Introduction to
Web services for RPG developers

Claus Weiss clausweiss22@gmail.com

TUG meeting March 2011

Acknowledgement

In parts of this presentation I am using work published by:

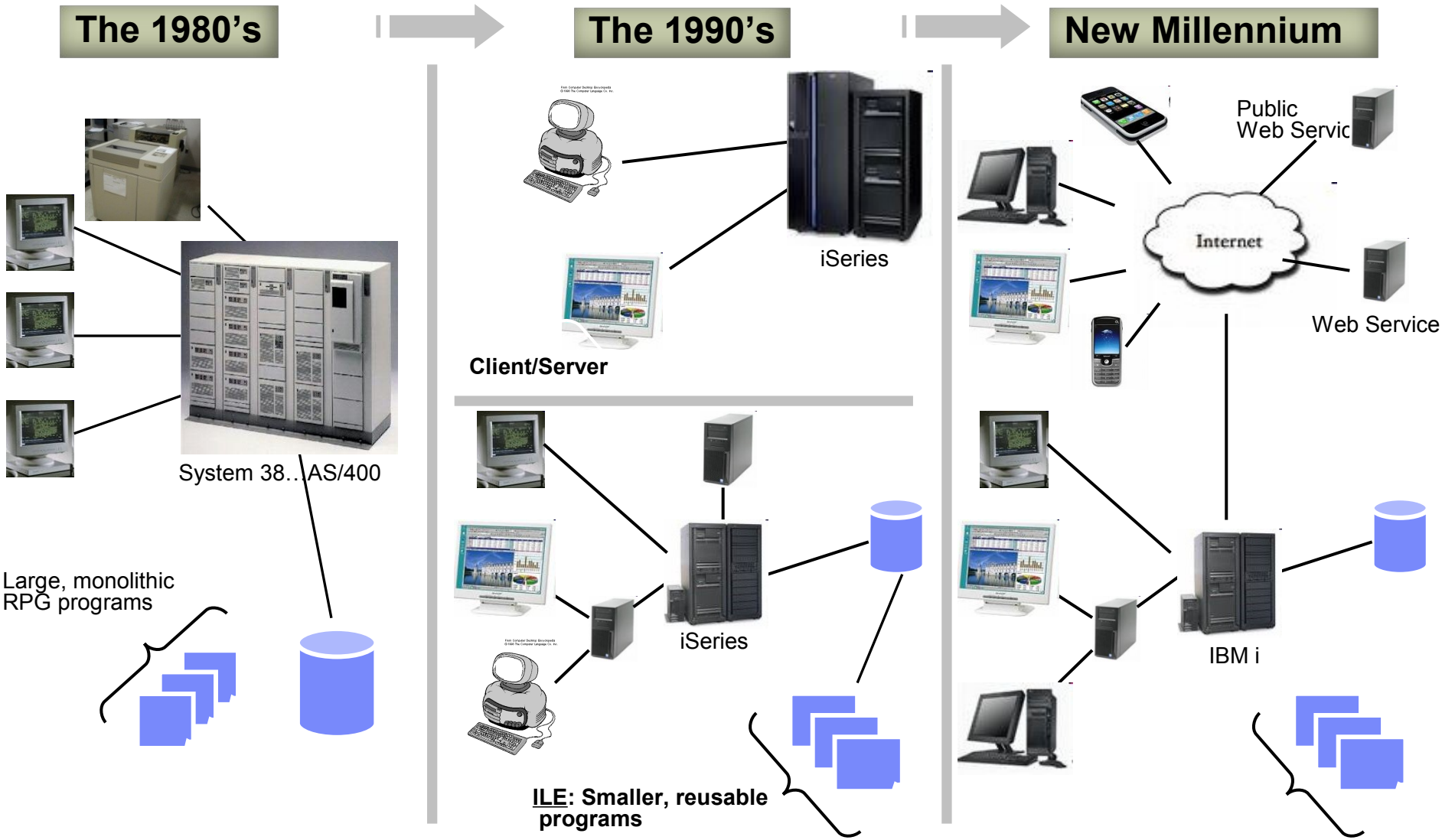
Linda Cole, IBM Canada

Scott Klement, Klement Sausage Co., Inc.

Agenda

- Web Services: What are they? Why use them?
- Creating a Web Service
  - Using RPG
  - Using EGL
- Consuming a web service
  - Using Web Service explorer
  - With EGL
  - With RPG using EGL or HTTPAPI

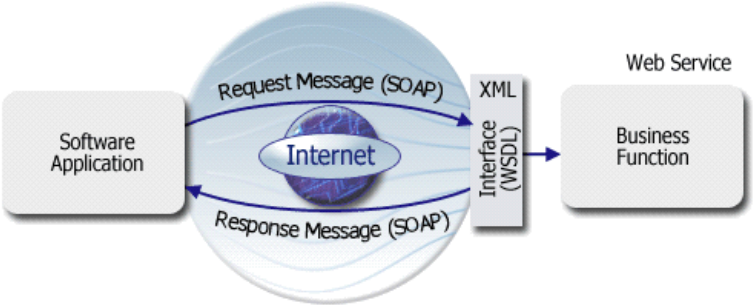
Change in the marketplace, IBM i, & App.Development



The Problem: Integration

- **Integrating software applications across multiple operating systems, programming languages, and hardware platforms is**
  - Difficult
  - Not something that can be solved by any one particular proprietary environment
- **Traditionally, the problem has been one of tight-coupling**
  - One application that calls a remote network is tied strongly to it by the function call it makes and the parameters it requests
  - Fixed interface to access remote programs or data, with little flexibility or adaptability to changing environments or needs

- **What are Web services:**
  - self-contained software components,
  - with well-defined interfaces (WSDL),
  - that can be invoked over a network using
    - XML and SOAP (for message formats)
    - XML Schema (for data types)
    - HTTP (for network transport)
    - WSDL (to describe Web service interface)



- **This allows applications to communicate independent of**
  - Hardware platforms
  - Operating systems
  - Programming languages

Types of web services

- SOAP **S**imple **O**bject **A**ccess **P**rotocol
  - Request is sent in SOAP document
  - Returns a SOAP document
- REST **RE**presentational **S**tate **T**ransfer
  - Request is send in URL
  - Returns a XML or json* document
- POX **P**lain o**l**d **X**ML
  - Request is sent in XML document
  - Returns a XML document

*JavaScript Object Notation

Web Service

- A Web Service is a special Web Application
  - A web application gets invoked by sending a request from a browser
  - A web service is a program that gets invoked by sending a request from a program
- They are both using the HTTP/HTTPS protocol to receive requests and return information to the requester.
  - Allowing to easily access the web service via the internet or intranet
- You can say a web service is a callable program that is accessible from anywhere.

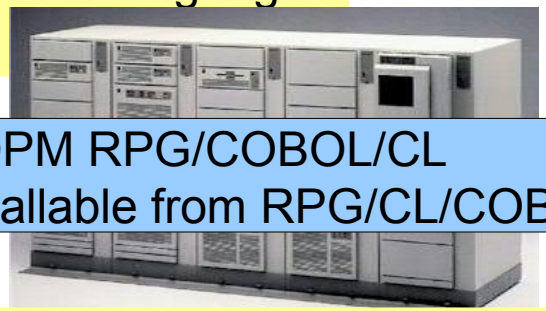
Evolution of accessing programs (services)

OPM

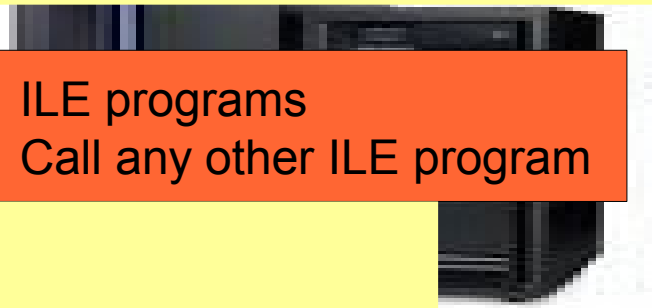
- Dynamic calls only
- Limited interlanguage calling

ILE

- Static and dynamic calls
- full interlanguage calling
- Modules and Serviceprograms



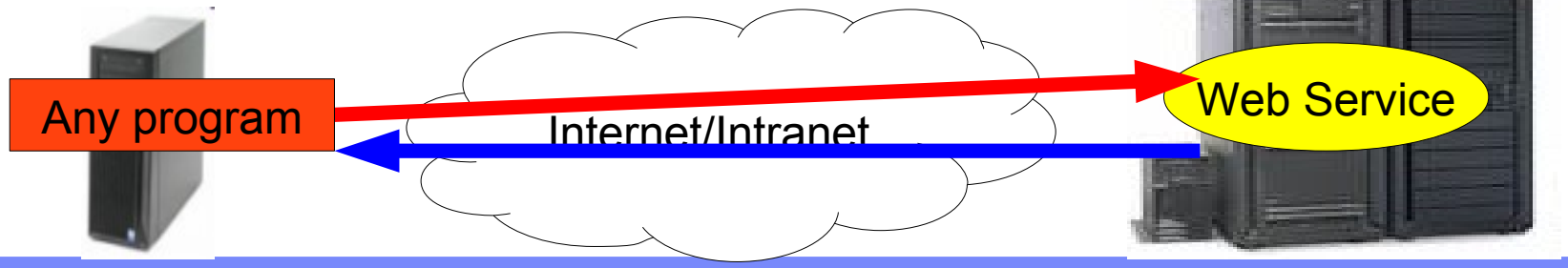
OPM RPG/COBOL/CL
Callable from RPG/CL/COBOL



ILE programs
Call any other ILE program

Web Service

- Dynamic invocation over the network
- Full interlanguage support



How does a SOAP Web Service work

Web Service requester (consumer client)

Send Request with document describing the request and data

```
<soapenv:Envelope xmlns:soapenv="http://schemas.....">
  <soapenv:Header/>
  <soapenv:Body>
    <web:NumberToDollars>
      <web:dNum> 095.00 </web:dNum>
    </web:NumberToDollars>
  </soapenv:Body>
</soapenv:Envelope>  ........
```

Web Service NumberToDollars

Send response back with document including data

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <m:NumberToDollarsResponse xmlns:m="http://www.dataaccess.com/webservicesserver/">
      <m:NumberToDollarsResult>ninety five dollars</m:NumberToDollarsResult>
    </m:NumberToDollarsResponse>
  </soap:Body>
</soap:Envelope>
```

SOAP message details

- A protocol defining how the input/output data of a web service is sent
- Send and receive data in XML documents
- XML Documents follow SOAP standard

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:web="http://www.dataaccess.com/webservicesserver/">
  <soapenv:Header/>
  <soapenv:Body>
    <web:NumberToDollars>
      <web:dNum> 095.00 </web:dNum>
    </web:NumberToDollars>
  </soapenv:Body>
</soapenv:Envelope>
```

Extra info: authentication, etc

Operation name: (program/sub procedure/function name and parameter data)

Documenting the Web Service

- How do you tell other people about your Web Service
  - Where is it located
  - What is the name
  - What input parameters does it except
  - What output parameters does it return

- You could create a document, a web page etc

- SOAP Web Services are described in a WSDL file
  - **W**eb **S**ervices Description Language
    - XML style to describe a Web Service

WSDL details

```
<definitions>
<types>
 data types the service uses........      </types>
```

```
<message name="NumberToDollarsSoapRequest">
   <part name="parameters" element="tns:NumberToDollars"/>
 </message>
<message name="NumberToDollarsSoapResponse">
   <part name="parameters" element="tns:NumberToDollarsResponse"/>
 </message>
<portType name="NumberConversionSoapType">
<operation name="NumberToDollars">
   <documentation>Returns the non-zero dollar amount of the passed number.</documentation>
   <input message="tns:NumberToDollarsSoapRequest"/>
   <output message="tns:NumberToDollarsSoapResponse"/>  </operation>  </portType>
```

Messages sent and received by service

Operations (programs/procedures) you can use/call in service

```
<binding>
Network protocol used in service          </binding>
<service>
A grouping of services/ports (like service program containing multiple procedures)
</service> </definitions>
```

Human readable, but more important tools can work with it easily

Terminology used so far

- ✔ Web Service
- ✔ SOAP
- ✔ WSDL
- UDDI Universal Description, Discovery and Integration
  - Registry standard for Service Oriented Architecture
  - A public repository containing web service descriptions

- **UDDI did not become accepted as the standard registry for Web Services**

UDDI overview

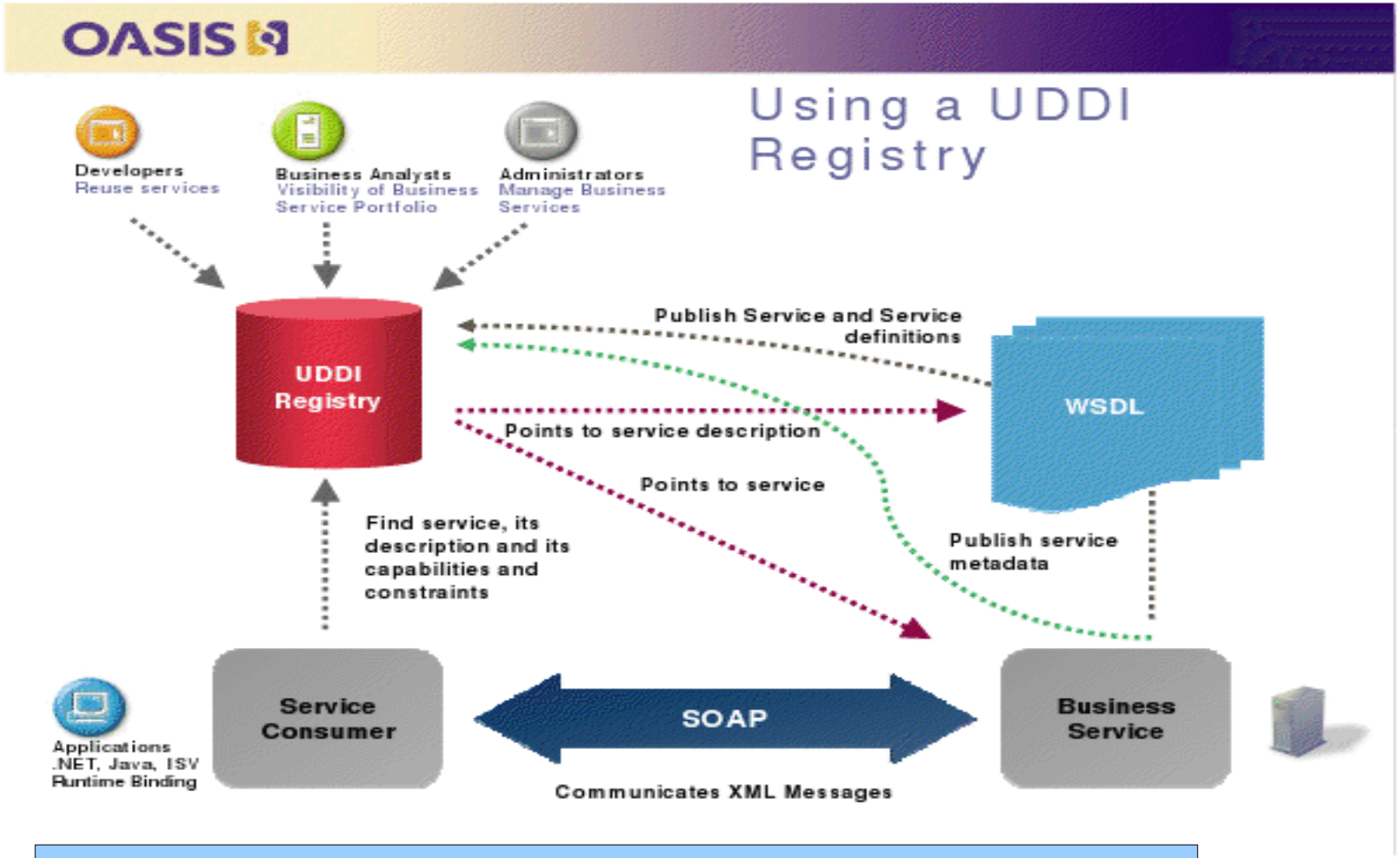


Chart created by UDDI OASIS Standard community

Why use a Web Service

- Somebody else created something that you want to use
  - Google services very popular (maps .....)
  - Yahoo for business information (show me realtors for that postal code)
  - Simple conversion routines (metric to .....)
  - Currency exchange rates (Dollar to Euro)
  - Shipment tracking (UPS, FedEx ....)
  - Your supplier implemented a web service (orders status inquiry)
  - You want to give your customers access to data (parts on hand)
  - ....
  - Many many other web services that might come in handy

Web 2.0 application using Web Services

- Sample application written in EGL

MainHandler.egl

Calculator

Loan Amount: $180,000.00

Interest Rate: 5.20000000

Term: 15

Calculate

$1,442.25

Results

Interest

Principal

Map

Search for local mortgage businesses

Zip code: 94101 Search

Integrated Mortgage

Certified Home Mortgage

Residential Pacific Mortgage Company

Zmax Mortgage Corporation

Guarantee Mortgage Corporation

California Financial Mortgage

My Low Mortgage

Alternative Mortgage Sources

Bank of America Mortgage

Map Satellite

Map data ©2011 Google - Terms of Use

YAHOO Web Service to show mortgage businesses In the area

Web Service to calculate monthly payment

GOOGLE Web Service to show locations on map

Creating and using a Web Service

- Now that you have an understanding what a Web Service is and which pieces make up a Web Service
- Let's look at
  - How to create a Web Service
  - How to use/consume a Web Service

Agenda

- **Web Services: What are they? Why use them?**
- **Creating a Web Service**
  - **Using RPG**
  - **Using EGL**
- **Consuming a web service**
  - **Using Web Service Explorer**
  - **With EGL**
  - **With RPG using EGL or HTTPAPI**

Use RPG to create a Web Service

- Write an RPG program or a sub procedure in a service program or use existing
- Use a tool to create a wrapper around the RPG code
  - Wrapper will create
    - The XML definition to handle a request that consumes your Web Service
    - The XML definition to send back the response from your Web Service
    - A WSDL file that describes your Web Service and its location
    - Code to call the RPG program and pass the parameter values from the Web Service request
    - Code to handle the return values from your RPG program and include them in the response XML document
  - Deploy your Web Service wrapper to a server

How it Works – Creating a Web Service with RPG

First: Create program

Zipcode in

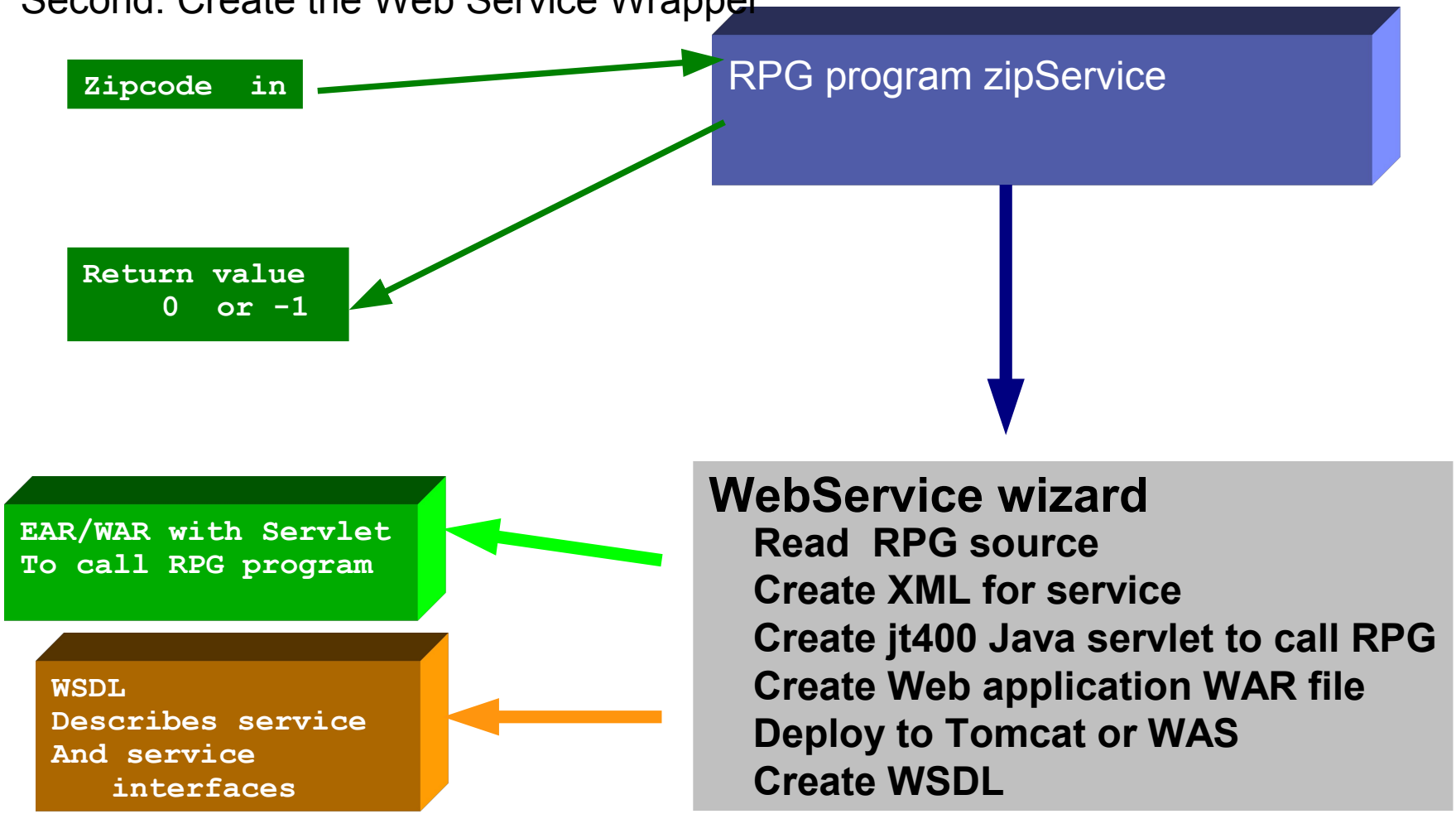
Return value
0 or -1

RPG program zipService

```
D ZIPSERVICE      PR
D zipin                          5a
D zipout                         5a
D ZIPSERVICE      PI
D zipin                          5a
D zipout                         5a
 /Free
  if zipin >= '90001' and zipin <= '96000';
     zipout = '0' ;
   else;
     zipout = '-1';
   endif;
   *inlr = *on;
```

How it Works – Creating a Web Service with RPG

Second: Create the Web Service Wrapper



How it Works – Creating a Web Service with RPG

Test Web Service

Web Service explorer
- Read WDSL
- Create front end to send request with data
- Display response

RPG program zipService

Zipcode in

Return value

Call to RPG program

Request with XML document containing Zipcode in

Response with Return value

App Server

WAR with Servlet To call RPG program

WSDL Describes service And service interfaces

Run the RPG Web Service

Input to service

Web Services Explorer

Navigator

WSDL Main
file:/C:/Users/Inge/IBM/rationalsdp/workspacerbc
ZipServiceService
ZipServiceBinding
zipServicefunctionName
file:/C:/Users/Inge/IBM/rationalsdp/workspacerbc
ZIPSERVICEServicesService
ZIPSERVICEServicesSoapBinding
zipservice_XML
zipservice

Actions

Endpoints

http://localhost:9081/rpgZipService/services/ RVICEServices

Body

zipservice

inputData nil?

ZIPIN string nil?

98000

Output from service

Status

Body

zipserviceResponse

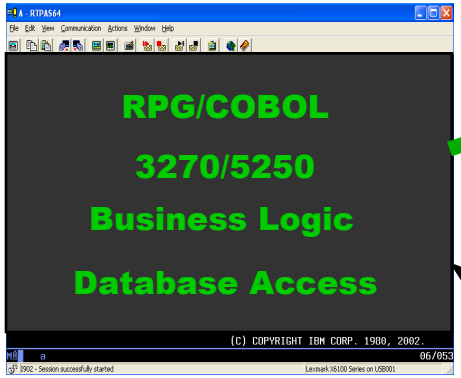
zipserviceReturn

ZIPOUT (string): -1

Web Service

Creating a Web Service using a 5250 Application

Existing Programs

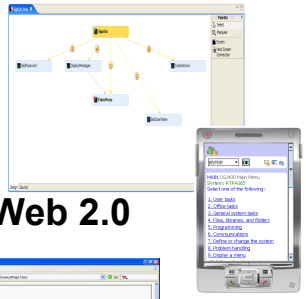


Generate Web Services from HATS and consume them with requests and responses

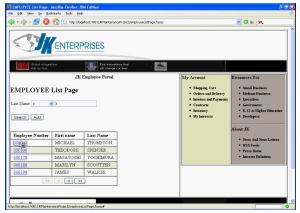
HATS Services Wizard

WSDL Describes service and service interfaces

Web Service



Web 2.0



Web Services clients

Input fields on screens map to request parameters
Output fields on screens map to response data

Use EGL to create a Web Service

- Enterprise Generation Language (EGL) **------> now open source !!!!!!!**
- Free download available !!!!!!!
- EGL contains a construct called a Service part
  - Like a function/subprocedure
- In the Deployment Descriptor you tell the generator what kind of service to create for the logic you code in the Service part
  - SOAP, REST, or EGL service

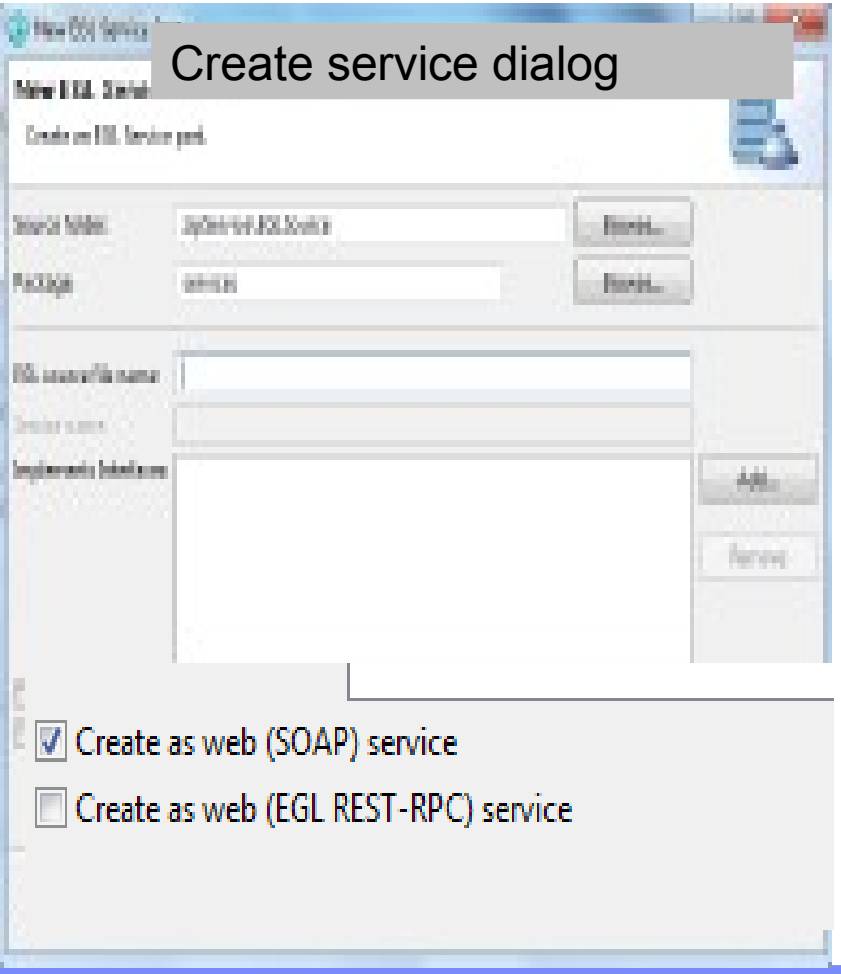
Steps for Creating a Web Service with EGL

- **Steps**
  - Create or use existing EGL project or EGL Web project
  - Create a new service in EGL
  - Tell EGL what kind of service you want to create (deployment descriptor)
  - Use EGL to generate Web Service and WSDL

- **Deploy the Web Service**
- **Use the Web Service wizard to test the service**

How it Works – Creating a Web Service with EGL

Create program/service



EGL program zipService

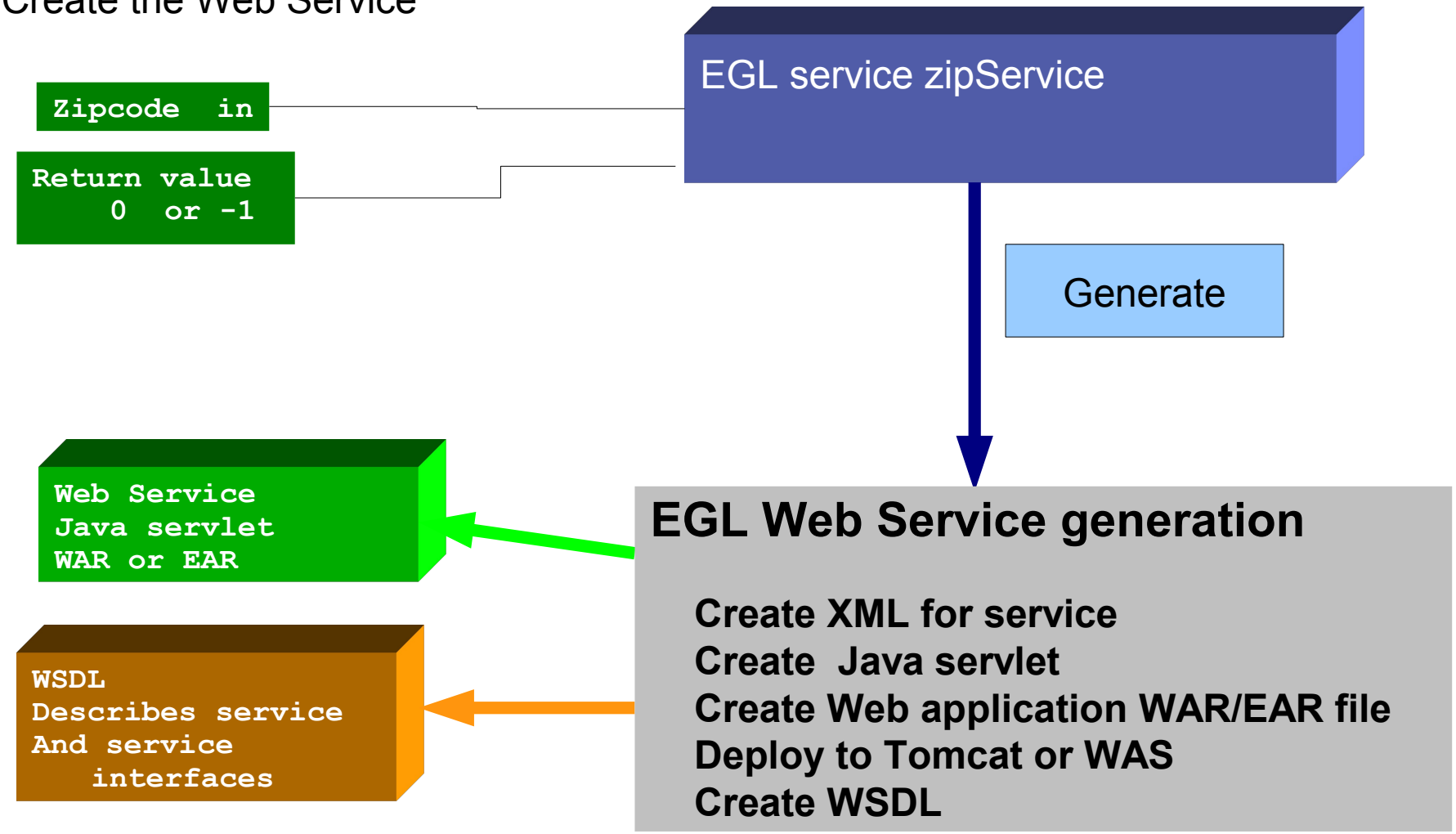
```
service ZipService

function zipServicefunction(zipin
                string in, zipout string out)

 if (zipin >= "90001" && zipin <= "96000");
   zipout = "0";
 else
   zipout = "=-1";
 end
end
end
```

Zipcode in

Return value
0 or -1

How it Works – Creating a Web Service with EGL

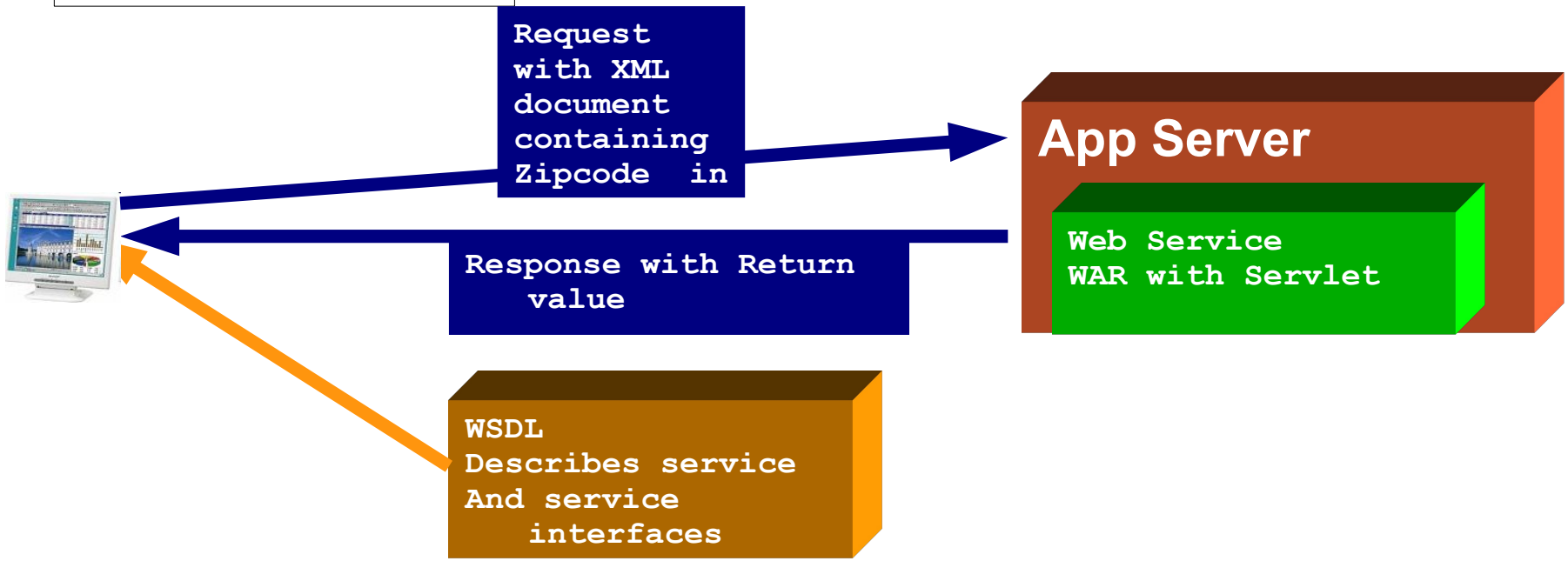
Create the Web Service



How it Works – Creating a Web Service with EGL

Test Web Service

Web Service Explorer
- Read WDSL
- Create front end
  - to send request
  - with data
- Display response



Run the EGL Web Service

Input to service

Web Services Explorer

Navigator

WSDL Main
file:/C:/Users/Inge/IBM/rationalsdp/workspacerbc
ZipServiceService
ZipServiceBinding
zipServicefunctionName

Web Service

Actions

Endpoints

http://localhost:9081/zipServ ervices/ZipService

Body

zipServicefunctionNam

zipin string

95000

Status

Output from service

Body

zipServicefunc nNameResponse

zipout (string): 0

Creating a Web Service

- **Use any programming language**
  - **Java, C, C++ supported via AXIS project**
  - **AXIS provides functionality for creating Web Services wrappers**
    - **Creates proxis for Web Service**
    - **You write the logic, AXIS provides the Web Services support**
    - **Apache open source project**

Overview on the Apache AXIS Web Page

The well known Apache Axis, and the second generation of it, the Apache Axis2, are two Web Service containers that help users to create, deploy, and run Web Services.

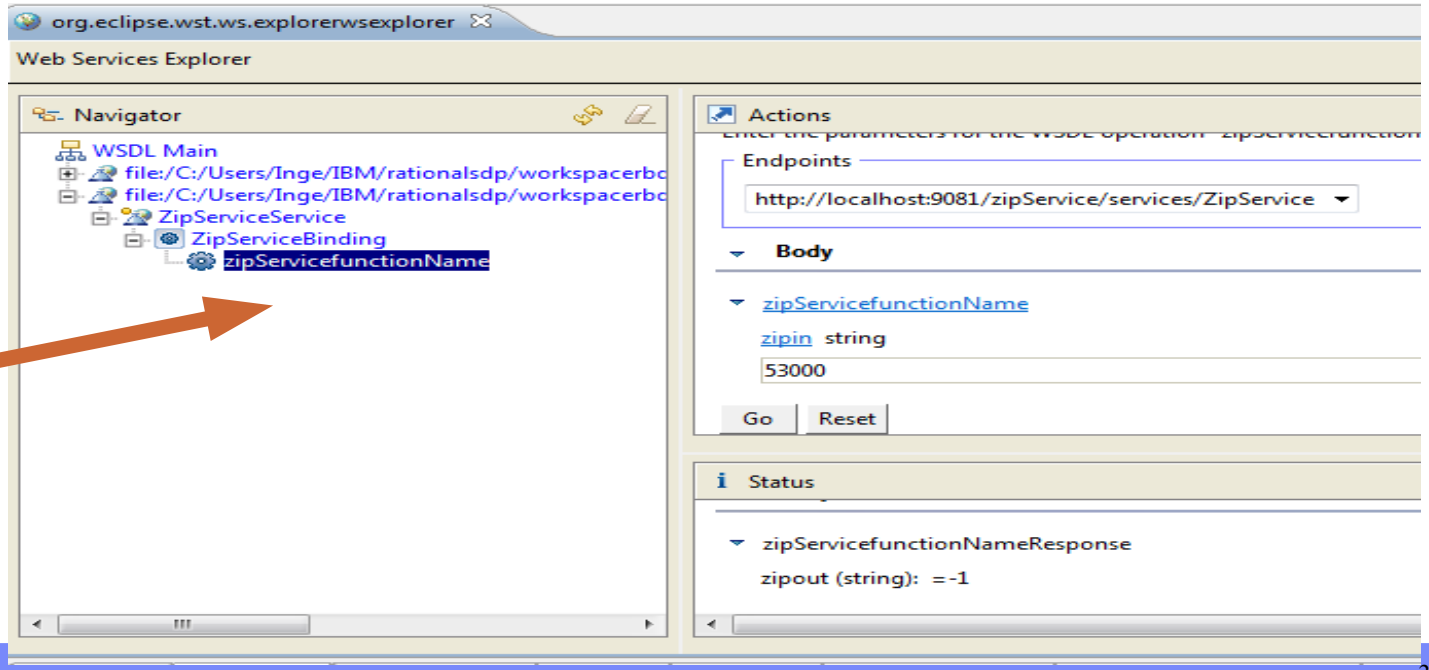
Agenda

- **Web Services: What are they? Why use them?**
- **Creating a Web Service**
  - **Using RPG**
  - **Using EGL**
- **Consuming a web service**
  - **Using Web Service Explorer**
  - **With EGL**
  - **With RPG using EGL or HTTPAPI**

Consuming a Web Service using a tool/wizard

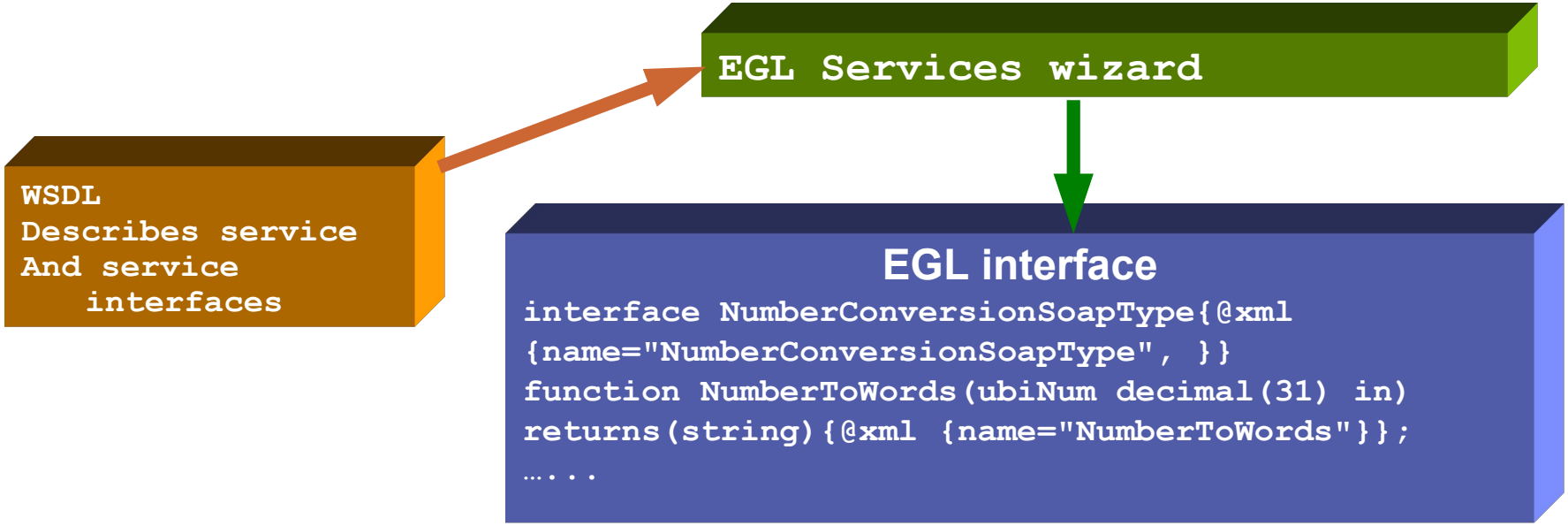
- Several tools on the web download and try
  - SOAPUI very popular
- Use Web Service Explorer, part of RBD
  - Point Explorer to WSDL
  - Extracts Input/Output parameter, invocation, and URI information
  - Prompts for input, sends request, shows response

WSDL
Describes
service
And service
interfaces



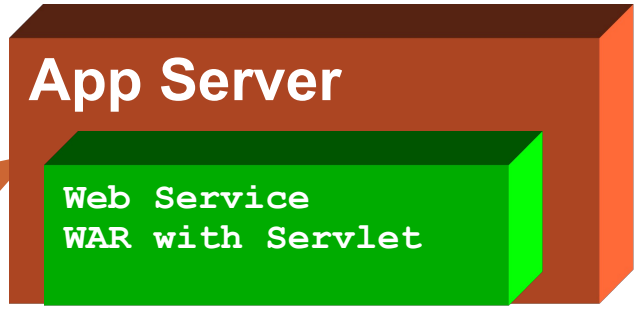
EGL consuming a service

- Using EGL Web Service wizard
  - Get service description from WSDL
  - Create interface (like prototype in RPG) from it
  - Interface can be invoked like a function to consume Web Service



EGL consuming a service

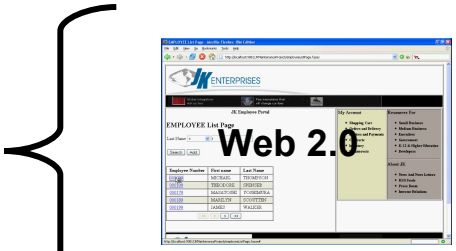
- Send request to Web Service via Interface
- Get response back from Web Service



EGL interface

`NumberConversionSoapType`

```
…..
numConv NumberConversionSoapType{@BindService {} };
result = numConv.NumberToWords(numberInput );

.....
```



RPG consuming Web Service using Java generated by EGL

- **Steps**
  - **Create EGL program consuming Web Service**
  - **Enable property, callable from native Java**
  - **Generate Java jar**
  - **Copy jar to IBM i**
  - **Write RPG program using Java call capability to consume Web Service**
  - **Compile**
  - **Try**

RPG consuming Web Service using Java generated by EGL

EGL logic

ILE business logic

Prototypes to call Java

Generated Java program sends request to Web Service and gets response

Wrapper program

Programc3Wrapper

2 input parms

1 response parm

```
D* this is the constructor for the EGL/JAVA program wrapper
D new_wrap          PR                O    EXTPROC(*JAVA : 'programs.Programc+
D                                           3Wrapper' : *CONSTRUCTOR)
D*
D* call method for wrapper program object
D*  2 input parameters, response will be in third parameter
D  callmeth         PR                     ExtProc(*JAVA:
D                                          'programs.Programc3Wrapper':
D                                          'call')
D parm1                            20i 0 value
D parm2                             O    Class(*JAVA:'java.math.BigDecimal')
D parm3                             O    Class(*JAVA:'java.lang.String')
```

RPG consuming Web Service using Java generated by EGL

EGL logic

ILE business logic

call to Java wrapper

Generated Java program sends request to Web Service and gets response

Wrapper program

Programc3Wrapper

2 input parms

1 response parm

```
D mypgm1            s                  o    class(*JAVA
D                                 'programs Programc3Wrapper')
D response          s                 50a    varying

    mypgm1 = new_wrap();
    //  send request to web service
    callmeth(mypgm1: parm1: parm2 :parm3);
    // response from Web Service is stored in parm3
    // method getmyparm3 gets the response value
    response = getbytes( getmyparm3(mypgm1));
```

App Server

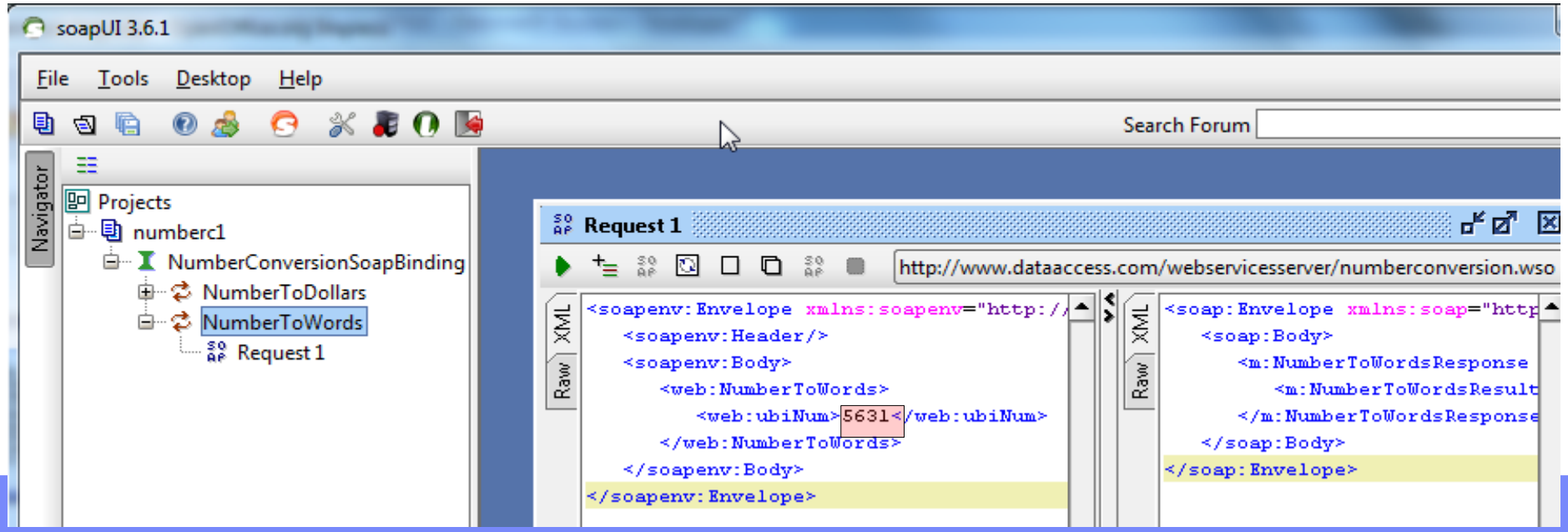
Web Service WAR with Servlet

Steps for consuming a Web Service using HTTPAPI

- **Steps**
  - **Download and restore HTTPAPI library**
  - **Create XML for request document in RPG (sounds more difficult than it is)**
  - **Write RPG code to call HTTPAPI program**
  - **Write RPG sub procedure to get data from response document**
  - **Compile**
  - **Try**

Use soapUI to extract XML document

- **Create a project in soapUI**
  - **Use Web Service WSDL**
  - **soapUI extracts the XML for request document**
  - **Copy/paste XML into RPG and edit XML**
  - **Assign string to variable to pass to HTTPAPI**



RPG consuming Web Service using HTTPAPI

The request XML document

RPG Variable SOAP gets document

pchar RPG variable contains data

```
SOAP = '<soapenv:Envelope +
       xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  +
       xmlns:web="http://www.dataaccess.com/webservicesserver/">  +
       <soapenv:Header/> +
       <soapenv:Body> +
         <web:NumberToWords> +
           <web:ubiNum>'  + pchar + '</web:ubiNum> +
          </web:NumberToWords> +
        </soapenv:Body> +
        </soapenv:Envelope> ';
```

RPG consuming Web Service using HTTPAPI

The prototype for http_post_xml

ILE business logic

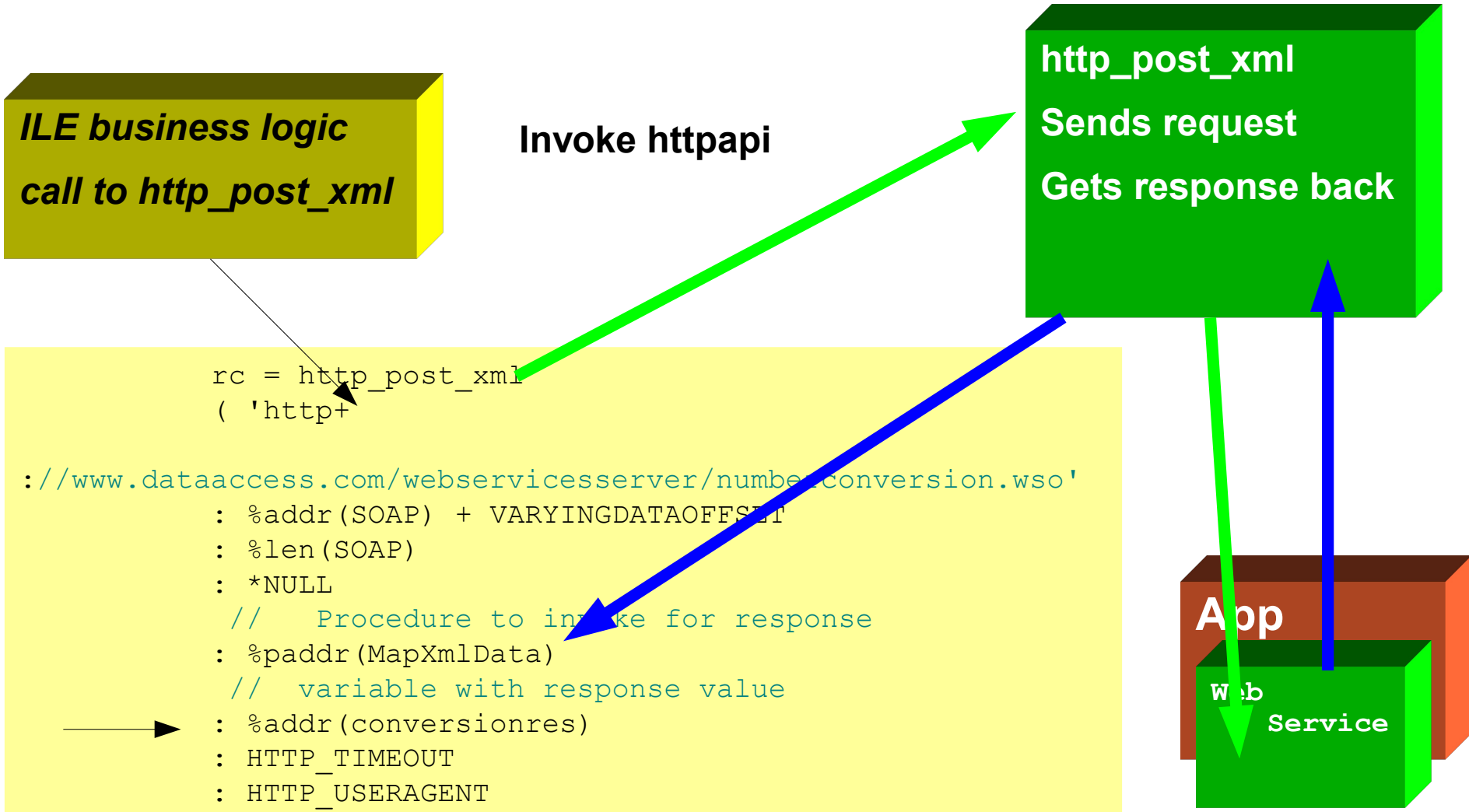
Prototypes to use HTTPAPI

HTTPAPI sends XML from RPG

Calls sub procedure with Response data

```
     D http_post_xml...
     D                 PR            10I 0 EXTPROC('HTTP_URL_POST_XML')
1    D  peURL                     32767A   varying const options(*varsize)
2    D  pePostData                     *   value
3    D  pePostDataLen              10I 0 value
4    D  peStartProc                    *   value procptr
5    D  peEndProc                      *   value procptr
6    D  peUsrDta                       *   value
7    D  peTimeout                  10I 0 value options(*nopass)
8    D  peUserAgent                64A   const options(*nopass:*omit)
9    D  peContentType              64A   const options(*nopass:*omit)
10   D  peSOAPAction               64A   const options(*nopass:*omit)
```

RPG consuming Web Service using HTTPAPI




```
          rc = http_post_xml
          ( 'http+

://www.dataaccess.com/webservicesserver/numberconversion.wso'
          : %addr(SOAP) + VARYINGDATAOFFSET
          : %len(SOAP)
          : *NULL
           //   Procedure to invoke for response
          : %paddr(MapXmlData)
           //  variable with response value
          : %addr(conversionres)
          : HTTP_TIMEOUT
          : HTTP_USERAGENT
          : 'text/xml; charset=UTF-8'
          : '""' );
```

RPG consuming Web Service using HTTPAPI

ILE business logic

callback sub procedure

Callback sub procedure

HTTPAPI gets response document

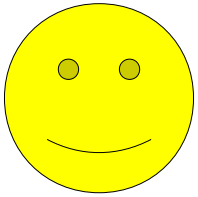
Calls sub procedure with Response data

```
P MapXmlData       B
D MapXmlData       PI
D   result                      52a   varying
D   depth                       10I 0 value
D   name                      1024A   varying const
D   path                     24576A   varying const
D   value                    65535A   varying const
D   attrs                         *   dim(32767)
D                                     const options(*varsize)
 /free
    if (name = 'm:NumberToWordsResult')
     result =value;
    endif;
```

Get this name from soapUI

Summary

- **RPG developers can**
  - **create Web Services on IBM i**
  - **consume Web Services on IBM i**
- **Choose from a variety of tools**
  - **Depending on your skills**
  - **Depending on your preferences**



Yes, you can do it
Come to TUG night school and try it out

IBM Community Sites for Business Developers

IBM. English Sign in (or register)

developerWorks. Technical topics Evaluation software Community Events Search developerWorks

developerWorks > Rational >

IBM Rational Cafés

Connecting Communities

Cafés home | C/C++ | COBOL | EGL | RPG

Join. Download. Learn.

developerWorks.

COBOL Get the latest technical tips, business news, conference, seminar, webcast and event information about COBOL and related topics.
→ Learn more

EGL Discover how this revolutionary technology can help you deliver innovative business solutions, fast.
→ Learn more

C/C++ Leverage the most from your IBM C/C++ compiler products by using blogs and on-line forums to facilitate conversations.
→ Learn more

RPG Interface with RPG and IBM developers to mingle and discuss our favorite language - RPG.
→ Learn more

COBOL
- COBOL group
- COBOL forum
- RDz hub
- RDz forum
- RDz Unit test hub
- RDz Unit test forum
- Learn COBOL and RDz
- RD Power hub
- RD Power forum

ibm.com/developerworks/rational/community/cafe/index.html


Questions


Thank
You