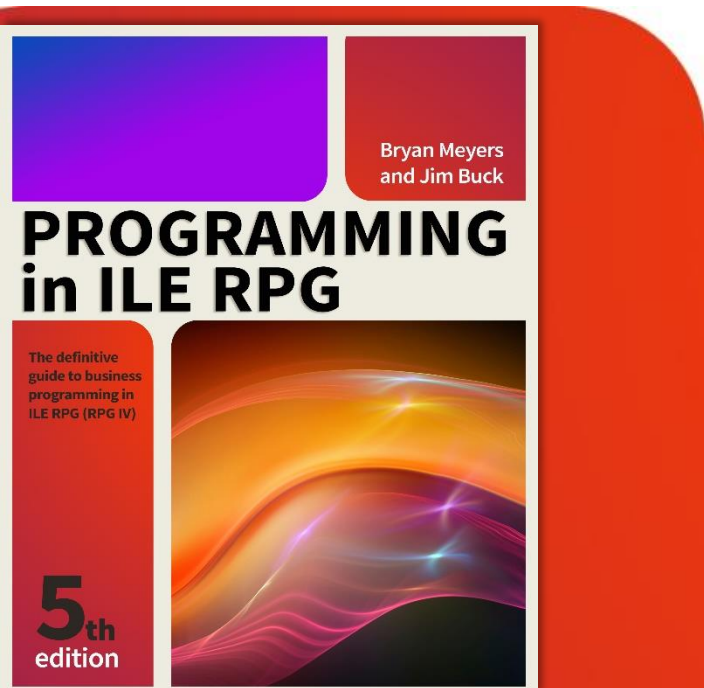





So Now What?

Using IWS Server and Service programs



Presentation © Copyright 2017
impowertechnologies.com



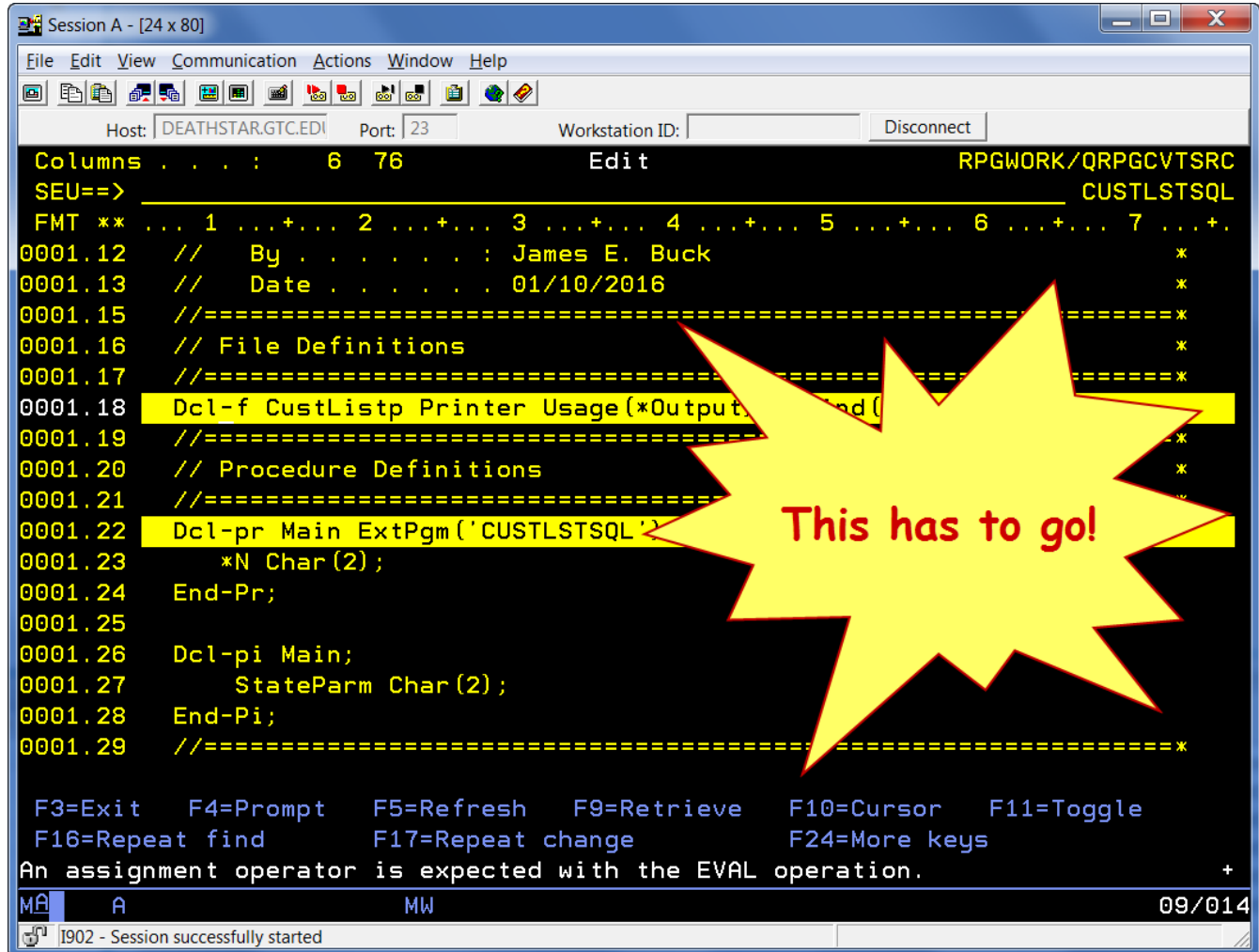
Jim Buck
Phone 262-705-2832
jbuck@impowertechnologies.com
Twitter - @j_buck51

5250 & SEU – Doesn't work anymore!

SEU doesn't support the latest version of RPG.

well I guess, you could turnoff Syntax Checking!

My students have a short introduction... in case of emergencies!

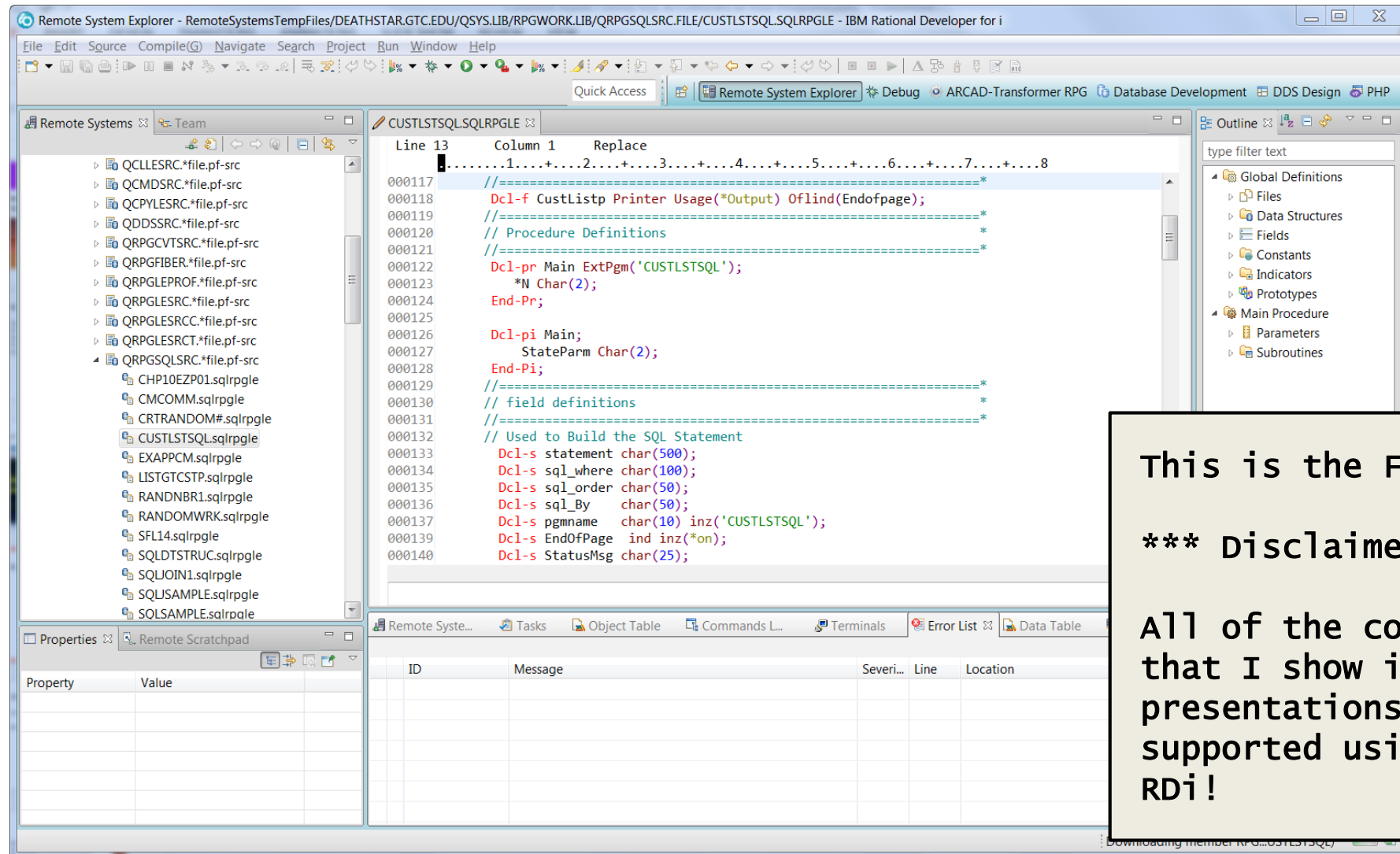


```
Session A - [24 x 80]
File Edit View Communication Actions Window Help
Host: DEATHSTAR.GTC.EDI Port: 23 Workstation ID: Disconnect
Columns . . . : 6 76 Edit RPGWORK/QRPGCVTSRC
SEU==> CUSTLSTSQL
FMT ** ... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+
0001.12 // By . . . . . : James E. Buck *
0001.13 // Date . . . . . 01/10/2016 *
0001.15 //===== *
0001.16 // File Definitions *
0001.17 //===== *
0001.18 Dcl-f CustListp Printer Usage (*Output) find ( *
0001.19 //===== *
0001.20 // Procedure Definitions *
0001.21 //===== *
0001.22 Dcl-pr Main ExtPgm('CUSTLSTSQL') *
0001.23 *N Char(2); *
0001.24 End-Pr; *
0001.25 *
0001.26 Dcl-pi Main; *
0001.27 StateParm Char(2); *
0001.28 End-Pi; *
0001.29 //===== *

F3=Exit F4=Prompt F5=Refresh F9=Retrieve F10=Cursor F11=Toggle
F16=Repeat find F17=Repeat change F24=More keys
An assignment operator is expected with the EVAL operation.
MA A MW 09/014
I902 - Session successfully started
```

This has to go!

Rational Developer for i – 9.5.1.2



This is the FUTURE!

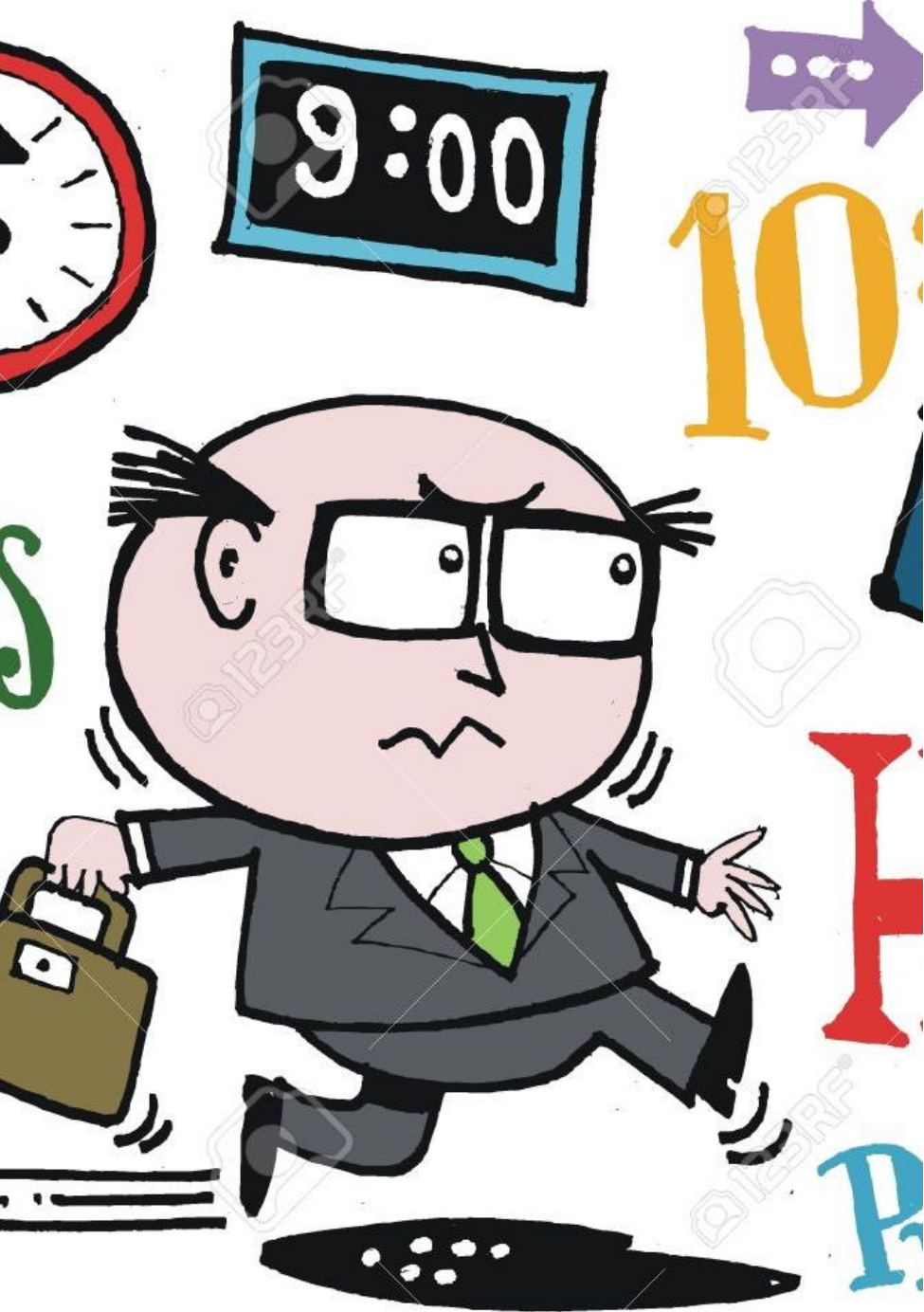
*** Disclaimer ***

All of the code that I show in my presentations is supported using RDi!



The Challenge!

- Monolithic programs
 - Attitude, “Look what I made this program do!”
 - Programs continued to grow.
- Maintenance hours continued to grow
 - We don’t have time to learn new techniques
 - Let’s get the job done...
 - We’ll change our methods tomorrow
- Cut and Paste from “Working” programs
 - Continue to propagate “Poor” code and techniques
- GOTO Monolithic programs



The Result.

- IT Department responsible for other systems
 - Usually small number of people in a department
 - Programmers responsible for Network, printers and PC's
- Productivity Challenges for IBM i IT departments
 - Reactive instead of Proactive
- Much time spent "Fixing" problems
 - Billing problems
 - Ordering problems
 - Month end and year end jobs
- Companies need to reevaluate responsibilities
 - Programmers fixing printers and networks?

Moving Forward...

IBM i Programmers:

- Need to broaden their IT Skills
 - New Tools and techniques
 - Learn skills that aren't necessarily traditional IBM i
 - CSS, HTML, JavaScript, NodeJS and PHP
 - Do a few Online Tutorials - <https://www.w3schools.com/>
- Pick a small High Profile project;
 - Something that a number of people would use
 - Will help raise awareness of your new skills
 - Also the capabilities of the IBM i
- Realize the first couple of projects will be “Freebies”

Writing today's RPG programs

Need to create modular applications:

- Many of RPG's problems were caused by:
 - Applications tied to a specific interface (5250)
 - IBM's continue support for outdated IDE's (PDM/SEU/SDA) Oh Yeah... RLU!
- IBM's Business Continuity has caused problems
 - Failure of IBM to force change
 - Why is today's OS capable of running System 36 code?
 - Try running Windows 98 Applications on Windows 10

The OLD Way!

```
IBM i RSE Getting Started | PROG122F.RPGLE
Line 16      Column 70      Replace
.....Ffilename++IPEASFRlen+LKlen+AIdevice+.Keywords+++++++
001200      F*      Author: J. Yaeger   Date Written: July 1, 1993   *
001300      F*      Converted to RPG IV Nov. 1995   *
001400      F*****
001500      FCSCstP      UF A E      K DISK
001600      FPROG122FD CF      E      WORKSTN
001700      C      DOW      *IN03 = *OFF
001800      C      WRITE      Footer
001900      C      EVAL      CustIn = 0
002000      C      EXFMT      Scrn1
002100      C      SELECT
002200      C      WHEN      *IN03 = *ON
002300      C      LEAVE
002400      C      WHEN      *IN12 = *ON
002500      C      ITER
002600      C      WHEN      Action = 'A'
002700      C      EXSR      AddRecord
002800      C      WHEN      Action = 'C'
002900      C      EXSR      ChgRecord
003000      C      WHEN      Action = 'D'
003100      C      EXSR      DltRecord
003200      C      ENDSL
003300      C      ENDDO
003400      C      EVAL      *INLR = *ON
003500      C      RETURN
```

Traditional RPG

- Tied to a specific interface
- Uses traditional DB2 I/O

```
A - 5250 Display
File Edit View Communication Actions Window Help
Host: 10.10.10.10 Port: 23 Workstation ID: Disconnect
Customer File Maintenance
Type choice, then Enter.
Action code . . . . . C
A = Add
C = Change
D = Delete
Enter Cust. number . . . _____
F3=Exit F12=Cancel
12/838
|DEVUSR030.IDEVCLOUD.COM:23
```

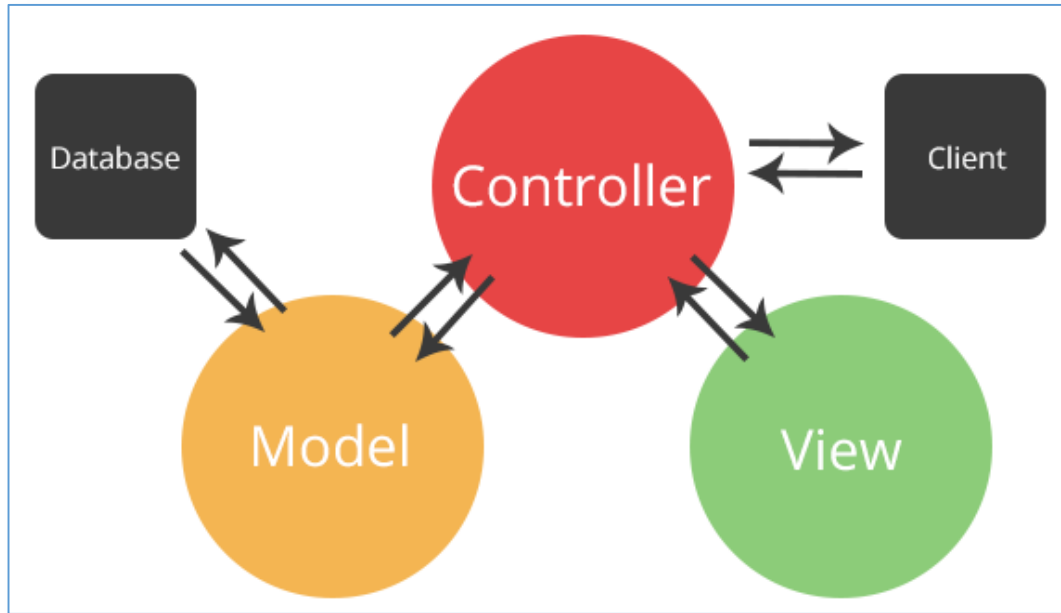

Today's Interfaces – What's next?



21st Century RPG programs - Modularity

- Developing code in small, independent units offers several advantages
 - Reusability
 - Fewer errors
 - Easier to test
 - Changes are unlikely to cause unwanted side effects
 - Easier team development
- Breakout the Interface from the database
 - Ready for the next User Interface
 - No need to rewrite your RPG Code for the next Interface

Model, View, Controller Architecture

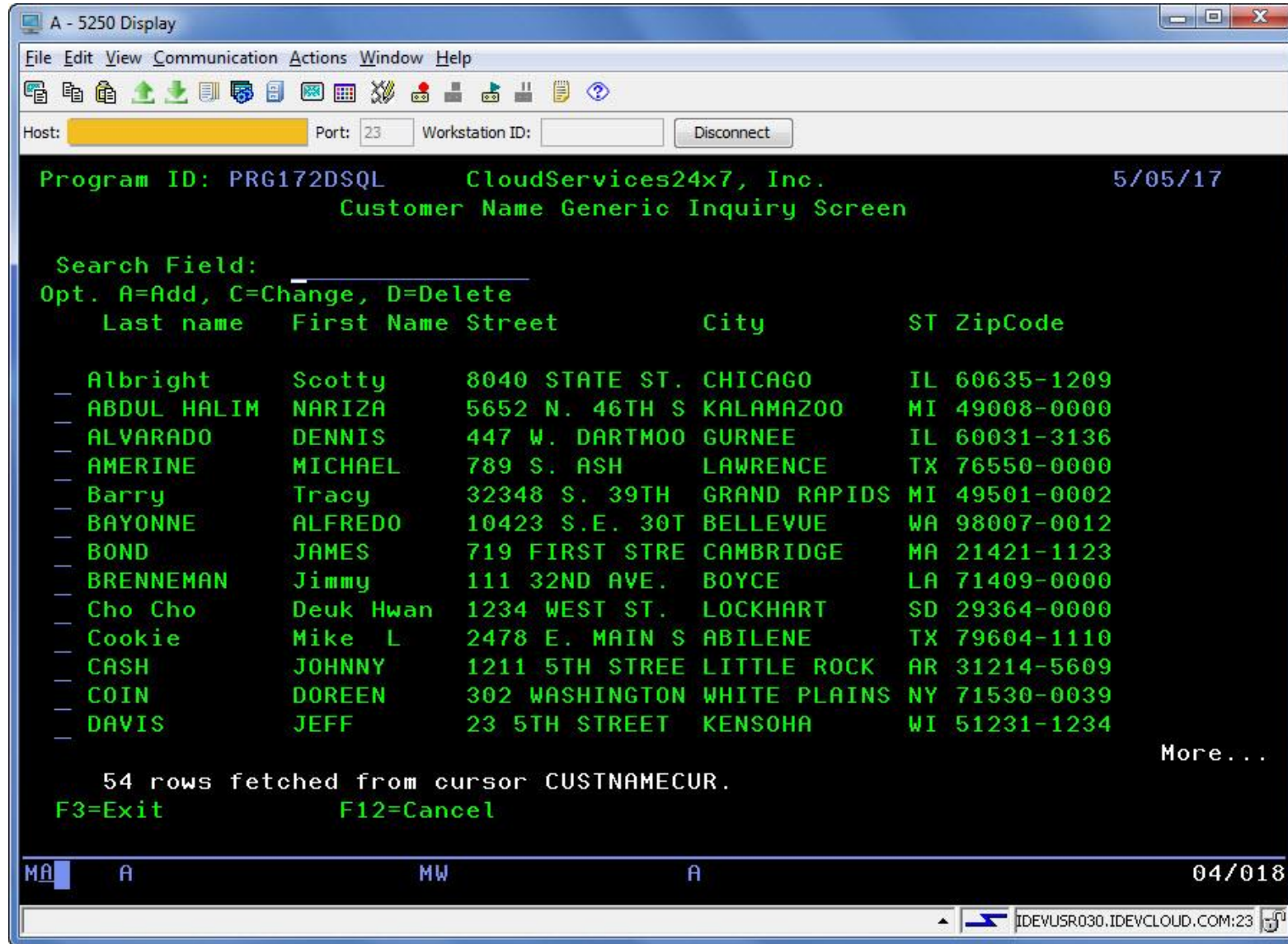


The process starts with a **CLIENT** request

The request reaches the **Controller**

- The **Controller** organizes the resources needed to process the request
- Updates the **MODEL** (Database) if needed
- Builds or changes the **VIEW** and sends the **VIEW** changes to the **CLIENT**

Subfile Application Example



Subfile Application Example

- **Comprised of three programs + copybook**
 - CUSTSFLPGM – Main Driver Program
 - Runs the 5250 screens
 - Handles the CREATE, READ, UPDATE and DELETE Logic
 - CUSTSRVPGM – Service program that handles SQL I/O
 - SQL INSERT, UPDATE, SELECT and DELETE Code
 - Returns data Structures (Customer and SQL Status)
 - GETSQLDIAG – Service program that:
 - Processes the GET DIAGNOSTICS command
 - Puts the results into a data structure
 - Returns this data structure to the calling program





CUSTSLFPGM – Driver program

The screenshot shows an IDE window with the following components:

- Menu Bar:** File, Edit, Source, Compile(G), Navigate, Search, Project, Run, Window, Help.
- Toolbar:** Standard development tools like save, run, and search.
- Tab Bar:** CUSTSLFPGM.SQRPGL, CUSTSRVCPY.RPGL, CUSTSRVPGM.SQRPGL, GETSQLDI.
- Main Editor:** Displays the source code for CUSTSLFPGM.SQRPGL. The code includes declarations for variables like SearchTerm, sql_where, sql_order, sql_By, and quote. It features a main loop with subroutines such as ClearFields(), LoadSFL(), DisplaySFL(), and BuildSQLStmt().
- Outline View:** Located on the right, it shows a hierarchical tree of the program's structure. A blue box labeled "Outline View" is overlaid on this panel. The tree includes categories like Data Structures, Fields, Constants, Indicators, Prototypes, Main Procedure, and Subprocedures. The Subprocedures list includes: LoadSFL, DisplaySFL, ProcessSFL, ClearSFL, BuildSQLStmt, HandleSQLMessages, AddRecord, AssignNextNbr, ChangeRecord, DeleteRecord, and ClearFields.



CUSTSRVPGM – SQL Database I/O

The screenshot displays an IDE window with a COBOL program named CUSTSRVPGM.SQLRPGLE. The main editor shows the following code:

```
Line 19      Column 1      Replace
...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+
000100      Ctl-opt nomain Option(*NoDebugIO:*SrcStmt:*NoUnRef); //
000101      Ctl-opt PGMINFO(*PCML : *MODULE : *DCLCASE);
000200      // *****
000300      // Handles SQL IO for CUSTOMER CRUD Application *
000400      // Written: Jim Buck Email: jbuck@impowertechnologies *
000500      // Copyright 2017 - imPower Technologies *
000600      // These examples are for demonstration purposes. There are *
000700      // NO Express of implied warranties. Have fun with the code! *
000800      // *****
000900      // Copy in the Prototypes
001000      /copy RPGTRAIN/QRPGWEBSRV,CUSTSRVCPY
001100
001200      // Data Structure for Database I/O
001300      Dcl-Ds CUSTOMER_IODataDS Ext ExtName('CUSTOMER') Qualified
001400      End-DS;
001500
001600      // Array Data Structure for Database I/O
001700      Dcl-Ds CUSTOMER_IODataRcdsDS Ext ExtName('CUSTOMER') Qualified
001800      Dim(9999) End-DS;
001900
001901      Dcl-ds IODataRcdsDSxx Qualified Dim(9999);
001902      CUSTNO ZONED(6:0);
001903      CFNAME CHAR(10);
001904      CLNAME CHAR(15);
001905      CSTREET CHAR(20);
001906      CCITY CHAR(15);
001907      CSTATE CHAR(2);
001908      CZIP CHAR(9);
001909      CPHONE Zoned(10:0);
001910      CEMAIL char(35);
001911      End-DS;
001912
002000      // Data Structure for SQL Results
002100      Dcl-Ds UtildSSQL inz;
002200      MessageId Char(10);
```

The Outline View on the right shows the following structure:

- Control Statements
- Global Definitions
 - Data Structures
 - Fields
 - Prototypes
- Subprocedures
 - GetCUSTOMER_Data : EXPORT
 - GetCUSTOMER_DataRcds : EXPORT
 - GetCUSTOMERRecs_DynSelect : EXPORT
 - DeleteCUSTOMER_Data : EXPORT
 - UpDateCUSTOMER_Data : EXPORT
 - WriteCUSTOMER_Data : EXPORT

A blue box with the text "Outline View" is overlaid on the right side of the IDE window.



GETSQLDIAG – SQL Database I/O

The screenshot displays an IDE window with a code editor on the left and an Outline View on the right. The code editor shows the following SQL code:

```
Line 1      Column 1      Replace
...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+
000100      Ctl-opt nomain Option(*NoDebugID:*SrcStmt:*NoUnRef); //
000101      // *****
000102      // Used to process SQL GET DIAGNOSTICS Service program *
000103      // Written: Jim Buck Email: jrbuck@impowertechnologies *
000104      // Copyright 2017 - imPower Technologies *
000105      // These examples are for demonstration purposes. There are *
000106      // NO Express of implied warranties. Have fun with the code! *
000107      // *****
000108      // *****
000109      Dcl-Pr GetDiagnostics;
000110      *n LikeDS(UtilDSSQL);
000111      End-Pr ;
000112
000113      // Data Structure for SQL Results
000114      Dcl-Ds UtilDSSQL inz;
000115      MessageId Char(10);
000116      MessageId1 Char(7);
000117      MessageId2 Char(7);
000118      MessageLength int(5);
000119      MessageText Char(120);
000120      ReturnedSQLCode int(5);
000121      ReturnedSQLState Char(5);
000122      RowsCount int(10);
000123      SuccessFlag Ind; // Operation was Successful
000124      End-Ds;
000125
000126      Dcl-proc GetDiagnostics Export;
000127      Dcl-Pi *N;
000128      DiagUtilDS LikeDS(UtilDSSQL);
000129      End-Pi ;
000130
000131      Clear DiagUtilDS;
000132
000133      Exec sql GET DIAGNOSTICS CONDITION 1
000134      :DiagUtilDS.MessageId = DR2 MESSAGE TD.
```

The Outline View on the right shows a tree structure of the code:

- Control Statements
 - Global Definitions
 - Data Structures
 - Fields
 - Prototypes
 - Subprocedures
 - GetDiagnostics : EXPORT
 - Parameters
 - Local Definitions
 - 10 (D)
 - 27



CUSTSRVCPY – Prototype Copybook

The screenshot displays a development environment with a code editor on the left and an outline view on the right. The code editor shows a copybook with several DCL-PR prototypes for a CUSTOMER CRUD application. The outline view lists these prototypes under 'Global Definitions'.

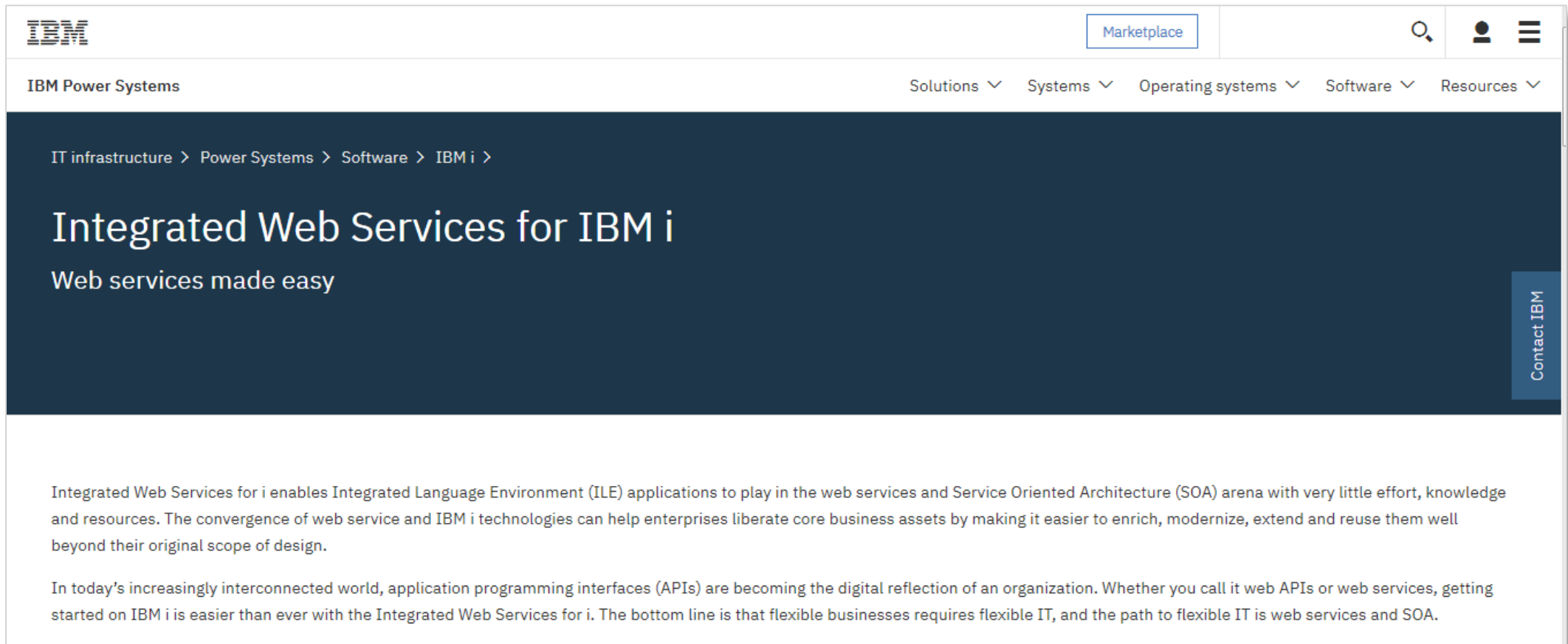
```
Line 1      Column 1      Replace
.....1.....2.....3.....4.....5.....6.....7.....
000101      // *****
000103      // Prototypes used for CUSTOMER CRUD Application      *
000104      // Written: Jim Buck Email: jbuck@impowertechnologies      *
000105      // Copyright 2017 - imPower Technologies      *
000106      // These examples are for demonstration purposes. There are      *
000107      // NO Express of implied warranties. Have fun with the code!      *
000108      // *****
000121
000135      DCL-PR GetCUSTOMER_Data;
000137      CUSTOMERDataDS LIKEDS(CUSTOMER_IODataDS);
000138      WrkCustNbr Zoned(6:0);
000139      WrkUtilDS LikeDS(UtilDSSQL);
000140      END-PR ;
000141
000142      DCL-PR GetCUSTOMER_DataRecds;
000143      CUSTOMER_IORcdsDS_LENGTH int(10);
000145      CUSTOMER_IORcdsDS LIKEDS(CUSTOMER_IODataDS) Dim(9999);
000146      WrkUtilDS LikeDS(UtilDSSQL);
000147      END-PR ;
000148
000149      DCL-PR GetCUSTOMERRecs_DynSelect;
000150      CUSTOMERDataDS_LENGTH int(10);
000151      CUSTOMERDataDS LIKEDS(CUSTOMER_IODataDS) Dim(9999);
000152      Statement Char(4096) ;
000153      SearchTerm Char(100);
000154      WrkUtilDS LikeDS(UtilDSSQL);
000155      END-PR ;
000156
000157      DCL-PR DeleteCUSTOMER_Data;
000158      WrkCustNbr Zoned(6:0);
000159      WrkUtilDS LikeDS(UtilDSSQL);
000160      END-PR ;
000161
000162      DCL-PR UpDateCUSTOMER_Data;
000163      CUSTOMERDataDS LIKEDS(CUSTOMER_IODataDS);
```

Outline View

- Global Definitions
 - Prototypes
 - GetCUSTOMER_Data: EXTPROC ('GETCUSTOMER_DATA')
 - GetCUSTOMER_DataRecds: EXTPROC ('GETCUSTOMER_DATAARECDS')
 - GetCUSTOMERRecs_DynSelect: EXTPROC ('GETCUSTOMERRECS_DYNSELECT')
 - DeleteCUSTOMER_Data: EXTPROC ('DELETECUSTOMER_DATA')
 - UpDateCUSTOMER_Data: EXTPROC ('UPDATECUSTOMER_DATA')
 - WriteCUSTOMER_Data: EXTPROC ('WRITECUSTOMER_DATA')
 - GetDiagnostics: EXTPROC ('GETDIAGNOSTICS')

IBM's IWS Server Solution

www-03.ibm.com/systems/power/software/i/iws/



The screenshot shows the IBM website interface. At the top left is the IBM logo. To the right is a 'Marketplace' button, a search icon, and a user profile icon. Below the logo is the text 'IBM Power Systems'. On the right side of the header, there are navigation menus for 'Solutions', 'Systems', 'Operating systems', 'Software', and 'Resources', each with a dropdown arrow. The main content area has a dark blue background with the breadcrumb 'IT infrastructure > Power Systems > Software > IBM i >'. The main heading is 'Integrated Web Services for IBM i' with the sub-heading 'Web services made easy'. A vertical 'Contact IBM' button is on the right. Below this is a white section with two paragraphs of text.

IBM

Marketplace

IBM Power Systems

Solutions ▾ Systems ▾ Operating systems ▾ Software ▾ Resources ▾

IT infrastructure > Power Systems > Software > IBM i >

Integrated Web Services for IBM i

Web services made easy

Contact IBM

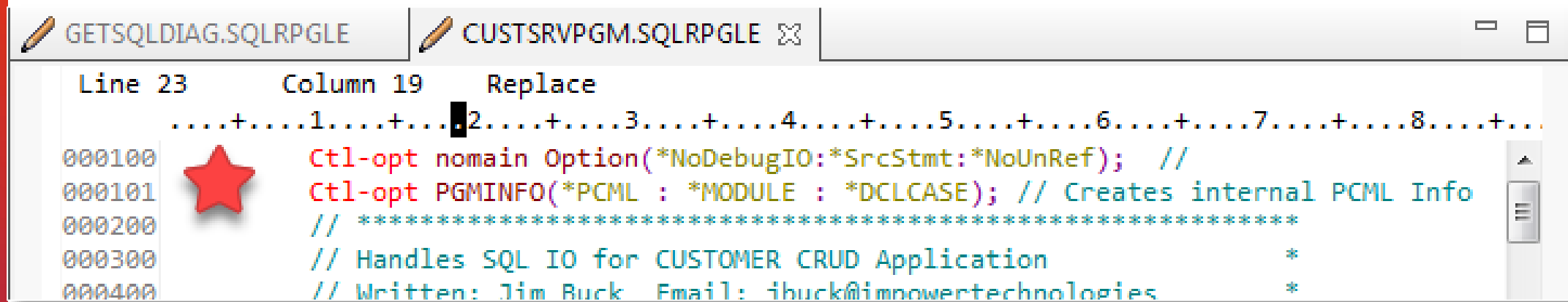
Integrated Web Services for i enables Integrated Language Environment (ILE) applications to play in the web services and Service Oriented Architecture (SOA) arena with very little effort, knowledge and resources. The convergence of web service and IBM i technologies can help enterprises liberate core business assets by making it easier to enrich, modernize, extend and reuse them well beyond their original scope of design.

In today's increasingly interconnected world, application programming interfaces (APIs) are becoming the digital reflection of an organization. Whether you call it web APIs or web services, getting started on IBM i is easier than ever with the Integrated Web Services for i. The bottom line is that flexible businesses requires flexible IT, and the path to flexible IT is web services and SOA.

Program Call Markup Language (PCML)

What is PCML and how are we using it?

- Describes the exported procedures of a Service Program for the Web Server.
- The Service Program CUSTSRVPGM uses the Ctrl-Opt Below
- This command adds the PCML information to the Program Object



```
GETSQLDIAG.SQLRPGLE | CUSTSRVPGM.SQLRPGLE
Line 23      Column 19      Replace
.....1.....2.....3.....4.....5.....6.....7.....8.....+...
000100      ★ Ctrl-opt nomain Option(*NoDebugIO:*SrcStmt:*NoUnRef); //
000101      Ctrl-opt PGMINFO(*PCML : *MODULE : *DCLCASE); // Creates internal PCML Info
000200      // *****
000300      // Handles SQL IO for CUSTOMER CRUD Application *
000400      // Written: Jim Buck Email: ihuck@impowertechologies *
```

PCML Restrictions - Important

https://www.ibm.com/support/knowledgecenter/ssw_ibm_i_72/rzasc/pcmlrestrict.htm

IBM i 7.2

PCML Restrictions

The following are restrictions imposed by PCML regarding parameter and return value types.

- » The following data types are not supported by PCML:
 - o Pointer
 - o Procedure Pointer
 - o 1-Byte Integer
- ◀
- Return values and parameters passed by value can only be 4 byte integers (10i 0).
- Varying-length arrays, and data structures containing varying-length subfields are not supported.
- When a data structure is used as a parameter for a *ENTRY PLIST, or a prototyped parameter is defined with LIKEDS, some PCML restrictions apply:
 - o The data structure may not have any overlapping subfields.
 - o The subfields must be coded in order; that is, the start position of each subfield must follow the end position of the previous subfield.
 - o If there are gaps between the subfields, the generated PCML for the structure will have subfields named "_unnamed_1", "_unnamed_2" etc, of type "char".
- RPG does not have the concept of output-only parameters. Any parameters that do not have CONST or VALUE coded have a usage of "inputoutput". For inputoutput parameters, the ProgramCallDocument class requires the input values for the parameter to be set before the program can be called. If the parameter is truly an output parameter, you should edit the PCML to change "inputoutput" to "output".

The compile will fail if you generate PCML for a program or module that violates one of the restrictions. The PCML will be generated, but it will contain error messages as comments. For example, if you use a Date field as a parameter, the PCML for that parameter might look like this:

```
<data name="DATE" type=" " length="10" usage="input" />
<!-- Error: unsupported data type -->
```

Contact Us

PCML Restrictions - Important

The screenshot displays the Source Prompter interface for a program named CUSTSRVTST. The code defines a data structure for SQL results with various fields. A red arrow points to the declaration of MessageText as VarChar(32). Below the code, the Error List shows three messages, with a red arrow pointing to the first one: RNS9308, indicating a compilation error.

```
Line 42      Column 45      Insert
.....1.....2.....3.....4.....5.....6.....7.....8.....9.....+
000159
000160      // Data Structure for SQL Results
000161      Dcl-Ds UtilDSSQL inz;
000162      MessageId Char(10);
000163      MessageId1 Char(7);
000164      MessageId2 Char(7);
000165      MessageLength int(5);
000166      MessageText VarChar(32);
000167      ReturnedSQLCode int(5);
000168      ReturnedSQLState Char(5);
000169      RowsCount int(10);
000170      End-Ds;
000171      Dcl-s WrkCustNbr Zoned(6:0);
000172
000173      // *****
000174      // * Retrieves DB2 Data For CUSTOMER
```

Commands Log | Error List | IBM i Service E... | Data Table | Source Prompter | Job Log | Field

ID	Message	Severity	Line	Location
RNS9308	Compilation stopped. Severity 30 errors found in program.	50	0	RPGTRA
RNF0320	Errors were found while generating the program information to be placed i...	30	0	RPGTRA
RNF7031	The name or indicator CSTREET is not referenced.	00	29	RPGTRA

Create a Web Services Server

IBM Web Administration for i

Setup **Manage** Advanced | Related Links

All Servers HTTP Servers | Application Servers | Installations

Common Tasks and Wizards

- Create Web Services Server
- Create HTTP Server
- Create Application Server

Create Web Services Server

Specify Web services server name - Step 1 of 4

Welcome to the Create Web Services Server wizard. A Web services server provides a set of Web services. Web service clients can then interact with these IBM i programs using protocols such as SOAP and REST. The clients can be implemented using various programming languages. For more information, please visit: <https://www.ibm.com/systems/power/software>

Specify a unique name for this server ?

Server name:

Server description:

Create HTTP server

Back Next Cancel

Create Web Server

1. Create Web Services Server
2. Name the Server
3. Create a HTTP Server
4. Click Next to proceed

Create a Web Services Server - cont.

Application Servers Installations

Server: WEBSERVS - V2.6 (web services)

Create Web Services Server

Specify network attributes for server - Step 2 of 4

Your server may listen for requests on specific IP addresses or on all IP addresses of the server.

Specify internet addresses and ports for server ?

Specify server command port: 10026

Specify internet address and port for the server

IP address: All IP addresses

Port: 10025

Specify internet address and port for the HTTP server

IP address: All IP addresses

Port: 10035

Back Next Cancel

Proceed to the next step of this task.

Assign the Server Ports

1. Ports for the IWS Server
2. Port for the HTTP Server
3. Click **Next**

Note: it's a good idea to check your applications and make sure there are not conflicts

Use **NETSTAT** to check

Create a Web Services Server - cont.

Application Servers | Installations

Create Web Services Server

Specify User ID for Server - Step 3 of 4

The server requires an IBM i user ID to run the server's jobs. It is recommended that a special server's objects, such as files and directories.

Specify user ID for this server: ?

Use **default** user ID

Specify an **existing** user ID

User ID:

Create a **new** user ID

Back Next Cancel

User ID for the Web Server

1. Use the default ID or use one created for the Server
2. Click **Next**

Create a Web Services Server - Click **Finish**

Application Servers | Installations

Create Web Services Server

Summary - Step 4 of 4

Servers Service

Web Services Server Information

Server name: **WEBSERVICE** **1**

Server description: Web services server created by the Create Web Services Server wizard.

Port: 10025

Command port: 10026

Server root: **/WWW/WEBSERVICE** **2**

Server URL: http://idevus030.idevcloud.com:10035

User ID for server: WEBSERVICE

Context root: /web

HTTP Server Information

HTTP server name: WEBSERVIC **3**

Back Finish Cancel

Review the Selections

1. IWS Server name

- Server Ports

2. Server Root

- Location of main IFS folder

3. HTTP Name Information

- Port information

Click **Finish**

- Server is created and started

The Web Services Application

Application Servers Installations

Server: WEBSERVS - V2.6 (web services)

[WEBSERVS](#) > Manage Deployed Services

Manage Deployed Services

Data current as of Sep 17, 2017 8:16:50 PM.

Deployed services: ?

	Service name	Status	Type	Startup type	Service definition
<input type="radio"/>	AddCustomer	Running	REST	Automatic	View Swagger
<input type="radio"/>	ConvertTemp	Running	SOAP	Automatic	View WSDL
<input type="radio"/>	DeleteCustomer	Running	REST	Automatic	View Swagger
<input type="radio"/>	GetAllCustomers	Running	REST	Automatic	View Swagger
<input type="radio"/>	GetCustomer	Running	REST	Automatic	View Swagger
<input checked="" type="radio"/>	UpdateCustomer	Running	REST	Automatic	View Swagger

This is the completed IWS Customer Application:

- Comprised of procedures from the **CUSTSRVPGM**
- Each Web Service is an individual procedure
- All are **REST** Web Services

Get a Customer Record.

```
// *****  
// * Retrieves DB2 Data For CUSTOMER  
// *****  
Dcl-Proc GetCUSTOMER_Data Export;  
Dcl-Pi *N;  
    CUSTOMERDataDS LIKEDS(CUSTOMER_IODDataDS);  
    WrkCustNbr Zoned(6:0);  
    WrkUtilDS LikeDS(UtilDSSQL);  
End-Pi ;  
  
SuccessFlag = *off;  
Clear CUSTOMERDataDS;  
Clear WrkUtilDS;  
  
EXEC SQL  
    SELECT CUSTNO, CFNAME, CLNAME, CSTREET, CCITY, CSTATE, CZIP,  
           CPHONE, CALPHONE, CEMAIL, ORDDAT, BALDUE  
           INTO :CUSTOMERDataDS FROM CUSTOMER  
           WHERE CUSTNO = :WrkCustNbr;  
  
GetDiagnostics(WrkUtilDS);
```

Get a Customer Record - cont.

```
If WrkUtilDS.ReturnedSQLCode = 0000;  
    WrkUtilDS.SuccessFlag = *on;  
Else;  
    WrkUtilDS.SuccessFlag = *off;  
    CUSTOMERDataDS.CUSTNO = 999999;  
    CUSTOMERDataDS.CFNAME = 'No record found';  
EndIf;  
End-Proc;
```

Deploying a Web Service – GetCustomer

IBM Web Administration for i
Setup **Manage** Advanced | Related Links
All Servers | HTTP Servers **Application Servers** Installations
Running Server: WEBSERVS - V2.6 (web services)

Manage Deployed Services
Data current as of Sep 16, 2017 8:10:21 PM.

Deployed services: ?

	Service name	Status	Type	Startup type	Service definition
<input type="radio"/>	ConvertTemp	Running	SOAP	Automatic	View WSDL

Deploy Refresh
Deploy a new service on this server.

Close

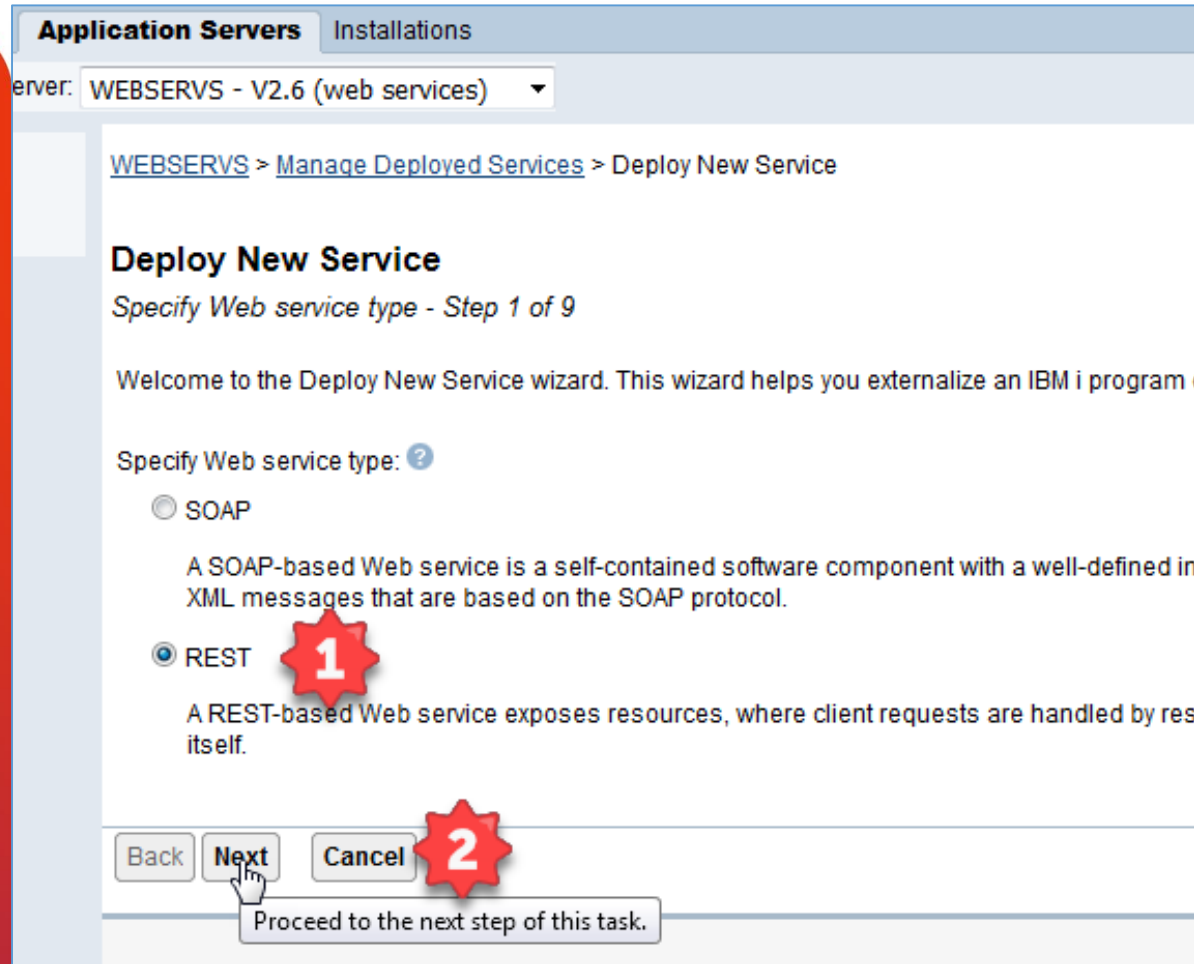
1. ConvertTemp

- Default application
- Easy way to test the new server

2. Click **Deploy**

- Start the process of installing a new service

Deploying a Web Service – GetCustomer



This presentation will only discuss REST Services

1. Select **REST** web Service
2. Then **NEXT**

Deploying a Web Service – GetCustomer

Application Servers Installations

Server: WEBSERVS - V2.6 (web services)

Deploy New Service

Specify Location of IBM i Program Object - Step 2 of 9

The IBM i object to be externalized as a Web service must be an existing ILE program (*PGM) or service program (*SRVPGM).

Specify the program object for the Web service. ?

Specify IBM i library and ILE program object name (Recommended)

You can specify the program object location by entering the name of the library that contains the program object. This is the recommended way to locate the program object.

Library name: **1**

ILE Object name:

ILE Object type: *SRVPGM *PGM

Browse the integrated file system for the IBM i program object

2

Proceed to the next step of this task.

1. Program information

- Enter the Library where your Service program is located
- Enter the name of your Service Program

2. Click **Next**

NOTE – Always be aware of authority with the Library and program... *PUBLIC *ALL is NOT the answer!

Deploying a Web Service – GetCustomer

Application Servers Installations

Server: WEBSERVS - V2.6 (web services)

WEBSERVS > Manage Deployed Services > Deploy New Service

Deploy New Service

Specify Name for Service - Step 3 of 9

The Web service to be externalized is a resource. The URI path template identifies a string or one or more template parameters that can contain regular expressions.

Resource name:

Service description: **1**

URI path template: e.g. /temperature

2

Back Next Cancel

Proceed to the next step of this task.

1. New Service Information

- Resource name used in the Request URL.
- Service Description for documentation
- URL path template
 - defines the portion of the URL for passing any parameters to the Web Service
 - Parameters are defined by braces ({ and })

2. Click **Next**

Deploying a Web Service – GetCustomer

Application Servers Installations

Server: WEBSERVS - V2.6 (web services)

The table below lists all the exported procedures found in the program object that can be exported to a Web Service. The Usage parameter attribute affects what data is sent by client applications.

Detect length fields

Use parameter name as element name for data structures

Export procedures: ?

Select	Procedure name/Parameter name	Usage	Data type
<input type="checkbox"/>	▶ WRITECUSTOMER_DATA		
<input type="checkbox"/>	▶ UPDATECUSTOMER_DATA		
<input type="checkbox"/>	▶ DELETECUSTOMER_DATA		
<input type="checkbox"/>	▶ GETCUSTOMERRECS_DYNSELECT		
<input type="checkbox"/>	▶ GETCUSTOMER_DATAARECDS		
<input checked="" type="checkbox"/>	▶ GETCUSTOMER_DATA		

Select All Deselect All Expand All Collapse All

Back Next Cancel

Proceed to the next step of this task.

Select a Service program Procedure

1. Click - **Deselect All**
 - Notice that all of the Procedures in the service program are listed
 - Only one procedure to a Web Service
2. Click – **GETCUSTOMER_DATA**
3. Click **Next**

Deploying a Web Service – GetCustomer

Application Servers Installations

Server: WEBSERVS - V2.6 (web services)

Specify Resource Method Information - Step 5 of 9

Procedures are mapped to resource methods. Each resource method needs to be defined to handle the request.

Specify resource method information. ?

Procedure name: GETCUSTOMER_DATA

URI path template for resource: /customer/{custno:\d+}

HTTP request method: **1** GET

URI path template for method: *NONE or...

HTTP response code output parameter: *NONE

HTTP header array output parameter: *NONE

Allowed input media types: **2** *JSON or...

Returned output media types: **3** *JSON or...

Back Next Cancel

Proceed to the next step of this task.

Select how parameters are handled:

1. Select the type of Request method. In this case we will use a GET method
2. In this application we are setting the Input/Output method types to JSON.
3. Click **Next**

Deploying a Web Service – GetCustomer

Application Servers Installations

Server: WEBSERVS - V2.6 (web services) ▾

Detect length fields

Use parameter name as element name for data structures

Export procedures: ?

Select	Procedure name/Parameter name	Usage	Data type
<input type="checkbox"/>	▶ WRITECUSTOMER_DATA		
<input type="checkbox"/>	▶ UPDATECUSTOMER_DATA		
<input type="checkbox"/>	▶ DELETECUSTOMER_DATA		
<input type="checkbox"/>	▶ GETCUSTOMERRECS_DYNSELECT		
<input checked="" type="checkbox"/>	▼ GETCUSTOMER_DATARECDS		
	CUSTOMER_IORcdsDS_LENGTH	output ▾	int
	CUSTOMER_IORcdsDS	output ▾	struct
	WrkUtilDS	output ▾	struct
<input type="checkbox"/>	▶ GETCUSTOMER_DATA		

Select Deselect All Expand All Collapse All

Back Next Cancel

Proceed to the next step of this task.

Define procedure parameters exported

1. Click on the Procedure that you will use for this Web Service.
 - Then the twisty to display the fields
2. Select whether the parameter is Input or Output
3. Click **Next**

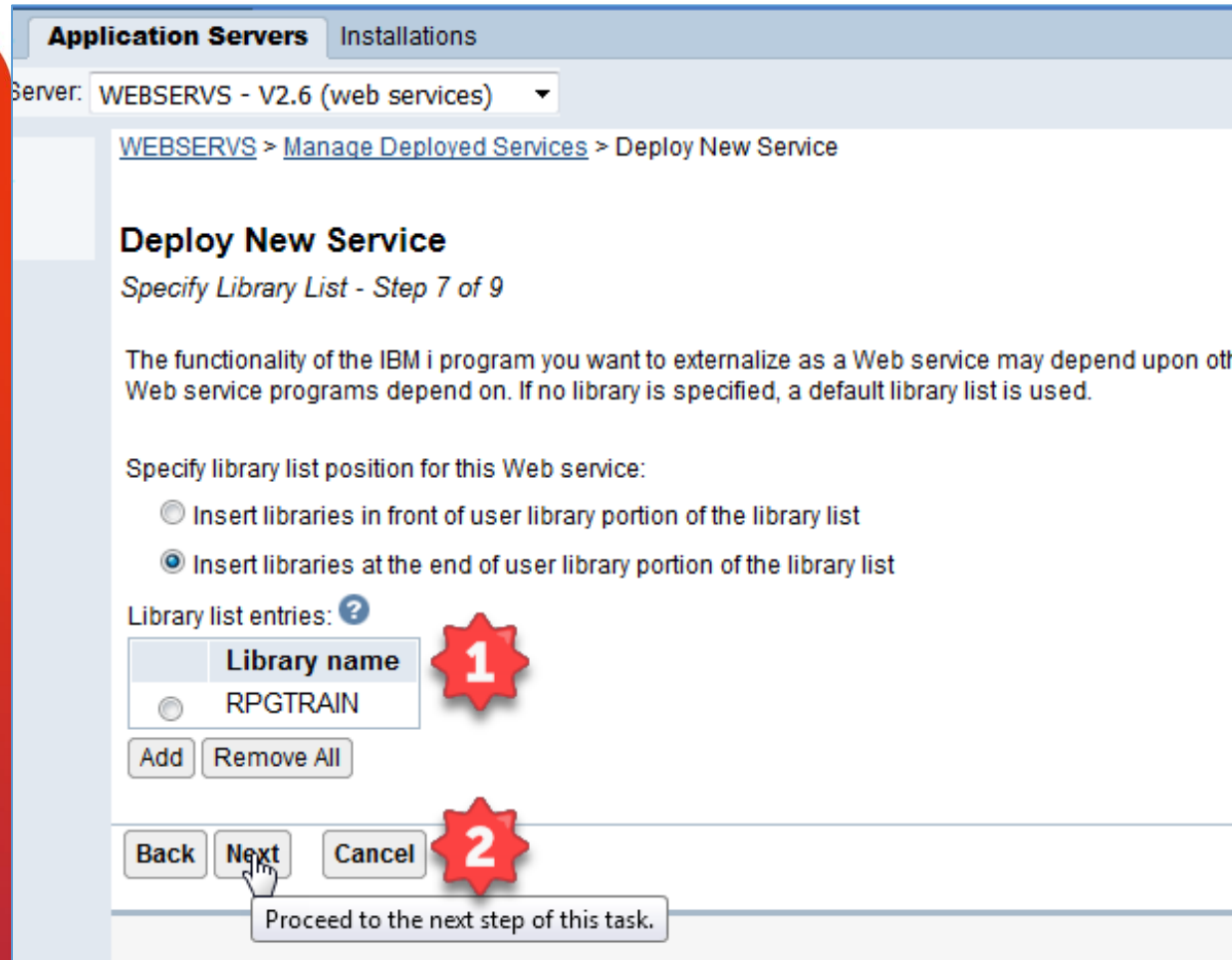
Deploying a Web Service – GetCustomer

Setting the User ID for the Web Service

1. I suggest you use a user ID with minimal authority and no Special Authorities¹.
2. Click **Next**

¹ **NOT QSECOFR!** 😊

Deploying a Web Service – GetCustomer



Application Servers Installations

Server: WEBSERVS - V2.6 (web services)

WEBSERVS > Manage Deployed Services > Deploy New Service

Deploy New Service

Specify Library List - Step 7 of 9

The functionality of the IBM i program you want to externalize as a Web service may depend upon other Web service programs depend on. If no library is specified, a default library list is used.

Specify library list position for this Web service:

- Insert libraries in front of user library portion of the library list
- Insert libraries at the end of user library portion of the library list

Library list entries: ?

Library name
<input type="radio"/> RPGTRAIN

Add Remove All

Back Next Cancel

Proceed to the next step of this task.

Specifying Library list

1. Depending on the complexity of the application here is where you set the web Services Library List
2. Click **Next**

Deploying a Web Service – GetCustomer

Application Servers Installations

Server: WEBSERVS - V2.6 (web services)

Specify transport information to be passed to the web service implementation code.

Information to be passed to web service implementation code ?

Specify Transport Metadata:

Transport Metadata
<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

Back Next Cancel

Proceed to the next step of this task.

Passing additional information to the Web Service

1. For example:
 - The remote IP address
 - Remote Server information
 - Passed in the environmental variables
2. Click **Next**

Deploying a Web Service – GetCustomer

Application Servers Installations

Server: WEBSERVS - V2.6 (web services)

Deploy New Service

Summary - Step 9 of 9

When you click **Finish** the web service is deployed.

1

Service Methods Request Information

Resource name: GetCustomer
Resource description: Get an Individual Customer Record
Service install path: /www/webservs/webservices/services/GetCustomer
URI path template: /customer/{custno:\d+}
User ID for service: WEBSERVICE
Program: /QSYS.LIB/RPGTRAIN.LIB/CUSTSRVPGM.SRVPGM
Library list for service: RPGTRAIN

2

Back Finish Cancel

Save all of the pending changes for this task.

Summary Section

1. This area has three sections
 - Allows you to review the settings you chose
 - Go **Back** or **Cancel**
 - **Care should be used here...** once you create the server some settings cannot be changed
2. Click **Finish** and the service will be created and started

Deploying a Web Service – GetCustomer

Application Servers Installations

Server: WEBSERVS - V2.6 (web services)

[WEBSERVS](#) > Manage Deployed Services


Manage Deployed Services

Data current as of Sep 16, 2017 9:04:20 PM.

Deployed services: ?

	Service name	Status	Type	Startup type	Service definition
<input type="radio"/>	ConvertTemp	Running	SOAP	Automatic	View WSDL
<input checked="" type="radio"/>	GetCustomer	Installing	REST	Automatic	View Swagger

Deploy



Application Servers Installations

Server: WEBSERVS - V2.6 (web services)


[WEBSERVS](#) > Manage Deployed Services

Manage Deployed Services

Data current as of Sep 16, 2017 9:05:17 PM.

Deployed services: ?

	Service name	Status	Type	Startup type	Service definition
<input type="radio"/>	ConvertTemp	Running	SOAP	Automatic	View WSDL
<input checked="" type="radio"/>	GetCustomer	Running	REST	Automatic	View Swagger



Finding the URL to run

Final URL format (Bottom) Comprised of:

- URL Template
- Base Resource URL

Application Servers Installations

Server: WEBSERVS - V2.6 (web services)

[WEBSERVS](#) > Manage Deployed Services

Manage Deployed Services

Data current as of Sep 17, 2017 12:34:30 PM.

Deployed services: ?

Service name	Status	Type	Start
ConvertTemp	Running	SOAP	Auto
GetCustomer	Running	REST	Auto

Buttons: Deploy, Stop, Properties, Uninstall

Properties tooltip: Display the properties.

Service Properties

General | Methods | Library List | Swagger | Connection Pool | Request Information

Service information ?

Resource Name: GetCustomer

Resource description: Get an Individual Customer Record

URI path template: /customer/{custno:ld+}

Startup type: Automatic

Service install path: /www/WEBSERVS/webservices/services/GetCustomer

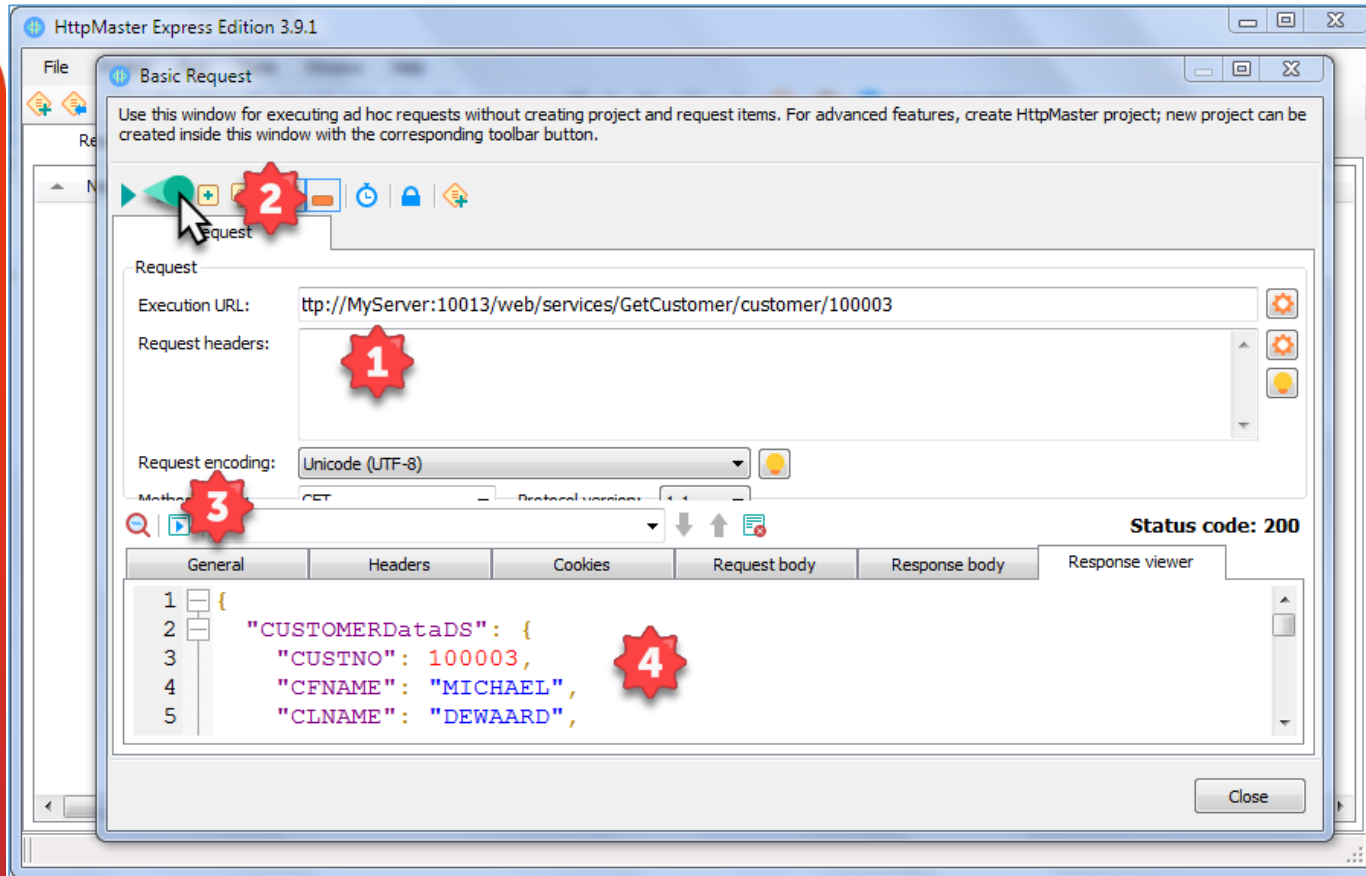
Program: /QSYS.LIB/RPGTRAIN.LIB/CUSTSRVPGM.SRVPGM

Base resource URL: http://MyServer.com:10024/web/services/GetCustomer

User ID for this service: *SERVER

<http://myserver.com:10024/web/services/GetCustomer/customer/100002>

Testing your new Web Service



I use HTTPMaster a free product to test my URL's.

1. Enter the URL
2. Click the RUN icon
3. Will display the Web request status and Data
4. Display the actual data returned

www.httpmaster.net

Delete a Customer Record.

```
// *****  
// * Delete DB2 Data For CUSTOMER  
// *****  
Dcl-Proc DeleteCUSTOMER_Data Export;  
Dcl-Pi *N;  
    WrkCustNbr Zoned(6:0);  
    WrkUtilDS LikeDS(UtilDSSQL);  
End-Pi ;  
SuccessFlag = *off;  
  
EXEC SQL  
    Delete from CUSTOMER  
        where CUSTNO = :WrkCustNbr;  
  
GetDiagnostics(WrkUtilDS);  
If WrkUtilDS.ReturnedSQLCode = 000;  
    WrkUtilDS.SuccessFlag = *on;  
    COMMIT;  
Else;  
    WrkUtilDS.SuccessFlag = *off;  
EndIf;  
  
End-Proc;
```

Deploying a Web Service – DeleteCustomer

Application Servers Installations

Server: WEBSERVS - V2.6 (web services)

WEBSERVS > Manage Deployed Services

Data current as of Sep 17, 2011

Deployed services: ?

Service name
<input type="radio"/> ConvertTemp
<input checked="" type="radio"/> GetCustomer

Deploy Stop Pro

Application Servers Installations

Server: WEBSERVS - V2.6 (web services)

WEBSERVS > Manage Deployed Services

Deploy New Service

Specify Web service

Welcome to the Deployment Wizard

Specify Web service type

SOAP

A SOAP-based XML message

REST

A REST-based service that is externalized as itself.

Back Next Cancel

Application Servers Installations

Server: WEBSERVS - V2.6 (web services)

WEBSERVS > Manage Deployed Services

Deploy New Service

Specify Location of IBM i Program Object - Step 2 of 9

The IBM i object to be externalized as a Web service must be located in an IBM i library.

Specify the program object for the Web service. ?

Specify IBM i library and ILE program object name (Recommended)

You can specify the program object location by entering the name of the program object. The recommended way to locate the program object is to specify the library and program object name.

Library name:

ILE Object name:

ILE Object type: *SRVPGM *PGM

Browse the integrated file system for the IBM i program object

Back Next Cancel

Proceed to the next step of this task.

First three steps are the same since the procedures are all in one service program

Deploying a Web Service – DeleteCustomer

Application Servers Installations

server: WEBSERVS - V2.6 (web services)

[WEBSERVS](#) > [Manage Deployed Services](#) > Deploy New Service

Deploy New Service

Specify Name for Service - Step 3 of 9

The Web service to be externalized is a resource. The URI path template identifies matching string or one or more template parameters that can contain regular expressions to further res

Resource name:

Service description:

URI path template: e.g. /temperature, /temperature

1

Proceed to the next step of this task.

Name the Web Service

1. Important thing to remember is to name fields to help identify the service
2. Click **Next**

Deploying a Web Service – DeleteCustomer

Application Servers Installations

Server: WEBSERVS - V2.6 (web services)

Detect length fields

Use parameter name as element name for data structures

Export procedures: ?

Select	Procedure name/Parameter name	Usage	Data type
<input type="checkbox"/>	▶ WRITECUSTOMER_DATA		
<input type="checkbox"/>	▶ UPDATECUSTOMER_DATA		
<input checked="" type="checkbox"/>	▼ DELETECUSTOMER_DATA		
	WrkCustNbr	input	zoned
	WrkUtilDS	output	struct
<input type="checkbox"/>	▶ GETCUSTOMERRECS_DYNSELECT		
<input type="checkbox"/>	▶ GETCUSTOMER_DATAECDS		
<input type="checkbox"/>	▶ GETCUSTOMER_DATA		

Select All Deselect All Expand All Collapse All

Back Next Cancel

Proceed to the next step of this task.

Define Parameters

1. Deselect all of the procedures
2. Select the DELETECUSTOMER_DATA procedure
3. Change the
 - **WrkCustNbr** to an input field, **WrkUtilDS** will still be an Output field

Click **Next**

Deploying a Web Service – DeleteCustomer

Application Servers Installations

Server: WEBSERVS - V2.6 (web services)

Procedure name: DELETECUSTOMER_DATA

URI path template for resource: /DeleteCustomer/{custno:\d+}

HTTP request method: DELETE **1**

URI path template for method: *NONE or...

HTTP response code output parameter: *NONE

HTTP header array output parameter: *NONE

Allowed input media types: *XML_AND_JSON or... **2**

Returned output media types: *XML_AND_JSON or... **2**

Whether to wrap input parameters:

Wrap input parameters

Do not wrap input parameters

Input parameter mappings:

Parameter name	Data type	Input source	Identifier	Def
WrkCustNbr	zoned	*PATH_PARAM	custno	*NONE

Back Next Cancel

Proceed to the next step of this task.

Resource Method Information

1. Change HTTP Method to Delete
2. Change Media types to JSON or XML
3. Mapping **WrkCustNbr**
 - Associate with custno
 - *PATH_PARAM will be in the URL of the request

Click **Next**

Deploying a Web Service – DeleteCustomer

The image displays four sequential screenshots of the IBM WebSphere Administration Console, illustrating the deployment process for a web service named 'DeleteCustomer'. The screenshots are overlaid, showing the progression from step 1 to step 4. Red starburst callouts numbered 1 through 4 highlight the 'Next' button in the first three steps and the 'Finish' button in the final summary step.

Step 1: 'Deploy New Service' - Specify User ID for the service. The user ID is 'WEB'. The 'Next' button is highlighted with a red starburst callout '1'.

Step 2: 'Deploy New Service' - Specify Library List. The library list is 'RPGTRAIN'. The 'Next' button is highlighted with a red starburst callout '2'.

Step 3: 'Deploy New Service' - Specify Transport Information. The transport information is 'QUERY_ST'. The 'Next' button is highlighted with a red starburst callout '3'.

Step 4: 'Deploy New Service' - Summary - Step 9 of 9. The summary shows the following details:

- Service: DeleteCustomer
- Methods: Methods
- Request Information: Request Information
- Resource name: DeleteCustomer
- Resource description: Delete a Customer Record
- Service install path: /www/webservers/webservices/services/DeleteCustomer
- URI path template: /DeleteCustomer/{custno:\d+}
- User ID for service: WEBSERVICE
- Program: /QSYS.LIB/RPGTRAIN.LIB/CUSTSRVPGM.SRVPGM
- Library list for service: RPGTRAIN

The 'Finish' button is highlighted with a red starburst callout '4'. A tooltip below the 'Finish' button reads: 'Save all of the pending changes for this task.'

Last four steps are the same.

Make sure to check your options before clicking Finish!

Add a Customer Record.

```
// *****  
// * Adds New DB2 Data For CUSTOMER  
// *****  
Dcl-Proc WriteCUSTOMER_Data Export;  
  
Dcl-Pi *N;  
    CUSTOMERDataDS LIKEDS(CUSTOMER_IODataDS);  
    WrkCustNbr Zoned(6:0);  
    WrkUtilDS LikeDS(UtilDSSQL);  
End-Pi ;  
SuccessFlag = *off;  
    EXEC SQL  
        INSERT INTO CUSTOMER  
            (CUSTNO, CFNAME, CLNAME, CSTREET, CCITY, CSTATE, CZIP, CPHONE,  
             CALPHONE, CEMAIL, ORDDAT, BALDUE)  
            VALUES(:CUSTOMERDataDS);  
    GetDiagnostics(WrkUtilDS);  
  
    If ReturnedSQLCode = 000;  
        WrkUtilDS.SuccessFlag = *on;  
        COMMIT;  
    Else;  
        WrkUtilDS.SuccessFlag = *off;  
    EndIf;  
End-Proc;
```

Deploying a Web Service – AddCustomer

The image displays three overlapping screenshots of the IBM i Application Servers console, illustrating the steps to deploy a web service:

- Step 1:** The 'Manage Deployed Services' page. The 'Service name' list shows 'ConvertTemp' and 'GetCustomer'. The 'GetCustomer' service is selected, and the 'Deploy' button is highlighted.
- Step 2:** The 'Deploy New Service' page. The 'Specify Web service type' section shows 'REST' selected. The 'Next' button is highlighted.
- Step 3:** The 'Specify Location of IBM i Program Object - Step 2 of 9' page. The 'Specify the program object for the Web service' section shows 'Specify IBM i library and ILE program object name (Recommended)' selected. The 'Library name' is 'RPGTRAIN' and the 'ILE Object name' is 'CUSTSRVPGM'. The 'Next' button is highlighted.

First three steps are the same since the procedures are all in one service program

Deploying a Web Service – AddCustomer

Application Servers Installations

Server: WEBSERVS - V2.6 (web services)

[WEBSERVS](#) > [Manage Deployed Services](#) > Deploy New Service

Deploy New Service

Specify Name for Service - Step 3 of 9

The Web service to be externalized is a resource. The URI path template identifies matching string or one or more template parameters that can contain regular expressions to further res

Resource name:

Service description: **1**

URI path template: e.g. /temperature, /temperature

2

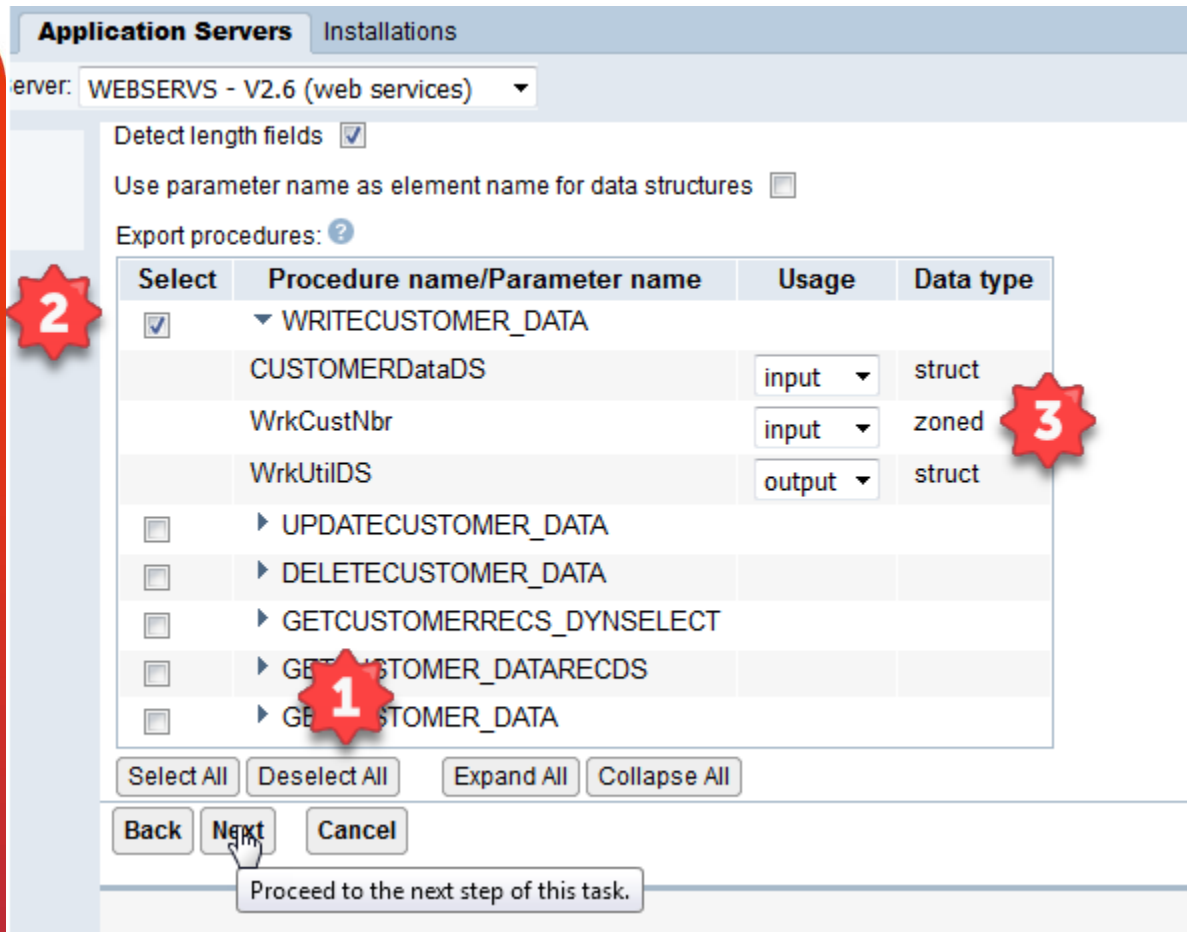
Proceed to the next step of this task.

1. Define Parameters

- Resource should make sense for the procedure
- Always include a description of the Service
- URL path Template

2. Click **Next**

Deploying a Web Service – AddCustomer



Define Parameters

1. Deselect All procedures
2. Select
WRITECUSTOMER_DATA
3. Set CUSTOMERDataDS as
an input parameter
4. Also WrkCustNbr As
input

Click **Next**

Deploying a Web Service – AddCustomer

Application Servers Installations

Server: WEBSERVS - V2.6 (web services)

Procedure name: WRITECUSTOMER_DATA

URI path template for resource: /customer/{custno: *+}

HTTP request method: PUT **1**

URI path template for method: *NONE or...

HTTP response code output parameter: *NONE

HTTP header array output parameter: *NONE

Allowed input media types: *JSON or... **2**

Returned output media types: *JSON or...

Whether to wrap input parameters:

Wrap input parameters

Do not wrap input parameters

Input parameter mappings:

Parameter name	Data type	Input source	Identifier	Default
CUSTOMERDataDS	struct	*NONE		
WrkCustNbr	zoned	*PATH_PARAM	custno	*NONE

3

Back Next Cancel

Proceed to the next step of this task.

Resource Method Information

1. Change HTTP Method to **PUT**
2. Change Media types to JSON or XML
3. Mapping **WrkCustNbr**
 - Associate with custno
 - *PATH_PARAM will be in the URL of the request

Click **Next**

Deploying a Web Service – ADDCustomer

The image displays four sequential screenshots of the IBM WebSphere Administration Console, illustrating the steps to deploy a web service. Red starburst numbers 1, 2, 3, and 4 are overlaid on the 'Next' buttons in each screenshot to indicate the progression.

Step 1: The 'Deploy New Service' wizard is shown. The 'Specify User ID for the service' section has 'Specify an existing user ID' selected. The 'User ID' field contains 'WEBSERVICE'. The 'Update the service' checkbox is checked. The 'Next' button is highlighted with a red starburst '1'.

Step 2: The 'Specify Library List' section is shown. The 'Specify library list position' section has 'Insert libraries' selected. The 'Library list entries' section shows 'RPGTRAIN' selected. The 'Next' button is highlighted with a red starburst '2'.

Step 3: The 'Specify Transport Information' section is shown. The 'Specify Transport Method' section has 'QUERY_STATEMENT' selected. The 'Next' button is highlighted with a red starburst '3'.

Step 4: The 'Summary - Step 9 of 9' section is shown. The 'Service' tab is selected. The 'Resource name' is 'DeleteCustomer', the 'Resource description' is 'Delete a Customer Record', the 'Service install path' is '/www/webservers/webservices/services/DeleteCustomer', the 'URI path template' is '/DeleteCustomer/{custno:\d+}', the 'User ID for service' is 'WEBSERVICE', the 'Program' is '/QSYS.LIB/RPGTRAIN.LIB/CUSTSRVPGM.SRVPGM', and the 'Library list for service' is 'RPGTRAIN'. The 'Finish' button is highlighted with a red starburst '4'.

A callout box on the right contains the text: "Last four steps are the same. Make sure to check your options before clicking Finish!"

Retrieve all Customer Records

```
// *****  
// Get Multiple CUSTOMER Records  
// *****  
Dcl-Proc GetCUSTOMER_DataRecds EXPORT;  
  
Dcl-Pi GetCUSTOMER_DataRecds;  
    CUSTOMER_IORcdsDS_LENGTH int(10);  
    CUSTOMER_IORcdsDS LikeDS(CUSTOMER_IODataRcdsDS) Dim(9999);  
    WrkUtilDS LikeDS(UtilDSSQL);  
End-Pi;  
  
Dcl-s NbrOfRows int(5) inz(%elem(CUSTOMER_IORcdsDS));  
Dcl-s RecordsNotFound Ind;  
SuccessFlag = *off;  
  
// Clear out the Data Structures  
Clear CUSTOMER_IORcdsDS;  
Clear WrkUtilDS;  
    Exec SQL Declare GetCUSTOMER_DataRecdsCur Cursor  
        for SELECT CUSTNO, CFNAME, CLNAME, CSTREET, CCITY, CSTATE, CZIP,  
            CPHONE, CALPHONE, CEMAIL, ORDDAT, BALDUE  
            FROM CUSTOMER  
            ORDER BY CLNAME, CFNAME;  
  
GetDiagnostics(WrkUtilDS);
```

Retrieve all Customer Records

```
Exec SQL Open GetCUSTOMER_DataRecdsCur;  
GetDiagnostics(WrkUtilDS);
```

```
Exec SQL Fetch GetCUSTOMER_DataRecdsCur FOR :NbrOfRows ROWS  
Into :CUSTOMER_IORcdsDS;  
GetDiagnostics(WrkUtilDS);
```

```
If WrkUtilDS.RowsCount > 0;  
    WrkUtilDS.SuccessFlag = *on;  
    CUSTOMER_IORcdsDS_LENGTH = WrkUtilDS.RowsCount;  
Else;  
    WrkUtilDS.SuccessFlag = *off;  
EndIf;
```

```
Exec SQL close GetCUSTOMER_DataRecdsCur;
```

```
End-Proc
```

Deploying a Web Service – GetAllCustomers

The image displays three overlapping screenshots of the IBM i Application Servers console, illustrating the steps to deploy a web service. The screenshots are numbered 1, 2, and 3.

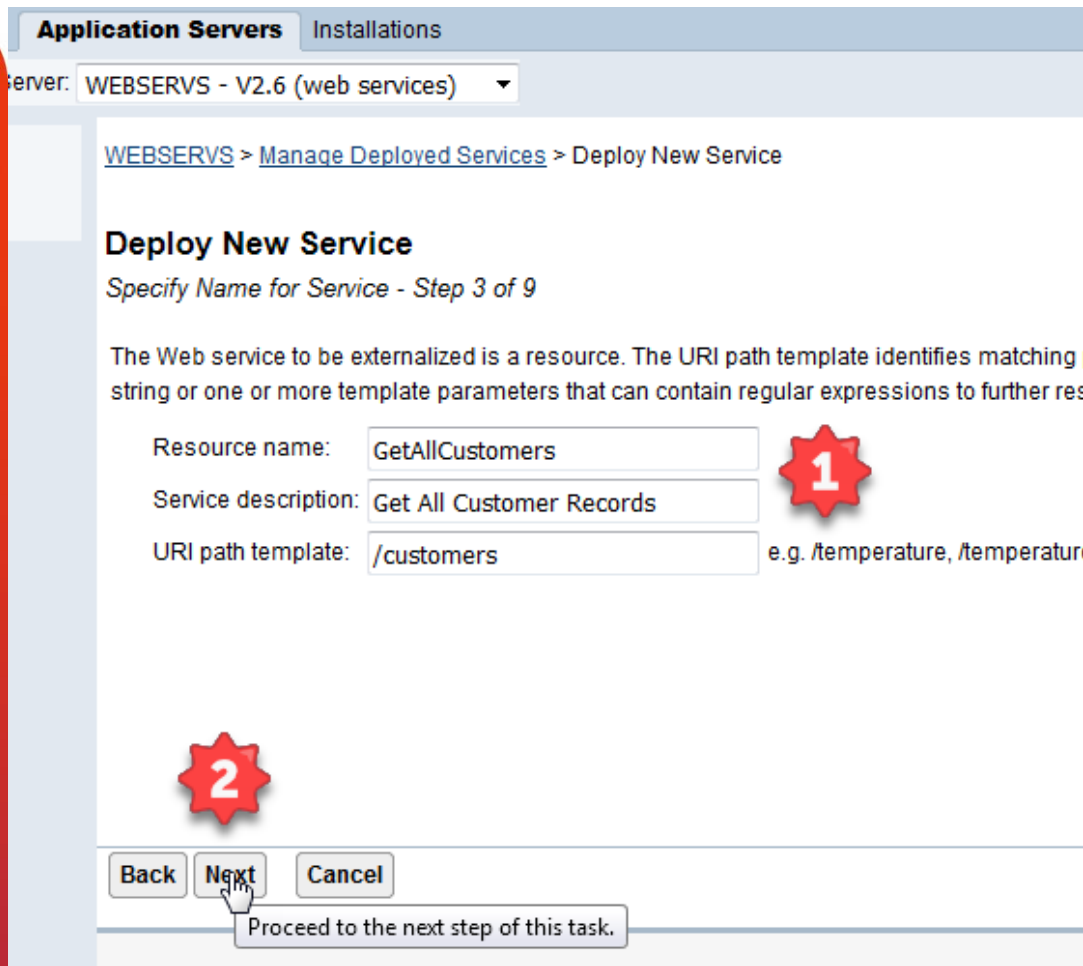
Step 1: The 'Manage Deployed Services' page is shown. The 'Service name' column lists 'ConvertTemp' and 'GetCustomer'. The 'GetCustomer' service is selected, and the 'Deploy' button is highlighted.

Step 2: The 'Deploy New Service' page is shown. The 'Specify Web service type' section has 'REST' selected. The 'Next' button is highlighted.

Step 3: The 'Specify Location of IBM i Program Object - Step 2 of 9' page is shown. The 'Specify the program object for the Web service' section has 'Specify IBM i library and ILE program object name (Recommended)' selected. The 'Library name' is 'RPGTRAIN' and the 'ILE Object name' is 'CUSTSRVPGM'. The 'Next' button is highlighted.

First three steps are the same since the procedures are all in one service program

Deploying a Web Service – GetAllCustomers



Application Servers Installations


Server: WEBSERVS - V2.6 (web services)

[WEBSERVS](#) > [Manage Deployed Services](#) > Deploy New Service

Deploy New Service


Specify Name for Service - Step 3 of 9

The Web service to be externalized is a resource. The URI path template identifies matching string or one or more template parameters that can contain regular expressions to further res

Resource name: 

Service description:

URI path template: e.g. /temperature, /temperature



Proceed to the next step of this task.

1. Define Parameters

- Resource should make sense for the procedure
- Always include a description of the Service
- URL path Template

2. Click **Next**

Deploying a Web Service – GetAllCustomers

Application Servers Installations

Server: WEBSERVS - V2.6 (web services)

Detect length fields **1**

Use parameter name as element name for data structures

Export procedures: ?

Select	Procedure name/Parameter name	Usage	Data type	Count
<input type="checkbox"/>	▶ WRITECUSTOMER_DATA			
<input type="checkbox"/>	▶ UPDATECUSTOMER_DATA			
<input type="checkbox"/>	▶ DELETECUSTOMER_DATA			
<input type="checkbox"/>	▶ GETCUSTOMERRECS_DYNSELECT			
<input checked="" type="checkbox"/> 2	▼ GETCUSTOMER_DATAARECDS			
	CUSTOMER_IORcdsDS_LENGTH	output	int	
	CUSTOMER_IORcdsDS	output	struct	CUSTOMER_IORcdsDS_L
	WrkUtilIDS	output	struct	
<input type="checkbox"/>	▶ GETCUSTOMER_DATA			

Select All Deselect All Expand All Collapse All

Back **Next** Cancel

Proceed to the next step of this task.

Resource Method Information

1. Uncheck Detect length fields
2. Check
GETCUSTOMER_DATAARECDS
3. Parameters
 - Discussed on the next Slide

Click **Next**

Deploying a Web Service – GetAllCustomers

Detect length fields **1**

Use parameter name as element name for data structures

Export procedures: ?

Select	Procedure name/Parameter name	Usage	Data type	Count
<input type="checkbox"/>	▶ WRITECUSTOMER_DATA			
<input type="checkbox"/>	▶ UPDATECUSTOMER_DATA			
<input type="checkbox"/>	▶ DELETECUSTOMER_DATA			
<input type="checkbox"/>	▶ GETCUSTOMERRECS_DYNSELECT			
<input checked="" type="checkbox"/>	▼ GETCUSTOMER_DATAARECDS			
2	CUSTOMER_IORcdsDS_LENGTH	output ▼	int	
	CUSTOMER_IORcdsDS	output ▼	struct	CUSTOMER_IORcdsDS_LENGTH ▼
	WrkUtilDS	output ▼	struct	

Resource Method Information

1. Detect length fields – Uncheck to allow you to associate:
2. CUSTOMER_IORcdsDS_LENGTH with CUSTOMER_IORcdsDS

The CUSTOMER_IORcdsDS_LENGTH must be set in the RPG program to tell the webservice how many records are returned

Deploying a Web Service – GetAllCustomers

Application Servers Installations

Server: WEBSERVS - V2.6 (web services)

Deploy New Service

Specify Resource Method Information - Step 5 of 9

Procedures are mapped to resource methods. Each resource method needs to be defined to

Specify resource method information. ?

Procedure name:	GETCUSTOMER_DATAARECDS
URI path template for resource:	/customers
HTTP request method:	1 GET
URI path template for method:	*NONE or...
HTTP response code output parameter:	*NONE
HTTP header array output parameter:	*NONE
Allowed input media types:	*ALL or...
Returned output media types:	2 *JSON or...

Back Next Cancel

Proceed to the next step of this task.

Resource Method Information

1. Change HTTP Method to GET
2. Change Media output type to JSON
 - There is no input parameters in the Web Service

Click **Next**

Deploying a Web Service – GetAllCustomers

1 Specify User ID for the service

2 Specify Library List

3 Specify Transport Information

4 Summary - Step 9 of 9

When you click **Finish** the web service will be deployed.

Service	Methods	Request Information
Resource name:	GetAllCustomers	
Resource description:	Get All Customer Records	
Service install path :	/www/webservs/webservices/services/GetAllCustomers	
URI path template:	/customers	
User ID for service:	webservice	
Program:	/QSYS.LIB/RPGTRAIN.LIB/CUSTSRVPGM.SRVPGM	
Library list for service:	RPGTRAIN	

Save all of the pending changes for this task.

Last four steps are the same.
Make sure to check your options before clicking Finish!

Update A Customer Record.

```
// *****  
// * Updates DB2 Data For CUSTOMER  
// *****  
  
Dcl-Proc UpDateCUSTOMER_Data Export;  
  Dcl-Pi *N;  
    CUSTOMERDataDS LIKEDS(CUSTOMER_IODataDS);  
    WrkCustNbr Zoned(6:0);  
    WrkUtilDS LikeDS(UtilDSSQL);  
  
  End-Pi ;  
  
  SuccessFlag = *off;  
    EXEC SQL UPDATE CUSTOMER  
      SET ROW = :CUSTOMERDataDS  
      WHERE CUSTNO = :WrkCustNbr;  
  
  GetDiagnostics(WrkUtilDS);
```

Update A Customer Record - cont.

```
If ReturnedSQLCode = 000;  
    WrkUtilDS.SuccessFlag = *on;  
    COMMIT;  
else;  
    WrkUtilDS.SuccessFlag = *off;  
EndIf;  
End-Proc;
```


Deploying a Web Service – UpdateCustomer

The image displays three overlapping screenshots of the IBM i Application Servers console, illustrating the steps to deploy a web service. The first screenshot shows the 'Manage Deployed Services' page with 'GetCustomer' selected. The second screenshot shows the 'Deploy New Service' dialog with 'REST' selected. The third screenshot shows the 'Specify Location of IBM i Program Object' dialog with 'RPGTRAIN' as the library name and 'CUSTSRVPGM' as the object name.

1

2

3

First three steps are the same since the procedures are all in one service program

Deploying a Web Service – UpdateCustomer

Application Servers Installations


Server: WEBSERVS - V2.6 (web services)

[WEBSERVS](#) > [Manage Deployed Services](#) > Deploy New Service

Deploy New Service


Specify Name for Service - Step 3 of 9

The Web service to be externalized is a resource. The URI path template identifies matching string or one or more template parameters that can contain regular expressions to further re:

Resource name: 

Service description:

URI path template: e.g. /temperature, /temperatur



Proceed to the next step of this task.

1. Define Parameters

- Resource should make sense for the procedure
- Always include a description of the Service
- URL path Template

2. Click **Next**

Deploying a Web Service – UpdateCustomer

Application Servers Installations

Server: WEBSERVS - V2.6 (web services)

Detect length fields

Use parameter name as element name for data structures

Export procedures: ?

Select	Procedure name/Parameter name	Usage	Data type
<input type="checkbox"/>	▶ WRITECUSTOMER_DATA		
<input checked="" type="checkbox"/>	▼ UPDATECUSTOMER_DATA		
	CUSTOMERDataDS	input	struct
	WrkCustNbr	input	zoned
	WrkUtilDS	output	struct
<input type="checkbox"/>	▶ DELETECUSTOMER_DATA		
<input type="checkbox"/>	▶ GETCUSTOMERRECS_DYNSELECT		
<input type="checkbox"/>	▶ GETCUSTOMER_DATAARECDS		
<input type="checkbox"/>	▶ GETCUSTOMER_DATA		

Select All Deselect Expand All Collapse All

Back Next Cancel

Proceed to the next step of this task.

Define Parameters

1. Deselect All procedures
2. Select
UPDATECUSTOMER_DATA
3. Set CUSTOMERDataDS as
an input parameter
4. Also WrkCustNbr As input

Click **Next**

Deploying a Web Service – UpdateCustomer

Application Servers Installations

Server: WEBSERVS - V2.6 (web services)

Procedure name: UPDATECUSTOMER_DATA

URI path template for resource: /UpdateCustomer/{custno:\d+}

HTTP request method: PUT

URI path template for method: *NONE or...

HTTP response code output parameter: *NONE

HTTP header array output parameter: *NONE

Allowed input media types: *JSON or...

Returned output media types: *JSON or...

Whether to wrap input parameters:

Wrap input parameters

Do not wrap input parameters

Input parameter mappings:

Parameter name	Data type	Input source	Identifier	Default
CUSTOMERDataDS	struct	*NONE		
WrkCustNbr	zoned	*PATH_PARAM	custno	*NONE

Back Next Cancel

Proceed to the next step of this task.

Resource Method Information

1. Change HTTP Method to **PUT**
2. Change Media types to JSON or XML
3. Mapping **WrkCustNbr**
 - Associate with custno
 - *PATH_PARAM will be in the URL of the request

Click **Next**

Deploying a Web Service – UpdateCustomer

1 Specify User ID for the service. The service requires a user ID. Specify User ID for the service.

2 Specify Library List. The functionality of the Web service program is determined by the library list. Specify library list position.

3 Specify Transport Information. Specify Transport Method.

4 Summary - Step 9 of 9. When you click Finish the web service will be deployed.

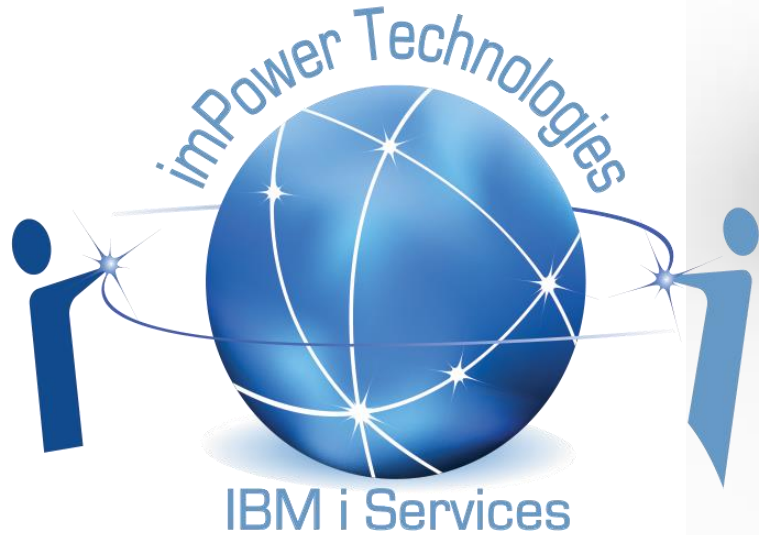
Service	Methods	Request Information
Resource name:	UpdateCustomer	
Resource description:	Update A Customer Record	
Service install path :	/www/webservers/webservices/services/UpdateCustomer	
URI path template:	/UpdateCustomer/{custno:ld+}	
User ID for service:	webservice	
Program:	/QSYS.LIB/RPGTRAIN.LIB/CUSTSRVPGM.SRVPGM	
Library list for service:	RPGTRAIN	

Save all of the pending changes for this task.

Last four steps are the same.

Make sure to check your options before clicking Finish!

Questions or Comments?



Jim Buck
Phone 262-705-2832
Email - jbuck@impowertechnologies.com
Twitter - @j_buck51

