Modernization

Getting Started

Brian May
IBM i Modernization Specialist
Profound Logic Software

Live Tweet

- My Twitter
  - @BMayTheYiP
- Company Twitter
  - @ProfoundLogic
- Hashtag for Event
  - #AppMod4i

Overview

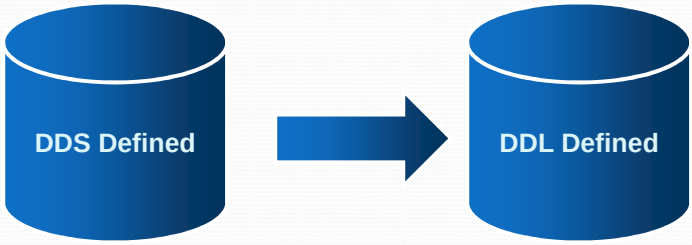
- Three critical areas of modernization
- The future of RPG and Rational Open Access, RPG Edition
- MVC
- Q & A

Three Critical areas

Application Modernization

Three Critical Areas

- Database



Three Critical Areas

- Database
  - Use DDL for new tables
  - No need to redefine existing tables just to be DDL defined
  - Make use of newer data types
    - Date, Time, Timestamp
    - CLOB, BLOB
    - Datalink
    - XML

Three Critical Areas

- Database

Record Level Access

```
Setll Key MyFile ;
Reade Key MyFile;
DoW Not %EOF;
  Update MyFile;
  Reade Key MyFile;
EndDo;
```

SQL Access

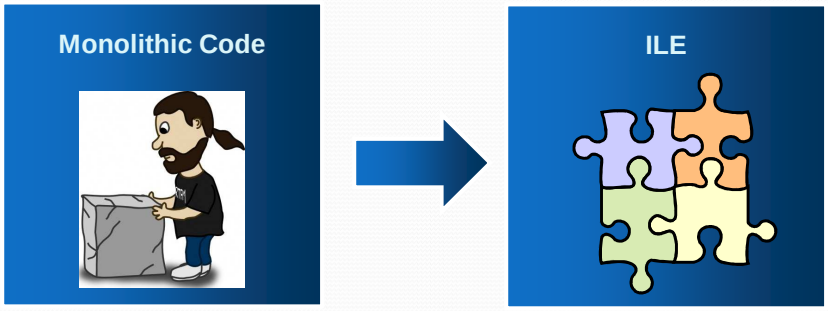
```
Update MyFile Set x= y
  Where KeyField = Key ;
```

Three Critical areas

- Database
  - RLA vs. SQL
    - Does not have to be exclusive
  - Triggers
    - Enforce business rules no matter where updates originate
  - Constraints
    - Ensure data integrity
    - Reduce code required in programs
  - Identity Columns
    - Every row in every table needs a unique key
    - Easily link tables with a single field

Three critical areas

- Code

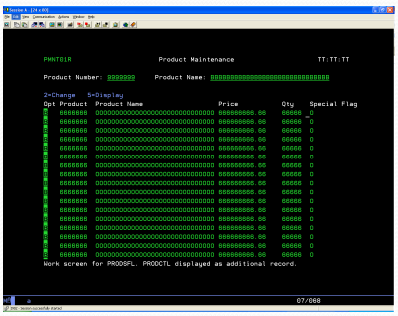


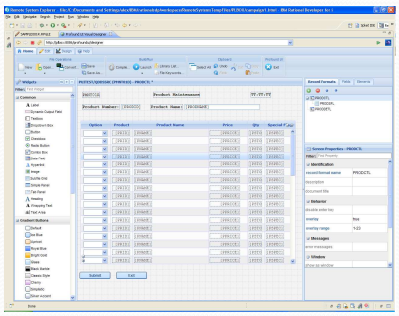
Three critical areas

- Code
  - Free Form vs. Fixed Form
    - Personal Choice
    - Free Form is more intuitive to new RPG developers
  - Named Indicators vs. Numeric Indicators
    - Numeric Indicators force maintenance developers to spend time determining meaning of indicator
    - Named Indicators can have descriptive names to save time and add clarity
  - Descriptive Field Names
    - Longer names allow variables to have meaningful names
    - Adds clarity for maintenance

Three critical areas

- User Interface





Three critical areas

- User Interface
  - Green Screens are immediately give an outdated impression
  - Users expect features that are not available on 5250 applications
  - Many different options available and none of them are necessarily wrong

Future of RPG

Application Modernization

FUTURE of rpg

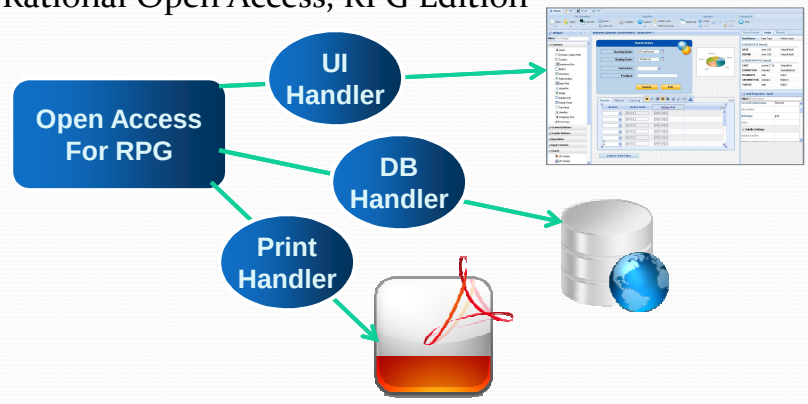
- IBM's commitment to IBM i (and RPG)
  - IBM recently released charts showing plans for support beyond 2025
    - http://ibmsystemsmag.blogs.com/you_and_i/2012/12/future-of-ibm-i-caac-buzz.html
  - IBM has consistently added significant functionality to RPG in each release of IBM i
    - Built-in XML support
    - Data Structure and Array enhancements
    - Local File support for subprocedures
    - Larger storage limits
    - Full Free Format

Future of RPG

- RPG is still the ideal language for business logic
  - Numbers, Numbers, Numbers
    - Business logic is all about numbers
      - Prices
      - Quantities
    - RPG handles decimal precision simply and elegantly
  - Database Integration
    - Both RLA and SQL access to data is simple in RPG
  - Date Data Type
    - Some languages require use of complex classes to use dates properly
    - In RPG, Date is a native data type

Future of RPG

- Rational Open Access, RPG Edition



Future of RPG

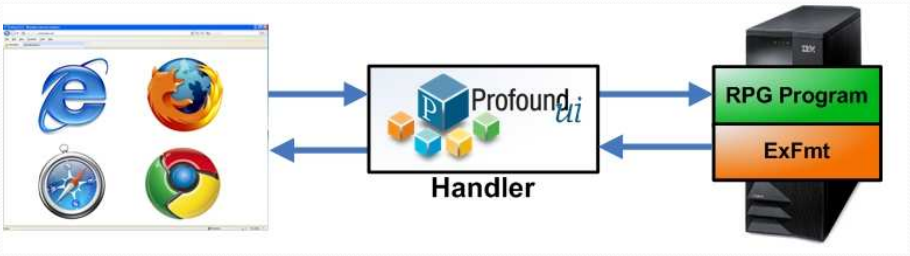
- Rational Open Access, RPG Edition
  - "Opens" the interface for file IO in RPG to allow 3rd parties to enhance the RPG language
  - Gives RPG the ability to interact with any type of input or output technology available now or in the future
  - "Handlers" are programs (or subprocedures) that replace the functionality of the RPG runtime for the file where the HANDLER keyword is specified

Future of RPG

- Rational Open Access, RPG Edition
  - It is the handler's responsibility to provide all functionality that the RPG runtime normally has
    - %FOUND, %EQUAL
    - INFDS
    - Error handling/reporting
    - Any opcode that can be used on file type

Future of RPG

- Rational Open Access, RPG Edition



MVC

Application Modernization

MVC

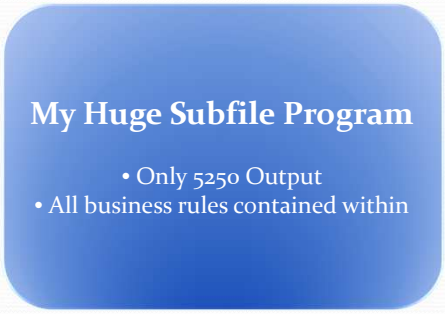
- Open Access can be your first step to MVC
  - What is MVC?
    - Model – Your data and business rules
    - View – Your UI
    - Controller – Your program to control your application
  - MVC is a mainstream approach to application design.
    - Allows for any of the 3 pieces to be replaced without major changes to the others.

MVC

- This is a very simplified definition of MVC
  - In fact, there are differing opinions on its meaning and implementation
- MVC architecture can be implemented in any programming language (even RPG)
  - All it takes is a change in mindset
  - RPG has all the tools needed via ILE to implement MVC effectively

MVC

- Typically, RPG applications are large monolithic programs

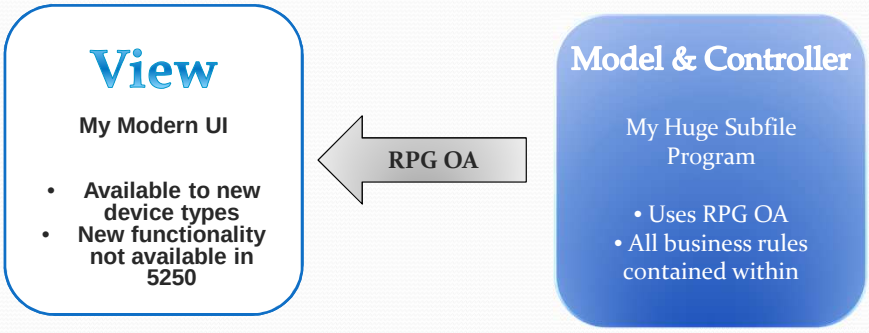


MVC

- Monolithic programs seem simple in theory
  - Everything used by the program is in one place
  - No need to worry about outside changes (other than data) impacting processing
- As you move to larger, more complex applications, monolithic programs become very complicated
  - When all logic is one entity, even the slightest change runs the risk of introducing bugs in any part of the program
  - As the source becomes larger, it becomes harder to maintain

MVC

- RPG Open Access allows us to separate the view from the RPG program using an open interface.

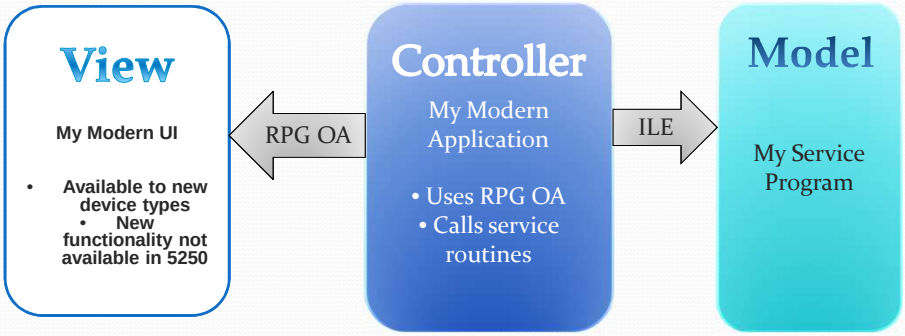


MVC

- Separate View Layer
  - RPG OA allows RPG to have presentation layers beyond the green screen
  - Presentation elements can populate themselves by querying database or calling procedures
  - Many validations can be made in the View layer instead of in the main program
  - Inclusion of remote content

MVC

- After we modernize our UI, all that's left is to move our business rules into ILE service programs to achieve an MVC design.



MVC

- Model Layer
  - Should include business rules and data access methods
  - Removal of business rules and data access simplify the code left in the controller
  - Individual business rules can be modified and tested in isolation without requiring retesting of all programs
  - Changes can be made to database without the need to recompile or modify all programs using the data

Q & A

Application Modernization

Thank You!

About the Presenter

Brian May is an IBM i Modernization Specialist for Profound Logic Software. He has also served as webmaster and coordinator for the Young i Professionals (http://www.youngiprofessionals.com). He is a husband and father of two beautiful girls. Brian can be reached at bmay@profoundlogic.com