Service-Oriented Architecture for the RPG Programmer

George Farr  
IBM  
WW System i tools and compilers products manager  
farr@ca.ibm.com

Joe Pluta  
Pluta Brothers Design  
joepluta@plutabrothers.com

Agenda

- Packaging and rebranding.
  - RDi and RDi SOA Value proposition
  - RDi Functional capabilities
  - RDi SOA Functional capabilities
  - EGL For RPG Programmers
  - Why is SOA Important to RPG Programmers?
  - A Quick Overview of EGL
    - Rich Data
    - EGL Syntax
  - Putting it All Together
  - Conclusion
What System i customers get today.

- ILE RPG
- S/36 Compatible RPGII
- S/38 Compatible RPGII
- RPG/400
- ILE RPG *PRV Compiler
- ILE COBOL
- S/36 Compatible COBOL
- S/38 Compatible COBOL
- OPM COBOL
- ILE COBOL *PRV Compiler
- ILE C
- ILE C++
- IXLC for C/C++
- ADTS
- WDSC

New Packaging for WDS

**WDS V5R4**
- ILE RPG
- S/36 Compatible RPGII
- S/36 Compatible RPGII
- RPG/400
- ILE RPG *PRV Compiler
- ILE COBOL
- S/36 Compatible COBOL
- S/38 Compatible COBOL
- OPM COBOL
- ILE COBOL *PRV Compiler
- ILE C
- ILE C++
- IXLC for C/C++
- ADTS
- WDSC

**WDS V6R1**
- Feature: ILE Compilers
  - ILE RPG
  - ILE RPG *PRV Compiler
  - ILE COBOL
  - ILE COBOL *PRV Compiler
  - ILE C
  - ILE C++
  - IXLC for C/C++

- Feature: Heritage Compilers
  - S/36 Compatible RPG II
  - S/36 Compatible RPG II
  - RPG/400 (RPG III)
  - S/36 Compatible COBOL
  - S/38 Compatible COBOL
  - OPM COBOL

- Feature: ADTS
  - ADTS

- Supply Feature
  - EOS 2010
New Client products: RDi & RDi SOA

- New Rational products
  - RDi – lightweight Eclipse based development tool for RPG, COBOL, and CL programs
    - Includes LPEX, RSE, and End to End Debugger
    - New features include: Application Diagram and the Screen Designer
  - RDi SOA – EGL, SOA, Web tools
    - RDi + Rational Business Developer (RBD) with EGL
    - Tools for Web Design
    - Web Services from RPG/COBOL
    - Bundle for developers who need end-to-end web and SOA solutions

Agenda

- Packaging and rebranding.
- RDi and RDi SOA Value proposition
- RDi Functional capabilities
- RDi SOA Functional capabilities
- EGL For RPG Programmers
- Why is SOA Important to RPG Programmers?
- A Quick Overview of EGL
  - Rich Data
  - EGL Syntax
- Putting it All Together
- Conclusion
Who could benefit from WDS, RDI and RDI SOA

–Customers who have...

• IBM i

<table>
<thead>
<tr>
<th>Category</th>
<th>New capability</th>
<th>How capability addresses client needs</th>
<th>Pre-Rerequisites</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notice Development</td>
<td>Sub-System language support</td>
<td>• Provides tooling to call, compile, and debug the build language specifications</td>
<td>RDS v7.1</td>
</tr>
<tr>
<td>SOA</td>
<td>Customizability of generated code</td>
<td>• Allows customizability of all generated language definitions</td>
<td>RDS SOA v7.1</td>
</tr>
<tr>
<td>Application Modernization</td>
<td>Program understanding</td>
<td>• Provides tooling to analyze, debug, and optimize generated programs with graphical representation</td>
<td>RDS v7.1</td>
</tr>
<tr>
<td>Application Modernization</td>
<td>RPG compiler</td>
<td>• Provides language enhancements to meet software requirements</td>
<td>RDS v6r1</td>
</tr>
<tr>
<td>Scalability</td>
<td>RPG Multi-threading</td>
<td>• Allows RPG modules to run concurrently in multiple threads</td>
<td>RDS v6r1</td>
</tr>
<tr>
<td>System integration</td>
<td>COBOL compatibility</td>
<td>• Provides easier migration for COBOL applications between IBM i &amp; z</td>
<td>RDS v6r1</td>
</tr>
<tr>
<td>Maintainability</td>
<td>Integrated object interface</td>
<td>• Provides tooling to call, compile, and debug the build language specifications for IBM i &amp; z</td>
<td>RDS v6r1</td>
</tr>
</tbody>
</table>

Who benefits

• Customer migration from/to IBM i & z
• Customer seeking easy way to integrate SOA with RPG and COBOL
• Customer seeking high transaction processing on System i through RPG multi-thread support
• Customer seeking to understand and re-architect existing IBM i applications

Key capabilities for WDS
Product Overview and What’s new in V6R1

• Highlights
  – Repacking IBM i based on roles and strategy
    – ILE/Heritage Compilers, and ADTS
    – RPG multi-threaded support
    – RPG XML consumability improvements
    – RPG & COBOL customer satisfaction improvements
    – Improved COBOL IBM i & z compatibility
    – Major restructuring of Rational tools offerings
    – Language and system integration
Key capabilities WDS

Modernize your business logic with the latest features of RPG IV
- Take advantage of more powerful hardware using multi-threaded
- Better integration with Service Oriented Architecture (SOA)
- Expanded XML capabilities
- Organize and modularize your code with new scoping capabilities

What’s new for 1Q08?
- Ability to run concurrently in multiple threads
- Specify Main procedure to be called
- Local File support
- Enhancements related to the use of files;
- Significantly higher limits for the size of variables
- Relaxation of some UCS-2 rules (available starting in V5R3 through PTFs).
- Miscellaneous enhancements

Key capabilities – Rational Developer for System i (RDi)

Product overview and What’s new

What’s new for 1Q08?
- Packaging
  - Repackage WDSC components into RDi
  - Renaming and rebranding
- V6R1 Support
- Modernization
  - Application Diagram enhancements
  - Screen Designer
- SOA*
  - White-Box Direct Program Call runtime built on top of toolbox
  - IBM involvement

Who benefits
- Low cost RPG/COBOL/Java solution for System i customers
- RPG/COBOL customers looking for a simple yet effective business development solution
- Customers looking for a low learning curve solution to exploit their existing business logic
Rational Developer for IBM i for SOA construction (RDi SOA)

• Highlights
  - RDi SOA combines Rational Developer for IBM i and Rational Business Developer (RBD) offering a single robust, productive and easy to learn development environment
  - New RBD packaging and pricing
  - Improved EGL IBM i integration
  - IBM i Java SOA improvements

• Why use it?
  - Improve performance, maintainability and scalability of Java SOA solution
  - Rational Developer for IBM I for SOA Construction (RDi SOA) offers a complete solution for rapid development of modern and rich web applications and web services that extend existing RPG or COBOL investments
  - IBM i developers seeking a simple yet effective development solution to rapidly design, develop and test an optimized Java SOA solution for IBM
  - Easy exploitation of existing System i logic
  - Primary web development tools platform for System i

• Who benefits
  - Customers seeking to build an optimized Java SOA solution for IBM i
  - Low cost SOA solution for IBM i customers
  - RPG,COBOL customers looking for a simple yet effective Web development solution
  - Customers looking for a low learning curve solution to exploit the latest Web technologies

• What’s new in 7.1
  - In addition to the RDi and RBD enhancements, also delivers powerful tooling for rendering legacy RPG and COBOL programs as Web Services.

New Product
  > RDi SOA v7.1

Supported Release
  > i V6R1

Session 20556

Agenda

• Packaging and rebranding.
• RDi and RDi SOA Value proposition

RDI Functional capabilities

• RDI SOA Functional capabilities
• EGL For RPG Programmers
• Why is SOA Important to RPG Programmers?
• A Quick Overview of EGL
  > Rich Data
  > EGL Syntax
• Putting it All Together
• Conclusion

Session 20556
RDi is the Only IDE that Supports RPG and COBOL

- RDi has features that aren't even in the Java editor, which is Eclipse's baby
- Add in the Application Diagrammer and the Screen Designer, and RDi is the top of the line editor for IBM i application development

And now there is the Remote System Explorer (RSE)
RDi by Itself is Enough for Core Business Application Maintenance

- RDi is not the only component of the Rational tooling strategy for the IBM i
- Other products, including RDi-SOA, are available, with others coming

---

Agenda

- Packaging and rebranding.
- RDi and RDi SOA Value proposition
- RDi Functional capabilities
- RDi SOA Functional capabilities
- EGL For RPG Programmers
- Why is SOA Important to RPG Programmers?
- A Quick Overview of EGL
  - Rich Data
  - EGL Syntax
- Putting it All Together
- Conclusion
The equation → RDi SOA = RDi + RBD

- RDi-SOA combines core business application maintenance with future architecture
- You can use RDi to maintain your existing applications
- Then you can use LPEX editor to write new business logic in ILE languages
- That business logic can then be exposed via RBD tooling to the web
- This is the "SOA" part of RDi-SOA, and represents the fastest way for ILE developers to being producing true web applications

RDi SOA is the Only IDE that Supports "i" Languages as SOA Components

Build entire multi-tiered service-oriented applications using ILE languages as a primary component
RDi-SOA is Designed to Combine Traditional IBM i Languages with the Advanced Features of EGL

- All the capabilities of ILE
  - Tight integration to database
  - High performance
  - Ease of development and debugging

- All the features of EGL
  - No plumbing
  - High-level data abstraction
  - Encapsulation of web technologies

- Nothing gets complex business logic to the web faster!

Agenda

- Packaging and rebranding.
- RDi and RDi SOA Value proposition
- RDi Functional capabilities
- RDi SOA Functional capabilities
- EGL For RPG Programmers
- Why is SOA Important to RPG Programmers?
- A Quick Overview of EGL
  - Rich Data
  - EGL Syntax
- Putting it All Together
- Conclusion
Drilling into the details... Why EGL?

EGL is IBM newest business oriented language for IBM i

It shields developers from technical intricacies of Web development and SOA

EGL tooling is delivered in RDI SOA

EGL:
- Is extremely easy to learn
- Is flexible & productive
- Multi Platform
- Web UI: Provides seamless integration with JSF
- RBD delivers Eclipse based tooling
- It comes with a “service” language element for SOA solution
- Is a core component of IBM Enterprise Modernization strategy

The Syntax is Straightforward

```plaintext
stations TRNSTN[]; // Define an array of records
line String; // Define a string

// Function Declarations
Function onPreRender() // Gets called before display
  line = J2EELib.getParameter("id"); // Get value from URL
  loadPage(); // Call internal function
End

Function loadPage() // Define internal function
  TrainLib.loadStations(line, stations); // Call a library function
End
```
Plenty of Real World Data Types

- Primitive types built around business
  - Primitive types includes money and date
  - Character types for every need
  - Fixed decimal types (no business language can be without this)

- Structured types also powerful
  - Records make it easy to aggregate data
  - Can define overlapping fields as needed
  - Arrays auto-extend

- Biggest issue is metadata
  - Meta-data allows definition of labels and characteristics
  - Takes a lot of the grunt work out of basic application programming

Libraries Make it Easy to Reuse Code

- Easy to build packages of business logic
- Business logic can hide implementation details from user interface
- For example, a library function may use SQL or might call ILE to fill an array
  - The user interface calls the function with an array, and doesn't care about the mechanics
- Package names make it easy to group many functions
- The user interface code then becomes almost trivial, especially when using ILE as the primary business logic
WYSIWYG Web Page Design is as Easy as SDA and code designer

- Drag and drop arrays, using metadata for headings
- Drop fields from program onto page

Logic Parts: EGL vs ILE RPG

<table>
<thead>
<tr>
<th>EGL</th>
<th>ILE RPG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program</td>
<td>Program</td>
</tr>
<tr>
<td>Library</td>
<td>ServiceProgram</td>
</tr>
<tr>
<td>Function</td>
<td>Procedure</td>
</tr>
<tr>
<td>PageHandler</td>
<td>No equivalent</td>
</tr>
</tbody>
</table>

[Web user interface]
The CALL Opcode

```plaintext
call "STNSVRR" (opts, station);
while (opts.QRYRTC == "5")
    stations.addElement(station);
call "STNSVRR" (opts, station);
end
```

- The CALL opcode is simple and easy to use
- It supports multiple bi-directional parameters
- Converts records to data structures and vice versa
- Can call programs on other machines

Persistent Connections

- EGL supports stateful connections between web session and ILE programs
- ILE programs can retain state between calls and even between pages
- This allows program call performance measured in milliseconds
- You can call a program dozens of times to fill a page and still get subsecond response
- We'll see more about this later in the presentation
AJAX Makes it Even Better

- IBM extensions to the JSF syntax make it easy to implement AJAX
- No JavaScript coding required
- Just identify AJAX-enabled sections of code
- Then write the AJAX handler in EGL
- RBD generates both the client and server-side code as well as the connection between them

The Whole … is Greater than the Sum of the Parts

By itself, RDi makes it easy to build high-performance ILE programs and debug them
By itself, RBD is one of the most productive tools for generating web applications

Together
EGL and RDi bring a new level of business application production to the table
Agenda

- Packaging and rebranding.
- RDi and RDi SOA Value proposition
- RDi Functional capabilities
- RDi SOA Functional capabilities
- EGL For RPG Programmers
- Why is SOA Important to RPG Programmers?
  - A Quick Overview of EGL
    - Rich Data
    - EGL Syntax
  - Putting it All Together
  - Conclusion

The Islands of Application Design

- Traditional green screen development
- Browser-based web applications
- UI-centric thick client applications

The Problems

- Duplicated Data - Synchronization
- Redundant Logic - Maintenance
- User Interface Integration - Complexity
Islands of Application Design

IBM i
- DB2 UDB Data
- Green Screen Applications

Web Application Servers
- Network Databases
- Browser-Based Applications

User Workstation
- Local PC Data
- Thick Client Applications

Complex UI Integration Strategies

Duplicated Data
Redundant Logic

The Goal of SOA

- Integrate the Islands
- Loosely coupled components
- Common application elements
  - Shared database
  - Encapsulated business logic
- Pervasive deployment
  - New UI does not require new business logic!
One Strategy

But we can start moving in that direction

- Phase I: encapsulate logic
  - Extract business logic from legacy applications
    - Service programs
      - Or even just called server programs
  - Stored procedures

- Phase II: Build industry standard wrappers
  - Web services
  - Stored procedures

Won't Happen Overnight
The Business Issue

- Phase I is hard enough
  - Rewriting monolithic applications into server-based design is not easy
  - Typically it provides no obvious benefits to users

- It takes too long to reach Phase II
  - Users want something NOW
  - This is when the idea of throwing out all existing systems often surfaces

- Answer: an interim solution
  - EGL provides a way to immediately leverage Phase I
  - EGL is also a bridge to future development

Business Programming

- Business programming can be broken down into two basic kinds of application:
  - Simple CRUD and QUERY
    - Master file maintenance
    - Inquiries and reports
  - Sophisticated business logic
    - Order entry
    - MRP generation
    - Pricing, inventory allocation

- These are different from utility functions
  - Transferring data (email, FTP)
  - Generating specialized output (e.g., PDF, XML)
The Midrange Silversmith Life Cycle

- The IBM i is an incredible system for creating business logic
  - RPG is the pre-eminent language for defining data-driven business rules
    - The COBOL folks might reasonably argue this point
  - ISAM access (the ability to CHAIN to a single record) is one of the most important capabilities of data-driven business logic
- Use skeleton programs for basic CRUD and QUERY

- In the days of the green screen, that's all that was necessary
  - Programmers started out creating reports
  - Moved up to maintenance programs
  - Next, inquiries
  - Eventually graduated to business logic

This Has Changed

- Presentation is much more important
  - Quick access to ad hoc data
  - Graphics capabilities
    - Business graphics, dashboards
    - External formats such as PDF much more important
- Development tools can help a lot more
  - Perfect for SQL access
  - A lot of work already built in
    - Data access: selecting rows, joining files and so on
    - Data presentation: paging, sorting
    - Interaction: drill down, presentation (table ↔ graph)
Different Programming Models

- Green screen development is very much one-off
  - Think silversmith
    - Analyze the need
    - Design the program
      - You may have a previous product in mind, but this one is new
    - Craft it
      - Even if you use a skeleton, still mostly manual effort
- Visual programming tools
  - Think assembly line
  - Analyze the need
  - Point at pre-defined components
  - Plug in values and let code be generated

The Development Problem

- No way to make the two styles work together
- The sophisticated business logic is trapped behind:
  - Green screens
  - Screen scrapers
  - Clunky hand-crafted CGI screens
- Meanwhile, the queries look really GOOD
  - Latest web graphics
    - Up to date CSS features
    - AJAX interactivity
  - Or glamorous thick client glitz
Software Schizophrenia

- It’s difficult to create server applications for every possible user request
- Creating a server program and the associated plumbing for a simple query is often not cost effective
- So you end up with a dual development environment:
  - The "tool of the day" for user interface design
  - RPG-CGI, JSP, JSF, PHP, even Rails
  - RPG or COBOL on the IBM i for back end business processing
- What is needed is an environment where both quick SQL-based applications and sophisticated business logic can interact
- EGL provides this interaction

A Road to the Future

- The idea is to continue down this road
- No more silversmithing
- The future is packaged software with semi-custom modifications
  - Standard components for standard functions (e.g., accounting)
  - Standard models to develop custom business rules
  - Standard language to drive all tiers
  - Standard interfaces such as web services and other SOA interfaces
- All of this driven from a model, rather than from hand-crafted code
- EGL provides just such an model-driven environment
Model Driven EGL Development

User Interactions:
- Start with UML class diagram or DB Schema (these both are "models")
- Provide additional information to customize code gen (opt)
- Generate EGL code from models, using transform tooling
- Use Page Designer and EGL to edit web pages & associated EGL business logic

Agenda

- Packaging and rebranding.
- RDi and RDi SOA Value proposition
- RDi Functional capabilities
- RDi SOA Functional capabilities
- EGL For RPG Programmers
- Why is SOA Important to RPG Programmers?
- A Quick Overview of EGL
  - Rich Data
  - EGL Syntax
- Putting it All Together
- Conclusion
What is EGL

- Made Up of Parts
- Uses “Rich Data”
  - You define data items and their metadata
  - You then build records which have operational characteristics
- A Very High Level Language
  - Platform-Independent Model
  - Generates Platform-Specific Model
  - But uses the same data constructs
- Removes “plumbing”
  - SQL, MQSeries, JavaServer Faces, and more …

EGL Parts

- Logic Parts
  - Function
  - PageHandler
  - Program
  - Library

- Data Parts
  - Data Item
  - Record
  - Data Table

- UI Parts
  - Form
  - Form Group

- Build Parts
  - Build Descriptor
  - Linkage Options
  - Resource Associations
Packages and Libraries

- A package is a named entity, similar to a Java package
  - Packages are not parts; they are a group of parts
  - Packages are imported into other parts
  - Packages are unique, and have dot-names
    - com.pbd.i5.access
  - Some packages (e.g., pagehandlers) are standard
- A library is a group of related functions
  - Libraries are conveniences
  - Programmers can group related functions this way
  - All libraries (and their functions) in a package are available to someone who imports the package

Rich Data

- Define the characteristics of the field
  - Standard things like type and length
- Specify metadata such as prompt (for building UIs)
- Specify validation to auto-generate logic
- Specify attributes (also for UI generation)
- Even specify some processing-specific values
  - Null capable, or read only
  - Used when generating datastore access
Records

- Records are the primary way you access things outside of EGL
- Defined as groups of fields, with optional characteristics
  - SQLRecord provides database access via SQL
  - MQRecord provides access to MQ Series data queues
  - Other types exist
    - IndexedRecord, RelativeRecord, SequentialRecord
  - Also, the type BasicRecord acts as a data structure
    - Defines fields, but no data access

EGL is a "VHLL" (Very High Level Language)

- EGL has many 4GL Features
  - It's also missing some
    - Round trip modifications – any tweaks you make to the generated code get lost if you have to generate again.
    - Percolations of changes – if you change a field's attributes, there's no automated way to get those changes to show up on pages.
- EGL is more of a generator
  - You define your data specifications
  - You drag and drop things onto screens
  - You write some code in your page handlers
  - Hit "Generate" and away it goes!
- EGL's real productivity boost comes from its tight integration to various middleware
EGL is Tightly Integrated to Various Middleware

- To read a list of records from an SQL database into an array, you write one line of code:
  - `get customers;`
  - Well, technically two lines, because you have to define the array
  - And prior to that, you have to define the record
  - But the point is that once you've defined the record, access to the database is fast and easy
  - And you can drag-n-drop the array onto a JSF page to quickly create a professional-looking user interface

External Integration

- If EGL was just another completely self-contained language, I wouldn't be interested
- But it's NOT!
- Great support for Java
  - By using external types, you can import your own Java packages
  - Including third-party packages
  - This allows you to essentially extend the language
- Support for calls to other languages
  - Especially ILE Languages on i
  - We'll address this in more detail later
Agenda

- Packaging and rebranding.
- RDi and RDi SOA Value proposition
- RDi Functional capabilities
- RDi SOA Functional capabilities
- EGL For RPG Programmers
- Why is SOA Important to RPG Programmers?
- A Quick Overview of EGL

  - Rich Data
  - EGL Syntax

- Putting it All Together
- Conclusion

Data Definition

- EGL has a concept called "Rich Data"
- Data types can be defined along with their metadata
  - Metadata includes all those things we need as programmers, including UI capabilities
  - Things like column heading and prompt and alignment and color
  - Also includes validation

<table>
<thead>
<tr>
<th>Account</th>
<th>Name</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>33211</td>
<td>Joe's Pizza</td>
<td>1124.52</td>
</tr>
<tr>
<td>63326</td>
<td>Bernie's Bakery</td>
<td>223.21</td>
</tr>
</tbody>
</table>
Data Items are Your Basic Building Block

- Data Items allow you to define dozens of different attributes
  - Basic data characteristics
    - A much larger choice than most languages
      - Int, Bigint, float, decimal, packed, you name it
      - Char, blob, clob, dbChar, string, Unicode
      - Special types like money, timestamp, interval
  - Bridges the SQL/fixed-field gap pretty well
  - Validation
  - Display attributes (GUI and text-based)

Publication Type Data Item

```java
dataItem PUB_TYPE char(1) {
    displayName = "Type",
    highlight = defaultHighlight,
    selectType = value,
    upperCase = yes,
    validValues = ["N", "D", "B"]
}
```

The Field Reference Technique

- You build data items, which are then used in the rest of your development
  - Data items can be defined in a data dictionary
  - Those items can then be inserted into records
  - They can also be used in programs

- Very much like field reference files
  - You can override attributes when you add a data item to a record
  - I'd like the same capabilities both when creating other data items or when creating standalone field

A DataItems Part

```java
package data;

DataItem STATE string (displayName = "STATE") end
DataItem CUSTOMER_ID int (displayName = "CUSTOMER_ID") end
DataItem STREET string (displayName = "STREET") end
DataItem CITY string (displayName = "CITY") end
DataItem EMAIL_ADDRESS string (displayName = "EMAIL_ADDRESS") end
DataItem APARTMENT string (displayName = "APARTMENT") end
DataItem FIRST_NAME string (displayName = "First Name") end
DataItem PHONE string (displayName = "PHONE") end
DataItem DIRECTIONS string (displayName = "DIRECTIONS") end
DataItem POSTALCODE string (displayName = "POSTALCODE") end
DataItem PASSWORD string (displayName = "PASSWORD") end
DataItem LAST_NAME string (displayName = "Last Name") end
```
The Next Level is the Record

- Records are used for two purposes:
  - Group together fields into related structures
    - Can use data items or fields
    - One syntax variant makes it relatively easy to define fields that overlap other fields
  - Provide a fast, programmer-friendly interface to various middleware
    - An SQLRecord, for example, will automatically generate the appropriate SQL code when asked
    - Support currently exists for SQL, MQSeries, and indexed and sequential files
    - I've asked for a special "JavaRecord"; we'll see

Temporary Data

- "Records" and fields can be temporary
  - A record is really more like a data structure
    - Allows COBOL-like overlapping of fields
    - The BasicRecord type doesn't have any special plumbing characteristics at all
  - You can define records and fields that are local to a single EGL source member ("part")
  - Each record generates its own Java class
    - Java can directly access fields
    - In fact, Java can do some really powerful manipulation of the code, as we'll see a little later
Structures, overlays, and dot naming

10 workAddress;
20 streetAddress1 CHAR(20);
  30 Line1 CHAR(10);
  30 Line2 CHAR(10);
20 streetAddress2 CHAR(20);
  30 Line1 CHAR(10);
  30 Line2 CHAR(10);
20 city CHAR(20);

workAddress.streetAddress2 =
  workAddress.streetAddress2.Line1 +
  workAddress.streetAddress2.Line2

Agenda

- Packaging and rebranding.
- RDi and RDi SOA Value proposition
- RDi Functional capabilities
- RDi SOA Functional capabilities
- EGL For RPG Programmers
- Why is SOA Important to RPG Programmers?
- A Quick Overview of EGL
  - Rich Data
  - EGL Syntax
- Putting it All Together
- Conclusion
Functions

- There are two basic types of functions
  - Those inside a Page Handler
    - Designed to respond to UI events
      - onLoad is invoked when the page is first loaded
      - onPreRender is invoked when before each time the page is displayed
    - Other functions are called in response to UI events
    - Have access to the "Faces context"
      - Allows them to update the UI easily
  - Those in libraries
    - Not used to update UI
    - Instead, should be used for business logic and database access

A Sample EGL Page Handler (Using JSF)

```java
package pagehandlers;

import com.ibm.egl.jsf.*;

pageHandler myPageHandler
    onPageLoadFunction = onPageLoad,
    view = "myPage.jsp",
    viewRootVar = "myViewRoot"

myViewRoot UIViewRoot;

function onPageLoad()
    myInputField HtmlInputText;
    myInputField = myViewRoot.findComponent("form1:text1");
    myInputField.setStyle(\"color : red\");
end
end
```
Another EGL Sample (a Simple Library Function)

```java
package com.mycomp.db;
Library CustomerLib

Function updateCustomer(newCustomer Customer)
    oldCustomer Customer;
    oldCustomer.ID = newCustomer.ID;
    get oldCustomer for update;
    move newCustomer to oldCustomer byname;
    update oldCustomer;
End

End
```

System Libraries

- **System Libraries** – The EGL System Libraries are a collection of shared functions and variables that are available to all EGL programs, pageHandlers, services, and custom libraries
  - **DateTimeLib** – The date and time system variables allow you to retrieve the system date and time in a variety of formats
  - **J2EELib** – The J2EE system words perform operations on Request and Session attributes
  - **StrLib** – The string handling system functions provide operations such as comparing or concatenating strings
  - **Syslib** – The system library functions provide support for such items as exception handling and status, native Java program access, options for file and database access, and data conversion
Integrating External Types

```java
package com.pbd.egl;
public class Util {
    public static boolean localData = false;
    public static boolean debug = false;
}

package externals;
externalType Util type JavaObject {
    javaName = "Util", packageName = "com.pbd.egl"
    static localData boolean;
    static debug boolean;
}
```

Agenda

- Packaging and rebranding.
- RDi and RDi SOA Value proposition
- RDi Functional capabilities
- RDi SOA Functional capabilities
- EGL For RPG Programmers
- Why is SOA Important to RPG Programmers?
- A Quick Overview of EGL
  - Rich Data
  - EGL Syntax
- Putting it All Together
- Conclusion
A Multi-Tiered Library Function – EGL Side

Function loadLines(lines TRNLIN[] out)

    lines = new TRNLIN[];
    opts.QRYOPC = "IN";
    call "LINSVRR" (opts, line);
    While (opts.QRYRTC == "0")
        lines.appendElement(line);
        opts.QRYOPC = "NX";
        call "LINSVRR" (opts, line);
    End

End

A Multi-Tiered Library Function – RPG ILE

PTNLIN IF K K DISK
D dsQRYOPT K DS EXTNAM(QRYOPT)'
D dsTRNLIN K DS EXTNAM(TRNLIN)'
C *ENTRY PLIST
C PARM dsQRYOPT
C PARM dsTRNLIN

/IF QRYOPC = 'IN';
    SETLL *LOCAL TRNLIN;
    ELSEIF QRYOPC = 'GT';
    SETLT LINID TRNLIN;
    ENDIF;
    READ TRNLIN;
    QRYRTC = %eof(TRNLIN);
    RETURN;
/END-FREE
Adding Debug Capabilities

Function loadLines(lines TRNLIN[] out)

    if (Util.debug)
        SysLib.writeStdout("TrainLib.loadLines");
    end

    if (Util.localData)
        lines = localLines();
        if (Util.debug)
            SysLib.writeStdout("TrainLib.loadLines: local lines: " + lines.getSize());
        end
        return;
    end

    lines = new TRNLIN[];
    opts.QRYOPC = "IN";
    call "LINSVRR" (opts, line);
    While (opts.QRYRTC == "0")
        lines.appendElement(line);
        opts.QRYOPC = "NX";
        call "LINSVRR" (opts, line);
    end

    if (Util.debug)
        SysLib.writeStdout("TrainLib.loadLines: external lines: " + lines.getSize());
    end

End

Setting up Test Data

localTrains TRNLIN[4] {
    this[1].LINID = "BNSF", this[1].LINDSC = "Burlington Northern Santa Fe",
    this[2].LINID = "UPN", this[2].LINDSC = "Union Pacific North",
    this[3].LINID = "UPNW", this[3].LINDSC = "Union Pacific Northwest",
    this[4].LINID = "UPW", this[4].LINDSC = "Union Pacific West"
};

Function localLines() returns (TRNLIN[])

    return (localTrains);
End

Other options include reading a stream file or accessing a local database
**Testing the Business Logic**

```java
Program TestTrain
function main()
    TrainLib.setUser();
    lines TRNLIN[0];
    Util.localData = true;
    TrainLib.loadLines(lines);
    temp String;
    syslib.convertToJSON(lines, temp);
    writeStdout(temp);
end
end
```

**CONSOLE OUTPUT**

```
[{"LINID" : "BNSF", "LINDESC" : "Burlington Northern Santa Fe"},
 {"LINID" : "UPN", "LINDESC" : "Union Pacific North Line"},
 {"LINID" : "UPNW", "LINDESC" : "Union Pacific Northwest Line"},
 {"LINID" : "UPW", "LINDESC" : "Union Pacific West Line"}]
```
Linking to Another Page

A Complete Page Handler

```java
package jsfhandlers;
import data.*;
import externals.*;

handler Trains type JSFHandler
    (onPreRenderFunction = onPreRender,
     view = "Trains.jsp")

// Function Declarations
function onPreRender()
    sysLib.writeStdout("DBG: Lines!";
    TrainLib.setUser();
    loadPage();
end

lines TNMLIN[0];
Function loadPage()
    //Util.localData = true;
    TrainLib.loadLines(lines);
End
end
```
The Finished Product

Trains Demo - Line List

<table>
<thead>
<tr>
<th>Line</th>
<th>Address</th>
<th>Phone Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chicago</td>
<td>Canal St between Adams &amp; Jackson (312) 322-4269</td>
<td></td>
</tr>
<tr>
<td>Beulah</td>
<td>14th &amp; Halsted St.</td>
<td></td>
</tr>
<tr>
<td>Western Ave</td>
<td>10th &amp; Western Ave.</td>
<td></td>
</tr>
<tr>
<td>Cicero</td>
<td>28th &amp; Cicero</td>
<td></td>
</tr>
<tr>
<td>Clyde</td>
<td>24th &amp; Austin Blvd</td>
<td></td>
</tr>
<tr>
<td>La Grange</td>
<td>Wabash &amp; Halsted Ave.</td>
<td></td>
</tr>
<tr>
<td>Beverly</td>
<td>6911 W. Wabash Ave.     (708) 475-6526</td>
<td></td>
</tr>
<tr>
<td>Elmhurst</td>
<td>7115 Winder Ave.        (708) 465-0214</td>
<td></td>
</tr>
<tr>
<td>Brothers</td>
<td>38 Bloomingdale Rd.    (708) 475-5757</td>
<td></td>
</tr>
<tr>
<td>Hollywood</td>
<td>Golf Road &amp; Hollywood Ave.</td>
<td></td>
</tr>
<tr>
<td>Brookfield</td>
<td>18155 Burbank Ave.    (708) 455-2207</td>
<td></td>
</tr>
<tr>
<td>Cook County</td>
<td>1000 Taylor St.</td>
<td></td>
</tr>
<tr>
<td>La Grange Rd</td>
<td>27 W. Burbank Rd.    (708) 475-0117</td>
<td></td>
</tr>
<tr>
<td>La Grange Stone Ave</td>
<td>751 East Ave.        (708) 475-0193</td>
<td></td>
</tr>
<tr>
<td>Western Springs</td>
<td>900 Burbank Ave.</td>
<td></td>
</tr>
<tr>
<td>Highlands</td>
<td>Cottage Lane Rd. &amp; 78th St.</td>
<td></td>
</tr>
<tr>
<td>Harlem</td>
<td>1 East Harlem Ave.       (312) 322-6292</td>
<td></td>
</tr>
<tr>
<td>West Side</td>
<td>1000 W. Chicago Ave.</td>
<td></td>
</tr>
<tr>
<td>Clarendon Hills</td>
<td>1 South Prospect Ave. (312) 322-1077</td>
<td></td>
</tr>
<tr>
<td>Westmore</td>
<td>1 West Quincy</td>
<td></td>
</tr>
<tr>
<td>Fargo Ave.</td>
<td>Fargo Ave. &amp; Burlington Ave.</td>
<td></td>
</tr>
<tr>
<td>Des Plaines Grove</td>
<td>5911 Main Street</td>
<td>(312) 322-0315</td>
</tr>
<tr>
<td>Enclave</td>
<td>9000 Belvedere Ave.     (312) 322-0012</td>
<td></td>
</tr>
<tr>
<td>South Side</td>
<td>1000 S. Dearborn St.</td>
<td></td>
</tr>
<tr>
<td>Sylmar</td>
<td>1001 S. Dearborn St.    (312) 322-3166</td>
<td></td>
</tr>
<tr>
<td>Norridge</td>
<td>1000 E. 46th Ave.       (312) 322-4209</td>
<td></td>
</tr>
<tr>
<td>North Side</td>
<td>700 N. Dearborn St.</td>
<td></td>
</tr>
<tr>
<td>Armour</td>
<td>2301 N. Dearborn         (312) 322-0473</td>
<td></td>
</tr>
</tbody>
</table>

The Standard Browser Interface

- JSF Page
- EGL Page Handler
- EGL Library Function
- ILE Business Logic
The Persistent Connection

By keeping the same QZRCSRVS job on the host, the ILE program on the host can stay resident. The program can thus maintain a persistent state, keeping cursors open and so on, making the calls incredibly fast. With this design, it's easy to create web pages that refresh a dozen times a second.

Throw in a Little AJAX

Trains Ajax Demo - Line List - Line UPW

Chicago:
- Sedgwick St. and Canal St. (312) 427-6977
- Oak Park
- 102 N. Market St. (312) 750-0124
- River Forest
- 1801 S. Oak Park Ave. (312) 750-0124
- Northbrook
- 101 N. Oak Park Ave. (312) 750-0124
- Mt. Prospect
- 101 N. Main St. (312) 750-0124
- Bartlett
- 101 S. Oak Park Ave. (312) 750-0124
- Elmhurst
- 100 E. Oakland Ave. (312) 750-0124
- Oak Park
- 101 N. Oak Park Ave. (312) 750-0124
- Cicero
- 101 S. Oak Park Ave. (312) 750-0124
- Whiteman
- 101 S. Oak Park Ave. (312) 750-0124
- Mundelein
- 101 S. Oak Park Ave. (312) 750-0124
- Ingleside
- 101 S. Oak Park Ave. (312) 750-0124
- Waukegan
- 101 S. Oak Park Ave. (312) 750-0124
curLine string;
rowClasses string;
Function loadCurrentLine() {
    command string = J2EELib.getParameter("command");
    if (command == "Prev") prevLine(); end
    if (command == "Next") nextLine(); end
    curLine = lines[lineIndex].LINID;
    rowClasses = UtilLib.formatRowClasses(lines.getSize(), lineIndex, "normal", "selected");
End

stations TRNSTN[0];
stationIndex int;
Function loadStations() {
    Trainlib.loadStations(curLine, stations);
    stationIndex = 1;
End

Function prevLine() {
    if (lineIndex <= 1) lineIndex = lines.getSize() + 1;
    end
    lineIndex -= 1;
End

Function nextLine() {
    if (lineIndex >= lines.getSize()) lineIndex = 0;
    end
    lineIndex += 1;
End
The Service Interface

The point here is that regardless of the interface, the business logic stays the same. And with the concept of a dynamic array of records, even the EGL library function can remain the same, leaving only the interface to change, which is the ultimate goal of a UI independent application development environment.

The Next Generation of Apps: Rich Web Services

EGL allows you to create user friendly mashups by combining ILE business logic, SQL queries, RESTful services, EGL widgets, third-party utilities and custom libraries.

The plumbing work is removed, letting you concentrate on the creative aspect.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BNSF</td>
<td>Burlington Northern Santa Fe</td>
</tr>
<tr>
<td>UPN</td>
<td>Union Pacific North</td>
</tr>
<tr>
<td>UPMX</td>
<td>Union Pacific Northwest</td>
</tr>
<tr>
<td>UPW</td>
<td>Union Pacific West</td>
</tr>
</tbody>
</table>
The Next Generation is Here Today

Bottom Line:

With the RDi-SOA toolset (RDi and RBD), EGL and IBM i Provide Web Elegance with Green Screen Performance at the Speed of ILE Development
Everything Because of This

Function loadLines(lines TRNLIN[] out)

lines = new TRNLIN[];
opts.QRYOPC = "FN";
call "LINSVRR" (opts, line);
While (opts.QRYRTC == "0")
    lines.appendElement(line);
    opts.QRYOPC = "NX";
    call "LINSVRR" (opts, line);
End

Agenda

- Packaging and rebranding.
- RDi and RDi SOA Value proposition
- RDi Functional capabilities
- RDi SOA Functional capabilities
- EGL For RPG Programmers
- Why is SOA Important to RPG Programmers?
- A Quick Overview of EGL
  - Rich Data
  - EGL Syntax
- Putting it All Together
- Summary
Bottom Line:

With the RDi-SOA toolset (RDi and RBD), EGL and IBM i Provide Web Elegance with Green Screen Performance at the Speed of ILE Development

QUESTIONS
THANK YOU

Learn more at:

- IBM Rational software
- IBM Rational Software Delivery Platform
- Process and portfolio management
- Change and release management
- Quality management
- Architecture management
- Rational trial downloads
- Leading Innovation Web site
- developerWorks Rational
- IBM Rational TV
- IBM Rational Business Partners