2016年度OSS協議会 第3回勉強会

2016年12月9日

株式会社オムニサイエンス　田中　昌宏

IBMiのNode.jsでAPIをつくるにあたって・・・

やるなら・・・

- Restfulな設計で作成してみる
- Toolkitを使ってRPGと連携してみる

その前に・・・Restとは（概要）

Representation State Transfer(REST)

- HTTP仕様に準拠したWebサービス

  RESTの原則に従って実装されている
  システムのことを、RESTfulと呼ぶ

- シンプルである

  「何を」「どうする」の簡単なインターフェース

Restの原則　～その１

URI（URL）の設計

- 一目でAPIと判る
- APIのバージョンを含める
- 複数形の名詞のみで構成

http://example.com/api/v1/customers

Restの原則　～その２

インターフェース(HTTPメソッドの利用)の統一

- RESTで用いられるHTTPメソッドは
  下記のようにデータのCRUD操作と対応付けられる

処理	HTTPメソッド	CRUD操作
登録	POST	CREATE
取得	GET	READ
更新	PUT	UPDATE
削除	DELETE	DELETE

Restの原則　～その３

ステートレスである

- ステート（状態）レス（無い）とは

  クライアント側とサーバー側で情報を共有しない
  常に同じ振る舞いとなる。

  →スケーラビリティに優れる
  　負荷を推測しやすい

Restの原則　～その４

処理結果をHTTPステータスコードで通知

- サーバはクライアントのリクエストに対するレスポンスに以下の適切なレスポンスコードを返さなければならない

コード	状態	説明
200	OK	リクエストが正常に処理された
201	Created	リクエストが正常に処理され、新規リソースが作成された
204	No Content	リクエストが正常に処理されたが、返す新規情報はない
400	Bad Request	サーバーが理解できない無効な要求である
401	Unauthorized	要求されたリソースには認証が必要である
403	Forbidden	要求されたリクエストは拒否された
404	Not Found	要求されたリソースはサーバーに存在しない
500	Internal Server Error	サーバーでエラーが発生した

Restの原則　～その５

JSONまたはXMLをレスポンス

- 標準的なデータフォーマットで連携が容易

RestとSoapの違い

- Rest
  - ✓ ネットワーク上のコンテンツ(リソース)を一意なURLで表すのが基本
  - ✓ 各リソース（URI）に対してGET,POST,PUT,DELETEで
    リクエストを送信しレスポンスをXMLやjsonなどで受け取る形式
  - ✓ URLがリソースに対応づけられるためURLは名詞的になることが多い
- Soap
  - ✓ SoapはSimpleObject Access Protocolの略
  - ✓ リクエストおよびレスポンスともにXMLフォーマットのデータで行う形式
  - ✓ SOAPのURLは操作と対応づけられるため、
    URLの命名が動詞的になることが多い

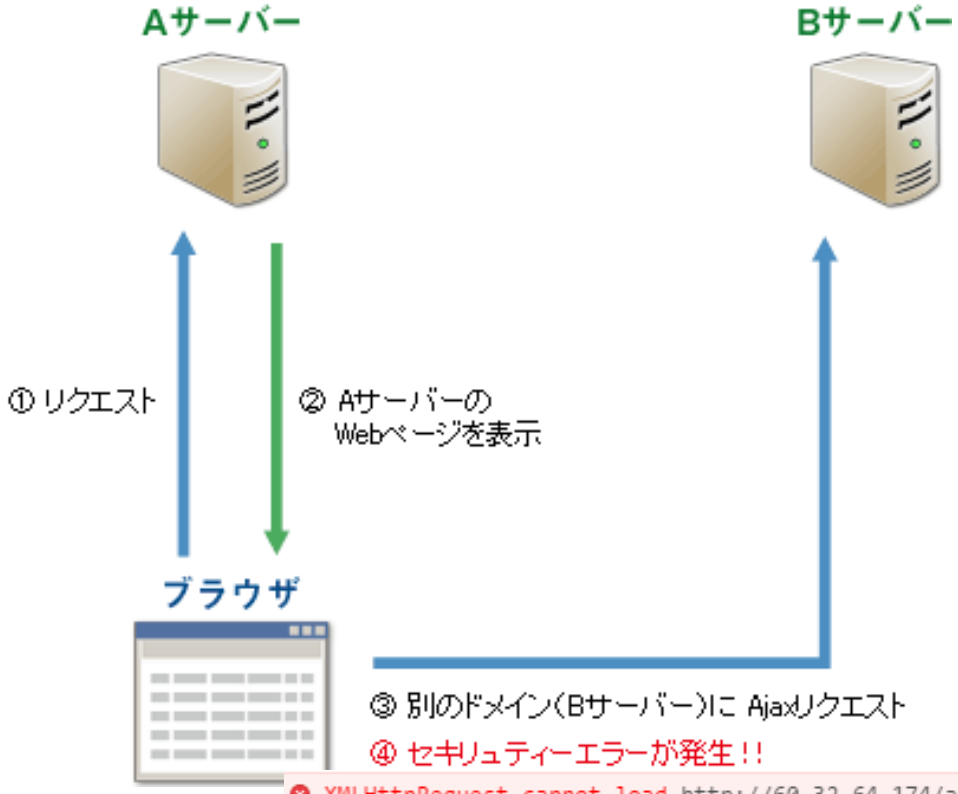
RestとSoapの棲み分け

- Rest
  - ✓ 不特定多数を対象にした、入力パラメータが少ない情報配信や検索サービス等での利用に向いている

- Soap
  - ✓ 複雑な入力を必要としたり、入出力に対してチェックを必要とするようなサービス等での利用に向いている

クロスドメインの制約について

- クロスドメインの制約
  →別ドメインのサーバーにAjax(XMLHttpRequest)
  リクエストすることは不可（ブラウザでエラー）




```
XMLHttpRequest cannot load http://60.32.64.174/api/v1/customers/. No 'Access-Control-Allow-Origin' header is
present on the requested resource. Origin 'http://192.168.1.7:10080' is therefore not allowed access.
```

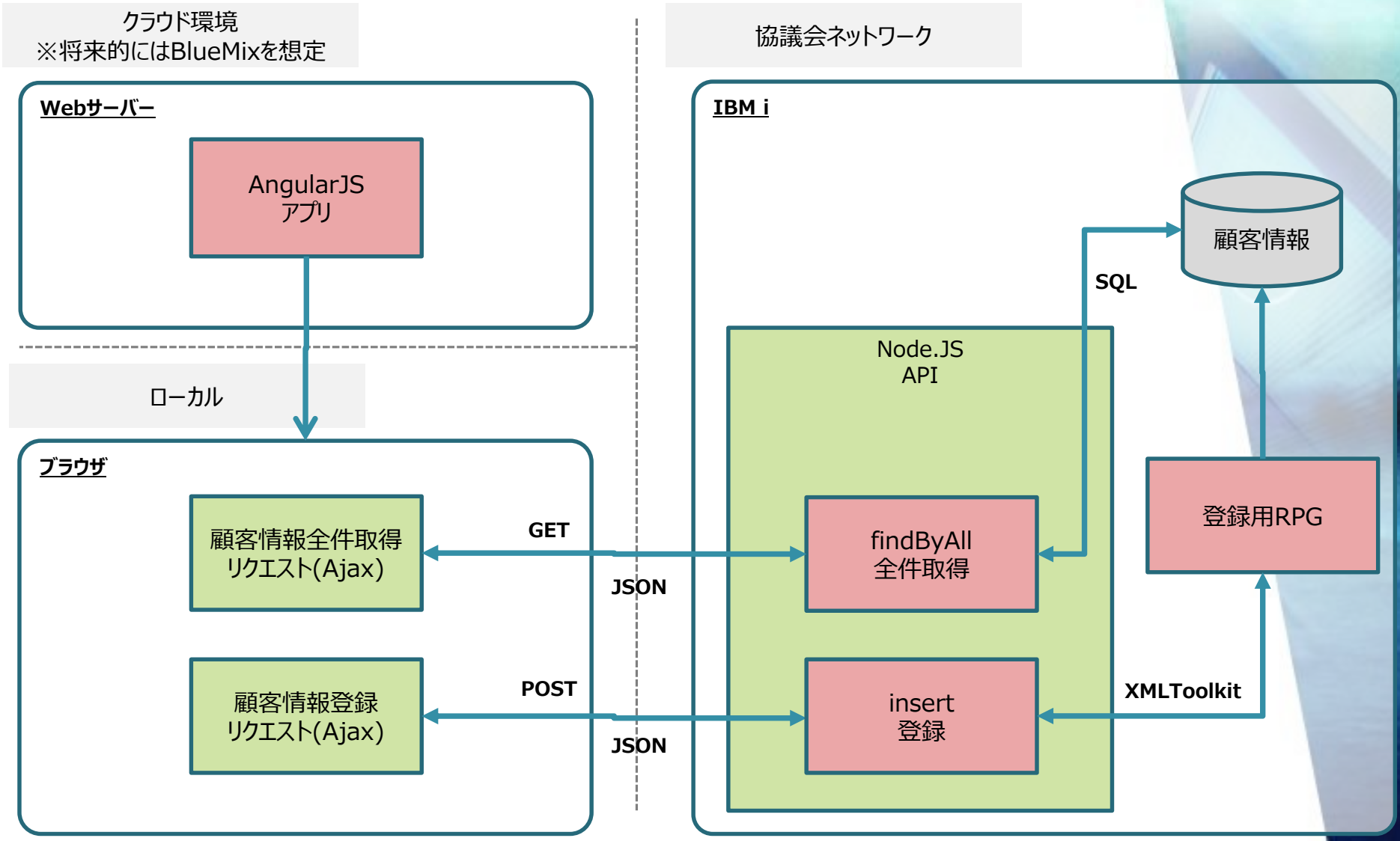
クロスドメインの制約について

どうするか・・・

・クライアント側（ブラウザ）で対応
　→JSONP（JSON With Padding）を利用

・サーバー側で対応
　→CORS（Cross-Origin-Resource-Sharing）
　　を許可

今回作ったものの紹介　～仕様編



今回作ったものの紹介　～環境編

- IBMi

  OS　　:V7R3

  Node.js :v4.6.1

  ①5733OPS Option 5導入
  ②PTF SI59404適用
  ③/QOpenSys/QIBM/ProdData/OPS/Node4
  　にインストールされる

  npm　　:2.15.9

今回作ったものの紹介　～環境編

- 開発準備

```
//PATHの設定
export PATH=/QOpenSys/QIBM/ProdData/OPS/Node4/bin:$PATH
export LIBPATH=/QOpenSys/QIBM/ProdData/OPS/Node4/bin:$LIBPATH

//ディレクトリを作成
mkdir /home/node/
cd /home/node/

//express-generatorをインストール
npm install express-generator

//express-generatorでexpressプロジェクト雛形を作成
/home/node/node_modules/express-generator/bin/express osssample
```

今回作ったものの紹介　～環境編

```
//package.jsonの編集
"dependencies": {
  "body-parser": "・1.13.2",
  "cookie-parser": "・1.3.5",
  "corser": "latest", //追加
  "forever": "latest", //追加
  "express": "・4.13.1"
}
```

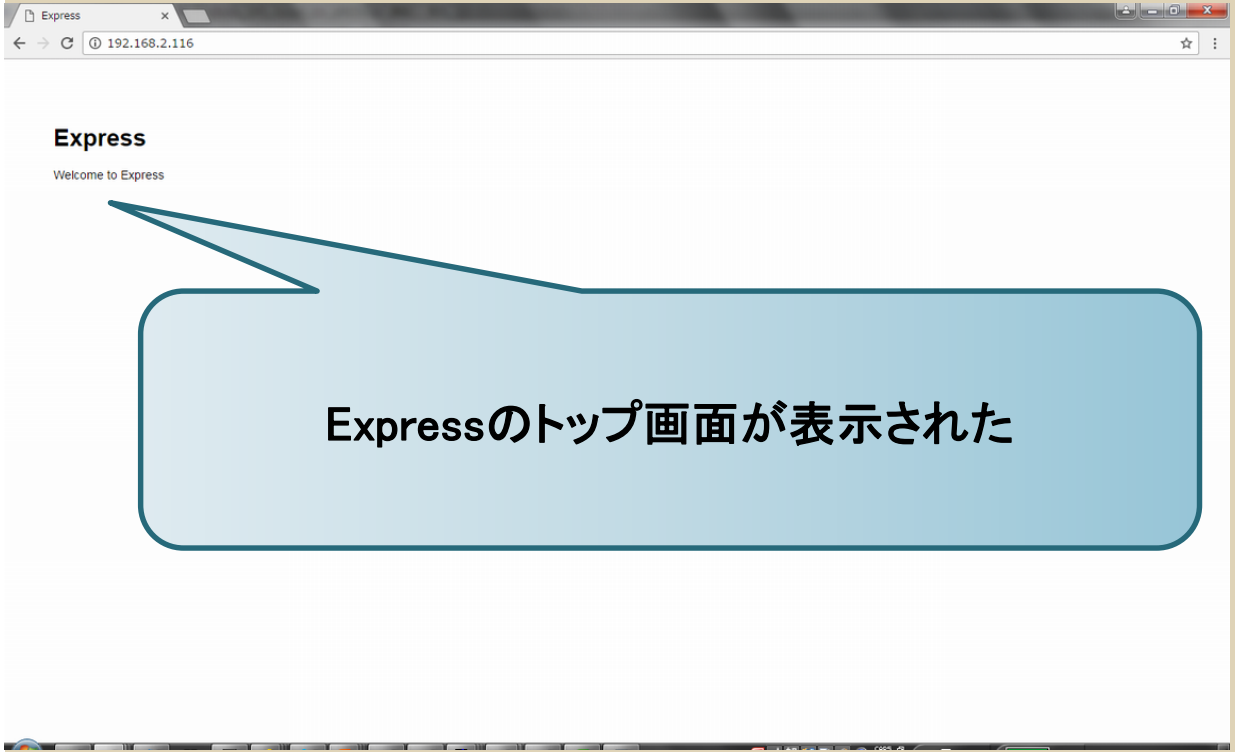
```
cd /home/node/osssample
//package.jsonに基づいてnpmインストール
npm install
```

```
//ポートの変更（bin/www）
var port = normalizePort(process.env.PORT || '80'); //←3000から変更
```

今回作ったものの紹介　～環境編

```
//デーモン起動（foreverで実行）
/home/node/osssample/node_modules/forever/bin/forever start bin/www
```

//ブラウザからアクセス

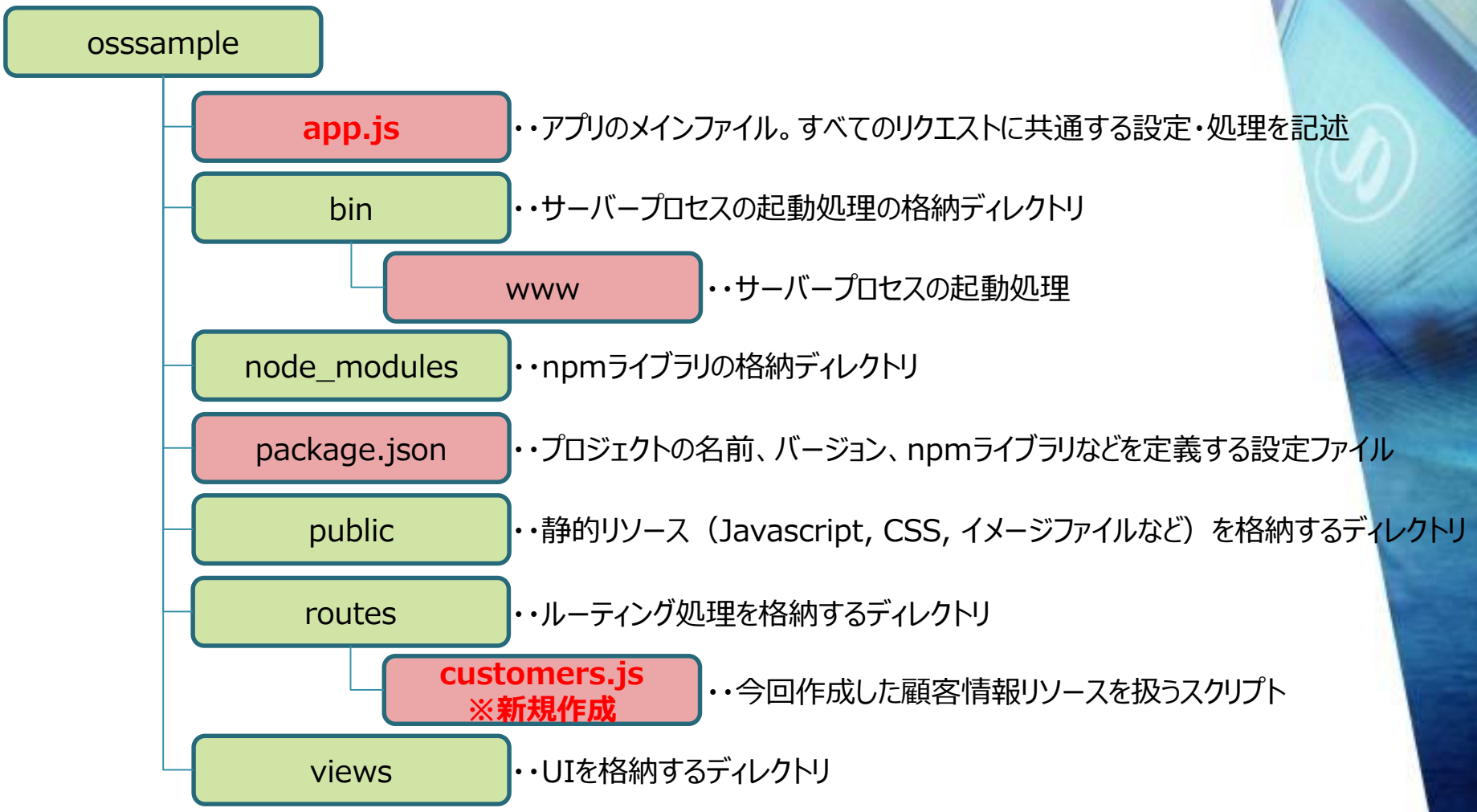


今回作ったものの紹介　～環境編

- 使用したnpmライブラリの説明
  - ・express
    → 『Express』とは、Node.js上で動作する
    MVCフレームワーク
  - ・express-generator
    → 『Express』のプロジェクト雛形を自動作成する
  - ・corser
    → クロスドメインをサーバー側で許可する
  - ・forever
    → node.jsスクリプトのデーモン化

今回作ったものの紹介　～ソース編

- ディレクトリ構成（Express4）



今回作ったものの紹介　～ソース編

app.js

```
var express = require('express');
var bodyParser = require('body-parser');
var corser = require("corser"); // CORS許可ライブラリの使用

var customers = require('./routes/customers'); // ルーティングの定義

var app = express();

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(corser.create()); // CORS許可

// リクエストURI毎のルーティング
app.get('/api/v1/customers', customers.index);          // 全件取得
app.get('/api/v1/customers/:id', customers.show);       // １件取得
app.post('/api/v1/customers', customers.create);        // 登録
app.put('/api/v1/customers/:id', customers.update);     // 変更
app.delete('/api/v1/customers/:id', customers.destroy); // 削除
```

今回作ったものの紹介　～ソース編

```
// catch 404 and forward to error handler
app.use(function(req, res, next) {
  var err = new Error('Not Found');
  err.status = 404;
  next(err);
});

// error handler
app.use(function(err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err : {};

  // render the error page
  res.status(err.status || 500);
  res.render('error');
});

module.exports = app;
```

今回作ったものの紹介　～ソース編

customers.js ※新規作成

```
// db2接続、XMLToolkitライブラリ使用
var db =require('/QOpenSys/QIBM/ProdData/OPS/Node4/os400/db2i/lib/db2');
var xt =
require('/QOpenSys/QIBM/ProdData/OPS/Node4/os400/xstoolkit/lib/itoolkit');

module.exports = {
  index : function (req, res) { // 全件取得
    db.init();
    db.conn("*LOCAL");
    db.exec("SELECT CMO1,CMO2,CMO3,CMO4,CMO5 FROM OSSSAMPLE.CUSINF",
      function(customers) {
        res.writeHead(200, {
          'Content-Type':'application/json; charset=utf-8',
        });
        res.end(JSON.stringify(customers));
      });
    db.close();
  },
```

今回作ったものの紹介　～ソース編

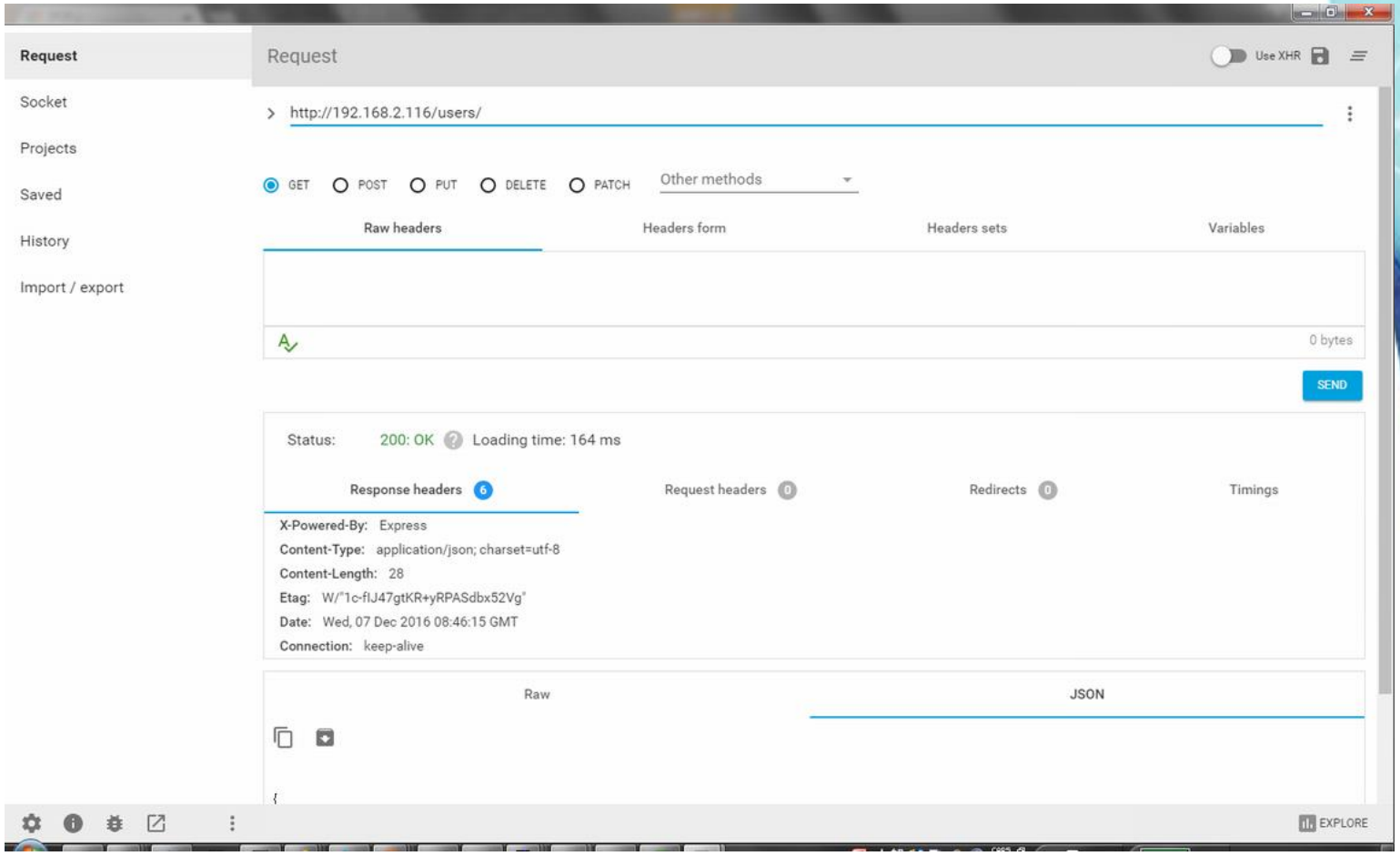
```
create : function (req, res) {  // 登録
  var conn = new xt.iConn("*LOCAL");
  conn.add(xt.iCmd("ADDLIBLE LIB(OSSSAMPLE) POSITION(*LAST)"));
  var pgm  = new xt.iPgm("CREATE", {"lib":"*LIBL"});
  pgm.addParam(req.body.cm01 ,"42a");
  pgm.addParam(req.body.cm02 ,"42a");
  pgm.addParam(req.body.cm03 ,"22a");
  pgm.addParam(req.body.cm04 ,"14a");
  pgm.addParam(req.body.cm05 ,"20a");
  conn.add(pgm.toXML());
  conn.run(function (rsp) {
    var results = xt.xmlToJson(rsp);
    res.writeHead(200, {
      'Content-Type':'application/json; charset=utf-8'
    });
    res.end(JSON.stringify(results[1]));
  });
},
```

今回作ったものの紹介　～ソース編

```
  show : function (req, res) {    // TODO: １件取得
    res.send('show');
  },
  update : function (req, res) {  // TODO: 変更
    res.send('update');
  },
  destroy : function (req, res) { // TODO: 削除
    res.send('destroy');
  }
};
```

今回作ったものの紹介　～実行編

- 確認ツールは以下を使用
  →Advanced REST client（Chrome拡張）



今回作ったものの紹介　～実行編（デモ）

- GET（全件取得）

今回作ったものの紹介　～実行編（デモ）

- POST（登録）

今回作ったものの紹介　～実行編（デモ）

- DELETE（削除）※未実装

APIの今後の可能性

- APIを使ったビジネス

  API公開によって自社だけでなく、
  他社のサービスも活用して広がっていく商圏（経済圏）
  →APIエコノミー

  例えば・・・
  今IBMiにあるリソース（データ、プログラム）を
  外部からAPIで活用し合うビジネスがくるかも？

API公開時の考慮点

- セキュリティー
  →誰もが使える？認証は？

- リクエスト数、データ量
  →制限が必要？サーバーリソースは大丈夫？

- インターフェース仕様の周知
  →手段、バージョン管理は？

IBMiのリソースをAPIを通じて安全に公開するには

- BlueMix SecureGatewayを利用した
  ハイブリッドクラウド

- IBM API Connect