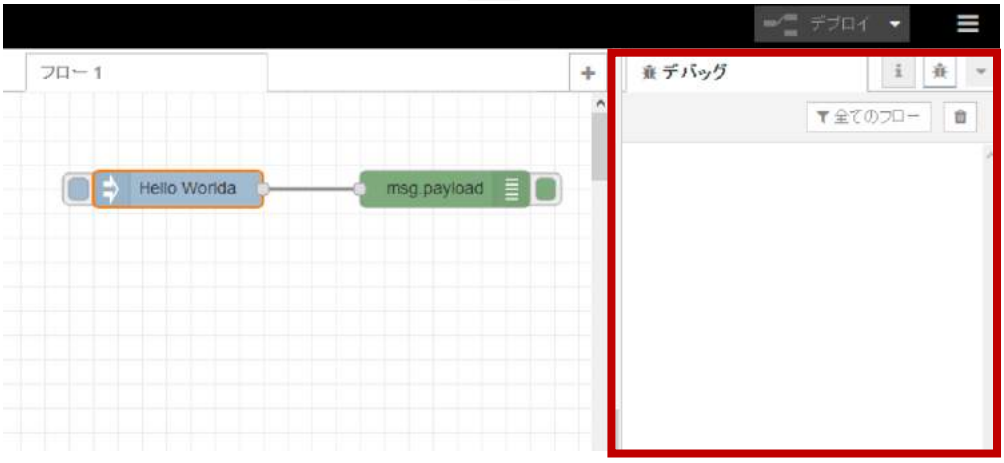
これで Hello World の作成は完了しました。作成したプログラムをサーバーに配置するため画面右上にある「デプロイ」をクリックします

成功すると 「デプロイが成功しました」 とメッセージが表示されます



6. コンソールを開く

画面右上にある 「デバッグ」 アイコン  をクリックし、コンソールを開きます



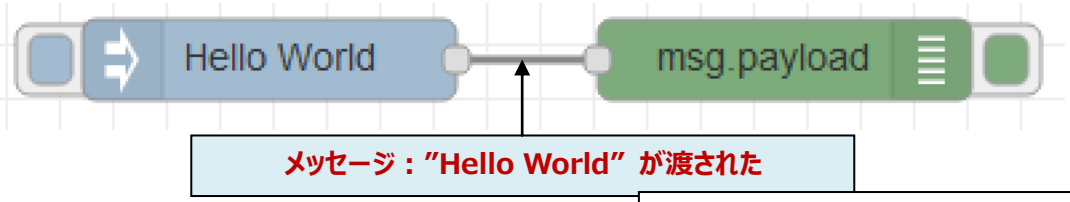
7. 実行

「Hello World」(Inject) の左端にある  をクリックします

作成した Node-RED プログラムが実行され結果がコンソールに表示されます

この様に Node-RED ではマウス操作により、以下を行う事ができます

- 必要な処理に合った「ノードの配置」
- 配置した「ノード間の接続」
- 「ノード属性」の設定
- ノード間で「メッセージ」を渡し、結果（msg.payload）を出力



メッセージ：”Hello World” が渡された

メッセージは以下の様な JSON オブジェクト

```
{
  _msgid: "7d73eb9e.500fd4",
  topic: "",
  payload: "Hello World"
}
```

以上で「ハンズオン１：Hello World」は完了となります

ハンズオン２ ：IBM i からデータを取り出してみる

ハンズオン２ では、IBM i のデータベースファイルからデータを取得しブラウザに表示します。
前回の様な「Inject」からの実行ではなく、ブラウザから URL で Node-RED プログラムを起動し結果をブラウザに表示します

ハンズオンの準備：DB2 for IBM i のノードを Node-RED に追加する

Node-RED の初期パレットには、DB2 for i のデータを参照するための「ノード」がありません。
DB2 for i のデータを参照するためには以下の手順によりノードを追加する必要があります。

追加するノードは以下となります

- Node-red-contrib-db2-for-i

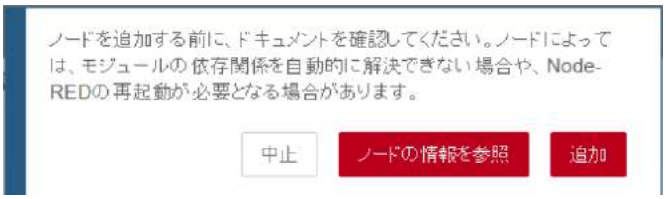
ノードの追加手順

1. 右上のハンバーガーメニュー  をクリックしメニューを表示します

2. 表示されたメニューの 「パレットの管理」 をクリックし 「ユーザー設定」 を開きます

3. 「ユーザー設定」 にある 「ノードを追加」 タブを開き 「ノードを検索」 に “node-red-contrib-db2-for-i” を入力し表示された 「node-red-contrib-db2-for-i」 の 「ノードを追加」 をクリックします

4. 確認が表示されるので 「追加」 をクリックします



5. ノードが追加されたメッセージが表示されます。表示が消えたら 「閉じる」 をクリックし 「ユーザー設定」 を終わらせます

6. 左の 「パレット」 にある 「ストレージ」 に 「DB2 for i」 ノードが追加されます

以上で「ハンズオン 2 」を行う準備が整いました

ハンズオンの手順

1. ノードの配置

「パレット」 から以下の 「ノード」 を 「フローエディタ」 にドラッグアンドドロップします

- 「入力」にある「http」
- 「機能」にある「function」
- 「ストレージ」にある「DB2 for i」
- 「機能」にある「json」
- 「出力」にある「http response」

2. ノードの接続

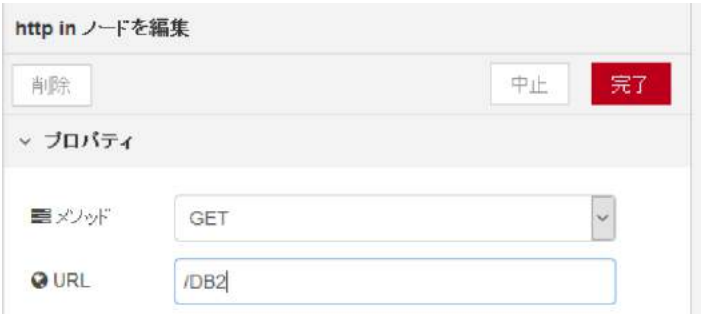
以下の順番でノードを接続します

1. 「http」
2. 「function」（フローエディタ上は 空白）
3. 「DB2 for i」
4. 「json」
5. 「http response」（フローエディタ上は http）

3. http ノードの設定

「http」ノードをダブルクリックし 「http in ノードを編集」 を開きます。以下を設定し 「完了」 をクリックします

- メソッド　：GET
- URL　　：/DB2



4. function ノードの設定

「function」ノードをダブルクリックし 「function ノードを編集」 を開きます。以下を設定し 「完了」 をクリックします

- 名前　　：query
- コード　　：以下を入力します

```
var sql = "select * from QEOL.TOKMSP where TKNAKJ like '%" + msg.payload.tknakj + "%'";

msg.payload = sql;

return msg;
```

5. DB2 for i ノードの設定

「DB2 for i」ノードをダブルクリックし「DB2 for i ノードを編集」を開きます。

次に 「Database」の右にあるボタンをクリックし「新規に DB2 for i Config ノードの設定を追加」 を開きます。

「新規に DB2 for i Config ノードの設定を追加」 の以下を設定し「追加」をクリックします。

- Connection Name : OSS
- User : NODERED
- Password : NODERED
- Database : *LOCAL

「DB2 for i ノードを編集」に戻るので、以下を設定し 「完了」をクリックします

- Single Array Result mode： チェック

6. json ノードの設定

「json」ノードをダブルクリックし 「json ノードを編集」 を開きます。以下を設定し 「完了」 をクリックします

- 動作 ： JSON 文字列とオブジェクト間の相互変換
- プロパティ ： payload
- オブジェクトから JSON へ変換 ： JSON 文字列フォーマットにチェック

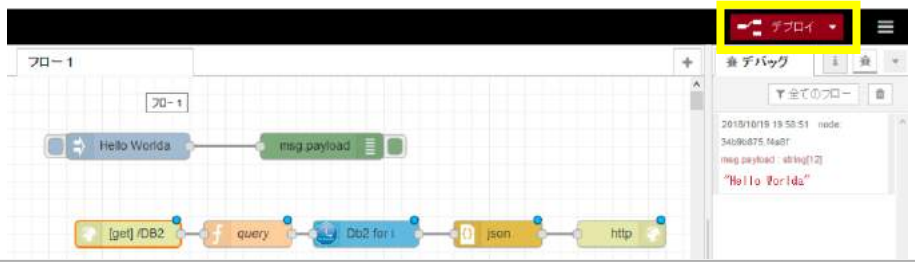
7. http response ノードの設定

設定は不要です

8. デプロイ

これでハンズオン 2 の作成は完了しました。作成したプログラムをサーバーに配置するため画面右上にある 「デプロイ」 をクリックします

成功すると 「デプロイが成功しました」 とメッセージが表示されます



9. 実行

ブラウザで別のタブを開き、以下の URL を入力し実行します

```
http://60.32.64.174:XXXX/DB2?tknakj=相川
```

ブラウザに以下の様に DB2 のデータが表示されます

[{ "TKBANG": "01030", "TKNAKN": "ｱｲｶﾜ ｺｳｷﾞﾖｳ ", "TKNAKJ": "相川工業 ", "TKADR1": "東京都世田谷区 ", "TKADR2": "若林
４－２４ ", "TKTIKU": "06", "TKPOST": "154 ", "TKTELE": "03-964-6406 ", "TKGURI": "136200", "TKNURI": "243000", "TKZURI":
"796600", "TKUZAN": "110000", "TKGEND": "1120000", "TKNYUK": "880619", "TKSIME": "1" }, { "TKBANG": "01070", "TKNAKN": "ｱｲｶﾜ ｶﾒﾗ
", "TKNAKJ": "相川カメラ ", "TKADR1": "東京都新宿区 ", "TKADR2": "四谷１－１６ ", "TKTIKU": "03", "TKPOST":
"160 ", "TKTELE": "03-354-3018 ", "TKGURI": "367900", "TKNURI": "2011700", "TKZURI": "3914500", "TKUZAN": "960000", "TKGEND":
"1000000", "TKNYUK": "880611", "TKSIME": "1" }, { "TKBANG": "01080", "TKNAKN": "ｱｲｶﾜ ｺｳｸｳK.K ", "TKNAKJ": "相川広告Ｋ．Ｋ ",
"TKADR1": "東京都渋谷区 ", "TKADR2": "広尾３－９ ", "TKTIKU": "02", "TKPOST": "150 ", "TKTELE": "03-368-6366 ",
"TKGURI": "318000", "TKNURI": "1461700", "TKZURI": "2205800", "TKUZAN": "290000", "TKGEND": "2000000", "TKNYUK": "880214",
"TKSIME": "2" }, { "TKBANG": "01090", "TKNAKN": "ｱｲｶﾜ ﾃﾞﾝｷK.K ", "TKNAKJ": "相川電機Ｋ．Ｋ ", "TKADR1": "東京都北区 ",
"TKADR2": "滝の川７－１７ ", "TKTIKU": "14", "TKPOST": "114 ", "TKTELE": "03-749-6271 ", "TKGURI": "877100", "TKNURI":

参考）

※ Google Chrome に拡張機能 「JSONView」 を導入すると以下の様に整形して表示されます

[
  - {
      TKBANG: "01030",
      TKNAKN: "ｱｲｶﾜ ｺｳｷﾞﾖｳ ",
      TKNAKJ: "相川工業 ",
      TKADR1: "東京都世田谷区 ",
      TKADR2: "若林４－２４ ",
      TKTIKU: "06",
      TKPOST: "154 ",
      TKTELE: "03-964-6406 ",
      TKGURI: "136200",
      TKNURI: "243000",
      TKZURI: "796600",
      TKUZAN: "110000",
      TKGEND: "1120000",
      TKNYUK: "880619",
      TKSIME: "1"
    },
  - {
      TKBANG: "01070",
      TKNAKN: "ｱｲｶﾜ ｶﾒﾗ ",

この様に Node-RED ではマウス操作により、以下を行う事ができます

- ブラウザからのリクエストを受け入れ、パラメータを取得できる
- パラメータを利用してデータベースのレコードを照会できる
- 照会結果のレコードを JSON などに整形しブラウザへ結果を返す事ができる

以上で「ハンズオン２ ：IBM i からデータを取り出してみる」は完了となります

ハンズオン３：外部の API サービスを利用してみる

ハンズオン３では、ハンズオン２で取り出した IBM i のデータ（住所）を利用して、外部の API サービスを利用します

外部 API サービスは以下を利用します

1. 住所情報から緯度/経度を入手する

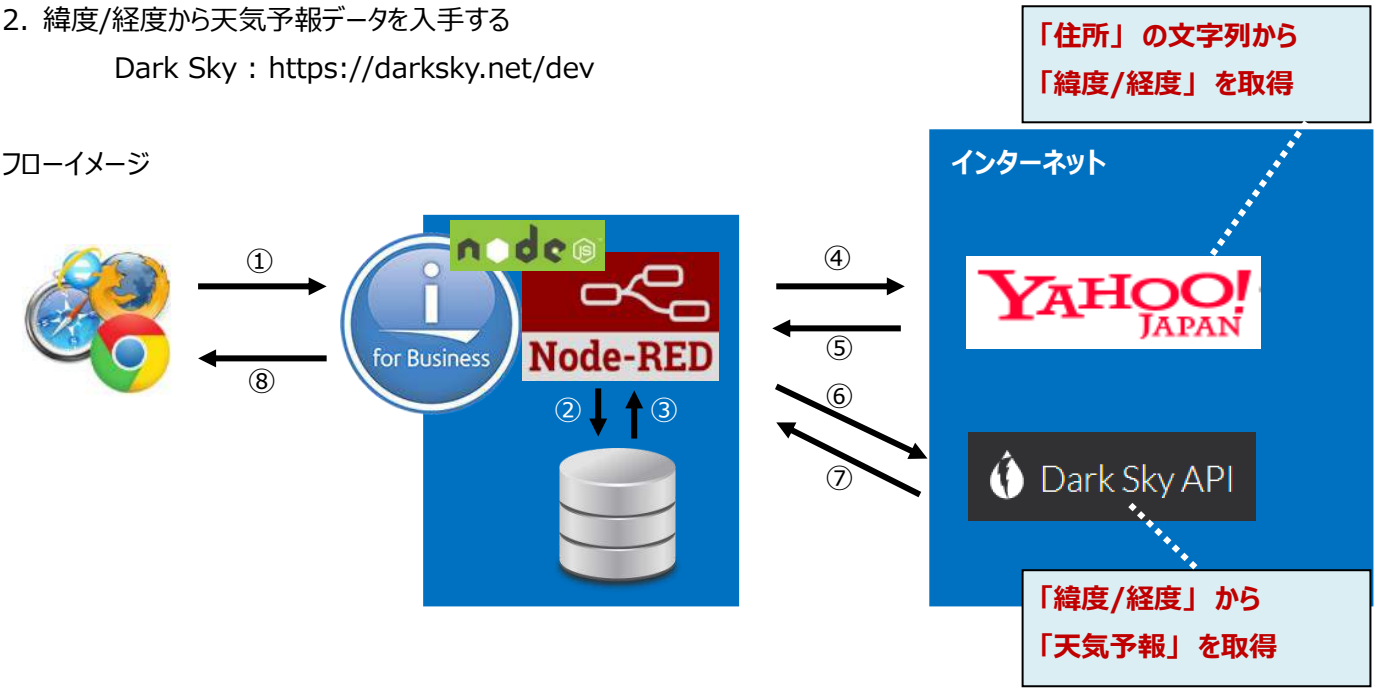
   Yahoo!ジオコーダ API：

   https://developer.yahoo.co.jp/webapi/map/openlocalplatform/v1/geocoder.html

2. 緯度/経度から天気予報データを入手する

   Dark Sky：https://darksky.net/dev

フローイメージ



ハンズオンの準備：http response への接続を除去する

ハンズオンを行う前に先ほど設定した「json」ノード →「http response」ノード は「ハンズオン 3」では不要なので、ノードと接続を除去します。

接続の除去手順

1. json ノードを選択し「delete」キーを打鍵します。json ノードが削除されます
2. http response ノードを選択し「delete」キーを打鍵します。http response ノードが削除されます
3. 削除後は以下の様になります

以上で「ハンズオン３」を行う準備が整いました

ハンズオンの手順

1. ノードの配置

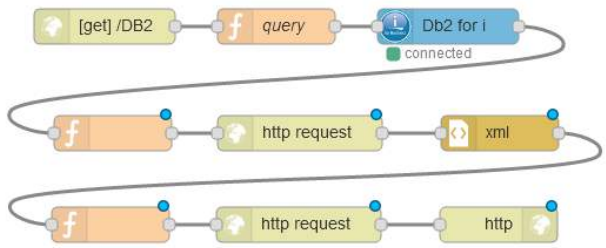
「パレット」 から以下の 「ノード」 を 「フローエディタ」 にドラッグアンドドロップします

- 「機能」にある「function」
- 「機能」にある「http request」
- 「機能」にある「xml」
- 「機能」にある「function」
- 「機能」にある「http request」
- 「出力」にある「http response」

2. ノードの接続

以下の順番でノードを接続します

1. 「DB2 for i」から「function」（フローエディタ上は 空白）
2. 「http request」
3. 「xml」
4. 「function」（フローエディタ上は 空白）
5. 「http request」
6. 「http response」



3. query (function) ノードの設定変更

ハンズオン２で作成した「query (function)」ノードをダブルクリックし 「function ノードを編集」 を開きます。以下を変更し 「完了」 をクリックします

- コード　：既存のコードを以下の様に変更します **（※※※ 必ず修正してください ※※※）**

```
var sql = "select * from QEOL.TOKMSP where TKNAKJ like '%" + msg.payload.tknakj + "%' limit 1";

msg.payload = sql;

return msg;
```

4. DB2 for i ノードの設定変更

ハンズオン２で作成した「DB2 for i」ノードをダブルクリックし「DB2 for i ノードを編集」を開きます。以下を設定し「完了」をクリックします

- Single Array Result mode： チェックを外す

5. function ノードの設定

「function」ノードをダブルクリックし「function ノードを編集」を開きます。以下を設定し「完了」をクリックします

- 名前　　：TKADR1
- コード　　：以下を入力します

```
msg.payload = encodeURI(msg.payload.TKADR1.trim());

return msg;
```

6. http request ノードの設定

「http request」ノードをダブルクリックし「http request ノードを編集」を開きます。以下を設定し「完了」をクリックします

- メソッド　　：GET
- URL　　：http://60.32.64.174:10090/yhapi.php?query={{payload}}
- 出力文字形式　　：文字列

7. xml ノードの設定

設定は不要です

8. function ノードの設定

「function」ノードをダブルクリックし 「function ノードを編集」 を開きます。以下を設定し 「完了」 をクリックします

- 名前　　：　Coordinates
- コード　　：　以下を入力します

```
var coordinates = (msg.payload.YDF.Feature[0].Geometry[0].Coordinates[0]).split(',');

msg.payload.latitude = coordinates[1];

msg.payload.longitude = coordinates[0];

return msg;
```

9. http request ノードの設定

「http request」ノードをダブルクリックし 「http request ノードを編集」 を開きます。以下を設定し 「完了」 をクリックします

- メソッド　　：　GET
- URL　　：　下記は 1 行です。

```
https://api.darksky.net/forecast/80ed6afe4867f226ca0b341d2edabb2a/{{payload.latitude}},{{payload.longitude}}
```

- 出力形式　　：　JSON

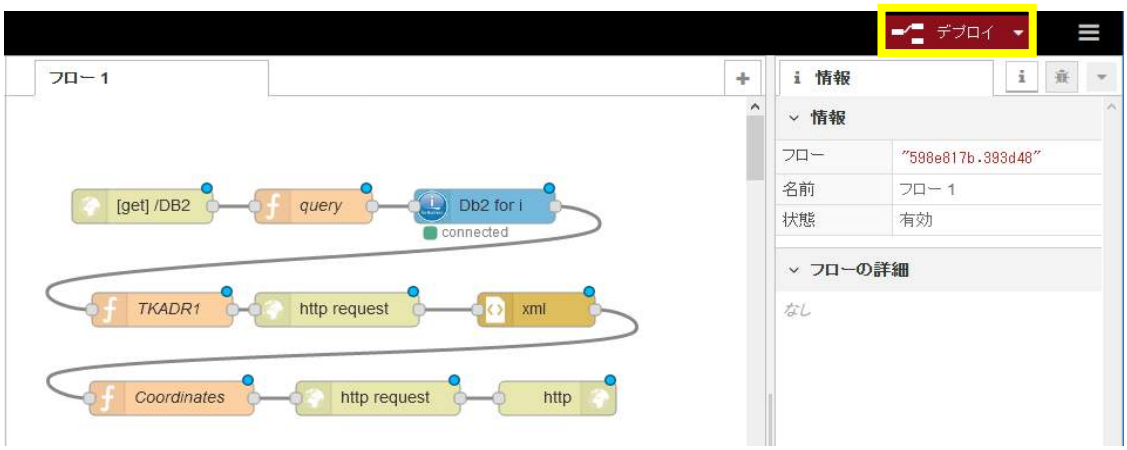
8. http response ノードの設定

設定は不要です

10. デプロイ

これでハンズオン 3 の作成は完了しました。作成したプログラムをサーバーに配置するため画面右上にある 「デプロイ」 をクリックします

成功すると「デプロイが成功しました」とメッセージが表示されます



11. 実行

ブラウザで別のタブを開き、以下の URL を入力し実行します

※※※　注意：実行前に講師が確認します。講師にお声がけください。　※※※

```
http://60.32.64.174:XXXX/DB2?tknakj=相川工業
```

ブラウザに以下の様に「天気予報」のデータが表示されます

{"latitude":35.646572,"longitude":139.653247,"timezone":"Asia/Tokyo","currently":{"time":1539996070,"summary":"Mostly Cloudy","icon":"partly-cloudy-day","precipIntensity":0.004,"precipProbability":0.2,"precipType":"rain","temperature":63.39,"apparentTemperature":63.39,"dewPoint":53.28,"humidity":0.7,"pressure":1018,"windSpeed":10.2,"windGust":14.33,"windBearing":293,"cloudCover":0.73,"uvIndex":3,"visibility":9.3,"ozone":286.4},"hourly":{"summary":"Partly cloudy throughout the day.","icon":"partly-cloudy-day","data":[{"time":1539993600,"summary":"Mostly Cloudy","icon":"partly-cloudy-day","precipIntensity":0.004,"precipProbability":0.21,"precipType":"rain","temperature":63.04,"apparentTemperature":63.04,"dewPoint":53.48,"humidity":0.71,"pressure":1018.23,"windSpeed":10.32,"windGust":14.5,"windBearing":339,"cloudCover":0.62,"uvIndex":2,"visibility":8.39,"ozone":286.25},{"time":1539997200,"summary":"Mostly Cloudy","icon":"partly-cloudy-day","precipIntensity":0.004,"precipProbability":0.2,"precipType":"rain","temperature":63.55,"apparentTemperature":63.55,"dewPoint":53.18,"humidity":0.69,"pressure":1017.89,"windSpeed":10.16,"windGust":14.24,"windBearing":272,"cloudCover":0.77,"uvIndex":3,"visibility":9.72,"ozone":286.48},{"time":1540000800,"summary":"Mostly Cloudy","icon":"partly-cloudy-day","precipIntensity":0.0048,"precipProbability":0.19,"precipType":"rain","temperature":64.56,"apparentTemperature":64.56,"dewPoint":52.76,"humidity":0.66,"pressure":1017.43,"windSpeed":9.88,"windGust":13.85,"windBearing":188,"cloudCover":0.92,"uvIndex":4,"visibility":10,"ozone":286.58},{"time":1540004400,"summary":"Overcast","icon":"cloudy","precipIntensity":0.0048,"precipProbability":0.19,"precipType":"rain","temperature":65.83,"apparentTemperature":65.83,"dewPoint":52.27,"humidity":0.62,"pressure":1017.04,"windSpeed":9.62,"windGust":13.54,"windBearing":118,"cloudCover":0.96,"uvIndex":3,"visibility":10,"ozone":287.16},{"time":1540008000,"summary":"Mostly Cloudy","icon":"partly-cloudy-day","precipIntensity":0.0022,"precipProbability":0.11,"precipType":"rain","temperature":67.48,"apparentTemperature":67.48,"dewPoint":51.47,"humidity":0.57,"pressure":1018.76,"windSpeed":9.19,"windGust":13.4,"windBearing":73,"cloudCover":0.89,"uvIndex":3,"visibility":10,"ozone":288.27}

参考）

※ Google Chrome に拡張機能 「JSONView」 を導入すると以下の様に整形して表示されます

```
{
    latitude: 35.646572,
    longitude: 139.653247,
    timezone: "Asia/Tokyo",
  - currently: {
        time: 1539997571,
        summary: "Mostly Cloudy",
        icon: "partly-cloudy-day",
        precipIntensity: 0.004,
        precipProbability: 0.2,
        precipType: "rain",
        temperature: 64.08,
        apparentTemperature: 64.08,
        dewPoint: 52.82,
        humidity: 0.67,
        pressure: 1017.84,
        windSpeed: 9.85,
        windGust: 14.2,
        windBearing: 235,
        cloudCover: 0.79,
        uvIndex: 3,
        visibility: 7.56,
        ozone: 286.49
    },
  - hourly: {
        summary: "Partly cloudy throughout the day.",
        icon: "partly-cloudy-day",
      - data: [
          - {
                time: 1539997200,
                summary: "Mostly Cloudy",
                icon: "partly-cloudy-day",
```

付録：Node-RED の IBM i へのインストール

準備 1 ：5733-OPS オープンソース for IBM i のインストール

1. 前提ライセンスの確認（すべて無償ライセンス）

- 5770-SS1 オプション 33： PORTABLE APP SOLUTIONS ENVIRONMENT
- 5733-SC1 オプション 1： IBM Portable Utilities for i
- 5770-DG1 *BASE：IBM HTTP SERVER FOR I

2. 5733-OPS の入手（※ダウンロードで入手する場合の手順です）

1. ESD へアクセス
2. https://www-304.ibm.com/servers/eserver/ess/ProtectedServlet.wss
3. Firefox（または Java をサポートするブラウザ）で開く
4. 左メニューの My Entitled Software>ソフトウェアのダウンロードを開く
5. オペレーティングシステムとバージョンを選択する
6. 5770-SS1 を選択して、「続行」をクリックする
7. 言語を選択して「続行」をクリックする
8. hide / show リンクを使用して、5770-SS1 および 5817：i7.3 B_GROUP1 v07.03.00,ENU,DVD を開く（V7R3 の場合）
9. ダウンロード方法を選択し、ダウンロードする
10. 拡張子 .udf がダウンロードできる

3. インストールの準備（仮想光メディア装置の準備：一例です）

1. IBM i にディレクトリ作成(5250) : MKDIR DIR('/5733OPS')
2. イメージ・カタログの作成(5250) : CRTIMGCLG IMGCLG(OPNSRC) DIR('/5733OPS')
3. ダウンロードした udf ファイルを FTP で IBM i に送る（windows）
4. 仮想光学ドライブの作成(5250) :
   CRTDEVOPT DEVD(OPTVRT01) RSRCNAME(*VRT) LCLINTNETA(*N)
5. 仮想光学ドライブの開始(5250) :
   VRYCFG CFGOBJ(OPTVRT01) CFGTYPE(*DEV) STATUS(*ON)
6. イメージカタログに UDF ファイルを追加(5250) :
   ADDIMGCLGE IMGCLG(OPNSRC) FROMFILE('5733OPS.UDF') TOFILE(*FROMFILE)
7. イメージカタログを仮想光学ドライブにロードする(5250)
   LODIMGCLG IMGCLG(OPNSRC) DEV(OPTVRT01)

4. 5733-OPS　のインストール

1. ライセンスプログラムの復元(5250)

   RSTLICPGM LICPGM(5733OPS) DEV(OPTVRT01) OPTION(*BASE)
   RSTLICPGM LICPGM(5733OPS) DEV(OPTVRT01) OPTION(1)
   RSTLICPGM LICPGM(5733OPS) DEV(OPTVRT01) OPTION(2)
   RSTLICPGM LICPGM(5733OPS) DEV(OPTVRT01) OPTION(3)
   RSTLICPGM LICPGM(5733OPS) DEV(OPTVRT01) OPTION(4)
   RSTLICPGM LICPGM(5733OPS) DEV(OPTVRT01) OPTION(5)
   RSTLICPGM LICPGM(5733OPS) DEV(OPTVRT01) OPTION(6)
   RSTLICPGM LICPGM(5733OPS) DEV(OPTVRT01) OPTION(7)
   RSTLICPGM LICPGM(5733OPS) DEV(OPTVRT01) OPTION(8)
   RSTLICPGM LICPGM(5733OPS) DEV(OPTVRT01) OPTION(9)
   RSTLICPGM LICPGM(5733OPS) DEV(OPTVRT01) OPTION(10)
   RSTLICPGM LICPGM(5733OPS) DEV(OPTVRT01) OPTION(11)
   RSTLICPGM LICPGM(5733OPS) DEV(OPTVRT01) OPTION(12)
   RSTLICPGM LICPGM(5733OPS) DEV(OPTVRT01) OPTION(13)
   RSTLICPGM LICPGM(5733OPS) DEV(OPTVRT01) OPTION(14)
   RSTLICPGM LICPGM(5733OPS) DEV(OPTVRT01) OPTION(15)
   ※ Node.js V6 は オプション 10
   ※ 各オプションの対応表は以下を参照
   https://www.ibm.com/developerworks/community/wikis/home?lang=en_us#!/wiki/IBM%20i%20Technology%20Updates/page/Open%20Source%20Technologies

2. 5733-OPS グループ PTF の適用（5733-OPS の実体はこの PTF により導入されます）
   - V7R3 : SF99225
   - V7R2 : SF99223

Node-RED のインストールと起動確認

1. Node.js V6 のインストール確認

1. Qshell または QP2TERM でシェルを開く
   CALL QSH または CALL QP2TERM
2. Node.js V6 のバイナリーにディレクトリを変更
   cd /QOpenSys/QIBM/ProdData/Node6/bin
3. node.js のバージョン確認（バージョンが表示されれば正しくインストールされている）
   node -v
4. nmp のバージョン確認（バージョンが表示されれば正しくインストールされている）
   ※ nmp は node package manager の略、後程この機能を利用して Node-RED をインストールする
   nmp -v

2. Node-RED のインストール

1. Qshell または QP2TERM でシェルを開く（すでに開いている場合は不要です）
   CALL QSH または CALL QP2TERM
2. Node.js V6 のバイナリーにディレクトリを変更
   cd /QOpenSys/QIBM/ProdData/Node6/bin
3. nmp で Node-RED をインストール
   npm install -g node-red

3. Node-RED の起動

1. Qshell または QP2TERM でシェルを開く（すでに開いている場合は不要です）
   CALL QSH または CALL QP2TERM
2. Node.js V6 のバイナリーにディレクトリを変更
   cd /QOpenSys/QIBM/ProdData/Node6/bin
3. node-red コマンドの実行
   ./node-red
4. 以下の様なログが表示されれば起動完了 （最後の “Started flows” を確認）

```
Welcome to Node-RED
===================

30 Sep 05:00:05 - [info] Node-RED version: v0.19.4
30 Sep 05:00:05 - [info] Node.js  version: v6.9.1
30 Sep 05:00:05 - [info] OS400 3 ppc BE
30 Sep 05:00:06 - [info] Loading palette nodes
30 Sep 05:00:07 - [warn] rpi-gpio : Raspberry Pi specific node set inactive
30 Sep 05:00:08 - [info] Settings file  : /home/n6r010/.node-red/settings.js
30 Sep 05:00:08 - [info] Context store  : 'default' [module=memory]
30 Sep 05:00:08 - [info] User directory : /home/n6r010/.node-red
30 Sep 05:00:08 - [warn] Projects disabled : editorTheme.projects.enabled=false
30 Sep 05:00:08 - [info] Flows file     : /home/n6r010/.node-red/flows_OSSTAT.TAT.CO.JP.json
30 Sep 05:00:08 - [info] Creating new flow file
30 Sep 05:00:08 - [warn]

---------------------------------------------------------------------
Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
---------------------------------------------------------------------

30 Sep 05:00:08 - [info] Server now running at http://127.0.0.1:1880/
30 Sep 05:00:08 - [info] Starting flows
30 Sep 05:00:08 - [info] Started flows
```

5. ブラウザから以下の URL でアクセス
   http://<IBM i の IP アドレス>:1880/
6. ハンズオンの準備時と同じ画面が表示されれば完了