



IBM i World 2021

IBM i

IBM i コンテンツ (10月版)

RPG IIIがメインのユーザー様における REST API利用方法について

日本アイ・ビー・エム株式会社
テクノロジー事業本部
Power Systems テクニカルセールス
澤田英寿

IBM iにおけるREST APIを利用したアプリ/データ連携

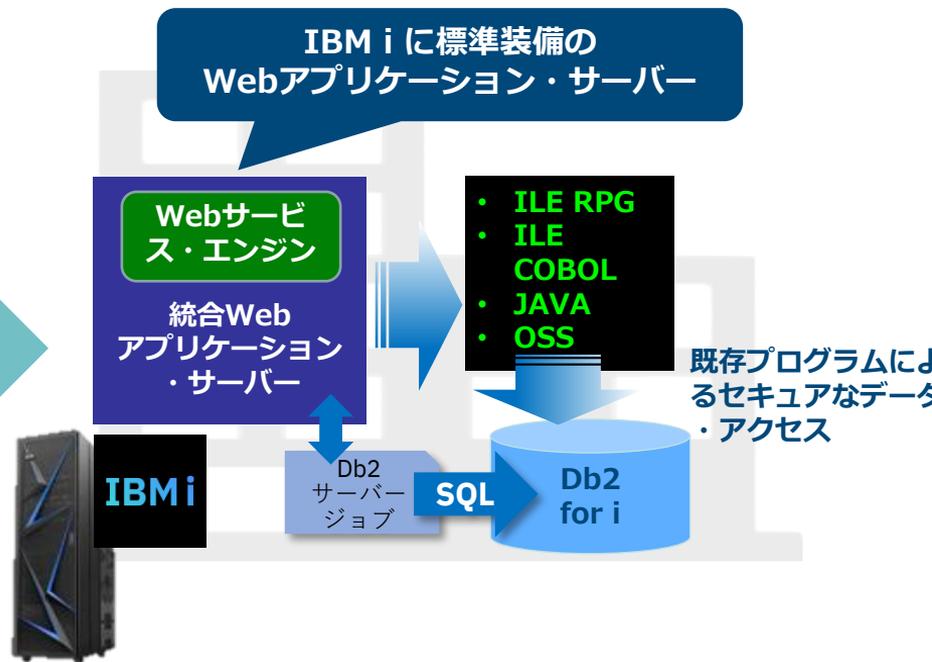
IBM iに標準装備されたWebアプリケーション・サーバーを利用して基幹データをAPI公開！
実績のある基幹業務ロジックを使用することでセキュアなデータ・アクセス環境を実現

REST APIを利用することで
クライアント・アプリケーションは
自由に開発可能



クライアント・アプリケーションの場所も
オンプレミス/クラウドを選ばず配置可能

API呼び出し
HTTP/HTTPS



IBM iユーザーからのFAQ

基幹業務ロジックが、RPGⅢをメインで利用している場合は、REST APIを利用したアプリ・データ連携ができないのでしょうか。



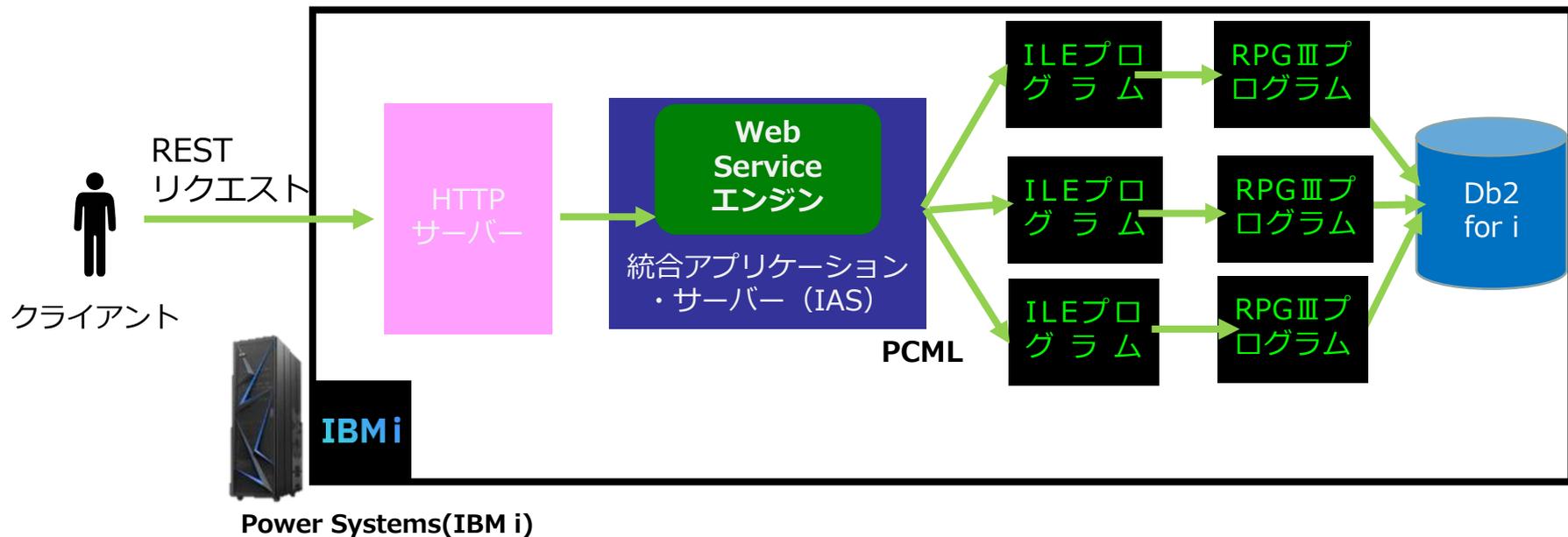
下記の2つの解決策があります

1. IBM i のプログラム (RPGⅢ) からILE RPGを経由して、REST APIサービス化する。
2. IBM i のプログラム (RPGⅢ) をILE RPGに変換して、REST API化する。

次ページ以降に、解説します。

1. IBM i のプログラム (RPGⅢ) からILE RPGを經由して、REST APIサービス化する

ILEプログラムで実装されたサービスをREST APIとして外部公開し、業務アプリのRPGⅢのPGMを呼び出す(全てのRPGをILE-RPGに変更する必要はない)
RPGⅢとILE RPGは普通に相互呼び出し可能です。



2. IBM i のプログラム (RPGⅢ) を ILE RPG に変換して、REST API化する

業務アプリのRPGⅢのソースをILE-RPGに変換して、REST API化する

RPGソース変換(CVTRPGSRC)コマンドは、RPGⅢまたはRPG/400ソース・コードをILE RPGソース・コードに変換することができます。

例 **RPGⅢ**ソースの**RPGⅣ**ソースへの変換（下記のようなコマンドで自動変換されます）
CVTRPGSRC FROMFILE(MYLIB/QRPGSRC) FROMMBR(XMPLE1) TOFILE(MYLIB/QRPGLESRC)
TOMBR(*FROMMBR) EXPCPY(*NO) CVTRPT(*YES) LOGFILE(MYLIB/QRNCVTLG)

- ・このコマンドはファイルMYLIB/QRPGSRC中のRPGⅢソース・メンバーXMPLE1を、ファイルMYLIB/QRPGLESRC中の同じ名前のRPGⅣソース・メンバーに変換します。
- ・RPGⅢプログラムの/COPYステートメントは展開されません。これらは、RPGⅣプログラムで/COPYステートメントとして残ります。報告書が印刷されます。それぞれの変換の状況は、ファイルMYLIB/QRNCVTLGに配置されます。

このページ以降は、

「IBM iのREST API使用法の簡易ガイド」

になります。

IBM i のREST API使用法の 簡易ガイド

「IBM i でのREST API使用法の簡易ガイド」

目次

1. ILE-RPGによるREST API活用

1. [外部 → IBM i] IBM i のプログラム（ILE RPG/ILE COBOL等）をREST APIサービス化し、外部のアプリケーションから呼び出す
2. [IBM i → 外部] IBM i からREST APIの外部アプリケーションを呼び出す
 1. Db2 for i のHTTPメソッド機能を用いたREST APIの呼び出し
 2. ILE RPGでの統合Webサービス・クライアントAPI (Transport API) の活用

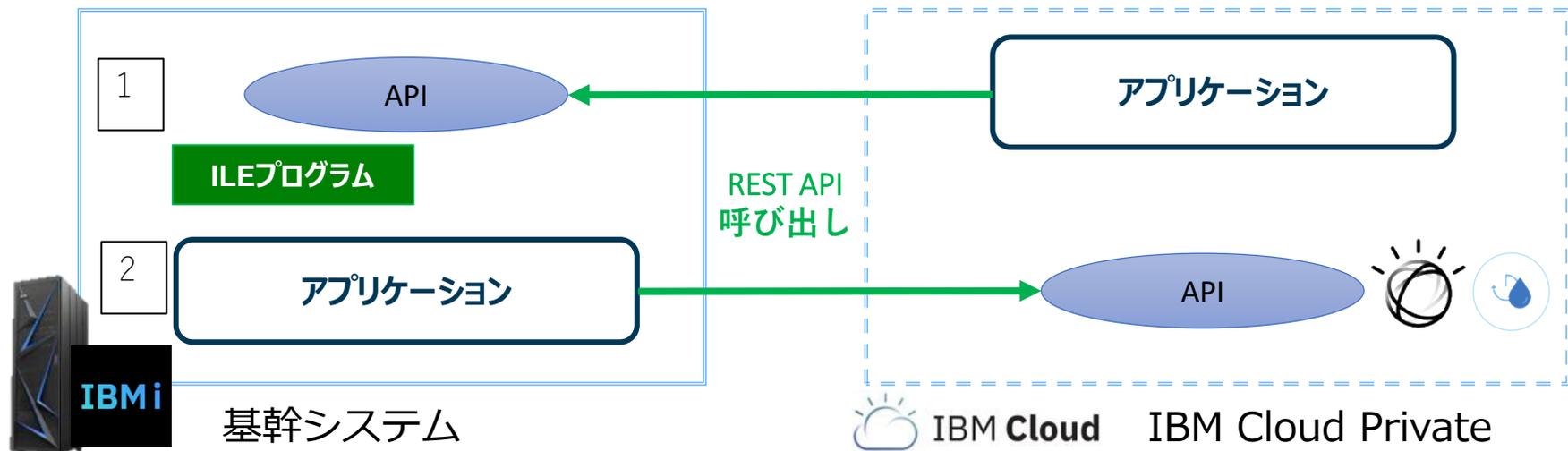
2. [外部 → IBM i] DB照会をSQL記述でREST APIサービス化

3. JSONデータとDb2 for i データの変換機能

1. ILE RPGによるREST API活用

IBM i がAPIサービスと連携する2つのアプローチ

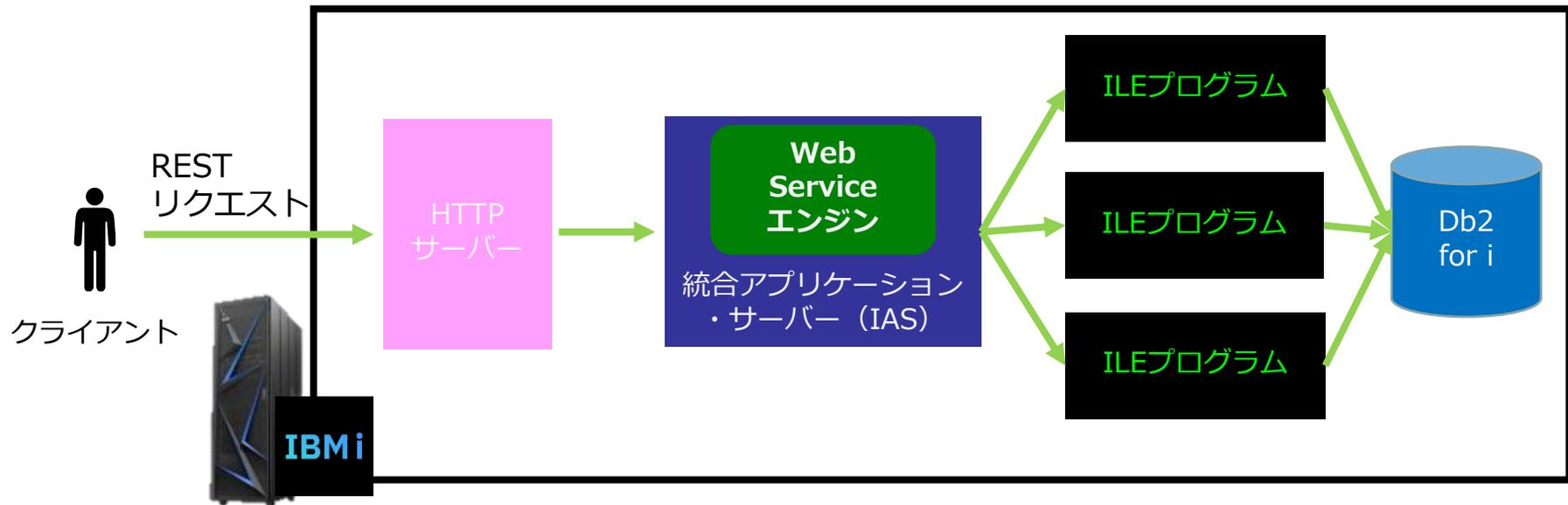
1. [外部 → IBM i] IBM i のILEプログラム(ILE RPG/ILE COBOL等)をREST APIサービス化し、外部のアプリケーションから呼び出す
2. [IBM i → 外部] IBM i のILEプログラム(ILE RPG/ILE COBOL等)から、外部のAPIを呼び出す



1. [外部 → IBM i]

IBM i の ILE プログラム (ILE RPG / ILE COBOL 等) を REST API サービス化し、外部のアプリケーションから呼び出す

IBM i に標準搭載されている統合アプリケーション・サーバー (IAS) で REST をサポート ILE プログラム で実装されたサービスを REST API として外部公開可能



IBM i統合アプリケーション・サーバーによる Web Serviceエンジンサポート

IBM i 7.2以降でサポート (IBM i OS標準機能)

IBM i 7.2 HTTP グループPTF レベル6以降

WebサービスをデプロイするIASインスタンスはWeb Administration for i
(http://<system>:2001/HTTPAdmin) からウィザードで作成可能
関連HTTPサーバー・インスタンスも自動作成

Web サービスのプログラム・オブジェクトを指定します。 ?

IBM i ライブラリー名および ILE プログラム・オブジェクト名の指定 (推奨)

プログラム・オブジェクトの場所は、プログラム・オブジェクトが入っているライブラリー・プログラム・オブジェクトの場所を指定する一番の早道でかつ推奨方法です。

ライブラリー名:

ILE オブジェクト名:

ILE オブジェクト・タイプ: *SRVPGM *PGM

統合ファイル・システムを参照して IBM i プログラム・オブジェクトを探します

呼び出し先 ILE プログラムの指定

リソース・メソッド情報を指定します。 ?

プロシージャー名: CONVERTTEMP
 リソースの URI パス・テンプレート: /temp/d+
 HTTP 要求メソッド: GET
 メソッドの URI パス・テンプレート: *NONE または...
 HTTP 応答コード出力パラメーター: *NONE
 HTTP ヘッダー配列出力パラメーター: *NONE
 許可される入力メディア・タイプ: ALL または...
 返される出力メディア・タイプ: *JSON または...

入力パラメーターをラップするかどうか:

- 入力パラメーターをラップする
 入力パラメーターをラップしない

入力パラメーター・マッピング:

パラメーター名	データ・タイプ	入力ソース	ID	デフォルト値
TEMPIN	char	*PATH_PARAM	temp	*NONE または...

APIアクセス・メソッド
の定義

統合Webサービス・サーバーへのRESTサービスの作成ステップ

4点について検討、ウィザードで指定

- ・ RESTサービスとして展開するILEプログラムの検討
- ・ プログラムでの処理に対するHTTPプロトコルのメソッドの検討
- ・ URIの検討 (URI : Uniform Resource Identifier)
- ・ 入出力に関する検討
 - 入出力パラメーターとして受け入れる形式
 - ILEプログラムの入力パラメーターとして渡す値の指定方法
 - ILEプログラムの出力パラメーターとして応答コードやHTTPヘッダーを含ませるか

統合Webサービス・サーバーへのRESTサービスの作成ステップ

4点について検討、ウィザードで指定

- RESTサービスとして展開するILEプログラムの検討
- プログラムでの処理に対するHTTPプロトコルのメソッドの検討
- URIの検討
- 入出力に関する検討

入出力パラメーターとして受け入れる形式

ILEプログラムの入力パラメーターとして渡す値の型を指定するかどうか

ILEプログラムの出力パラメーターとして応答コードを返すかどうか

- 対象プログラムでのCRUD操作の特定
 - **Create:** データの作成
 - **Read:** データの参照
 - **Update:** データの更新
 - **Delete:** データの削除
- リソースに対するCRUD操作とHTTPメソッドを対応付ける
 - **Create: POST**
 - **Read: GET**
 - **Update: PUT**
 - **Delete: DELETE**

統合Webサービス・サーバーへのRESTサービスの作成ステップ

4点について検討、ウィザードで指定

- ・ RESTサービスとして展開するILEプログラムの検討
- ・ プログラムでの処理に対するHTTPプロトコルのメソッドの検討
- ・ URIの検討

・ 入出力に関する検討

入出力パラメーターとして受け入れる
ILEプログラムの入力パラメーターとして渡す値の指定方法
ILEプログラムの出力パラメーターとして渡す値の指定方法

URI検討のポイント

- ・ 操作対象のリソース (データ) を名詞で表現する
- ・ 処理内容は使用するHTTPメソッドで判別できる

・ 容易に推測可能な語句を採用する
ヘッダーを含ませるか

統合Webサービス・サーバーへのRESTサービスの作成ステップ

4点について検討、ウィザードで指定

- RESTサービスとして展開するILEプログラム
- プログラムでの処理に対するHTTP
- URIの検討
- 入出力に関する検討
 - 入出力パラメーターとして受け入れる形式
 - ILEプログラムの入力パラメーターとして渡す値の指定
 - ILEプログラムの出力パラメーターとして応答コードを返すか
- RESTサービスとしてデプロイする各プロシージャー単位で検討が必要
 - 入出力パラメーターとして受け入れる形式
 - JSON (application/json)
 - XML (application/xml)
 - 入力パラメーターの受け渡し方式
 - パス・セグメント (/cars/{color})
 - マトリックス・パラメーター (/cars;color=blue)
 - クエリー・パラメーター (/cars?color=blue)
 - 出力パラメーターとして応答コードやHTTPヘッダーを含めるか
 - アプリとしてHTTPヘッダーを返す場合、HTTPキャッシュに関するヘッダー情報を返すことが多い

統合Webサービス・サーバーへのRESTサービスの作成検討例

学生情報を処理するILEサービス・プログラム STUDENTRSC

学生情報はテーブル STUDENTRSCライブラリーのSTUDENTDB に格納

提供プロシージャーは5つ

プロシージャー	処理内容
GETALL	学生情報を全件参照
GETBYID	指定した学生コードに該当の学生情報を参照
CREATE	新規の学生情報を登録
UPDATE	既存の学生情報を更新
REMOVE	指定した学生コードの学生情報を削除



統合Webサービス・サーバーへのRESTサービスの作成検討例

検討例

- ・ プログラムでの処理に対するHTTPプロトコルのメソッドの検討
- ・ URIの検討
- ・ 入出力に関する検討

URI			
/{context-root}/students			
プロシージャー	HTTPメソッド	入出力	URI
GETALL	GET	<ul style="list-style-type: none"> ・ 結果はJSON形式で出力 ・ HTTP応答コードとヘッダーも出力に含める 	/students
GETBYID	GET	<ul style="list-style-type: none"> ・ 学生コードをパスセグメントで受け取る ・ 結果はJSON形式で出力 ・ HTTP応答コードとヘッダーも出力に含める 	/students/{id}
CREATE	POST	<ul style="list-style-type: none"> ・ HTTP応答コードとヘッダーも出力に含める 	/students
UPDATE	PUT	<ul style="list-style-type: none"> ・ HTTP応答コードを出力に含める 	/students
REMOVE	DELETE	<ul style="list-style-type: none"> ・ 学生コードをパスセグメントで受け取る ・ HTTP応答コードを出力に含める 	/students/{id}

RESTサービスとして作成するILE RPGプログラム

RESTサービスプログラムのソースは固定フォーマットでも、フリー・フォーマットでもOK

```
//*****
// getAll
//*****
P getAll      B          EXPORT
D getAll      PI
D students_...
D LENGTH      10i 0
D students    Likeds(studentRec) dim(1000)
D options(*varsize)
D httpStatus  10i 0
D httpHeaders 100a dim(10)
/FREE
clear httpHeaders;
clear students;
students_LENGTH = 0;

openStudentDB();

setll *loval STUDENTDB;

read(e) studentR;
if (%ERROR);
  httpStatus = H_SERVERERROR;
  return;
endif;

dow (NOT %eof);
  students_LENGTH = students_LENGTH+1;
  students(students_LENGTH).studentID = studentID;
  students(students_LENGTH).firstName = firstName;
  students(students_LENGTH).lastName = lastName;
  students(students_LENGTH).gender = gender;

  read(e) studentR;
  if (%ERROR);
    httpStatus = H_SERVERERROR;
```

```
//*****
// getByID
//*****
P getByID     B          EXPORT
D getByID     PI
D studentID   9a const
D student     Likeds(studentRec)
D httpStatus  10i 0
D httpHeaders 100a dim(10)
/FREE
clear httpHeaders;
clear student;

openStudentDB();

chain(e) studentID STUDENTDB;
if (%ERROR);
  httpStatus = H_SERVERERROR;
  return;
elseif %FOUND;
  student.studentID = studentID;
  student.firstName = firstName;
  student.lastName = lastName;
  student.gender = gender;

  httpStatus = H_OK;
else;
  httpStatus = H_NOTFOUND;
endif;

httpHeaders(1) = 'Cache-Control: no-cache, no-store';

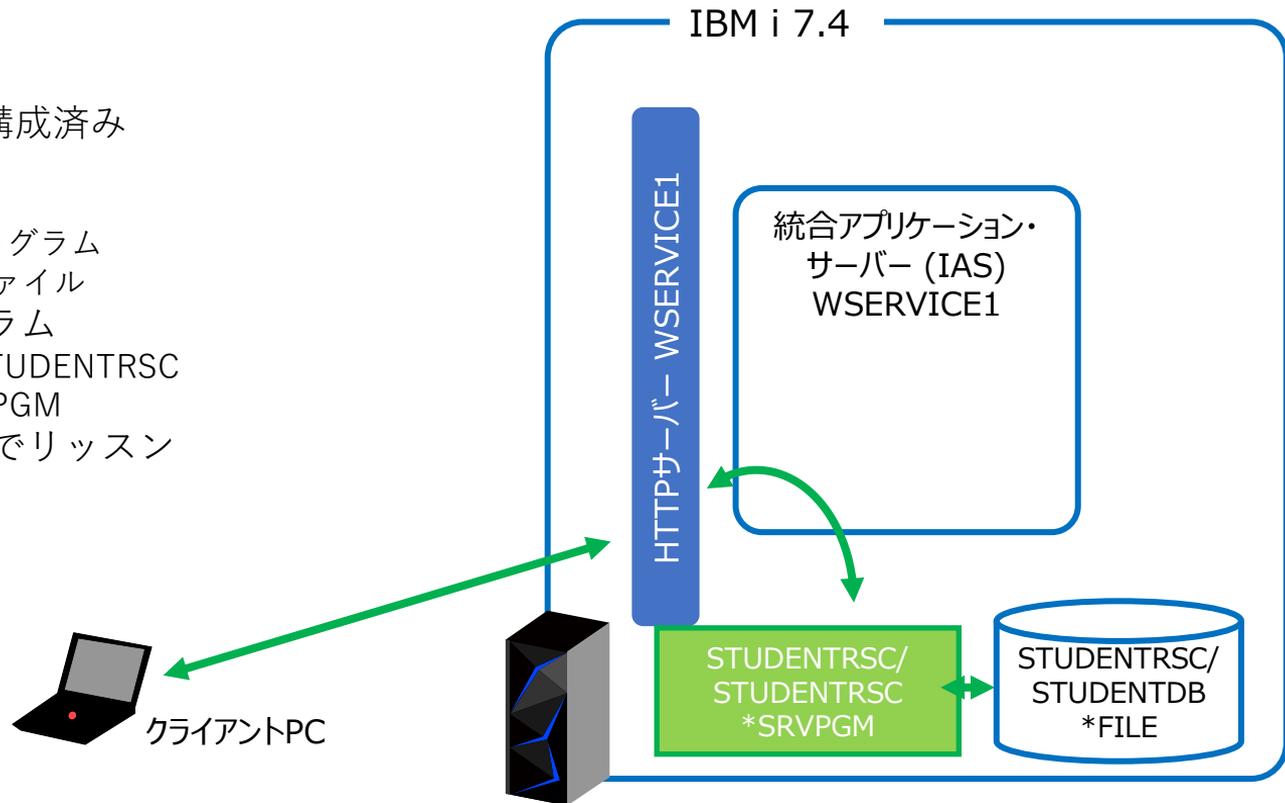
closeStudentDB();
/END-FREE
P getByID     E
```

統合Webサービス・サーバーへのRESTサービスの作成

開発環境

IBM i 7.4

- ・ 全て単一のOS環境に構成済み
 - HTTPサーバー
 - IASインスタンス
 - ILEサービス・プログラム
 - データベース・ファイル
- ・ ILEサービス・プログラム
 - STUDENTRSC/STUDENTRSC
 - 前掲検討例のSRVPGM
- ・ HTTPサーバーは以下でリッスン
 - HTTP: 10073



<参考>Webサービス用IASインスタンスの作成手順

1. http://<system>:2001/HTTPAdmin にブラウザでアクセス
2. 「セットアップ」タブの「新規 Web Serviceサーバーの作成」を選択

The screenshot shows the IBM Web Administration for i interface. The top navigation bar includes 'セットアップ' (Setup), '管理' (Management), '拡張' (Expansion), and '関連したリンク' (Related Links). The main content area is titled 'IBM Web Administration for i' and contains a list of options for creating new servers. The option '新規 Web Services サーバーの作成' (Create New Web Services Server) is highlighted with a red rectangular box. Below it are two other options: '新規 HTTP サーバーの作成' (Create New HTTP Server) and '新規アプリケーション・サーバーの作成' (Create New Application Server).

IBM Web Administration for i ウェルカム SAWADA

セットアップ | 管理 | 拡張 | 関連したリンク

▶ 共通タスクおよびウィザード

IBM Web Administration for i
始めに - Web コンテンツの実行に必要なサーバーの作成およびその学習.

新規 Web Services サーバーの作成
「Web Services サーバーの作成」ウィザードは、IBM i上で実行される既存のプログラム (RPG や COBOL など) を Web サービスとして外部化する便利な方法を提供します。このため、Web サービス・クライアントは、インターネットやイントラネットから、SOAP など Web サービス・ベースの業界標準通信プロトコルを使用して IBM i プログラム・ベースのサービスと対話することができます。

新規 HTTP サーバーの作成 ⓘ
HTTP Web コンテンツを実行するための新規 HTTP サーバー (powered by Apache) を作成します。このウィザードは、単純な Web サービスを開始するために必要なすべてのものを作成します。

新規アプリケーション・サーバーの作成 ⓘ
動的 Web アプリケーションを実行するための新規アプリケーション・サーバーを作成します。i 用 IBM 統合 Web アプリケーション・サーバーまたは WebSphere Application Server のいずれかを作成します。

<参考>Webサービス用IASインスタンスの作成手順（続き）

3. IASインスタンス名（およびテキスト）を入力、「Next」ボタンをクリック
4. IASインスタンスのポートを指定、「Next」ボタンをクリック
5. IASインスタンスのサブシステムを指定、「NEXT」ボタンをクリック

<p>Web Services サーバーの作成 Web サービス・サーバー名の指定 - ステップ 1 / 4</p> <p>Welcome to the Create Web Services Server wizard. A Web Services Server creates programs and SQL statements. This wizard creates everything you need. For more details, see the following URL: http://www.ibm.com</p> <p>このサーバーに固有の名前を指定 ?</p> <p>サーバー名: <input type="text" value="WSERVICE"/></p> <p>サーバー記述: 「Web サービス・サーバーの作成」ウィザードには</p> <p><input checked="" type="checkbox"/> HTTP サーバーの作成</p>	<p>Web Services サーバーの作成 サーバーのネットワーク属性の指定 - ステップ 2 / 5</p> <p>サーバーは、システムの特定の IP アドレスで、あるいはすべての IP アドレスで実行されます。</p> <p>サーバーの IP アドレスおよびポートの指定 ?</p> <p>サーバー・コマンド・ポートの指定: <input type="text" value="10086"/></p> <p>サーバーの IP アドレスおよびポートの指定</p> <p>IP アドレス: <input type="text" value="すべての IP アドレス"/></p> <p>ポート: <input type="text" value="10085"/></p> <p>HTTP サーバーの IP アドレスおよびポートの指定</p> <p>IP アドレス: <input type="text" value="すべての IP アドレス"/></p> <p>ポート: <input type="text" value="10095"/></p>	<p>Web Services サーバーの作成 Specify subsystem for server - ステップ 3 / 5</p> <p>Specify the operating environment for the server's jobs by specifying work management attributes the controls what server jobs running in subsystem QHTTSPSVR. ?</p> <p>Path to job description: <input type="text" value=" QSYS.LIB/QHTTSPSVR.LIB/QZHBHTTP.JOBDD"/> または... <input type="button" value="参照"/></p> <p>Path to job queue: <input type="text" value="*JOBDD"/> または... <input type="button" value="参照"/></p> <p>Routing data: <input type="text" value="*JOBDD"/> または... <input type="button" value="参照"/></p>
--	--	---

<参考>Webサービス用IASインスタンスの作成手順（続き）

6. IASインスタンスのユーザーIDを指定、「NEXT」ボタンをクリック

7. サマリーを確認して「Finish」ボタンをクリック、作成が完了

関連のHTTPサーバー・インスタンスも自動作成

HTTPポートも確認可能

Web Services サーバーの作成

サーバーのユーザー ID の指定 - ステップ 4 / 5

サーバーのジョブを実行するためには、サーバーに IBM i ユーザー ID が必要です。プロジェクトに対する権限がこのユーザー ID に付与されるからです。

このサーバーのユーザー ID を指定: ?

- デフォルトのユーザー ID を使用
- 既存のユーザー ID を指定
- 新規ユーザー ID の作成

注: デフォルトのサーバー・ユーザー ID は QWSERVICE です。

Web Services サーバーの作成

要約 - ステップ 5 / 5

サーバー サービス

Web サービス・サーバー情報

サーバー名:	WSERVICE
サーバー記述:	「Web サービス・サーバーの作成」ウィザードによって作成された Web サービス・サーバー。
ポート:	10085
コマンド・ポート:	10086
サーバー・ルート:	/www/WSERVICE
サーバー URL:	http://demo00:10095
サーバーのユーザー ID:	QWSERVICE
コンテキスト・ルート:	/web
Job queue:	*JOB
Job description:	QHTTSPVR/QZHBHTTP
Subsystem:	QHTTSPVR/QHTTSPVR
Routing data:	*JOB

HTTP サーバー情報

HTTP サーバー名:	WSERVICE
HTTP サーバー記述:	「Web サービス・サーバーの作成」ウィザードによって作成された Web サービス・サーバー。
ポート:	10095
文書ルート:	/www/WSERVICE/htdocs
サーバー・ルート:	/www/WSERVICE
サーバー関連:	WSERVICE
Job queue:	*JOB
Job description:	QHTTSPVR/QZHBHTTP
Subsystem:	QHTTSPVR/QHTTSPVR
Routing data:	*JOB

<参考>RESTサービスの作成手順

1. `http://<system>:2001/HTTPAdmin` にブラウザでアクセス
2. 「管理」タブ→「Application Servers」タブ→サーバーを選択
3. 「Web Services ウィザード」→「新規サービスの配置」を選択
4. 「REST」を選択し、ILE PGM Objectに書きを入力し、「Next」ボタンをクリック

セットアップ 管理 拡張 | 関連したリンク

すべてのサーバー | HTTP サーバー Application Server インストール済み環境

実行中 [停止] [リフレッシュ] [ヘルプ] サーバー: WSERVICE1 - V2.6 (Web サービス)

共通タスクおよびウィザード

WSERVICE1 > 配置済みサービスの管理 > 新規サービスの配置

新規サービスの配置

Web サービス・タイプの指定 - ステップ 1 / 9

Welcome to the Deploy New Service wizard. This wizard helps you create Web services using IBM i objects and data. SOAP ベースの Web サービスは、自己完結型ソフトウェア・コンポーネントでアクセスできる操作および SOAP プロトコルに基づいた XML メッセージを交換する操作を記述する、明確に定義されたインターフェースを備えています。REST ベースの Web サービスでは、クライアント要求がリソース・メソッドによって処理され、交換されるメッセージの形式はリソース自体によって定義されます。

Web サービス・タイプの指定: REST

Specify Web service implementation:

ILE Program Object

Specify path to ILE program or service program:

プログラム・オブジェクトのパス: /QSYS.LIB/STUDENTRSC.LIB/STUDENTRSC.SRVPGM 参照 e.g. /QSYS.LIB/MYLIB.LIB/MY.SRVPGM

注: *PGM または *SRVPGM オブジェクトを指定してください。

Web サービスとしての SQL

<参考>RESTサービスの作成手順（続き）

- 「Resource name」にURI「students」を指定し、「Next」ボタンをクリック
「URI path template」にはサービス判別のための正規表現を指定可能だが、今回は「/context-root/students」とするため指定不要

WSERVICE1 > [配置済みサービスの管理](#) > 新規サービスの配置

新規サービスの配置

サービス名の指定 - ステップ 2 / 9

外部化される Web サービスはリソースです。URI パス・テンプレートが着信 HTTP 要求のマッチング・パターンを識別します。パス化されるのかをさらに制限する正規表現を含むことのできる 1 つ以上のテンプレート・パラメーターとすることができます。?

リソース名:	<input type="text" value="students"/>
サービス記述:	<input type="text" value="STUDENTRSC"/>
URI パス・テンプレート:	<input type="text" value="/"/> 例: /temperature, /temperature/{temp:\d+}

<参考>RESTサービスの作成手順（続き）

6. ILEオブジェクト内のプロシージャーの入出力パラメーターを調整・設定後、「Next」ボタンをクリック

宣言時にCONSTを指定したパラメーターをinputとして自動認識

新規サービスの配置

Web サービスとして外部化するエクスポート・プロシージャーの選択 - ステップ 3 / 9

エクスポートされたプロシージャーはプログラム・オブジェクトへのエントリー・ポイントであり、V型高水準言語ステートメントです。サービス・プログラムは1つ以上のプロシージャーからなります。

下記の表は、プログラム・オブジェクトにある、このWebサービスによって外部化できるすべてのエクスポートされたプロシージャーのリストです。「使用方法」パラメーター属性は、クライアントが送信するデータ構造体の要素名としてパラメーター名を使用するかどうかを示します。

長さフィールドの検出

データ構造体の要素名としてパラメーター名を使用

エクスポート・プロシージャー: ?

選択	プロシージャー名/パラメーター名	使用方法	データ・タイプ
<input checked="" type="checkbox"/>	▶ REMOVE		
<input checked="" type="checkbox"/>	▶ UPDATE		
<input checked="" type="checkbox"/>	▶ CREATE		
<input checked="" type="checkbox"/>	▶ GETBYID		
<input checked="" type="checkbox"/>	▶ GETALL		

すべて選択

すべて選択解除

すべて展開表示

すべて省略表示

エクスポート・プロシージャー: ?

選択	プロシージャー名/パラメーター名	使用方法	データ・タイプ
<input checked="" type="checkbox"/>	▶ REMOVE		
	studentID	入力	char
	httpStatus	出力	int
<input checked="" type="checkbox"/>	▶ UPDATE		
	student	出力	struct
	httpStatus	出力	int
<input checked="" type="checkbox"/>	▶ CREATE		
	student	出力	struct
	httpStatus	出力	int
	httpHeaders	出力	char
<input checked="" type="checkbox"/>	▶ GETBYID		
	studentID	入力	char
	student	出力	struct
	httpStatus	出力	int
	httpHeaders	出力	char
<input checked="" type="checkbox"/>	▶ GETALL		
	students_LENGTH	出力	int
	students	出力	struct
	httpStatus	出力	int
	httpHeaders	出力	char

すべて選択

すべて選択解除

すべて展開表示

すべて省略表示

<参考>RESTサービスの作成手順（続き）

7. 各プロシージャーに対して以下を設定

- ・使用するHTTPメソッド
- ・URIパス・テンプレート（入力パラメーターをパス・セグメント等で渡す場合はここで指定）
- ・出力パラメーターにHTTPステータス、ヘッダーを含む場合、該当パラメーターの設定
- ・受け入れるメディア・タイプ（JSON, XML）
- ・入力パラメーターの受け渡し方法とILE側パラメーターのマッピング設定

<REMOVEプロシージャーの例>

プロシージャー名: REMOVE
 リソースの URI パス・テンプレート: /
 HTTP 要求メソッド: DELETE
 メソッドの URI パス・テンプレート: {id:\w(9)} または...
 HTTP 応答コード出力パラメーター: httpStatus
 HTTP ヘッダー配列出力パラメーター: *NONE
 HTTP header information: *NONE

Error response output parameter: *NONE または...
 許可される入力メディア・タイプ: *ALL または...
 返される出力メディア・タイプ: *JSON または...
 Identifier for input wrapper element: removeinput または...
 Identifier for output wrapper element: removeResult または...

入力パラメーターをラップするかどうか:
 入力パラメーターをラップする
 入力パラメーターをラップしない

入力パラメーター・マッピング:

パラメーター名	データ・タイプ	入力ソース	ID	デフォルト値
studentID	char	*PATH_PARAM	id	*NONE

<参考>RESTサービスの作成手順（続き）

7. 各プロシージャーに対して以下を設定（続き）

- ・使用するHTTPメソッド
- ・URIパス・テンプレート（入力パラメーターをパス・セグメント等で渡す場合はここで指定）
- ・出力パラメーターにHTTPステータス、ヘッダーを含む場合、該当パラメーターの設定
- ・受け入れるメディア・タイプ（JSON, XML）
- ・入力パラメーターの受け渡し方法とILE側パラメーターのマッピング設定

<UPDATEプロシージャーの例>

プロシージャー名:	UPDATE
リソースの URI パス・テンプレート:	/
HTTP 要求メソッド:	<input type="text" value="PUT"/>
メソッドの URI パス・テンプレート:	<input type="text" value="*NONE"/> または... <input type="button" value="v"/>
HTTP 応答コード出力パラメーター:	<input type="text" value="httpStatus"/> <input type="button" value="v"/>
HTTP ヘッダー配列出力パラメーター:	<input type="text" value="*NONE"/> <input type="button" value="v"/>
HTTP header information:	<input type="text" value="*NONE"/>
Error response output parameter:	<input type="text" value="*NONE"/> または... <input type="button" value="v"/>
許可される入力メディア・タイプ:	<input type="text" value="*JSON"/> または... <input type="button" value="v"/>
返される出力メディア・タイプ:	<input type="text" value="*JSON"/> または... <input type="button" value="v"/>
Identifier for input wrapper element:	<input type="text" value="updateInput"/> または... <input type="button" value="v"/>
Identifier for output wrapper element:	<input type="text" value="updateResult"/> または... <input type="button" value="v"/>

<参考>RESTサービスの作成手順（続き）

7. 各プロシージャーに対して以下を設定（続き）

- ・使用するHTTPメソッド
- ・URIパス・テンプレート（入力パラメーターをパス・セグメント等で渡す場合はここで指定）
- ・出力パラメーターにHTTPステータス、ヘッダーを含む場合、該当パラメーターの設定
- ・受け入れるメディア・タイプ（JSON, XML）
- ・入力パラメーターの受け渡し方法とILE側パラメーターのマッピング設定

<CREATE プロシージャーの例>

プロシージャー名:	CREATE
リソースの URI パス・テンプレート:	/
HTTP 要求メソッド:	POST
メソッドの URI パス・テンプレート:	*NONE または...
HTTP 応答コード出力パラメーター:	httpStatus
HTTP ヘッダー配列出力パラメーター:	httpHeaders
HTTP header information:	*NONE
Error response output parameter:	*NONE または...
許可される入力メディア・タイプ:	*JSON または...
返される出力メディア・タイプ:	*JSON または...
Identifier for input wrapper element:	createInput または...
Identifier for output wrapper element:	createResult または...

<参考>RESTサービスの作成手順（続き）

7. 各プロシージャーに対して以下を設定（続き）

- ・使用するHTTPメソッド
- ・URIパス・テンプレート（入力パラメーターをパス・セグメント等で渡す場合はここで指定）
- ・出力パラメーターにHTTPステータス、ヘッダーを含む場合、該当パラメーターの設定
- ・受け入れるメディア・タイプ（JSON, XML）
- ・入力パラメーターの受け渡し方法とILE側パラメーターのマッピング設定

<GETBYIDプロシージャーの例>

プロシージャー名: GETBYID
 リソースの URI パス・テンプレート: /
 HTTP 要求メソッド: GET
 メソッドの URI パス・テンプレート: {id:w(9)} または...
 HTTP 応答コード出力パラメーター: httpStatus
 HTTP ヘッダー配列出力パラメーター: httpHeaders
 HTTP header information: *NONE

Error response output parameter: *NONE または...
 許可される入力メディア・タイプ: *ALL または...
 返される出力メディア・タイプ: *JSON または...
 Identifier for input wrapper element: getByIDInput または...
 Identifier for output wrapper element: getByIDResult または...

入力パラメーターをラップするかどうか:
 入力パラメーターをラップする
 入力パラメーターをラップしない

入力パラメーター・マッピング:

パラメーター名	データ・タイプ	入力ソース	ID	デフォルト値
studentID	char	*PATH_PARAM	id	*NONE

<参考>RESTサービスの作成手順（続き）

7. 各プロシージャーに対して以下を設定（続き）

- ・使用するHTTPメソッド
- ・URIパス・テンプレート（入力パラメーターをパス・セグメント等で渡す場合はここで指定）
- ・出力パラメーターにHTTPステータス、ヘッダーを含む場合、該当パラメーターの設定
- ・受け入れるメディア・タイプ（JSON, XML）
- ・入力パラメーターの受け渡し方法とILE側パラメーターのマッピング設定

<GETALLプロシージャーの例>

プロシージャー名:	GETALL
リソースの URI パス・テンプレート:	/
HTTP 要求メソッド:	GET
メソッドの URI パス・テンプレート:	*NONE または...
HTTP 応答コード出力パラメーター:	HttpStatus
HTTP ヘッダー配列出力パラメーター:	httpHeaders
HTTP header information:	*NONE
Error response output parameter:	*NONE または...
許可される入力メディア・タイプ:	*ALL または...
返される出力メディア・タイプ:	*JSON または...
Identifier for input wrapper element:	getAllInput または...
Identifier for output wrapper element:	getAllResult または...

<参考>RESTサービスの作成手順（続き）

- サービスが呼び出された際に利用されるユーザー・プロファイルを指定し、「Next」ボタンをクリック
ホスト・サーバー・ジョブQZRCRSRV5の現行ユーザーとなり、実行時のオブジェクト権限等の基点となる。
今回はデモのためIASインスタンスと同一ユーザー・プロファイルとする

新規サービスの配置

このサービスのユーザー ID を指定 - ステップ 6 / 9

The service requires an IBM i user ID to run the Web service business logic. The user ID must have the necessary authority to any resources that the Web service requires.

このサービスのユーザー ID を指定: ?

- サーバーのユーザー ID を使用
- 既存のユーザー ID を指定
- 認証済みユーザー ID を使用

<参考>RESTサービスの作成手順（続き）

9. サービスが呼び出された際のライブラリー・リストを設定し、「Next」ボタンをクリック
今回は特に必要ないため、追加指定はなし

新規サービスの配置

ライブラリー・リストの指定 - ステップ7/9

Web サービスとして外部化する IBM i プログラムの機能性は、システム内の他の IBM i プログラムに依存することがあります。ライブラリーを指定しないと、デフォルトのライブラリー・リストが使用されます。

この Web サービスのライブラリー・リストの位置を指定:

- ライブラリー・リストのユーザー・ライブラリー部分の前にライブラリーを挿入
- ライブラリー・リストのユーザー・ライブラリー部分の末尾にライブラリーを挿入

ライブラリー・リスト項目: ?

ライブラリー名
<input type="radio"/> STUDENTRSC

追加

すべて除去

<参考>RESTサービスの作成手順（続き）

10. 受け渡すメタデータを選択し、「Next」ボタンをクリック
今回はデフォルトのまま、特に指定しない

受け渡すトランスポート情報の指定 - ステップ 8 / 9

Web サービス実装コードに渡されるトランスポート情報を指定します。 ?

トランスポート・メタデータの指定:

トランスポート・メタデータ	
<input type="checkbox"/>	QUERY_STRING
<input type="checkbox"/>	REMOTE_ADDR
<input type="checkbox"/>	REMOTE_USER
<input type="checkbox"/>	REQUEST_METHOD
<input type="checkbox"/>	REQUEST_URI
<input type="checkbox"/>	REQUEST_URL
<input type="checkbox"/>	SERVER_NAME
<input type="checkbox"/>	SERVER_PORT

HTTP ヘッダーの指定:

HTTP ヘッダー	
このテーブルに項目がありません。	

追加 すべて除去

<参考>RESTサービスの作成手順（続き）

11. サマリーを確認し、「Finish」ボタンをクリック
デプロイされると、自動的に開始される
下記は作成後の、プロパティ

サービス・プロパティ

一般 メソッド ライブラリー・リスト **Swagger** 接続プール 要求情報

サービス情報

リソース名: students

リソース説明: STUDENTRSC

URI パス・テンプレート: /

開始タイプ:

サービス・インストール・パス: /www/WSERVICE1/webservices/services/students

プログラム:

ベース・リソース URL: http://9.188.29.26:10073/web/services/students

このサービスのユーザー ID: または... 

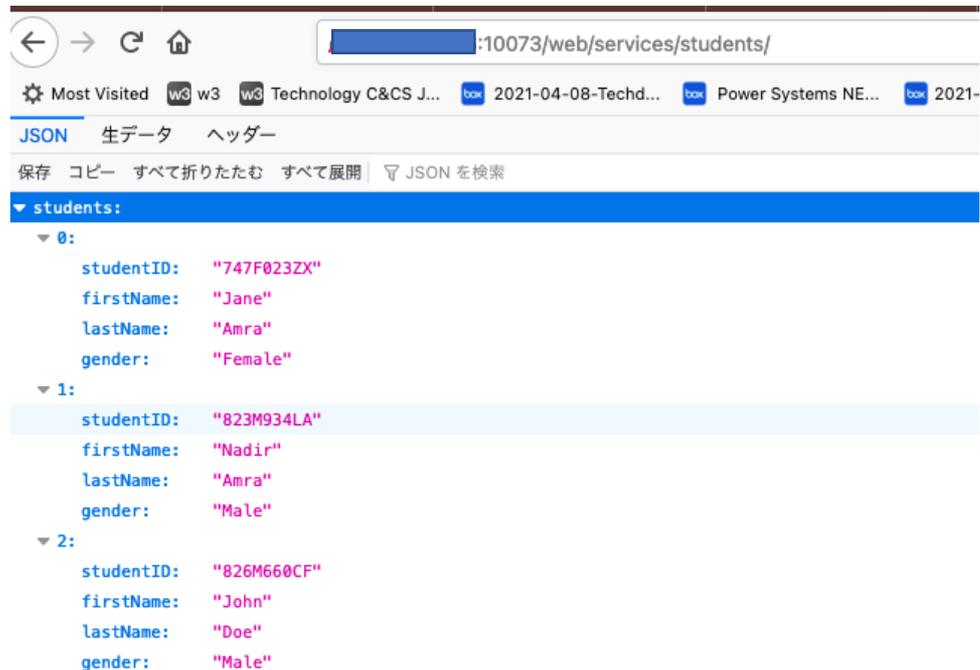
このユーザー ID に対して *USE 権限を持つようにサーバーのユーザー ID を更新します。

<参考>IAS上のRESTサービスの呼び出し

デフォルトではコンテキスト・ルートが/web/servicesとなる

例: `http://<system>:10073/web/services/students`

Webブラウザを用いて確認



The screenshot shows a web browser window with the address bar containing `...:10073/web/services/students/`. The browser's developer tools are open, displaying the JSON response for the `students` endpoint. The response is a JSON array with three elements, each representing a student with the following fields: `studentID`, `firstName`, `lastName`, and `gender`.

```
students:  
  0:  
    studentID: "747F023ZX"  
    firstName: "Jane"  
    lastName: "Amra"  
    gender: "Female"  
  1:  
    studentID: "823M934LA"  
    firstName: "Nadir"  
    lastName: "Amra"  
    gender: "Male"  
  2:  
    studentID: "826M660CF"  
    firstName: "John"  
    lastName: "Doe"  
    gender: "Male"
```

参考情報

IBM i アプリケーション開発

IBM Redbooks: Modernizing IBM i Applications from the Database up to the User Interface and Everything in Between

<http://www.redbooks.ibm.com/abstracts/sg248185.html?Open>

REST webサービス

IBM developerWorks: Building a REST service with integrated web services server for IBM i: Part 1

<https://developer.ibm.com/tutorials/i-rest-web-services-server1/>

IBM developerWorks: Building a REST service with integrated web services server for IBM i: Part 2

華氏を摂氏に変換するRESTサービスのサンプル

<https://developer.ibm.com/tutorials/i-rest-web-services-server2/>

IBM developerWorks: Building a REST service with integrated web services server for IBM i: Part 3

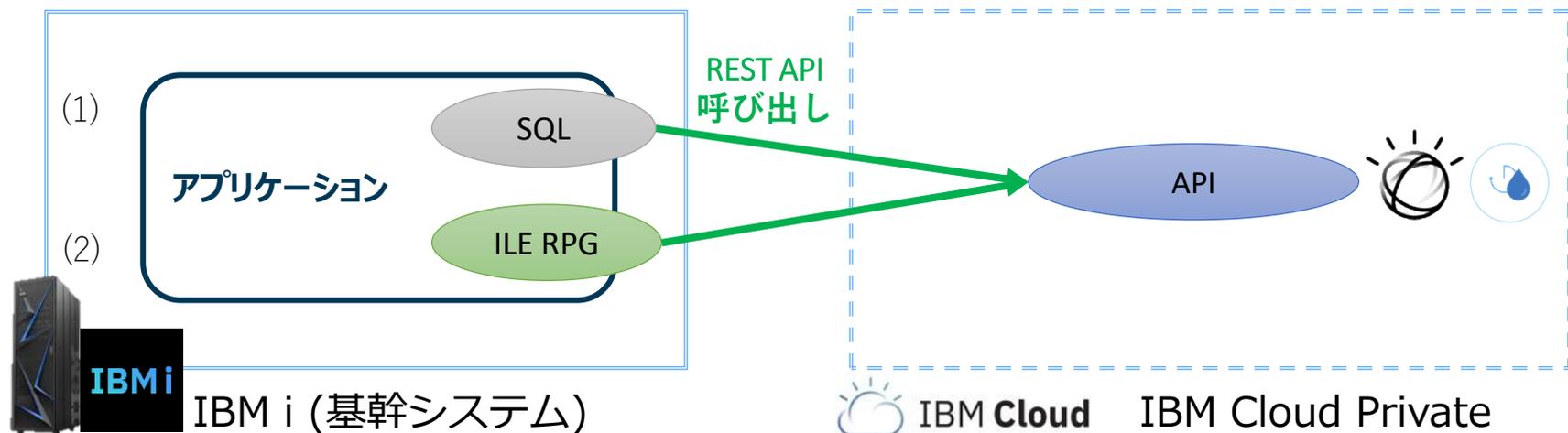
学生情報をメンテナンスするRESTサービスのサンプル

<https://developer.ibm.com/tutorials/i-rest-web-services-server3/>

2. [IBM i → 外部] IBM i アプリケーション側から REST API サービスを呼び出す

IBM i アプリケーション側から REST API サービスを呼び出す2つの方法

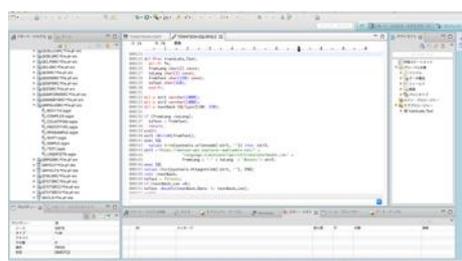
1. Db2 for i HTTP スカラー関数の活用
2. ILE RPG での Transport API の活用



IBM i 上のプログラムからクラウド・サービスをコール

Db2 for i のHTTPメソッド機能を活用したREST API呼び出し
「SYSTOOLS」スキーマでRESTのHTTPメソッドを呼び出すスカラー関数を提供

例) ILE RPGの組み込みSQLを使って、HTTPGETBLOB関数をREST APIをGETリクエストし、リソースを取得



開発ツール

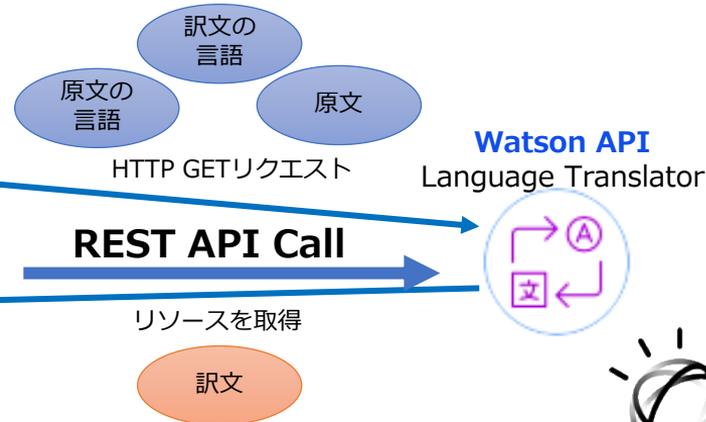
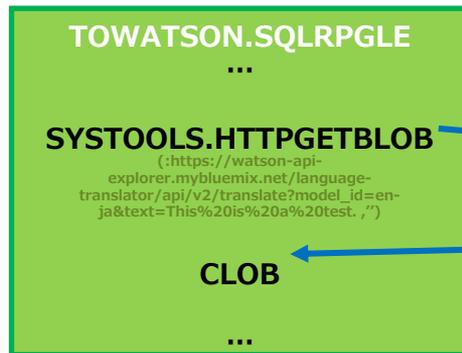
Rational Developer for i



IBM i



IBM Power



IBM Watson®

詳細はこちら : IBM i セミナー資料「ILE RPGからWatson APIを使用」
<https://ibm.ent.box.com/s/e0zcytg2r6iw4v6op7ujpxzvryd7yht2>

Db2 for i HTTPスカラー関数

IBM i のデータベース機能 (Db2 for i) でHTTPメソッドを呼び出す以下のスカラー関数をサポート

- httpGetClob
- httpGetBlob
- httpPostClob
- httpPostBlob
- httpPutClob
- httpPutBlob
- httpDeleteClob
- httpDeleteBlob

ライブラリーSYSTOOLSで提供しており、SQLで利用可能
アクセスしたいURLを入力として関数に引き渡し、CLOB、BLOBで値を受け取る

Db2 for i のHTTPメソッド機能の前提条件

- サポートOSバージョン
IBM i 7.2以降
- 前提ライセンス・プログラム
Java™ 1.6 以降 (5761-JV1 Option 11, 12, 14, 15)

【ご参考】エラー「SYSTOOLSのオブジェクトが見つからない」 場合の対処法

日本語環境の場合、下記前提PTFを適用

R720 SI64111

R730 SI64112

APAR SE66747 - OSP-DB UPDATES FOR SYSTOOLS NOT
HAPPENING IN CCSID 290

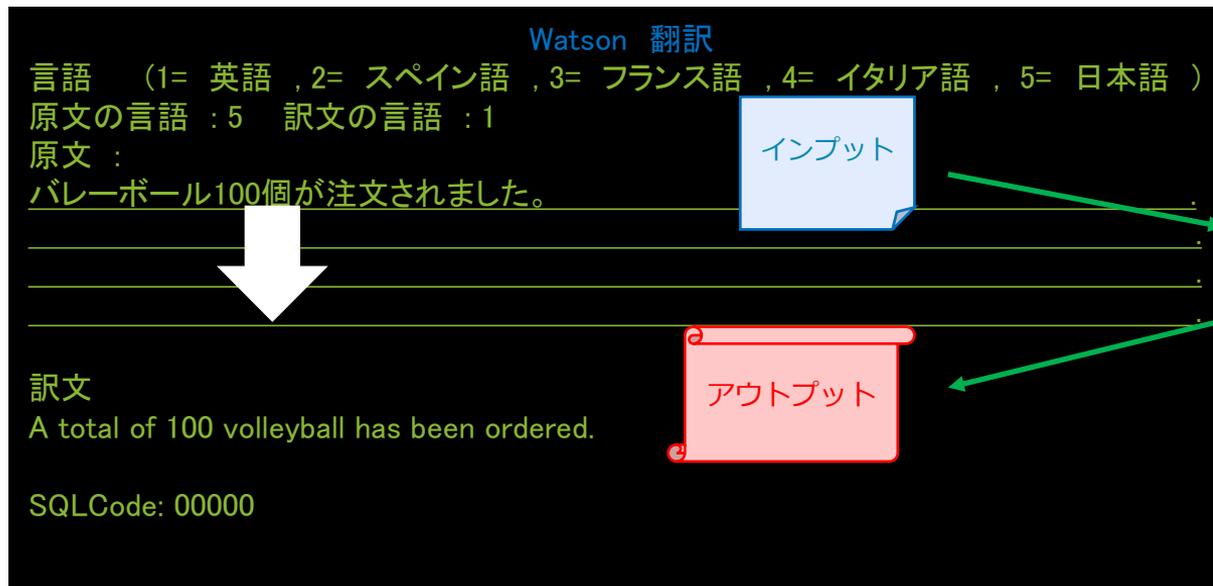
<https://www-304.ibm.com/support/docview.wss?uid=nas2SE66747>

注：PTF適用後、SYSTOOLSのオブジェクトが追加されていなかった場合、"CALL QSYS/QSQSYSIBM"を実行

Watson Language Translatorを使ったILE RPGアプリケーションの例

このILE RPGアプリケーションでは、Watson APIの「Language Translator」（翻訳機能）を使用

5250画面で入力した原文を「Language Translator」で翻訳し、5250画面に結果を出力



Language Translator API

Language Translator APIのURL構成

必須パラメーター

text : 例) This is a test.

model_id : 例) en-ja

Request URL

https://watson-api-explorer.mybluemix.net/language-translator/api/v2/translate?model_id=en-ja&text=This%20is%20a%20test.

上記URLの青太字部分をILE RPGアプリケーション内で3つのパラメータとして、REST呼び出しをします。

The screenshot shows the API interface with the following parameters and values:

Parameter	Value	Description	Parameter Type	Data Type
text	this is a test.	Input text in UTF-8 encoding. Multiple text query parameters indicate multiple input paragraphs, and a single string is valid input.	query	string
model_id	en-ja	The unique model_id of the translation model that is used to translate text. The model_id	query	string

Callouts from the image:

- text**: 原文を入力
- Model_id**: 原文の言語—訳文の言語を指定
- Request URL**: REST API呼び出しのURL
- Response Body**: 訳文の結果 (The response body contains the text: "これはテストです。")

Watson API を使った RPG サンプル・アプリケーション

- 表示装置ファイル

表示装置ファイル「TOWATSOND.DSPF」のサンプル・ソース

入力フィールド

FROMLANG：原文の言語

TOLANG：訳文の言語

FROMTEXT：原文

出力フィールド

TOTEXT：訳文

SQLCODEO：エラー時の

SQLコード

F3キー：プログラム終了

```

A          INDARA
A          DSPSIZ(24 80 *DS3)
A          CF03(03)
A          R DATAR
A          1 27'Watson翻訳'
A          COLOR(BLU)
A          3 1'言語 (1=英語, 2=スペイン語,'
A          3 35'3=フランス語, 4=イタリア語,'
A          3 67' 5=日本語)'
A          4 1'原文の言語:'
A          FROMLANG  1 0B 4 15VALUES(1 2 3 4 5)
A          EDTCDE(X)
A          4 20'訳文の言語:'
A          TOLANG    1 0B 4 34VALUES(1 2 3 4 5)
A          EDTCDE(X)
A          6 1'原文:'
A          FROMTEXT  320O B 8 1CHECK(LC)
A          13 1'訳文'
A          TOTEXT   320A O 14 1
A          19 1'SQLCode:'
A          SQLCODEO  5 0O 19 10
A          24 2'F3=Exit'
A          COLOR(BLU)

```

Watson API を使った RPG サンプル・アプリケーション - メイン・プロシージャ -

ILERPG 「TOWATSON.SQLRPGLE」のサンプル・ソース (1/3)

- A) データ構造は言語コードの配列を定義（コードは、画面上の原文/訳文言語（1=英語(en)、2=スペイン語(es)など）に入力された番号に対応）
- B) F3が押されるまで画面を表示しながらループ
- C) `transLate_Text()` サブプロシージャを呼び出し、原文言語コード、訳文言語コード、原文、および、訳文のパラメータを渡す

```
**free
ctl-opt option(*srcStmt: *noDebugIO) dftactGrp(*no);
dcl-F toWatsonD workstn(*ext) usage(*input: *output) indDs(WSI);
dcl-Ds WSI qualified;
      F3Exit ind pos(3);
end-Ds;

A) dcl-Ds *n;
   *n char(10) inz('enesfritja');
   lang char(2) dim(5) pos(1);
end-Ds;
exec SQL

B)   set option naming = *SQL;
     exfmt dataR;
     dow not WSI.F3Exit;

C)   transLate_Text(lang(fromLang) :
                   lang(toLang) :
                   fromText :
                   toText);
     SQLCodeO = SQLCODE;
     exfmt dataR;
endDo;
*inLR = *on;
```

Watson API を使った RPG サンプル・アプリケーション - transLate_Text() サブプロシージャ -

ILERPG 「TOWATSON.SQLRPGLE」のサンプル・ソース (2/3)

- A) RPGはCLOBデータ型を認識しないため、textBackをSQLTypeのCLOB変数として定義
- この定義はコンパイル後2つのサブフィールド textBack-Len (データの長さ) と textBack_Data (データ) を持つデータ構造になる
 - ※ 「systools.httpgetclob」はデータタイプ「CLOB」で返す
- B) 「systools.urlencode」を使用して、入力されたテキストをエンコード
- 問題の原因となる特殊文字 (& や < など) を同等のものに変換

```
dcl-Proc transLate_Text;
  dcl-Pi *n;
  fromLang char(2) const;
  toLang char(2) const;
  fromText char(320) const;
  toText char(320);
  end-Pi;
  dcl-s str1 varchar(1000);
  dcl-s str2 varchar(1000);
  dcl-s textBack SQLType(CLOB: 320);
  if (fromLang = toLang);
    toText = fromText;
    return;
  endif;
  A str1 =%trimR(fromText);
  B exec SQL
    values trim(systools.urlencode(:str1, '')) into :str2;
```

Watson API を使った RPG サンプル・アプリケーション - transLate_Text()サブプロシージャ 続き -

ILE RPG 「TOWATSON.SQLRPGLE」のサンプル・ソース (3/3)

- C) Watson APIをREST呼び出しし、翻訳を実行するURLを生成
- D) 「`systools.httpgetclob`」を使用して、Watson APIをREST呼び出し。
 - 戻り値は、先に定義した`textBack`に格納される
- E) 結果が返された場合は、データの長さ(`textBack_Len`)分のデータ(`textBack_Data`)を戻り値に代入

```
C str1 ='https://watson-api-explorer.mybluemix.net/' +  
    'language-translator/api/v2/translate?model_id=' +  
    fromLang + '-' + toLang + '&text='+ str2;  
  
exec SQL  
D values char(systools.httpgetclob(:str1, ''), 256)  
into :textBack;  
toText = *blanks;  
E if (textBack_Len > 0);  
toText =%subSt(textBack_Data: 1: textBack_Len);  
endif;  
return;  
end-Proc;
```

当プログラム例では、下記リンク先の内容、サンプル・アプリケーションを使用しています
Paul Tuohy 著 「RPG TALKS TO WATSON」 Copyright © 2017 IT Jungle
<https://www.itjungle.com/2016/09/27/fhg092716-story01/>

RPG TALKS TO WATSON

September 27, 2016 Paul Tuohy

Note: The code accompanying this article is available for download [here](#).

Yes, RPG can talk to Watson. No special software required, nothing to install, nothing to configure. You just need to be on V7R1, have the ability to use embedded SQL and write just a few lines of code—none of which are complicated. To see how it works, all you have to do is copy/paste the display file and RPG code in this article, compile and call.

On the off chance that you don't know what Watson is, Watson is the IBM computer that, in 2011, competed on the U.S. quiz show *Jeopardy!* against former winners Brad Rutter and Ken Jennings. Watson won by a mile.

REST利用APIのその他の活用例

– The Weather Company

気象データと基幹業務データを連携し、天候が影響を与えるビジネスに活用

StoreID	Name	City	Country	Lat	Lon
1	Store_1	Montpellier	France	43.614934	3.908162
2	Store_2	Rochester	USA	44.061604	-92.504532
3	Store_3	Osaka	Japan	34.4111	135.3112



JSON_TABLE

StoreID	Name	City	Country	Lat	Lon	Observation	Feel like
1	Store_1	Montpellier	France	43.614934	3.908162	快晴	11
2	Store_2	Rochester	USA	44.061604	-92.504532	所により曇り	17
3	Store_3	Osaka	Japan	34.4111	135.3112	小雨	21

【サンプルSQL】 Db2 for i テーブル上の店舗ごとの 現在の日と今後3日間の天気予報の取得

既存テーブルと結合しデータの
活用が可能

ILEプログラムで組み込み
SQLを使用しREST APIを
コール

```

1 SELECT storeid, name, city, country, lat, lon,
2       SUBSTR(dailydate, 1, 10) as "Date", DAILYOBS, DAILYTEMP
3 FROM storebot.store A,
4       JSON_TABLE(
5         SYSTOOLS.HTTPGETCLOB('https://[redacted]@' ||
6         'twcservice.mybluemix.net' ||
7         '/api/weather/v1/geocode/' ||
8         '/' || trim(char(cast(a.lat as decfloat))) ||
9         '/' || trim(char(cast(a.lon as decfloat))) ||
10        '/forecast/daily/3day.json?language=ja-JP&units=m',''),
11        '$'
12        COLUMNS(
13          NESTED '$.forecasts[*]'
14          COLUMNS ( DAILYDATE VARCHAR(22) CCSID 1208 PATH '$.day.fcst_valid_local' ,
15                   DAILYOBS VARCHAR(22) CCSID 1208 PATH '$.day.phrase_32char' ,
16                   DAILYTEMP VARCHAR(10) CCSID 1208 PATH '$.day.temp'
17          )
18        )
19        ) AS X
20 WHERE storeid = 1 OR storeid = 3;

```

STOREID	NAME	CITY	COUNTRY	LAT	LON	Date	DAILYOBS	DAILYTEMP
1	Store_1	東京	日本	35.67915	139.786883	2017-08-03	大体曇り	28
1	Store_1	東京	日本	35.67915	139.786883	2017-08-04	曇り	29
1	Store_1	東京	日本	35.67915	139.786883	2017-08-05	午前、雷雨	30
1	Store_1	東京	日本	35.67915	139.786883	2017-08-06	所により曇り	31
3	Store_3	Rochester	USA	44.061604	-92.504532	-	-	-
3	Store_3	Rochester	USA	44.061604	-92.504532	2017-08-03	雷雨	20
3	Store_3	Rochester	USA	44.061604	-92.504532	2017-08-04	大体晴れ	23
3	Store_3	Rochester	USA	44.061604	-92.504532	2017-08-05	午前、にわか	23

完了: 8 行を検索しました。

SELECT storeid, name, city, country, lat, lon, SUBSTR(dailydate, 1, 10) as "Date", DAILYOBS, DAILYTEMP

詳細はこちら：IBM i 基幹業務データと気象データの連携_201708.pdf

<https://ibm.ent.box.com/s/cv2i7opvvzr2bu2zb3n4t1ei0sidf19p>

Weather Company Data APIsを使用したサンプルSQL 実行時の前提条件

- IBM i からインターネットへの接続経路があること
 - HTTPsのポートが使用可能

JSON_TABLEを使用する場合

IBM i のライセンス・プログラム、グループPTFレベルが以下を満たすこと

IBM i 7.2 Db2 PTF Group PTF SI99702 level 14以降

IBM i 7.3 Db2 PTF Group PTF SI99703 level 3 以降

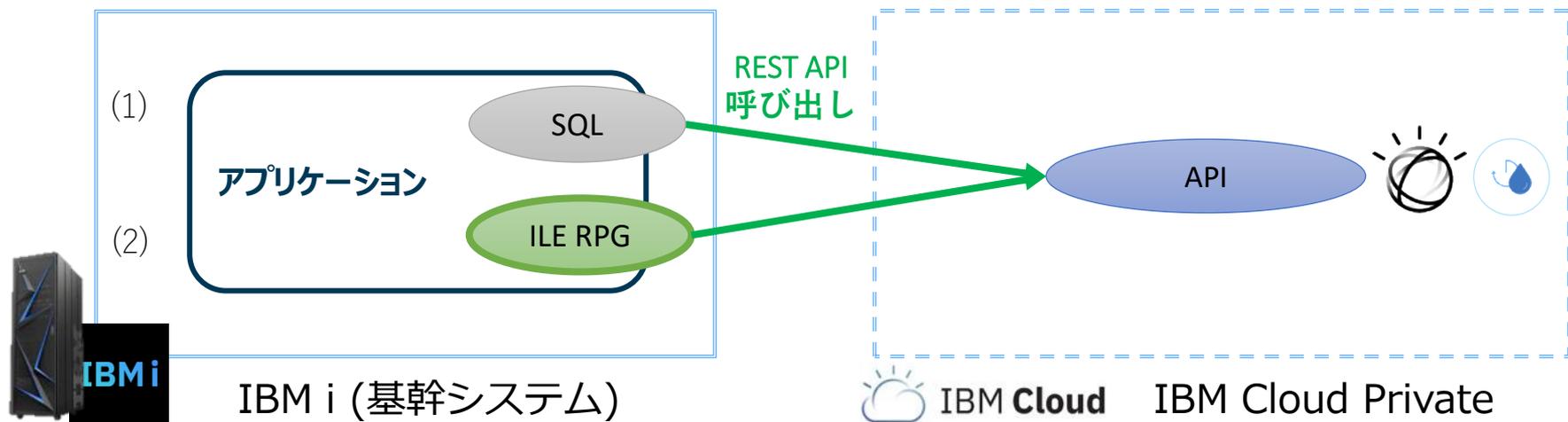
参考資料

<https://developer.ibm.com/articles/i-json-table-trs/>

2. [IBM i → 外部] IBM i アプリケーション側から REST APIサービスを呼び出す

IBM i アプリケーション側からREST APIサービスを呼び出す2つの方法

1. Db2 for i HTTPスカラー関数の活用
2. ILE RPGでのTransport APIの活用



ILE RPGでの統合Webサービス・クライアントAPIの活用

統合Webサービス・クライアント

Apache Axisを利用したSOAP webサービスをILEプログラムから呼び出すための機能として以前よりIBM iで提供しており、新たにREST webサービスを呼び出す機能を追加

Transport API

追加されたREST webサービスの呼び出し機能

Transport API	用途
axiscTransportCreate()	トランスポート・オブジェクトの生成
axiscTransportDestroy()	トランスポート・オブジェクトの破棄
axiscTransportReset()	トランスポート・オブジェクトを初期状態にリセット
axiscTransportSetProperty()	トランスポート属性への値のセット
axiscTransportGetProperty()	トランスポート属性値の取得
axiscTransportSend()	トランスポートを用いたデータの送信 (バッファ).
axiscTransportFlush()	バッファ・データのトランスポートを介した送付.
axiscTransportReceive()	トランスポートからの受信.
axiscTransportGetLastErrorCode()	直近で失敗したトランスポート処理のトランスポート・エラー・コードの取得
axiscTransportGetLastError()	直近で失敗したトランスポート処理のトランスポート・エラー・メッセージの取得

Transport APIの基本的な利用手順

Transport APIを利用したILE RPGからのREST webサービスの呼び出しの基本的な手順

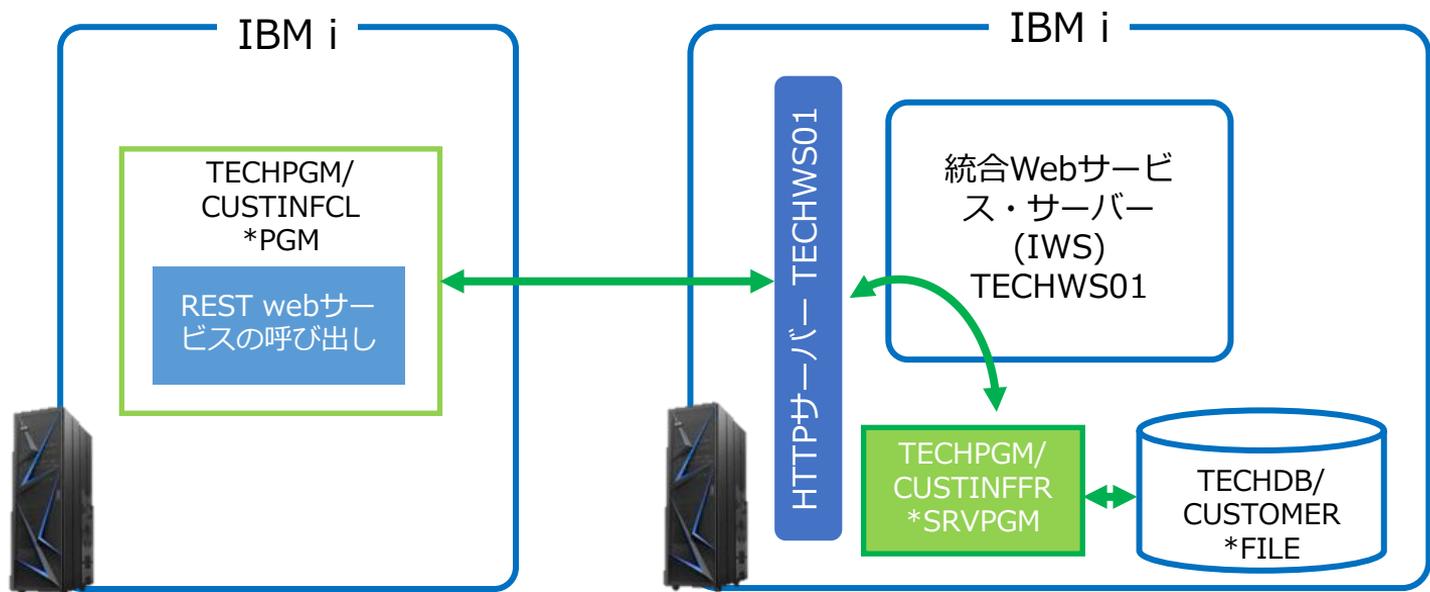


実体はライブラリーQSYSDIRにあるサービス・プログラムQAXIS10CC

上記APIのプロトタイプ定義が/QIBM/ProdData/OS/WebServices/V1/client/include/Axis.rpgleincとして提供されており、ILE RPGのソースに/COPYで取り込めば簡単に利用できる

【サンプル】 Transport APIの利用手順

学生情報を処理するサービス・プログラムCUSTINFFRのREST webサービスを他のIBM i上のILE RPGプログラムTECHPGM/CUSTINFCLから呼び出し



サンプル・プログラムの実行

TECHPGM/CUSTINFCLを
実行時の5250画面での表示
例

REST webサービス経由で
JSON形式のデータを取得
データがない場合はHTTPス
テータス・コード404を取得

コマンド入力

要求レベル : 4

すべての前のコマンドおよびメッセージ :

```
3 > CALL QCMD
4 > CALL PGM(TECHPGM/CUSTINFCL) PARM('00010315')
    DSPLY == 学生情報の取得 ==
    DSPLY 読み取りデータ・サイズ : 198
    DSPLY 読み取りデータ : {"customerDs":{"custCode":"0001031
    DSPLY 5","lastName":"西郷","firstName":"隆盛","zipCode
    DSPLY "":"892-0846","address":"鹿児島県鹿児島市加治屋町 99
    DSPLY 9-999 加治屋ハイツ 999 号室","telephone":"099-999
    DSPLY -9999"}}
    DSPLY HTTP ステータス・コード : 200
4 > CALL PGM(TECHPGM/CUSTINFCL) PARM('99999999')
    DSPLY == 学生情報の取得 ==
    DSPLY 読み取りデータなし
    DSPLY HTTP ステータス・コード : 404
```

終わり

コマンドを入力して、実行キーを押してください。
==>

サービス・プログラム QSYSDIR/QAXIS10CC のバインド

Transport APIを使用するために、提供されているプロトタイプ定義を取り込む
下記2つのどちらかの方法を選択

1. CRTRPGMODコマンドでモジュールを作成した後に、CRTPGMコマンドでQAXIS10CCをバインド
2. バインド・ディレクトリー TECHPGM/CUSTINFCL を作成、QSYSDIR/QAXIS10CC を登録し、ソースの制御仕様書で作成したバインド・ディレクトリーを指定
CRTBNDRPGコマンドで、QSYSDIR/QAXIS10CC をバインドした状態でプログラムを作成

以降、
2の方法を
ご紹介

コマンド入力

KIWI
要求レベル : 4

前のコマンドおよびメッセージ :

- > CRTBNDDIR BNDDIR(TECHPGM/CUSTINFCL)
バインド・ディレクトリー CUSTINFCL がライブラリー TECHPGM に作成された
- > ADDBNDDIRE BNDDIR(TECHPGM/CUSTINFCL) OBJ((QSYSDIR/QAXIS10CC))
ライブラリー TECHPGM のバインド・ディレクトリー CUSTINFCL に、1項目が追加され、0項目が追加されなかった。

<ソース> 制御仕様書でバインド・ディレクトリーの指定

mainプロシージャをメイン・プロシージャとして定義
制御仕様書で作成したバインド・ディレクトリーを指定

```
**free
// 全体制御 =====
ctl-opt dftactgrp(*no);
ctl-opt main(main);
ctl-opt pgminfo(*pcml:*dclcase);
ctl-opt bnddir('TECHPGM/CUSTINFCL');

// システム提供のプロトタイプ定義をコピー
/COPY /QIBM/ProdData/OS/WebServices/V1/client/include/Axis.rpgleinc

// グローバル変数
dcl-s rc int(10) inz(0);
dcl-s tranHandle pointer;

// -----
// メイン・プロシージャ
// 学生情報のREST webサービスを呼び出して取得
// -----
```

<ソース> mainプロシージャ (1/3)

変数 uri にREST webサービスのURIを指定
サンプル・プログラムCUSTINFCLは学生
コードを入力パラメーターとし、URIに入
力された値をつなげる

QAXIS10CCはCPPLE (ILE C++) で作られ
ているため、変数値は忘れずにヌル終端
(x'00'で終わる) にする。

指定したURIを入力パラメーターとし、API
“axiscTransportCreate”でトランスポート
・ハンドルを生成

```
dcl-proc main;
  dcl-pi *n;
    p_custCode char(8) const;
  end-pi;

  dcl-s uri char(200);
  dcl-s response char(32768);
  dcl-s propBuf char(100);
  dcl-s header pointer;
  dcl-s bytesRead int(10) inz(0);

  // トレース取得時はコメント解除
  // axiscAxisStartTrace('/tmp/axistransport.log': *NULL);

  // グローバル変数の初期化
  clear rc;
  clear tranHandle;

  // 学生情報取得のURIを指定
  uri = 'http://systemname:20080/web/services/customers/' + p_custCode + x'00';

  // HTTPトランスポート・ハンドルの生成
  tranHandle = axiscTransportCreate(uri:AXISC_PROTOCOL_HTTP11);
  if (tranHandle = *NULL);
    displayMessage('TransportCreate() failed');
    return;
  endif;
```

<ソース> mainプロシージャ (2/3)

属性値としてHTTPメソッドをAPI
“axiscTransportSetProperty”を使用して指定
今回は指定した学生コードの情報照会のため、
HTTPメソッドとしてはGETを指定

GETでの情報取得で、REST webサービス側
にデータを送信しないため、今回はAPI
“axiscTransportSend”は使用しない

属性の指定後 API “axiscTransportFlush”を用
いてHTTPリクエストを送信
リクエストの送信後は API
“axiscTransportReceive”を使用してレスポ
ンスを受け取る
受け取るデータが存在する限りループして受
け取る

```
// HTTPメソッドを定義し、トランスポートの属性に指定
propBuf = 'GET' + X'00';
axiscTransportSetProperty (tranHandle: AXISC_PROPERTY_HTTP_METHOD: %addr (propBuf));

displayMessage ('==学生情報の取得==');

// HTTPリクエストを送信
rc = axiscTransportFlush (tranHandle);
if (rc = -1);
    checkError ('TransportFlush()');
    return;
endif;

// HTTPレスポンスからデータを受け取り、画面に表示
rc = axiscTransportReceive (tranHandle:%addr (response):%size (response):0);
if (rc = 0);
    displayMessage ('読み取りデータなし');
else;
    dow rc > 0 AND bytesRead < %size (response);
        bytesRead = bytesRead + rc;
        rc = axiscTransportReceive (tranHandle:
                                        %addr (response)+bytesRead:
                                        %size (response)-bytesRead:
                                        0);
    enddo;
endif;
```

<ソース> mainプロシージャ (3/3)

今回の例では、受け取ったデータの画面表示はdisplayMessageプロシージャを別途作成、利用している
データ表示が完了すると、API “axiscTransportDestroy”を用いて不要になったトランスポート・ハンドルを破棄し、処理を終了

```
if (rc = -1);
  checkError ('TransportReceive()');
elseif (bytesRead > 0);
  displayMessage(' 読み取りデータ・サイズ: ' + %char(bytesRead));
  displayMessage(' 読み取りデータ: ' + response);
endif;

if (rc > -1);
  rc = axiscTransportGetProperty(tranHandle
                                :AXISC_PROPERTY_HTTP_STATUS_CODE
                                :%addr(header));

  if (rc = -1);
    checkError ('TransportGetProperty()');
  else;
    displayMessage(' HTTPステータス・コード: ' + %str(header));
  endif;
endif;

// トランスポート・ハンドルの破棄
axiscTransportDestroy(tranHandle);

return;
end-proc;
```

<ソース>メッセージ表示プロシージャ

displayMessageプロシージャを作成し、
受け取ったデータを画面に表示

```
// -----  
// メッセージ表示プロシージャ  
// -----  
dcl-proc displayMessage;  
  dcl-pi *n;  
    p_msg varchar(5000) const;  
  end-pi;  
  
  dcl-c lenPartMsg const(52);  
  
  dcl-s i int(10) inz(0);  
  dcl-s numOfLine int(10) inz(0);  
  dcl-s pos int(10) inz(0);  
  dcl-s len int(10) inz(0);  
  dcl-s partMsg char(lenPartMsg) inz(*blank);  
  
  numOfLine = %len(%trim(p_msg)) / lenPartMsg + 1;  
  
  for i = 1 to numOfLine;  
    pos = (lenPartMsg * (i - 1)) + 1;  
    if (i = numOfLine);  
      len = %len(p_msg) - lenPartMsg * (i - 1);  
    else;  
      len = lenPartMsg;  
    endif;  
  
    partMsg = %subst(p_msg:pos:len);  
    dsply partMsg;  
  endfor;  
end-proc;
```

<ソース>エラー・チェック・プロシージャー

- ・ Transport API使用時のエラーやHTTPステータス・コードのハンドリングを行うプロシージャーとしてcheckErrorプロシージャーを使用

- ・ トランスポート・エラー・コード、またコードに関するメッセージを取得するAPIとしてそれぞれ
axiscTransportGetLastErrorCode、
axiscTransportGetLastErrorが用意されている

- ・ HTTPステータス・コードの取得は
axiscTransportGetPropertyで属性から読み取る形で実装している

```
// -----  
// エラー・チェック・プロシージャー  
// -----  
dcl-proc checkError;  
  dcl-pi *n;  
    p_msg varchar (5000) const;  
  end-pi;  
  
  dcl-s axisCode int(10);  
  dcl-s statusCode pointer;  
  dcl-s rc int(10);  
  
  // Axisエラー・コードを取得  
  axisCode = axiscTransportGetLastErrorCode(tranHandle);  
  
  // 画面にエラー・コードおよびメッセージを表示  
  displayMessage(p_msg + ' call failed: ' +  
                %char(axisCode) + ':' +  
                %str(axiscTransportGetLastError(tranHandle)));  
  
  // HTTPでのエラーの場合はHTTPステータス・コードを取得、表示  
  if (axisCode = EXC_TRANSPORT_HTTP_EXCEPTION);  
    rc = axiscTransportGetProperty(tranHandle  
                                  :AXISC_PROPERTY_HTTP_STATUS_CODE  
                                  :%addr(statusCode));  
  
    displayMessage(' HTTPステータス・コード: ' + %str(statusCode));  
  endif;  
end-proc;
```

Transport APIの前提条件

前提PTF

IBM i 7.2: SI76961, SI77246

IBM i 7.3: SI76960, SI77245

参考情報

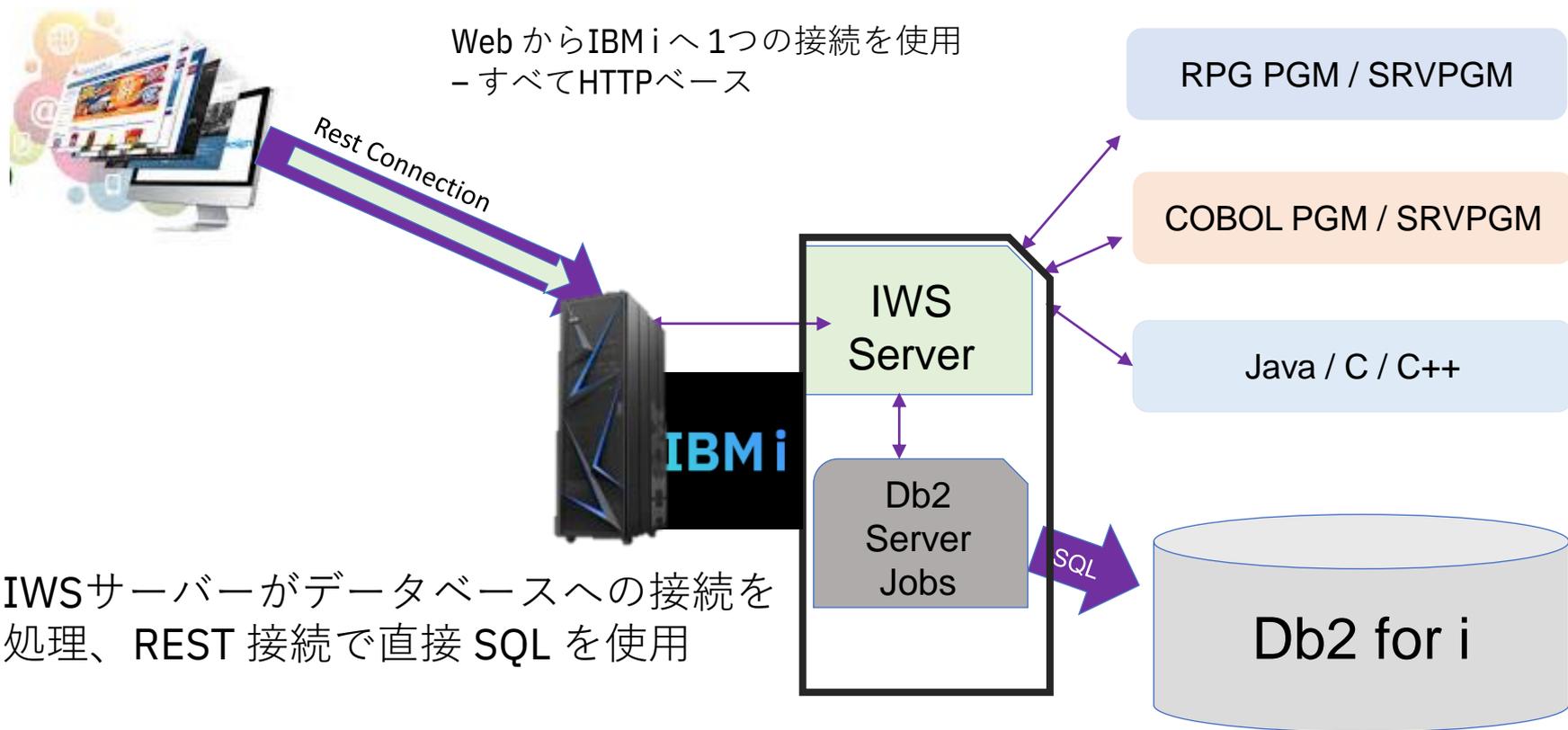
<developerWorks: Send and receive user-defined SOAP and REST messages from RPG>

<https://developer.ibm.com/articles/i-send-receive-user-defined-soap-rest-messages-trs/>

2. [外部 → IBM i] DB照会をSQL記述で REST APIサービス化

(IBM i 7.4 又は、IBM i 7.3 TR6以降の機能になります)

REST API を使用した新しい SQL アクセス



DB照会をSQL記述でRESTサービス化する手順 (1)

これは、Navigator for iのApplication ServerメニューにあるWeb Serviceウィザードの「新規サービスの配置」から作成できる。

IBM i 7.4とIBM i 7.3 TR6ではウィザードステップのSpecify Web service implementationパラメータで「WebサービスとしてのSQL」を選択することができる

The screenshot shows the IBM Web Administration for i interface. The main content area displays the 'New Service Configuration' wizard for 'WSERVICE1 - V2.6 (Web サービス)'. The current step is 'Specify Web service implementation:'. The 'Web service type' is set to 'REST'. Under 'Specify database properties that will be used to process SQL statements:', the following values are entered:

- Database system: localhost
- Default schema: *LIBL
- Naming convention: *SQL
- Library list: *LIBL

The 'Specify Web service implementation' section has two radio buttons: 'ILE Program Object' (unselected) and 'Web サービスとしての SQL' (selected). At the bottom of the wizard, there are buttons for '戻る' (Back), '次へ' (Next), and 'キャンセル' (Cancel).

DB照会をSQL記述でRESTサービス化する手順 (2)

では下記に示すDb2 for iのサンプルデータ (QEOLライブラリーのTOKMSPテーブルを例に、外部からデータを参照するREST APIサービスの構成手順を追って紹介する。

TKBANG	TKNAKN	TKNAKJ	TKADR1	TKADR2	TKTIKU	TKPOST	TKTELE	TKGURI	TKI
01010	アイ リヨカ	阿井旅館	東京都渋谷区	桜ヶ丘 2 9	02	150	03-504-9293	698500	
01020	アイ ヲギ ヲ	阿井工業	東京都渋谷区	渋谷 1-3	02	150	03-535-5951	452800	
01030	アイカ ヲギ ヲ	相川工業	東京都世田谷区	若林 4-2 4	06	154	03-964-6406	136200	
01040	アイ リヨカヤ	阿井旅行社	東京都品川区	東五反田 1-1 1	01	140	03-934-7946	3028300	1
01050	アイ ショク ヲK.K	阿井食品K. K	東京都荒川区	荒川 5-4 2	17	116	03-360-6701	541600	
01060	アイ シド ヲカヤ	阿井自動車	東京都港区	芝公園 1-2	14	105	03-860-2932	541100	
01070	アイカ カラ	相川カメラ	東京都新宿区	四谷 1-1 6	03	160	03-354-3018	367900	
01080	アイカ ヲカヤK.K	相川広告K. K	東京都渋谷区	広尾 3-9	02	150	03-368-6366	318000	
01090	アイカ デンキK.K	相川電機K. K	東京都北区	滝の川 7-1 7	14	114	03-749-6271	877100	
01100	アイカ ガクヤ	相川楽器店	東京都港区	虎ノ門 3-2 1	22	105	03-922-1801	501300	
01110	アイカ セツインジ ヲ	相川設計事務所	東京都文京区	本郷 1-2 5	13	113	03-909-9124	446300	
01120	アイカ ショウガ	相川商事	東京都葛飾区	新小岩 2-2 0	21	124	03-856-7896	315600	
01130	アイ ヲギ イヤ	愛工芸社	東京都港区	南麻布 1-1 0	14	106	03-187-2308	635400	
01140	アイカ トリヤ	相川塗装店	東京都練馬区	高野台 2-1 4	12	177	03-397-7993	159600	
01150	アイカ ヲカK.K	相川運輸K. K	東京都杉並区	西荻 3-2 3	04	167	03-920-5933	84200	
01160	アイカ ビヨウ	相川病院	東京都目黒区	目黒 4-2 3	15	153	03-897-8588	146900	
01170	アイカ カケイジ ヲ	相川会計事務所	東京都中央区	東銀座 7-1 5	08	103	03-895-5141	464100	
01180	アイカ ショウカイ	相川商会	東京都調布市	若葉 2-2 2	28	182	03-231-1579	674800	
01190	アイカ カセツK.K	相川建設K. K	東京都江東区	罐峨 1-1 1	23	135	03-751-8216	899000	
01200	アイ イヤカギ ヲカK.K	阿井印刷工業K. K	東京都千代田区	神田神保町 1-1	09	101	03-384-3427	149100	
01210	アイヤ エレクトロニクス ヲ	アイシーエレクトロ	東京都台東区	東上野 3-2 4	07	110	03-353-7744	1282500	
01220	アイカ デパート	相川デパート	東京都中央区	入船 2-5	08	104	03-731-1670	242600	
01230	アイカ ヲカヤ	相川広告社	東京都新宿区	西新宿 4-3 1	03	160	03-783-4171	487800	
01240	アイカ 納メK.K	相川包装K. K	東京都新宿区	大久保 2-1 8	03	160	03-893-2399	227000	
01250	アイデア サギ ヲK.K	アイデア産業K. K	東京都渋谷区	神宮前 1-1 1	02	150	03-318-2857	70200	
01260	アイカ ヲギ ヲK.K	相川産業K. K	東京都北区	田端 1-1 5	22	114	03-249-6244	46400	
01270	アイ シド ヲカヤ ヒヨギ ヲ	阿井自動車部品工業	東京都目黒区	大橋 2-7	15	153	03-886-4312	4220900	2
01280	アイデア キョウカK.K	アイデア協会K. K	東京都杉並区	荻窪 4-2 1	04	167	03-471-8810	1799100	1
01290	アイ デンキヨウカイ	愛電器商会	東京都杉並区	上高井戸 1-5	04	168	03-190-4691	1554900	
01300	アイ ヤコカヤ	阿井鉄工所	東京都豊島区	池袋 3-3 1	10	171	03-656-9641	479100	
01310	アイカ センター	相川センター	東京都江東区	新大塚 1-4	22	136	03-512-0478	705200	

DB照会をSQL記述でRESTサービス化する手順 (3)

サービス化の対象とするテーブルのライブラリ(QEOL)を指定

新規サービスの配置

Web サービス・タイプの指定 - ステップ 1 / 9

Welcome to the Deploy New Service wizard. This wizard helps you create Web services using IBM i objects and data. SOAP ベースの Web サービスでは、クライアント要求がリソース・メソッドによって処理され、交換されるメッセージの形式はリソース自体によって定義され、REST ベースの Web サービスでは、クライアント要求がリソース・メソッドによって処理され、交換されるメッセージの形式はリソース自体によって定義され、明確に定義されたインターフェースでは、クライアント要求がリソース・メソッドによって処理され、交換されるメッセージの形式はリソース自体によって定義され、

Web サービス・タイプの指定: REST  

Specify Web service implementation:

- ILE Program Object
- Web サービスとしての SQL

Specify database properties that will be used to process SQL statements: 

Database system: localhost  または... 

Default schema: QEOL  または... 

Naming convention: *SQL 

Library list: *LIBL

DB照会をSQL記述でRESTサービス化する手順 (4)

リソース名とURIパス・テンプレートを指定する

新規サービスの配置

サービス名の指定 - ステップ 2 / 9

外部化される Web サービスはリソースです。URI パス・テンプレートが着信 HTTP 要求のマッチング・パターンを識別します。 / されるのかをさらに制限する正規表現を含むことのできる 1 つ以上のテンプレート・パラメーターとすることができます。 ?

リソース名:

サービス記述:

URI パス・テンプレート: 例: /temperature, /temperature/{temp:\d+}

DB照会をSQL記述でRESTサービス化する手順 (5)

下記は2つのプロシージャを定義している例。`getAll`プロシージャは全レコードデータを返すメソッドとして定義している。`getByID`プロシージャはTKBANGを指定して特定のレコードを返すメソッドとして定義している。

SQL ステートメントの指定: ?

	プロシージャ名	SQL ステートメント/パラメーター名	使用法	データ・タイプ
<input checked="" type="checkbox"/>	getAll	select * from TOKMSP;		
<input checked="" type="checkbox"/>	getbyid	SELECT * FROM TOKMSP where TKBANG = ?;		

戻る 次へ キャンセル

DB照会をSQL記述でRESTサービス化する手順 (6)

ウィザードのステップ5ではSQL処理をカスタマイズする。ここではgetByIDプロシージャのSQL result typeを「Single-row result set」に変更している。この記事の例では、getALLプロシージャを含め、それ以外のパラメータはデフォルトの設定。

新規サービスの配置

Specify SQL Information - ステップ 5 / 8

Customize how each procedure will process SQL statements. For query statements, this includes the type of result sets that may be returned and h

プロシージャ名: getall
SQL Statement: select * from TOKMSP;

SQL result type: Multi-row result set

Trim mode for output fields: Trailing

SQL state information in response: On errors

Treat warnings as SQL Errors: はい

User-defined error message:

HTTP status code on SQL success: 200 または...

HTTP status code on SQL failure: 500 または...

新規サービスの配置

Specify SQL Information - ステップ 5 / 8

Customize how each procedure will process SQL statements. For query statements, this includes the type of result sets that may be returned and h

プロシージャ名: getbyid
SQL Statement: SELECT * FROM TOKMSP where TKBANG = ?;

SQL result type: Single-row result set

Trim mode for output fields: Trailing

SQL state information in response: On errors

Treat warnings as SQL Errors: はい

User-defined error message:

HTTP status code on SQL success: 200 または...

HTTP status code on SQL failure: 500 または...

DB照会をSQL記述でRESTサービス化する手順(7)

ウィザードのステップ6では各プロシージャのメソッド情報を指定する。**getAll**プロシージャはデフォルトのパラメータのままとし、**getByID**プロシージャは下記のように、入力パラメータを設定してレコード番号を特定できるように定義する。

新規サービスの配置

リソース・メソッド情報の指定 - ステップ 6 / 8

プロシージャはリソース・メソッドにマップされます。HTTP 要求メソッドをリソース・メソッドにマッピングする

プロシージャ名:

リソースの URI パス・テンプレート:

HTTP 要求メソッド:

メソッドの URI パス・テンプレート: または...

HTTP header information:

許可される入力メディア・タイプ: または...

返される出力メディア・タイプ:

Identifier for input wrapper element: または...

Identifier for output wrapper element: または...

プロシージャはリソース・メソッドにマップされます。HTTP 要求メソッドをリソース・メソッドにマッピングすることによって、各リソース。

プロシージャ名:

リソースの URI パス・テンプレート:

HTTP 要求メソッド:

メソッドの URI パス・テンプレート: または...

HTTP header information:

許可される入力メディア・タイプ: または...

返される出力メディア・タイプ:

Identifier for input wrapper element: または...

Identifier for output wrapper element: または...

入力パラメーターをラップするかどうか:

入力パラメーターをラップする

入力パラメーターをラップしない

入力パラメーター・マッピング:

パラメーター名	データ・タイプ	入力ソース	ID	デフォルト値
PARM00001	CHAR	<input type="text" value="*PATH_PARAM"/>	<input type="text" value="id"/>	<input type="text" value="*NONE"/> または...

DB照会をSQL記述でRESTサービス化する手順(8)

ウィザードのステップ7ではサービスの実行ユーザーIDを指定する

[WSERVICE1](#) > [配置済みサービスの管理](#) > [新規サービスの配置](#)

新規サービスの配置

このサービスのユーザー ID を指定 - ステップ 7 / 8

The service requires an IBM i user ID to run the Web service business logic. The user ID must have the nec

このサービスのユーザー ID を指定: ?

- サーバーのユーザー ID を使用
- 既存のユーザー ID を指定
- 認証済みユーザー ID を使用

DB照会をSQL記述でRESTサービス化する手順 (9)

ウィザードのステップ8の要約画面で設定内容を確認して完了すると、サービスが起動する

新規サービスの配置

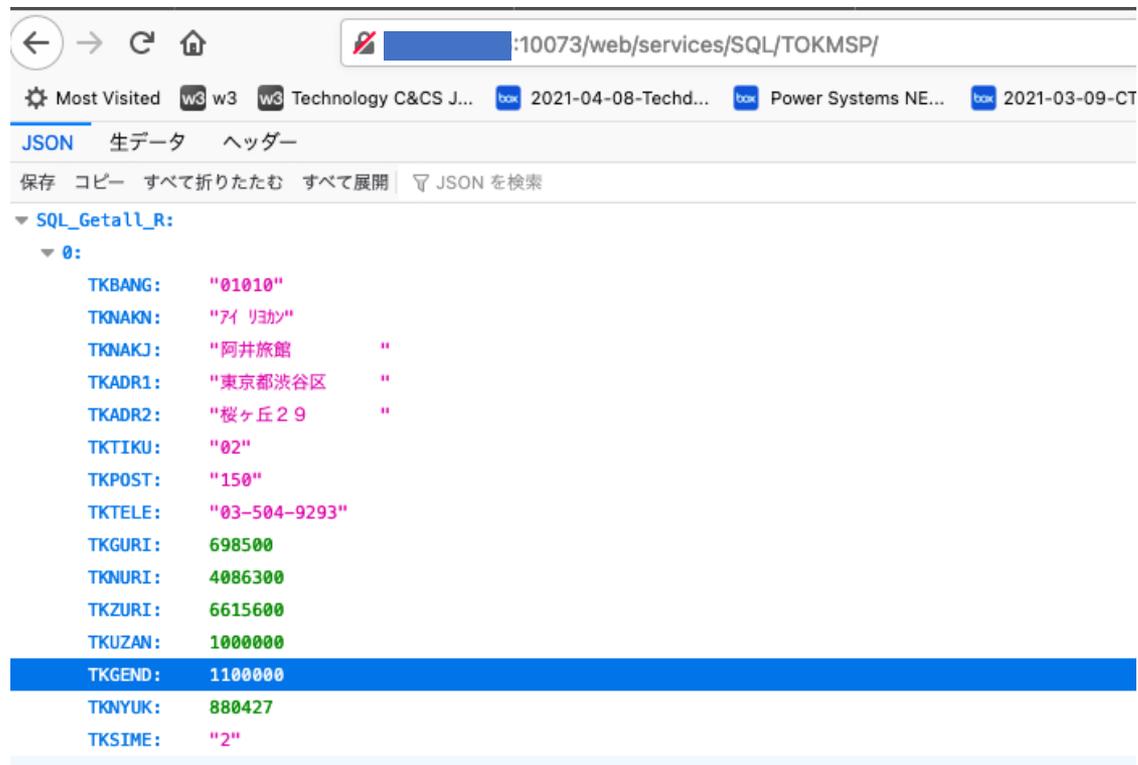
要約 - ステップ 8 / 8

「完了」をクリックすると、Web サービスが配置されます。

サービス	JDBC Properties	メソッド
リソース名:	SQL	
リソース説明:	SQL	
サービス・インストール・パス:	/www/wservice1/webservices/services/SQL	
URI パス・テンプレート:	/TOKMSP	
サービスのユーザー ID:	*SERVER (QWSERVICE)	

DB照会をSQL記述でRESTサービス化する手順 (10)

生成したREST APIサービスはWebブラウザでアクセスして稼働を確認できる。Db2 for iのデータをJSON形式で受け取ることができる。下記はGETALLメソッド。



The screenshot shows a web browser interface with the following details:

- Address bar: `...:10073/web/services/SQL/TOKMSP/`
- Page title: `JSON`
- Navigation: `生データ`, `ヘッダー`
- Actions: `保存`, `コピー`, `すべて折りたたむ`, `すべて展開`, `JSON を検索`
- Response content (JSON):

```
SQL_Getall_R:  
  0:  
    TKBANG: "01010"  
    TKNAKN: "アイ リオン"  
    TKNAKJ: "阿井旅館" "  
    TKADR1: "東京都渋谷区" "  
    TKADR2: "桜ヶ丘29" "  
    TKTIKU: "02"  
    TKPOST: "150"  
    TKTELE: "03-504-9293"  
    TKGURI: 698500  
    TKNURI: 4086300  
    TKZURI: 6615600  
    TKUZAN: 1000000  
    TKGEND: 1100000  
    TKNYUK: 880427  
    TKSIME: "2"
```

DB照会をSQL記述でRESTサービス化する手順 (11)

生成したREST APIサービスはWebブラウザでアクセスして稼働を確認できる。Db2 for iのデータをJSON形式で受け取ることができる。下記はGETBYIDメソッド。



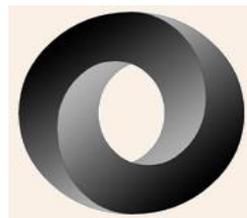
```
JSON 生データ ヘッダー
保存 コピー すべて折りたたむ すべて展開 🔍 JSONを検索
▼ SQL_Getbyid_R:
TKBANG: "01010"
TKNAKN: "アイ リヨク"
TKNAKJ: "阿井旅館" ""
TKADR1: "東京都渋谷区" ""
TKADR2: "桜ヶ丘29" ""
TKTIKU: "02"
TKPOST: "150"
TKTELE: "03-504-9293"
TKGURI: 698500
TKNURI: 4086300
TKZURI: 6615600
TKUZAN: 1000000
TKGEND: 1100000
TKNYUK: 880427
TKSIME: "2"
```

3. JSONデータと Db2 for iデータの変換機能

Db2 for i と JSON オブジェクトの双方向データ交換

Webアプリケーションで広く使われる JSON オブジェクトへの対応を強化

- Db2 for i から JSON オブジェクトをSQLで生成する機能
- JSONオブジェクトからDb2 for iへのRDBデータへ挿入する機能



1. JSON 出力機能
JSON_OBJECT
JSON_ARRAY



2. JSON テーブル機能
JSON_TABLE

CUSTOMER

```
{"BANGO": "01010", "CUSTNAME": "アイリヨカン",  
  "CUSTNAMEJ": "阿井旅館" }
```

```
SELECT JSON_OBJECT (KEY 'BANGO' VALUE TKBANG,  
                    KEY 'CUSTNAME' VALUE TRIM(TKNAKN),  
                    KEY 'CUSTNAMEJ' VALUE TRIM(TKNAKJ)  
                    ) AS CUSTOMER  
FROM QEOL.TOKMSP LIMIT 1;
```

1. JSON形式への出力機能

JSON_OBJECT (JSONオブジェクトの生成機能)

```
SELECT JSON_OBJECT (KEY 'BANGO' VALUE TKBANG,  
KEY 'CUSTNAME' VALUE TRIM(TKNAKN),  
KEY 'CUSTNAMEJ' VALUE TRIM(TKNAKJ)  
) AS CUSTOMER  
FROM QEOL.TOKMSP LIMIT 5;
```



(例)

QEOLライブラリー
のTOKMSPファイル



```
SELECT JSON_OBJECT (KEY 'BANGO' VALUE TKBANG, KEY 'CUSTNAME' VALUE TRIM(TK
```

CUSTOMER
{"BANGO": "01010", "CUSTNAME": "アイ リヨク", "CUSTNAMEJ": "阿井旅館" }
{"BANGO": "01020", "CUSTNAME": "アイ コク'ヨク", "CUSTNAMEJ": "阿井工業" }
{"BANGO": "01030", "CUSTNAME": "アイカ'コク'ヨク", "CUSTNAMEJ": "相川工業" }
{"BANGO": "01040", "CUSTNAME": "アイ リヨク'ヤ", "CUSTNAMEJ": "阿井旅行社" }
{"BANGO": "01050", "CUSTNAME": "アイ ショク'ウK.K", "CUSTNAMEJ": "阿井食品K. K" }

JSON_ARRAY (JSON配列の生成機能)

```
SELECT JSON_ARRAY (  
    JSON_OBJECT(KEY 'BANGO' VALUE TKBANG),  
    JSON_OBJECT(KEY 'CUSTNAME' VALUE TRIM(TKNAKN)),  
    JSON_OBJECT(KEY 'CUSTNAMEJ' VALUE TRIM(TKNAKJ))  
    ) AS CUSTOMER  
FROM QEOL.TOKMSP LIMIT 5;
```



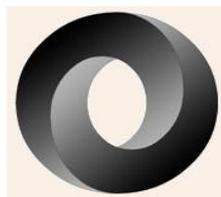
CUSTOMER
[{"BANGO": "01010"}, {"CUSTNAME": "アイ リヨカン"}, {"CUSTNAMEJ": "阿井旅館"}]
[{"BANGO": "01020"}, {"CUSTNAME": "アイ コギ ヨク"}, {"CUSTNAMEJ": "阿井工業"}]
[{"BANGO": "01030"}, {"CUSTNAME": "アイカワ コギ ヨク"}, {"CUSTNAMEJ": "相川工業"}]
[{"BANGO": "01040"}, {"CUSTNAME": "アイ リヨカン"}, {"CUSTNAMEJ": "阿井旅行社"}]
[{"BANGO": "01050"}, {"CUSTNAME": "アイ ショク うえん. K"}, {"CUSTNAMEJ": "阿井食品 K. K"}]

(例)

QEOLライブラリー
のTOKMSPファイル

2. JSONテーブル機能

- JSONをRDB変換処理することができるSQL関数
JSONデータをRDBの表形式に変換
変換された表をSQLで処理可能
- SELECTステートメントのFROMで使用
IFS上のJSONファイル进行处理
Webサービス等で直接取得して処理



JSON



2. JSON テーブル機能
JSON_TABLE



列1	列2	列3

JSONテーブルの利用方法

基本構文

```
SELECT *FROM JSON_TABLE (  
    JSON形式データ,  
    'SQL JSON パス形式'  
    COLUMNS (カラム名 データタイプ PATH カラム・パス  
    )  
    ) AS X
```

- Db2 for i HTTP機能を使用
SYSTOOLS.HTTPGETCLOBでJSON形式データを取得

(例)

```
SYSTOOLS.HTTPGETCLOB(  
'http://express.heartrails.com/api/json?method=getStations&line=JR山手線',null)
```

(参考) JSONテーブルでの駅名取得 (1)

HeartRails Express API (駅名などをAPI でJSON形式で公開)

<http://express.heartrails.com/api/json?method=getStations&line=JR山手線>

```

← → ↻ 保護されていない通信 | express.heartrails.com/api/json?method=getStations&line=JR山手線 ☆ 👤 :
{"response":{"station":[{"name":"品川","prefecture":"東京都","line":"JR山手線","x":139.738999,"y":35.62876,"postal":"1080075","prev":null,"next":"大崎"},{"name":"大崎","prefecture":"東京都","line":"JR山手線","x":139.728439,"y":35.619772,"postal":"1410032","prev":"品川","next":"五反田"},{"name":"五反田","prefecture":"東京都","line":"JR山手線","x":139.723822,"y":35.625974,"postal":"1410022","prev":"大崎","next":"目黒"},{"name":"目黒","prefecture":"東京都","line":"JR山手線","x":139.715775,"y":35.633923,"postal":"1410021","prev":"五反田","next":"恵比寿"},{"name":"恵比寿","prefecture":"東京都","line":"JR山手線","x":139.701238,"y":35.658871,"postal":"1500002","prev":"恵比寿","next":"原宿"},{"name":"原宿","prefecture":"東京都","line":"JR山手線","x":139.702592,"y":35.670646,"postal":"1500001","prev":"渋谷","next":"代々木"},{"name":"代々木","prefecture":"東京都","line":"JR山手線","x":139.702042,"y":35.683061,"postal":"1510051","prev":"原宿","next":"新宿"},{"name":"新宿","prefecture":"東京都","line":"JR山手線","x":139.700464,"y":35.689729,"postal":"1600022","prev":"代々木","next":"新大久保"},{"name":"新大久保","prefecture":"東京都","line":"JR山手線","x":139.700261,"y":35.700875,"postal":"1690073","prev":"新宿","next":"高田馬場"},{"name":"高田馬場","prefecture":"東京都","line":"JR山手線","x":139.703715,"y":35.712677,"postal":"1690075","prev":"新大久保","next":"目白"},{"name":"目白","prefecture":"東京都","line":"JR山手線","x":139.706228,"y":35.720476,"postal":"1710031","prev":"高田馬場","next":"池袋"},{"name":"池袋","prefecture":"東京都","line":"JR山手線","x":139.711085,"y":35.730256,"postal":"1710021","prev":"目白","next":"大塚"},{"name":"大塚","prefecture":"東京都","line":"JR山手線","x":139.728584,"y":35.731412,"postal":"1700005","prev":"池袋","next":"巣鴨"},{"name":"巣鴨","prefecture":"東京都","line":"JR山手線","x":139.739303,"y":35.733445,"postal":"1700002","prev":"大塚","next":"駒込"},{"name":"駒込","prefecture":"東京都","line":"JR山手線","x":139.748053,"y":35.736825,"postal":"1700003","prev":"巣鴨","next":"田端"},{"name":"田端","prefecture":"東京都","line":"JR山手線","x":139.761229,"y":35.737781,"postal":"1140013","prev":"駒込","next":"西日暮里"},{"name":"西日暮里","prefecture":"東京都","line":"JR山手線","x":139.766857,"y":35.731954,"postal":"1160013","prev":"田端","next":"日暮里"},{"name":"日暮里","prefecture":"東京都","line":"JR山手線","x":139.771287,"y":35.727908,"postal":"1100001","prev":"西日暮里","next":"鶯谷"},{"name":"鶯谷","prefecture":"東京都","line":"JR山手線","x":139.778015,"y":35.721484,"postal":"1100003","prev":"日暮里","next":"上野"},{"name":"上野","prefecture":"東京都","line":"JR山手線","x":139.777043,"y":35.71379,"postal":"1100005","prev":"鶯谷","next":"御徒町"},{"name":"御徒町","prefecture":"東京都","line":"JR山手線","x":139.774727,"y":35.707282,"postal":"1100005","prev":"上野","next":"秋葉原"},{"name":"秋葉原","prefecture":"東京都","line":"JR山手線","x":139.773288,"y":35.698619,"postal":"1010028","prev":"御徒町","next":"神田"},{"name":"神田","prefecture":"東京都","line":"JR山手線","x":139.770641,"y":35.691173,"postal":"1010044","prev":"秋葉原","next":"東京"},{"name":"東京","prefecture":"東京都","line":"JR山手線","x":139.766103,"y":35.681391,"postal":"1000005","prev":"神田","next":"有楽町"},{"name":"有楽町","prefecture":"東京都","line":"JR山手線","x":139.763806,"y":35.675441,"postal":"1000006","prev":"東京","next":"新橋"},{"name":"新橋","prefecture":"東京都","line":"JR山手線","x":139.758587,"y":35.666195,"postal":"1050004","prev":"有楽町","next":"浜松町"},{"name":"浜松町","prefecture":"東京都","line":"JR山手線","x":139.757135,"y":35.655391,"postal":"1050022","prev":"新橋","next":"田町"},{"name":"田町","prefecture":"東京都","line":"JR山手線","x":139.747575,"y":35.645737,"postal":"1080023","prev":"浜松町","next":"高輪ゲートウェイ"},{"name":"高輪ゲートウェイ","prefecture":"東京都","line":"JR山手線","x":139.740651,"y":35.635476,"postal":"1080075","prev":"田町","next":null}]}

```

(参考) JSONテーブルでの駅名取得 (2)

JSON_TABLEを含むSQLを実行 (ACSのSQLスクリプトで下記を実行) で公開URLのJSONファイルを取得

```
Select * from JSON_TABLE(
  SYSTOOLS.HTTPGETCLOB(
    'http://express.heartrails.com/api/json?method=getStations&line=JR山手線',null),
  '$.response.station[*]'
  COLUMNS( prev VARGRAPHIC(10) PATH '$.prev',
            name VARGRAPHIC(10) PATH '$.name',
            next VARGRAPHIC(10) PATH '$.next'
  )
) AS X;
```



PREV	NAME	NEXT
-	品川	大崎
品川	大崎	五反田
大崎	五反田	目黒
五反田	目黒	恵比寿
目黒	恵比寿	渋谷
恵比寿	渋谷	原宿
渋谷	原宿	代々木
原宿	代々木	新宿
代々木	新宿	新大久保
新宿	新大久保	高田馬場
新大久保	高田馬場	目白
高田馬場	目白	池袋
目白	池袋	大塚
池袋	大塚	巣鴨
大塚	巣鴨	駒込
巣鴨	駒込	田端
駒込	田端	西日暮里
田端	西日暮里	日暮里
西日暮里	日暮里	鷺谷
日暮里	鷺谷	上野
鷺谷	上野	御徒町
上野	御徒町	秋葉原
御徒町	秋葉原	神田
秋葉原	神田	東京
神田	東京	有楽町
東京	有楽町	新橋
有楽町	新橋	浜松町
新橋	浜松町	田町
浜松町	田町	高輪ゲートウェイ
田町	高輪ゲートウェイ	-

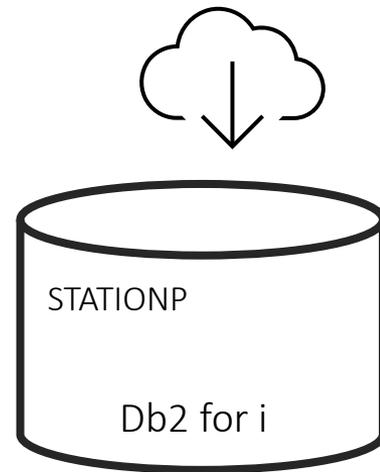
(参考) JSONテーブルでの駅名取得 (3)

- API公開しているJSONデータを、IBM iのデータベースに保存)

```

**free
  ctl-opt dftactgrp(*no) main(main);
  dcl-proc main;
  exec sql insert into stationp
  select * from JSON_TABLE(
    SYSTOOLS.HTTPGETCLOB(
      'http://express.heartrails.com/api/json?method=getStations&line=JR山手線
',null),
    '$.response.station[*]'
    COLUMNS( prev VARGRAPHIC(10) PATH '$.prev',
              name VARGRAPHIC(10) PATH '$.name',
              next VARGRAPHIC(10) PATH '$.next'
            )
  ) AS X;
end-proc;

```



前の駅名	駅名	次の駅名
000001	-	大崎
000002	品川	大反田
000003	大崎	目黒
000004	五反田	恵比寿
000005	目黒	渋谷
000006	恵比寿	渋谷
000007	渋谷	原宿
000008	原宿	代々木
000009	代々木	新大久保
000010	新大久保	新大久保
000011	新大久保	高田馬場
000012	高田馬場	目白
000013	目白	池袋
000014	池袋	大塚
000015	大塚	巣鴨
000016	巣鴨	駒込
000017	駒込	田端
000018	田端	西日暮里
000019	西日暮里	日暮里

IBM i 統合アプリケーション・サーバーでのRESTサービス・サポートのメリット

- IBM i 同梱の標準機能なので**経済的**。
- RPG/COBOLメンバーもRESTサービス開発ができるので、**すぐ開始**。
- 既存のRPG/COBOL資産を活かしてDXができるので**生産的**。
- IBM i 内のアプリ・アーキテクチャーもマイクロサービス化してアジャイル対応できるのでDX対応インフラへの変換が**短期間**。
- Db2とJSON間のやりとりはSQLで一括変換できるのでプログラミングが**簡単**。

A large, black square is centered on the slide. Inside the square, the text 'IBM i' is written in a bold, blue, sans-serif font. The 'i' is lowercase and has a distinct dot.

ワークショップ、セッション、および資料は、IBMによって準備され、IBM独自の見解を反映したものです。それらは情報提供の目的のみで提供されており、いかなる読者に対しても法律的またはその他の指導や助言を意図したのではなく、またそのような結果を生むものでもありません。本資料に含まれている情報については、完全性と正確性を期するよう努力しましたが、「現状のまま」提供され、明示または暗示にかかわらずいかなる保証も伴わないものとします。本資料またはその他の資料の使用によって、あるいはその他の関連によって、いかなる損害が生じた場合も、IBMは責任を負わないものとします。本資料に含まれている内容は、IBMまたはそのサプライヤーやライセンス交付者からいかなる保証または表明を引き出すことを意図したもので、IBMソフトウェアの使用を規定する適用ライセンス契約の条項を変更することを意図したものでなく、またそのような結果を生むものでもありません。

本資料でIBM製品、プログラム、またはサービスに言及している場合、IBMが営業活動を行っているすべての国でそれらが使用可能であることを暗示するものではありません。本資料で言及している製品リリース日付や製品機能は、市場機会またはその他の要因に基づいてIBM独自の決定権をもっていつでも変更できるものとし、いかなる方法においても将来の製品または機能が使用可能になると確約することを意図したものではありません。本資料に含まれている内容は、読者が開始する活動によって特定の販売、売上高の向上、またはその他の結果が生じると述べる、または暗示することを意図したもので、またそのような結果を生むものでもありません。パフォーマンスは、管理された環境において標準的なIBMベンチマークを使用した測定と予測に基づいています。ユーザーが経験する実際のスループットやパフォーマンスは、ユーザーのジョブ・ストリームにおけるマルチプログラミングの量、入出力構成、ストレージ構成、および処理されるワークロードなどの考慮事項を含む、数多くの要因に応じて変化します。したがって、個々のユーザーがここで述べられているものと同様の結果を得られると確約するものではありません。

記述されているすべてのお客様事例は、それらのお客様がどのようにIBM製品を使用したか、またそれらのお客様が達成した結果の実例として示されたものです。実際の環境コストおよびパフォーマンス特性は、お客様ごとに異なる場合があります。

IBM、IBM ロゴ、ibm.com、Db2、Power Systems、POWER6、POWER6+、POWER7、POWER7+、POWER8、POWER9は、世界の多くの国で登録された International Business Machines Corporationの商標です。

他の製品名およびサービス名等は、それぞれIBMまたは各社の商標である場合があります。

現時点での IBM の商標リストについては、www.ibm.com/legal/copytrade.shtmlをご覧ください。

インテル、Intel、Intelロゴ、Intel Inside、Intel Insideロゴ、Centrino、Intel Centrinoロゴ、Celeron、Xeon、Intel SpeedStep、Itanium、およびPentium は Intel Corporationまたは子会社の米国およびその他の国における商標または登録商標です。

Linuxは、Linus Torvaldsの米国およびその他の国における登録商標です。

Microsoft、Windows、Windows NT および Windowsロゴは Microsoft Corporationの米国およびその他の国における商標です。

ITILはAXELOS Limitedの登録商標です。

UNIXはThe Open Groupの米国およびその他の国における登録商標です。

JavaおよびすべてのJava関連の商標およびロゴは Oracleやその関連会社の米国およびその他の国における商標または登録商標です。