So Now What?

Using IWS Server and Service programs
SEU doesn’t support the latest version of RPG.

Well I guess, you could turn off Syntax Checking!

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

5250 & SEU – Doesn’t work anymore!

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/](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  • IBM’s continue support for outdated IDE’s (PDM/SEU/SDA) Oh Yeah… RLU!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!

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

<table>
<thead>
<tr>
<th>Last Name</th>
<th>First Name</th>
<th>Street Address</th>
<th>City</th>
<th>State</th>
<th>Zip Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Albright</td>
<td>Scotty</td>
<td>8040 STATE ST.</td>
<td>CHICAGO</td>
<td>IL</td>
<td>60635-1289</td>
</tr>
<tr>
<td>Alvarado</td>
<td>Dennis</td>
<td>447 W. DARTMOO</td>
<td>GURNEE</td>
<td>IL</td>
<td>60031-3136</td>
</tr>
<tr>
<td>Amerine</td>
<td>Michael</td>
<td>789 S. ASH</td>
<td>LAWRENCE</td>
<td>TX</td>
<td>76550-0600</td>
</tr>
<tr>
<td>Barry</td>
<td>Tracy</td>
<td>32348 S. 39TH</td>
<td>GRAND RAPIDS</td>
<td>MI</td>
<td>49501-0002</td>
</tr>
<tr>
<td>Bayonne</td>
<td>Alfredo</td>
<td>10423 S.E. 30TH</td>
<td>BELLEVUE</td>
<td>WA</td>
<td>98007-0012</td>
</tr>
<tr>
<td>Bond</td>
<td>James</td>
<td>719 FIRST STRE</td>
<td>CAMBRIDGE</td>
<td>MA</td>
<td>21421-1123</td>
</tr>
<tr>
<td>Breneman</td>
<td>Jimmy</td>
<td>111 32ND AVE.</td>
<td>BOYCE</td>
<td>LA</td>
<td>71409-0008</td>
</tr>
<tr>
<td>Cho Cho</td>
<td>Deuk Hwan</td>
<td>1234 WEST ST.</td>
<td>LOCKHART</td>
<td>SD</td>
<td>29364-0000</td>
</tr>
<tr>
<td>Cook</td>
<td>Mike L</td>
<td>2478 E. MAIN S</td>
<td>ABILENE</td>
<td>TX</td>
<td>79604-1110</td>
</tr>
<tr>
<td>Cash</td>
<td>Johnny</td>
<td>1211 5TH STRE</td>
<td>LITTLE ROCK</td>
<td>AR</td>
<td>31214-5609</td>
</tr>
<tr>
<td>Coin</td>
<td>DOREEN</td>
<td>302 WASHINGTON</td>
<td>WHITE PLAINS</td>
<td>NY</td>
<td>11530-0039</td>
</tr>
<tr>
<td>Davis</td>
<td>Jeff</td>
<td>23 5TH STREET</td>
<td>KENOSHA</td>
<td>WI</td>
<td>51231-1234</td>
</tr>
</tbody>
</table>

54 rows fetched from cursor CUSTNAMECUR.

F3=Exit    F12=Cancel
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
CUSTSRVPGM – SQL Database I/O
GETSQLDIAG – SQL Database I/O
CUSTSRVCPY – Prototype Copybook
IBM’s IWS Server Solution

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

Integrated Web Services for IBM i

Web services made easy

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.
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
PCML Restrictions - Important

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

PCML Restrictions

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

- The following data types are not supported by PCML:
  - Pointer
  - Procedure Pointer
  - 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:
  - The data structure may not have any overlapping subfields.
  - The subfields must be coded in order; that is, the start position of each subfield must follow the end position of the previous subfield.
  - 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 compiler 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:

```xml
<data name="DATE" type="  length="19" usage="Input" />
</l.- Error: unsupported data type -->
```
PCML Restrictions - Important

```
// Data Structure for SQL Results
Dcl-De UtilDSSQL Int;
  MessageId Char(10);
  MessageId1 Char(7);
  MessageId2 Char(7);
  MessageLength Int(5);
  MessageText VarChar(50);
  ReturnedSQLCode Int(5);
  ReturnedSQLState Char(5);
  RowsCount Int(10);
End-De;
Dcl-s WrkCustMbr Zoned(6:0);

// Retrieves DB2 Data For CUSTOMER
```

Error! VarChar
Create a Web Services 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.

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.

**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**

**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

This is the completed IWS Customer Application:

- Compromised 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_IODataDS);
    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);
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

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

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

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

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

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

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 Authorities.¹
2. Click **Next**

¹ NOT QSECOFR! 😊
Deploying a Web Service – GetCustomer

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

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

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
Finding the URL to run

Final URL format (Bottom) Comprised of:
- URL Template
- Base Resource URL

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.

```sql
// ************************************************************
// * 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

First three steps are the same since the procedures are all in one service program.
Deploying a Web Service – DeleteCustomer

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

Define Parameters
1. Deselect all of the procedures
2. Select the DELETECUSTOMER_DATA procedure
3. Change the
   • WrkCustNbr to an input field, WrkUtilIDS will still be an Output field

Click Next
Deploying a Web Service – DeleteCustomer

1. Change HTTP Method to \textit{Delete}
2. Change Media types to JSON or XML
3. Mapping \textit{WrkCustNbr}:
   - Associate with custno
   - *PATH PARAM will be in the URL of the request

Click \textit{Next}
Deploying a Web Service – DeleteCustomer

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

First three steps are the same since the procedures are all in one service program.
Deploying a Web Service – AddCustomer

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 WRITECSTOMER_DATA
3. Set CUSTOMERDataDS as an input parameter
4. Also WrkCustNbr As input

Click Next
Deploying a Web Service – AddCustomer

**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

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(UtlDSSQL);
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

First three steps are the same since the procedures are all in one service program.
Deploying a Web Service – GetAllCustomers

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

Resource Method Information
1. Uncheck Detect length fields
2. Check
   GETCUSTOMER_DATARECDS
3. Parameters
   - Discussed on the next Slide

Click Next
Deploying a Web Service – GetAllCustomers

**Resource Method Information**

1. **Detect length fields** – Uncheck to allow you to associate:
   - 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

**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

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

First three steps are the same since the procedures are all in one service program.
Deploying a Web Service – UpdateCustomer

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

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

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

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