Three Approaches to Web with RPG

Presented by

Scott Klement

http://www.scottklement.com

"A computer once beat me at chess, but it was no match for me at kick boxing." — Emo Philips

The Agenda

Agenda for this session:

1. Review of Web Technology
   - HTML, CSS, JavaScript

2. The Original Philosophy & History of web applications.
   - Original programs that output HTML
   - Smart page tools, like PHP
   - Why use RPG?
   - CGIDEV2 with example.

3. The New Philosophy
   - Frameworks
   - Example

4. The Open Access Approach

Review of HTML

HTML is the primary language of web browsers. It is a tag language (similar to XML.)

A tag consists of `<the-tag-name>` and `</the-tag-name>`

- without the slash, it's a "start tag," indicating the start of something.
- with the slash, it's an "end tag," indicating the end of something.
- used properly, HTML indicates what something is, not explicitly how to display it.

```
<html>
  <head>
    <link href="style.css" rel="stylesheet">
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is text in a paragraph, and will
      be <strong>formatted differently</strong>
      than the heading is.</p>
  </body>
</html>
```

Review of HTML, continued.

On the previous slide, the HTML tags indicated:

- HTML = the part of the document that is HTML (all of the document)
- HEAD = the part of the document that indicates header information.
- LINK = a link to (in this case) a CSS style sheet (more info in a moment)
- BODY = the body, or content, of the document.
- H1 = a level-1 heading (or title) of a section of text. There are also H2-H6.
- P = a paragraph
- STRONG = text that should be given strong emphasis.

At one time (before CSS was invented) there were HTML tags that gave explicit information about how to format data as well. *These are now deprecated.*

There are hundreds of HTML tags, far too much to cover in this presentation!

- There is an HTML tutorial, here:
  http://w3schools.com/html/default.asp
- There is a complete reference of the HTML tags, here:
  http://w3schools.com/tags/default.asp

Review of CSS

CSS (Cascading Style Sheets) are used to provide explicit formatting for HTML data.
- Works together with HTML
- Is the proper way to provide formatting

```
p {
  font-family: Arial;
  color: black;
  background-color: white;
}
strong {
  font-weight: bold;
  font-size: 150%;
}
```

Like HTML, there's far too many options in CSS to cover them all here. Instead, see:
http://www.w3schools.com/css/default.asp

JavaScript Review

JavaScript is an interpreted language that (usually) runs in a web browser.
- lets you do program logic inside a web page
- can add/remove/change the contents of HTML tags
- can do calculations as the user types
- can create HTTP requests to communicate back to the server (AJAX)

Please do not confuse JavaScript with Java!
- JavaScript = interpreted, weakly typed language, usually run in a browser, created by Netscape
- Java = compiled, strongly typed language, usually on the server created by Sun
- Syntax is quite different (not similar)

Confusing these is a big pet-peeve of mine, because I love working in JavaScript and try to avoid Java.

JavaScript Review, continued.

```
<script type="text/javascript">
  function addFields() {
     var field1 = document.getElementById("field1");
     var field2 = document.getElementById("field2");
     var result = document.getElementById("result");

     result.innerHTML = Number(field1.value)
                      + Number(field2.value);
  }
</script>

<form>
  <input type="text" id="field1" onblur="addFields()"><br />
  <input type="text" id="field2" onblur="addFields()"><br />
  <div id="result"></div>
</form>
```

A beginner's introduction to JavaScript is found here:
http://w3schools.com/js/default.asp

History: Original CGI (Early 1990's)

In the very early days of the web, when you wrote applications that interacted with a web browser, you would typically write a program that outputs HTML data. This way, it could output different data depending on the outcome of it's program logic, database values, etc.

The earliest examples were written in C, and were a pain to maintain. It is cumbersome coding HTML inside a HLL. Soon other languages like perl were used because they were a little easier, but still tricky.

```
#include <stdio.h>

int main(int argc, char **argv) {
     char var[6] = "World";

     printf("<html>\n");
     printf("<body>\n");
     printf("<p>Hello %s!</p>\n", var);
     printf("</body>\n");
     printf("</html>\n");

     return 0;
}
```

History: 'Smart Pages' (mid-late 1990s)

The PHP language was created in the mid-1990s to solve the awkwardness of coding HTML tags in your program code. With PHP whatever you type is HTML by default, so no need to escape tags properly to fit into HLL strings.

Soon other languages such as Java (with Java Server Pages), Net.Data, and Microsoft's ASP did the same thing. This became the new norm for web programming – and is a major reason why people think that these languages are better for web programming.

```
<html>
  <head>
    <title>My Web Page</title>
  </head>
  <body>

<?php
   $var = "World";
   echo "<p>Hello $var</p>";
?>

  </body>
</html>
```

Why Use RPG?

- Originally, it was difficult to create web applications in RPG
  - Concatenating HTML into a web page was difficult, awkward, much like early CGI programs in C or perl.
  - No longer true!
- RPG developers not proficient in Web skills
  - Still a problem, but less relevant today.
  - With frameworks, less knowledge is needed.
  - Many more Java or PHP programmers available on the market than RPGers, but these Java or PHP developers are easily taught RPG.
- Your business logic is in RPG. Existing talent is RPG.
  - Best database handling of any language
  - Best number handling of any language (try a 17-digit number in other languages)
  - RPG is simply a better language for writing business rules/logic!
  - Has modern features, procedures, service programs, free format, SQL, etc.
- So, now there are pros and cons
  - Are developers available?
  - Best tool? Or most mainstream tool?

Smart Pages in RPG?

There are commercial tools, such as RPG Smart Pages (RPGsp) from Profound Logic or RPG Server Pages (RSP) from ProData that use a precompiler to give you the same environment in RPG that you had in languages like PHP or Java's JSP.

There is also the absolutely free, open source, CGIDEV2 toolkit that also works well. You cannot code your HTML and RPG logic in the same source, but you can get very similar results by putting the HTML into an IFS file, and filling in variables from your RPG code.

CGIDEV2

You can break your document into sections (that can be output before/after RPG code) by using `/$section-name` in your IFS file.

You can insert variable data into one of your sections by inserting `/%variable name%/` into your HTML data.

```
/$Top
Content-type: text/html

<html>
  <head>
    <title>My Web Page</title>
  </head>
  <body>
/$Detail
  <p>Hello /%var%/!</p>
/$Bottom
  </body>
</html>
```

CGIDEV2, continued…

CGIDEV2 provides ILE subprocedures you can call to interact with the web page from your RPG code.

- `getHtmlIfs` tells CGIDEV2 which IFS HTML file to use.
- `zhbGetInput` tells CGIDEV2 to interpret input data from the browser.
- `zhbGetVar` gets the contents of one variable sent from the browser.
- `updHtmlVar` fills in the contents of one variable in the HTML file.
- `wrtsection` writes a section of data from your HTML file

Since /$ and /% are not always the best characters to use (especially in international applications), you can also configure CGIDEV2 to use different characters by adding parameters to the `getHtmlIfs` procedure.

As a simple example, let's take a look at a program that outputs a product listing.

CGIDEV2 Web 1.0 Example (1 of 4)

```
/$Keywords
Content-type: text/html

/$Top
<html>
<head>
  <title>Original Philosophy</title>
  <link href="/original/prodlist.css" rel="stylesheet" type="text/css">
</head>
<body>
  <table>
    <tr>
      <th>Product Id</th>
      <th>Product Name</th>
      <th>Product Price</th>
      <th>Stock Qty</th>
    </tr>
/$Detail
    <tr>
      <td>/%prid%/</td>
      <td>/%prname%/</td>
      <td class="number">/%pprice%/</td>
      <td class="number">/%pstock%/</td>
    </tr>
```

This is an excerpt. (Some sections omitted for brevity.)

CGIDEV2 Web 1.0 Example (2 of 4)

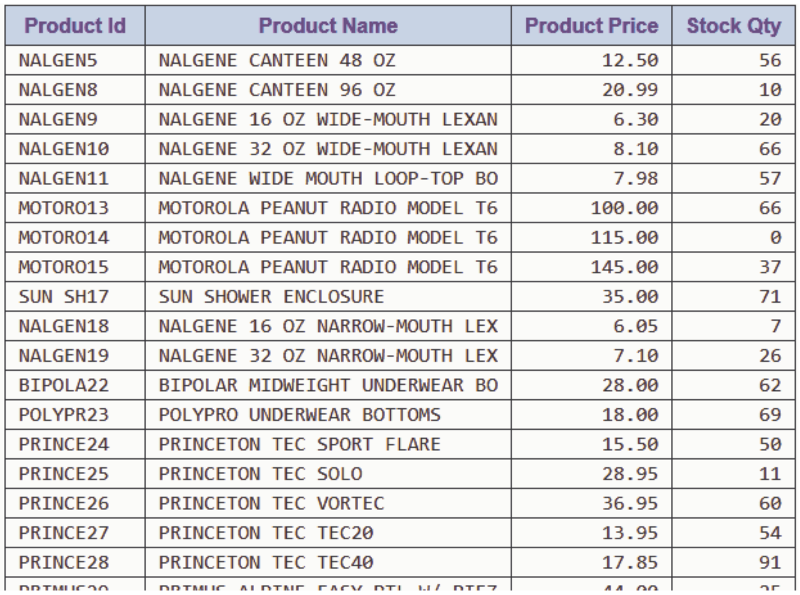
```
H DFTACTGRP(*NO) ACTGRP('KLEMENT')
H BNDDIR('TEMPLATE2')

 * CGIDEV2
 /COPY QRPGLESRC,PROTOTYPEB
 /COPY QRPGLESRC,USEC

D QueryString     S          32767A   Varying
D First           s              1n   inz(*on)
D PRID            S             10A
D PRNAME          S             30A
D PPRICE          S             11P 2
D PSTOCK          S              7P 0

 /Free

   getHtmlIfs('/www/skwebsrv/templates/prodlist.html');
   ZhbGetInput(QueryString : qusec);
   wrtsection('keywords');

   Exec SQL DECLARE cursor1 CURSOR FOR
        SELECT PRID, PRNAME, PPRICE, PSTOCK
          FROM PRODUCTSP;
   Exec SQL Open cursor1;
   Exec SQL Fetch cursor1 INTO :PRID, :PRNAME,
          :PPRICE, :PSTOCK;
```

CGIDEV2 Web 1.0 Example (3 of 4)

```
wrtsection('Top');

Dow %subst(sqlstt:1:2)='00'
   or %subst(sqlstt:1:2)='01';

  updHtmlVar('PRID': PRID);
  updHtmlVar('PRNAME': PRNAME);
  updHtmlVar('PPRICE': %char(PPRICE));
  updHtmlVar('PSTOCK': %char(PSTOCK));

  wrtsection('Detail');

  Exec SQL Fetch cursor1 INTO
       :PRID, :PRNAME, :PPRICE, :PSTOCK;
EndDo;
Exec SQL Close cursor1;

wrtsection('Bottom *fini');
*inlr = *on
```

CGIDEV2 Web 1.0 Example (4 of 4)

- When run from a browser, the output looks something like this
- (part of formatting is done in the stylesheet (not shown)



Product Id	Product Name	Product Price	Stock Qty
NALGEN5	NALGENE CANTEEN 48 OZ	12.50	56
NALGEN8	NALGENE CANTEEN 96 OZ	20.99	10
NALGEN9	NALGENE 16 OZ WIDE-MOUTH LEXAN	6.30	20
NALGEN10	NALGENE 32 OZ WIDE-MOUTH LEXAN	8.10	66
NALGEN11	NALGENE WIDE MOUTH LOOP-TOP BO	7.98	57
MOTORO13	MOTOROLA PEANUT RADIO MODEL T6	100.00	66
MOTORO14	MOTOROLA PEANUT RADIO MODEL T6	115.00	0
MOTORO15	MOTOROLA PEANUT RADIO MODEL T6	145.00	37
SUN SH17	SUN SHOWER ENCLOSURE	35.00	71
NALGEN18	NALGENE 16 OZ NARROW-MOUTH LEX	6.05	7
NALGEN19	NALGENE 32 OZ NARROW-MOUTH LEX	7.10	26
BIPOLA22	BIPOLAR MIDWEIGHT UNDERWEAR BO	28.00	62
POLYPR23	POLYPRO UNDERWEAR BOTTOMS	18.00	69
PRINCE24	PRINCETON TEC SPORT FLARE	15.50	50
PRINCE25	PRINCETON TEC SOLO	28.95	11
PRINCE26	PRINCETON TEC VORTEC	36.95	60
PRINCE27	PRINCETON TEC TEC20	13.95	54
PRINCE28	PRINCETON TEC TEC40	17.85	91

Problems with Web 1.0 Programs

This page is obviously a very simple example. In a real-world example:

- The HTML can get complex and unwieldy (and no less so in languages like PHP/Java!)

- Coding everything to work the way you want requires a lot of time spent writing the 'plumbing'
  - What if you want a numeric-only field? Write JavaScript.
  - What if you want only capital letters (uppercase) field? Write JavaScript.
  - What if you want data to be formatted in a particular way (such as dashes in a phone number) Write JavaScript.

- How do you allow sorting the list by column? Write JavaScript. Or sort on the server side and re-load the whole page.

- Generally, Web 1.0 applications are not very interactive.
  - Page cannot change without re-loading everything.
  - Server-side stuff cannot be done without the user clicking a button, and waiting for a new page.

AJAX Technology Spawns Web 2.0

AJAX stands for Asynchronous JavaScript And XML.

- You write a base HTML page.
- JavaScript runs when user interacts with page.
- JavaScript can make HTTP requests to server for more information, as needed.
- JavaScript simply updates parts of the page on-the-fly. No need to re-load.

Today, the term AJAX is not only used with XML, but also with other data formats, notably JSON, which serves the same purpose, but can be interpreted much faster in JavaScript than XML can.

AJAX was the first step in Web 2.0 technology. The next step was modifying your page on-the-fly, aka Dynamic HTML (DHTML).

Together, they revolutionized the web:
- Made social web sites, Facebook, LinkedIn, Twitter, YouTube, etc possible!
- business applications using a web interface became much more practical, due to the richer, more interactive UI

Frameworks

Now we have a lot of JavaScript being used:
- Validating/formatting data as you type.
- Interacting on-the-fly with the server (AJAX)
- Dynamically building the web page (DHTML)
- Providing the user with real-time changes (such as Facebook notifying you when someone posts on your wall)

Does it make sense to write all of that JavaScript yourself, individually for every application? No! Create a library of reusable JavaScript routines!

A library with JavaScript that you can use in your applications is called a "Framework" and there are many frameworks (often available for free) that you can download and use in your applications to:

- Build pages dynamically, with much more functionality than straight HTML
- Make AJAX requests to interact with server
- And much more!

The Profound UI Framework

Let's re-do our Product Listing example using a framework.

For my example, I will use the Profound UI Framework, because it's the one that I know the best.

- Completely free, open-source JavaScript framework.
- Released under the LGPL license.
- Available from github: https://github.com/ProfoundLogic/profoundui-framework
- Used as a component in Profound Logic's commercial products.
- Uses JSON format for communication
- Designed by RPG programmers to be natural for business applications.
- Can be used to create mobile applications as well.

Of course, you don't have to use this framework. Popular frameworks that I recommend:

- ExtJS and Sencha Touch
- jQuery and jQuery Mobile

Introducing JSON

JSON = JavaScript Object Notation

JSON is used for the types of applications as XML would be. However, JSON has one big advantage over XML: It's natively understood by Javascript. This means that web applications using JSON are easier to write, and run faster than those using XML.

JSON is used to describe data structures in Javascript. Including complex situations with data structures and arrays nested inside each other.

- An array in JSON begins/ends with [and] characters.
- An object (basically, a data structure) begins/ends with { and } characters.
- The : (colon) character separates a field name from it's value.
- The , (comma) character separates one element of an array from the next, as well as one field of an object from the next.
- Character data in JSON is enclosed in double quotes.

Framework Overview

We will use the following files:

- An HTML file that is first requested by the browser.
  - really just a skeleton.
  - primary purpose is to connect the other files together.
- The CSS stylesheet (not shown)
  - to set the specific fonts, colors, etc.
- A JSON file that describes the layout of the screen.
  - similar to how DDS describes the layout of a green screen.
  - what types of fields go where
  - configuration for the fields/screen
  - connect fields to variables from the RPG program.
- Your program
  - can be written in any environment that outputs web data (PHP, Java, RPG/CGIDEV2, etc)
  - we use RPG and CGIDEV2 for this example.
- A CGIDEV2 template that describes the JSON format for the variable data.

Framework Example – HTML file

Really just ties everything together. Is the same for every application, you can just copy this file and change the program name in the "controller"

```
<html>

<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Framework Example</title>
  <link href="../profoundui/proddata/css/profoundui.css"
rel="stylesheet" type="text/css">
  <script type="text/javascript" src="runtime.js"></script>
  <script type="text/javascript">
    window.onload = function() {
      pui.controller = "/rpg/prodlist";
      pui.start();
    };
  </script>
</head>

<body>
  <div id="pui"></div>
</body>

</html>
```

Framework Example – Format (1 of 2)

With the Profound UI framework, your screen layout is described not in "code", but rather as a "list of configuration settings" kept in a file that's in JSON format.

Excerpt:

```
"items":[
  {
    "id":"MySubfile",
    "field type":"grid",
    "description":"Product Subfile",
    "css class":"crystal-grid",
    "number of rows":"20",
    "number of columns":"4",
    "column widths":"150,342,153,99",
    "column headings":"Product Id,Product Name,Price,Stock Qty",
    "header height":"26",
    "row height":"26",
    "height":"520px",
    "width":"745px",
    "record format name":"PRODSFL"
  },
... Other screen elements follow ...
```

These settings describe the layout of the screen, much like DDS would in 5250.

Framework Example – Format (2 of 2)

Beneath the layout of the grid itself, you define the fields that go into it. Notice how the "value" property is connected to variables.

```
{
  "id":"ProdId",
  "field type":"output field",
  "css class":"outputField",
  "grid":MySubfile",
  "column":"0",
  "value":{
     "fieldName":"PRID", "dataType":"char", "dataLength":"10"
  }
},
{
  "id":"ProdName",
  "field type":"output field",
  "css class":"outputField",
  "grid":"MySubfile",
  "column":"1"
  "value":{
     "fieldName":"PRNAME", "dataType":"char", "dataLength":"30"
  },
},
... More subfile fields follow ...
```

Framework Example – RPG Code

RPG code is unchanged

- RPG code is *exactly the same* as it was in the previous CGIDEV2 example
- Connected to CGIDEV2 template file.
- Load input fields from browser
- Run an SQL statement
- Output 'top', then 'detail' for each record found, then 'bottom' section.
- Only difference is the template file used.

```
getHtmlIfs('/www/skwebsrv/templates/prodlist.json');
```

Framework Example – JSON Template

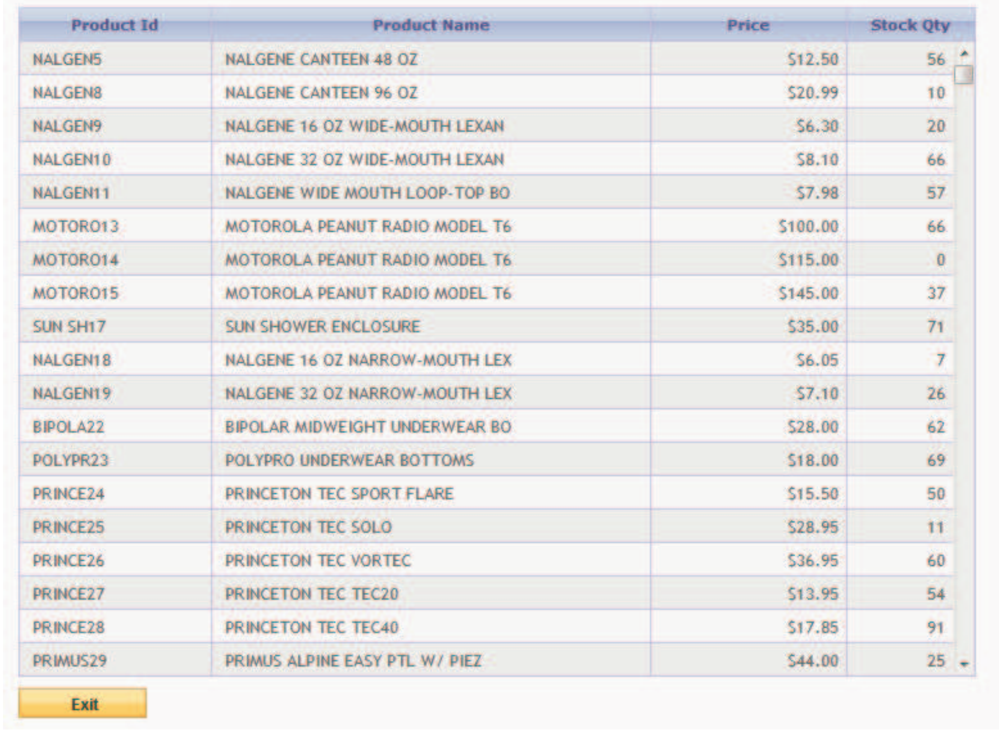
We are still using CGIDEV2, but now the template file will output JSON data connected to the variables in our screen format.

```
/$Keywords
Content-Type: text/plain

/$Top
{
  "view": "prodlist.json",
  "screen": "PRODCTL",
  "data": { "PRODSFL": [
/$Detail
         ,{
           "PRID": "/%PRID%/",
           "PRNAME": "/%PRNAME%/",
           "PPRICE": "/%PPRICE%/",
           "PSTOCK": "/%PSTOCK%/"
         }
/$Bottom
   ]}
}
```

Screen file and record formats to output

Variable values to fill in into the screen

Framework Example: Output



Product Id	Product Name	Price	Stock Qty
NALGEN5	NALGENE CANTEEN 48 OZ	$12.50	56
NALGEN8	NALGENE CANTEEN 96 OZ	$20.99	10
NALGEN9	NALGENE 16 OZ WIDE-MOUTH LEXAN	$6.30	20
NALGEN10	NALGENE 32 OZ WIDE-MOUTH LEXAN	$8.10	66
NALGEN11	NALGENE WIDE MOUTH LOOP-TOP BO	$7.98	57
MOTORO13	MOTOROLA PEANUT RADIO MODEL T6	$100.00	66
MOTORO14	MOTOROLA PEANUT RADIO MODEL T6	$115.00	0
MOTORO15	MOTOROLA PEANUT RADIO MODEL T6	$145.00	37
SUN SH17	SUN SHOWER ENCLOSURE	$35.00	71
NALGEN18	NALGENE 16 OZ NARROW-MOUTH LEX	$6.05	7
NALGEN19	NALGENE 32 OZ NARROW-MOUTH LEX	$7.10	26
BIPOLA22	BIPOLAR MIDWEIGHT UNDERWEAR BO	$28.00	62
POLYPR23	POLYPRO UNDERWEAR BOTTOMS	$18.00	69
PRINCE24	PRINCETON TEC SPORT FLARE	$15.50	50
PRINCE25	PRINCETON TEC SOLO	$28.95	11
PRINCE26	PRINCETON TEC VORTEC	$36.95	60
PRINCE27	PRINCETON TEC TEC20	$13.95	54
PRINCE28	PRINCETON TEC TEC40	$17.85	91
PRIMUS29	PRIMUS ALPINE EASY PTL W/ PIEZ	$44.00	25

Framework Example – The Payoff

The framework uses todays Web 2.0 concept of building the page dynamically with JavaScript.

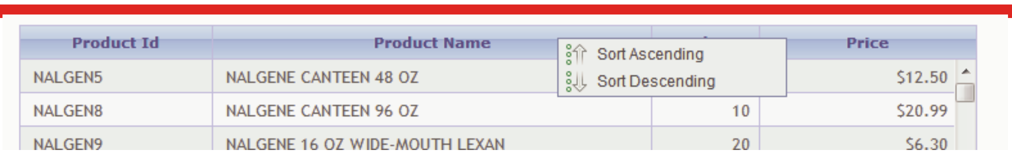
So far this seems like a little more work, but the result is approximately the same thing!! *Wasn't this going to save me work?!*

Now it will. Go back into the display format file, and add these lines to the subfile grid:

```
"persist state":"true",
"sortable columns":"true",
"movable columns":"true",
"resizable columns":"true"
```

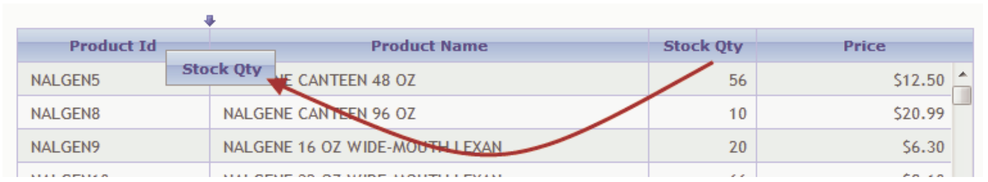
Persist state means these settings will be saved in your browser, and next time you view this screen, they will remain in effect.

Framework Example – The Payoff



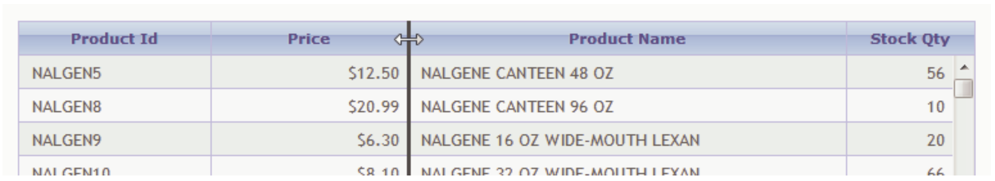
Product Id	Product Name		Price
NALGEN5	NALGENE CANTEEN 48 OZ		$12.50
NALGEN8	NALGENE CANTEEN 96 OZ	10	$20.99
NALGEN9	NALGENE 16 OZ WIDE-MOUTH LEXAN	20	$6.30

Sortable columns: Just click (or right-click for menu) to sort a column.



Product Id	Product Name	Stock Qty	Price
NALGEN5	NALGENE CANTEEN 48 OZ	56	$12.50
NALGEN8	NALGENE CANTEEN 96 OZ	10	$20.99
NALGEN9	NALGENE 16 OZ WIDE-MOUTH LEXAN	20	$6.30

Movable columns: Just drag a heading to move a column.



Product Id	Price	Product Name	Stock Qty
NALGEN5	$12.50	NALGENE CANTEEN 48 OZ	56
NALGEN8	$20.99	NALGENE CANTEEN 96 OZ	10
NALGEN9	$6.30	NALGENE 16 OZ WIDE-MOUTH LEXAN	20
NALGEN10	$8.10	NALGENE 32 OZ WIDE-MOUTH LEXAN	66

Resizable columns: Just drag a border to resize a column

Yes, it Saves Work.

Think of what it would take to code that yourself, to add:

- Sorting, both ascending and descending on all columns.
- JavaScript to allow resizing of columns in a table
- The ability to move columns around (not even possible with a standard HTML table)
- Save the settings.

1000 lines of code? More? With this framework, they only took 4.

This is only a simple example. There are bells and whistles for every single field type that you can enable. All for no charge.

Open Access Intro (1 of 2)

In 2010, IBM introduced a new feature.

Rational Open Access for RPG.

- "Open Access" or just "OA" for short.
- at the time, it was a for-money add-on.
- IBM later changed that decision, and made it an included component of the RPG compiler.
- works with F-spec defined files.
- can be any sort of file (printer, tape, workstation, etc)
- but, the most exciting use of open access is to use it for more modern displays, and today, that means a browser-based UI

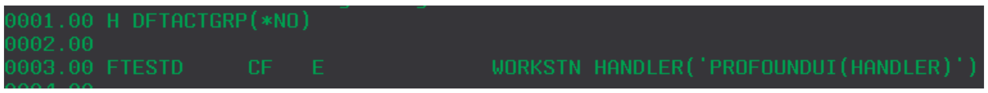
How does it work?

Think about what happens when you do an operation on a file (EXFMT, READ, WRITE, etc.) Under the covers, your RPG program is really calling a routine in the operating system that takes the definitions in your display file, and sends them out over the 5250 data stream…

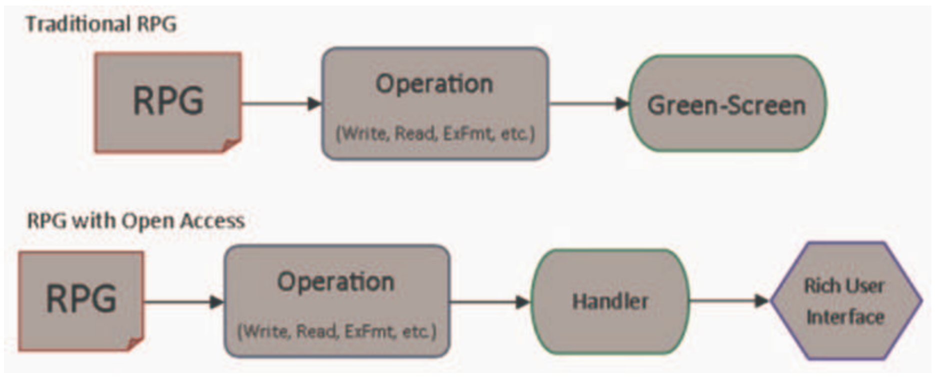
Open Access Intro (2 of 2)

Now imagine that you could replace that routine with a different one…

- Display in a browser?
- Display in a mobile app?



```
H DFTACTGRP(*NO)

FTESTD     CF   E             WORKSTN HANDLER('PROFOUNDUI(HANDLER)')
```



OA Pros and Cons

Advantages:

- Existing programs can be retro-fitted very easily. *Preserves your investment!*
  - o only required change, in most cases, is HANDLER keyword.
- OA tools (at least, the current crop) come with a visual screen editor
  - o simply replace SDA/RDi with the design tool. Keep existing workflow.
  - o dramatically lowers the learning curve
  - o lets you see what you're designing while you design it (WYSIWYG)
  - o boosts productivity
- You can concentrate on the business rules rather than the 'plumbing'.
  - o Ever feel like you don't want to fight with it, trouble shoot it, or work out all the details?
  - o You want it to "just work"?

Disadvantages:

- OA handlers are low-level (systems-type) programming.
  - Difficult to write, so most people buy from vendor.
- Can be expensive
  - but much, much less so than buying a new ERP package!
- Uses more memory than stateless (such as earlier examples herein)
  - In extremely high-volume environments, may not scale?
  - We have handler 10,000 users without problems, but this may be pushing the limits.
  - On the other hand, you are probably not Google or Facebook!!

Our Example using OA

Currently, all of the Open Access implementations that I know of are commercial products – sorry, I cannot show you a free OA implementation.

- *I work for Profound Logic*, who makes a commercial implementation.
- I will show you ours, as an example, since that's what I'm familiar with.

Profound Logic Software's *Profound UI*

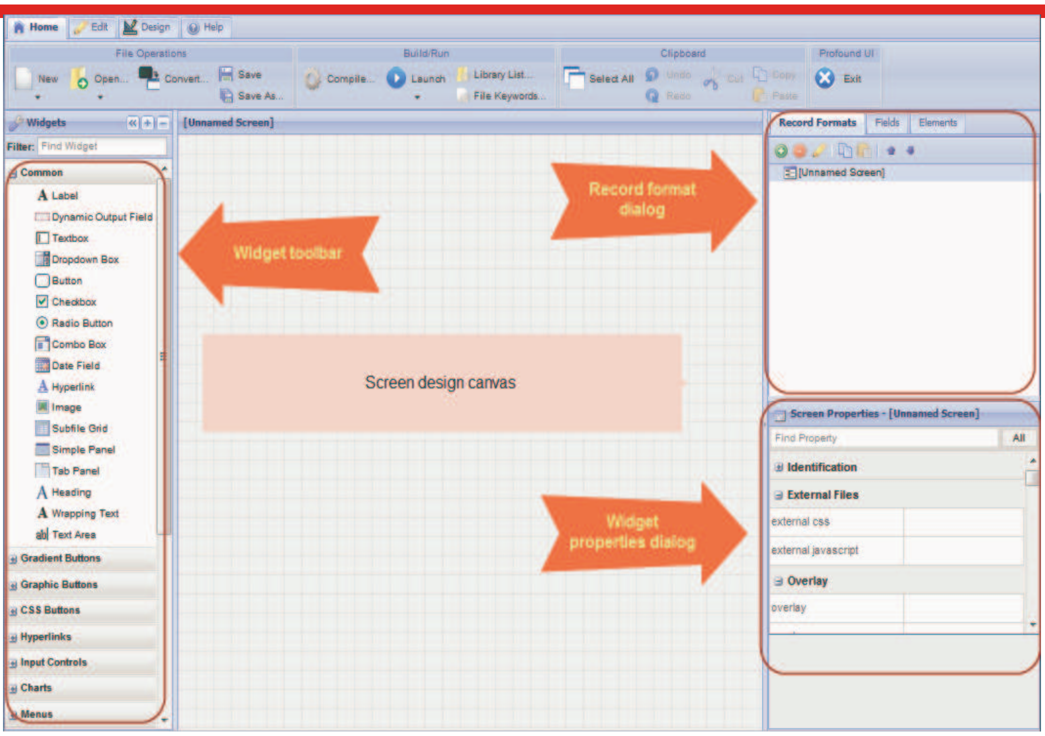
- http://www.profoundlogic.com

There are other vendors who make OA packages as well.

- LookSoftware: http://looksoftware.com
- ASNA: http://asna.com

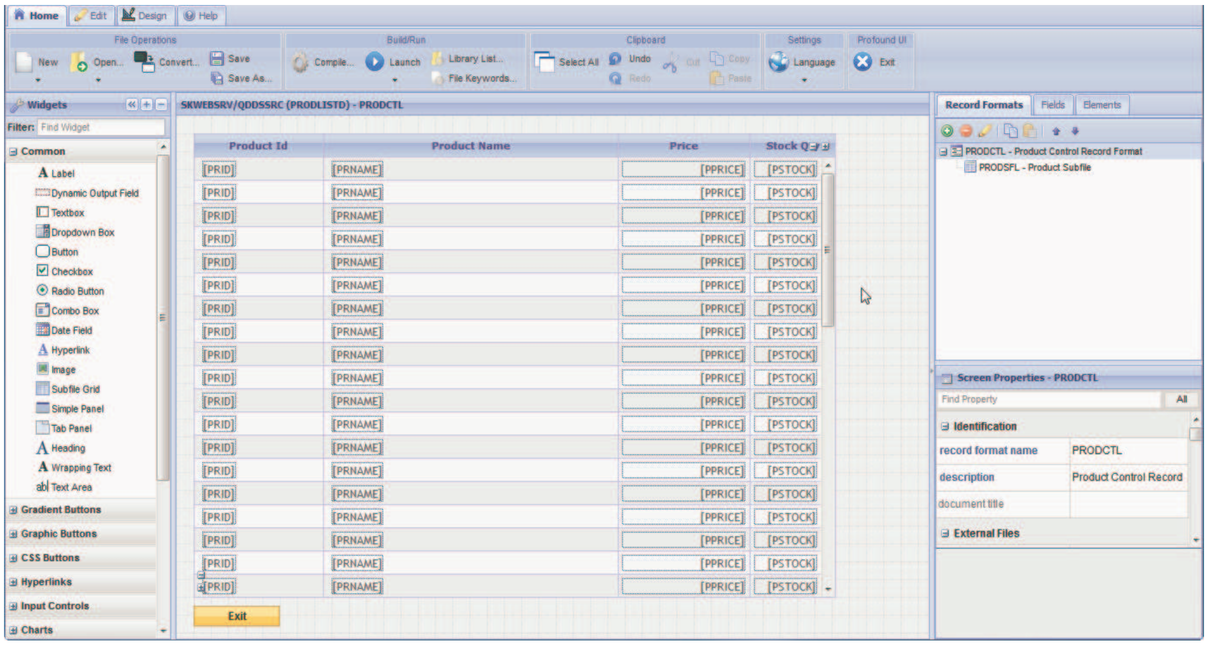
I do not want this to seem like a "sales pitch", because I would not like that if someone did it to me – but until there's a good open source OA offering, I must demonstrate a commercial tool. And Profound's is the one I'm familiar with.

Visual Designer



Open Access Screen Design

Instead of editing the JSON code, you drag & drop from the widgets toolbar, and point & click on the properties dialog to build the screen.

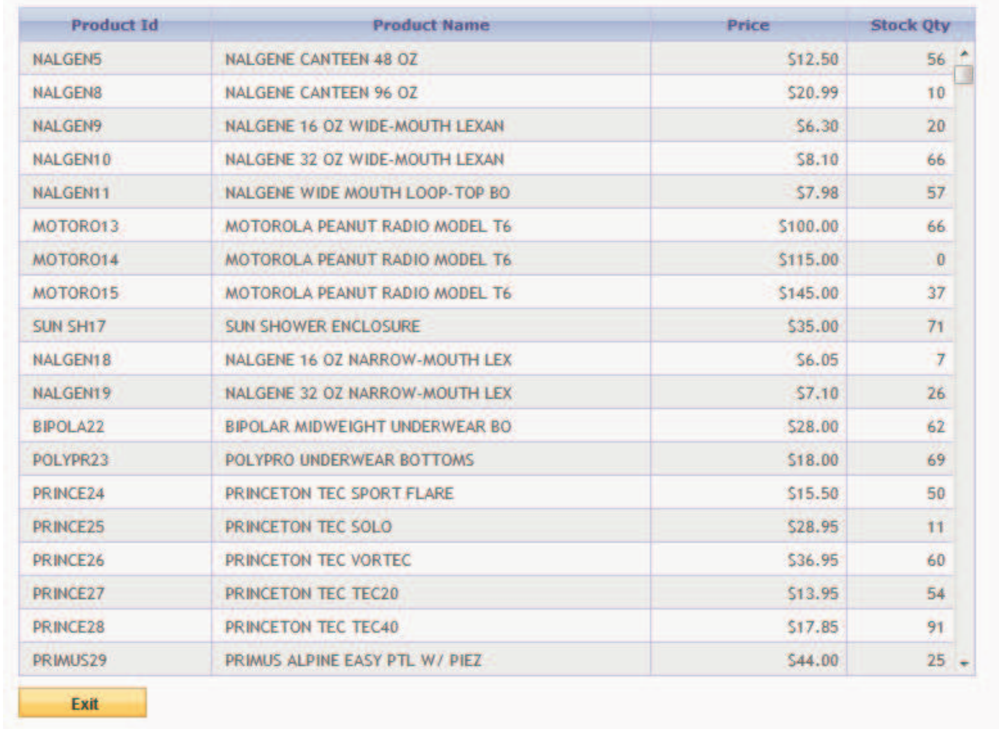


OA Example – RPG code

```
FPRODLISTD CF   E             WORKSTN SFILE(PRODSFL : RRN)
F                                     HANDLER('PROFOUNDUI(HANDLER)')
  . . .
   Exec SQL DECLARE cursor1 SCROLL CURSOR FOR
       SELECT PRID, PRNAME, PPRICE, PSTOCK FROM PRODUCTSP;
   Exec SQL Open cursor1;
   Exec SQL Fetch cursor1 INTO :PRID, :PRNAME, :PPRICE, :PSTOCK;

   Dow %subst(sqlstt:1:2)='00' or %subst(sqlstt:1:2)='01';
      RRN +=1;
      Write PRODSFL;
      Exec SQL Fetch cursor1 INTO :PRID, :PRNAME, :PPRICE, :PSTOCK;
   EndDo;

   Exec SQL Close cursor1;

   ExFmt PRODCTL;
```

This is standard free-format RPG. (Works with fixed format as well.)

OA Output (same as framework)



Product Id	Product Name	Price	Stock Qty
NALGEN5	NALGENE CANTEEN 48 OZ	$12.50	56
NALGEN8	NALGENE CANTEEN 96 OZ	$20.99	10
NALGEN9	NALGENE 16 OZ WIDE-MOUTH LEXAN	$6.30	20
NALGEN10	NALGENE 32 OZ WIDE-MOUTH LEXAN	$8.10	66
NALGEN11	NALGENE WIDE MOUTH LOOP-TOP BO	$7.98	57
MOTORO13	MOTOROLA PEANUT RADIO MODEL T6	$100.00	66
MOTORO14	MOTOROLA PEANUT RADIO MODEL T6	$115.00	0
MOTORO15	MOTOROLA PEANUT RADIO MODEL T6	$145.00	37
SUN SH17	SUN SHOWER ENCLOSURE	$35.00	71
NALGEN18	NALGENE 16 OZ NARROW-MOUTH LEX	$6.05	7
NALGEN19	NALGENE 32 OZ NARROW-MOUTH LEX	$7.10	26
BIPOLA22	BIPOLAR MIDWEIGHT UNDERWEAR BO	$28.00	62
POLYPR23	POLYPRO UNDERWEAR BOTTOMS	$18.00	69
PRINCE24	PRINCETON TEC SPORT FLARE	$15.50	50
PRINCE25	PRINCETON TEC SOLO	$28.95	11
PRINCE26	PRINCETON TEC VORTEC	$36.95	60
PRINCE27	PRINCETON TEC TEC20	$13.95	54
PRINCE28	PRINCETON TEC TEC40	$17.85	91
PRIMUS29	PRIMUS ALPINE EASY PTL W/ PIEZ	$44.00	25

Summary

- We have come a long way in terms of user interfaces with web
- RPG is a better language for business rules
- RPG can create web applications just as well as other languages
  - though, other languages will help if you want to be cross-platform
  - and in some cases, might make it easier to find talent
  - the challenges of writing web applications (html, css, javascript) are the same no matter what the server-side language is!
- Writing applications that output HTML pages is passé (in any language)
- Frameworks can really improve functionality, ease-of-use, and maintenance
- Open Access by itself is just a way to add a "handler" to your F-spec.
- There are Open Access packages on the market that make the job easier
- Open Access can also help preserve your investment in your applications

This Presentation

You can download a PDF copy of this presentation from:

http://www.scottklement.com/presentations/

Thank you!