



Microsoft®
.net™ +



=



(those are *rock fingers* for us laymen)



THE INTRO

The next generation user interface for RPG!

by Aaron Bartell
of KregelTech

aaronbartell@mowyourlawn.com

ABSTRACT

I have always been intrigued with how RPG can create modern looking applications – in particular, Internet browser based applications. Along with that I have found there is an **art** of sorts in getting **RPG to talk with the browser** and finding the right tools to use in doing that. This sessions describes the journey I took in my most recent open source solution titled “OpenRPGUI” (aka RPG User Interface). I will walk through the motions I took to discover what tools are available and how I put them together to create an easy to use framework (**free and open source!!**). Many things will be discussed including how to setup a simple **Apache** configuration, talk about the high-level concept of **communicating with the browser**, and how messages are passed back and forth.

THE STATE OF THE UI

UI is in a state of flux and it will most definitely change from year to year.

Considerations:

- What is required to be downloaded to the client.
- What platforms/browsers support it.
- New technologies survive based on **popularity**.

Processing is moving back **into “the cloud”** (i.e. back onto the server).



WHAT WE HAVE

Programming On IBM i

- RPG - Modular language with simple syntax
- Second to none integration with DB and operating system
- Very low maintenance DB
- Debug jobs on the server in real-time, one line at a time (i.e. STRSRVJOB and Service Entry Points). Try to do that with Java - the most popular language in the world.
- Environment controlling mechanism (i.e. Library Lists)
- Extensive Process/Job control (WRKACTJOB)
- Call stacks (great for a program reuse as a called program doesn't know or care who called it, it just receives in data and passes back data)
- Wait based technologies to alleviate polling which saves on system resources (i.e. data queues, job queues)
- Robust command line interface that you can ALWAYS count on for doing nearly EVERYTHING.



WHAT WE LACK

- **A modern User Interface** that has simplicity in development, deployment, and is similar to traditional DSPF approach where we can configure a screen and subsequently “talk” to it.
- Easy to pull from, and extensive, open source repositories.
- An “IBM i App Store”
- Ability for new people to be exposed to the IBM i and RPG.

What if you could...

...lease an IBM i in the cloud?

... www.iDevCloud.com



ENTER OPENRPGUI

OpenRPGUI (Open RPG User Interface) aims to address the point of not having a good way to convey a modern looking interface from the RPG language.

- Open source (LGPL) - free as in beer and in liberty.
- Hosted on the **worlds most prominent open source site** – www.SourceForge.net which facilitates a source repository, user forum, developer forum, MediaWiki for documentation, place to log bugs, place to log feature requests.
- **Marketed** through the YiPs (COMMON), article writing, and news updates.
- Aims to **re-use all** of the great features of the **RPG+DB2+IBMi** programming stack (stateful jobs, library lists, etc).
- Aims to build a **UI abstraction layer** so you aren't tied to a particular UI technology that may die in the future.
- **ExtJS** is currently the main UI technology being implemented.
- Uses **JSON *SRVPGM** from **Mihael Schmidt** of www.RPGNextGen.com

OpenRPGUI Home Page:

www.OpenRPGUI.com

... articles, downloads, updates, support.

open
RPGUI



ExtJS is the current UI implementation of OpenRPGUI. ExtJS is a cross-browser JavaScript library for building rich internet applications.

Download: www.ExtJS.com

Search morl Display 5 Go Reset

GeoNames

| Name ^ | Geoname ID | Modification Date | Latitude | Longitude |
|------------------------|------------|-------------------|------------|-----------|
| Dalton in Westmorla... | 6288100 | 21-SEP-2006 | 54.17142 | -2.70767 |
| Llanmorlais | 2643976 | 22-SEP-2004 | 51.6302778 | |
| Loch Morlich | 2642187 | 08-OCT-2004 | 57.1666667 | |
| Morlais Castle | 6286663 | 21-SEP-2006 | 51.7767 | |
| Morland | 2642190 | 15-JUN-2007 | 54.5833333 | |

Page 1 of 3 Export to Excel

GeoNames 2nd

| Country Code ^ | Name | Modification Date | Latitude | Lon |
|----------------|----------------|-------------------|------------|------|
| GB | | 21-09-2006 | 54.17142 | -2.7 |
| GB | | 22-09-2004 | 51.6302778 | -4.1 |
| GB | | 08-10-2004 | 57.1666667 | -3.7 |
| GB | Morlais Castle | 21-09-2006 | 51.7767 | -3.2 |
| GB | Morland | 15-06-2007 | 54.5833333 | -2.6 |

Country Code: GB

- Sort Ascending
- Sort Descending
- Columns
- Group By This Field
- Show in Groups

Page 1 of 3 Export to Excel

Fantasy Football League Teams

Add a Team Delete selection(s)

| # | Team Name | Owner | Wins ^ | Losses |
|--------|----------------------|-----------|--------|--------|
| E816E4 | Baked Snack Crackers | BSC | 10 | 0 |
| 11FAA | Pink Ladies | Hellerman | 9 | 1 |
| 47E447 | World Wide Willy | WWW | 5 | 5 |
| B19E20 | Dude wheres my Team | Traut | 5 | 5 |
| 8A9D5 | Bodiack | Kolwoman | 1 | 9 |
| ED017 | DeanEyde | Niel | 0 | 10 |

Page 1 of 1 Displaying 1 - 6 of 6

WHY EXTJS?



Pros

- Provides rich and configurable UI components
- Focuses on configuration out-of-the-box so you don't have to write your own HTML, JavaScript* or CSS.
- Screen Designer (this is still something that needs more)
- ExtJS offers a variety of paid support options - a must for high profile sites.
- Extensive formal documentation and community supplied examples.
- Most popular HTML+CSS+Javascript framework. Stands a high chance of sticking around. Lots of online help and code.
- **Has dual license (Open source and Commercial version).

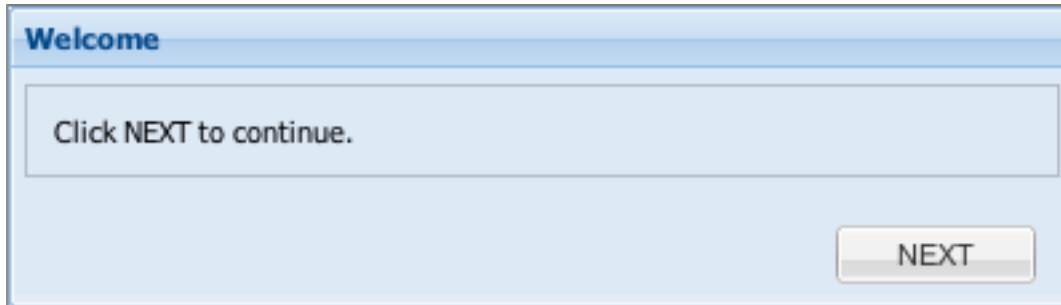
Cons

- **Has dual license (Open source and Commercial version).
- Initial download of ExtJS can take awhile on slower connections (i.e. 300k to 400k)
- Doesn't work on all handheld devices (i.e. Blackberry)

* If you want features that go significantly outside of the framework then custom Javascript can be a reality.

** Some view the dual license as a pro and others as a con.

DEFINING AN EXTJS PANEL



1 – User goes to a static document URL like <http://company.com/dspf/custmaint.html>. File ***custmaint.html*** is downloaded to the browser along with any references to other resources (i.e. the ExtJS files).

2 – The browser executes the ExtJS javascript and discovers an initial panel to display and subsequently renders it.

In this scenario the ***pnl1*** definition is set to **'hidden: false'** so it will be displayed.

Note the reference to ***btnHandler*** and ***submitForms***.

ExtJS Panel definition syntax

```
var pnl1 = new Ext.Panel({
  id: 'pnl1',
  frame: true,
  hidden: false,
  title: 'Welcome',
  width: 400,
  items: [{
    xtype: 'fieldset',
    html: 'Click NEXT to continue.'
  }],
  buttons: [{
    id: 'btnNxtPnl1',
    text: 'NEXT',
    handler: btnHandler,
    submitForms: ['pnl1']
  }]
});
```

PROCESSING A BUTTON

Function ***btnHandler*** is very generic in nature so it can handle all button interactions. The ***button*** parm was passed into ***btnHandler*** and that is what we use to access the ***submitForms*** config option that is passed into ***getFormVars()***.

When the communication with the server is successful it will then go to the ***success*** config option and invoke the code in there.

Ext.util.JSON.decode is taking the JSON response and converting it into a Javascript object. Similar to taking an RPG string and applying it to an RPG data structure.

```
function btnHandler(button, event){
  Ext.Ajax.request({
    url: '/oru12web/oruintro',
    params: getFormVars(button.submitForms) +
      '&action=' + button.getId() + back2me(),
    success: function(r, o) {
      rsp = Ext.util.JSON.decode(r.responseText);
      uiActn(rsp);
    },
    failure: function(o,r){
      if(actn.failureType == 'server'){
        svrRsp = Ext.util.JSON.decode(actn.response.responseText);
        Ext.Msg.alert('Oops!', svrRsp.msg);
      }else{
        Ext.Msg.alert('Oops!', 'Server is unreachable');
      }
    }
  });
}
```

The RPG program

WHAT IS JSON?

JSON = JavaScript Object Notation (www.JSON.org)

- Lightweight data-interchange format.
- It is easy for humans to read and write.
- It is easy for machines to parse and generate.

JSON Response From Server

```
{  
  "msg" : "" ,  
  "uiactn" : [ {  
    "hide" : "pn1"  
  } , {  
    "show" : "pn2"  
  } ]  
}
```

JSON

JavaScript Object Notation

HOW UIACTN WORKS (Javascript side)

Function **uiActn** is invoked from **btnHandler** and dynamically iterates through the JSON response and determines what actions it needs to take against the current UI (user interface). In the below case the server said to **hide pnl1** and to **show pnl2**.

JSON Response

```
{
  "msg" : "",
  "uiactn" : [ {
    "hide" : "pnl1"
  }, {
    "show" : "pnl2"
  } ]
}
```

Javascript for uiActn

```
function uiActn(jsonObj){
  Ext.iterate(jsonObj,function(key, value){
    if(key == 'uiactn'){
      Ext.iterate(value,function(uiCmpNam, value2){
        for(var uiActn in uiCmpNam){
          var cmp = Ext.getCmp(uiCmpNam[uiActn]);
          if(cmp != undefined){
            if(uiActn == 'hide'){
              cmp.hide();
            }else if(uiActn == 'show'){
              cmp.show();
            }else if(uiActn == 'disable'){
              cmp.setDisabled(true);
            }else if(uiActn == 'enable'){
```

HOW UIACTN WORKS (Javascript side)

...continued

Function uiActn also can "set" a value of a field. In this case it will set the field named **msg** to blanks.

JSON Response

```
{
  "msg" : "",
  "uiactn" : [ {
    "hide" : "pn11"
  }, {
    "show" : "pn12"
  } ]
}
```

Javascript for uiActn

```
}else{
  try{
    var cmp = Ext.getCmp(key);
    if(cmp != undefined){
      if(cmp.getXType() == 'combo'){
        cmp.setValue(trim(value));
      } else if(cmp.getXTypes().indexOf('datefield') > 0){
        cmp.setValue(value);
      } else if(cmp.getXType() == 'numberfield'){
        cmp.setValue(value);
      } else if(cmp.getXType() == 'textfield'){
        cmp.setValue(trim(value));
      } else if(cmp.getXType() == 'label'){
        cmp.setText(trim(value), false);
      } else if(cmp.getXType() == 'checkbox'){
        cmp.setValue(value);
      }
    }
  }
}catch(err){
  alert('Name:' + cmp.getId() + ' typeof:' + typeof value);
}
}
```

RPG *ORUINTRO*

The general flow behind each interaction with the ***ORUINTRO*** RPG program is as follows:

1 - Apache invokes ***ORUINTRO*** the same as if it was called from the command line.

2 - A sub routine named ***initpgm*** is invoked which reads in the request and obtains the action that is to be taken.

3 - There is a big ***select*** statement that checks the ***action*** and subsequently invokes a same named locally defined sub procedure.

4 - The ***btnNxtPnl1*** sub procedure is where you put your DB2 table access, business logic, and controller logic.

5 - The last thing performed is the ***endpgm*** sub routine. This will send the JSON request back down to the client. More on JSON in a few slides!

```
monitor;  
  
exsr initpgm;  
  
select;  
when gIn.action = 'btnNxtPnl1';  
    btnNxtPnl1();  
  
when gIn.action = 'btnBckPnl2';  
    btnBckPnl2();  
  
when gIn.action = 'btnNxtPnl2';  
    btnNxtPnl2();  
  
when gIn.action = 'btnCnclPnl3';  
    btnCnclPnl3();  
  
when gIn.action = 'btnStayPnl3';  
    btnStayPnl3();  
  
when gIn.action = *blank;  
    gMsgTxt = 'No action sent.';  
other;  
    gMsgTxt =  
        'Action ' + gIn.action + ' not supported.';  
endsl;  
  
on-error;  
    gMsgTxt =  
        'Unexpected error occurred. Please review joblog.';  
endmon;  
  
exsr endpgm;  
  
*inlr = *on;  
return;
```

RPG *initpgm*

The ***initpgm*** sub routine does many things including creating the two “JSON objects” and reading in the content from the browser.

gJRsp was created to hold the entire JSON response that will be sent back down to the browser. We initialize it here because we will incrementally be adding to it throughout the invocation of this program.

gJuiActnArr was created to hold the various user interface actions we wish to relay back down to the browser.

gIn.stdIn holds the entire request that was sent up from the browser.

gIn.action is occupied by calling the ***http_getCgiVal*** to retrieve the ***action*** currently residing in **gIn.stdIn**.

```
begsr initpgm;  
  clear gIn;  
  clear gMsgTxt;  
  gJRsp = json_create();  
  gJuiActnArr = jsona_create();  
  gIn.stdIn = http_inStr();  
  gIn.action = http_getCgiVal('action': gIn.stdIn);  
endsr;
```

RPG *btnNxtPnl1* AND *uiActn*

The ***btnNxtPnl1*** sub procedure tells the user interface to ***hide pnl1*** and ***show pnl2*** using the ***uiActn*** locally defined sub procedure. The ***uiActn*** sub procedure simply adds a new entry into the ***gJuiActnArr*** JSON "object". Basically saves three lines of code each time - an ease of use sub procedure.

```
P uiActn          b
D uiActn          pi
D pAct            10a  value varying
D pCmp            30a  value varying

D je              s      *
/free

  je = json_create();
  json_putString(je : pAct: pCmp);
  jsona_putObject(gJuiActnArr : je);
  return;

/end-free
P                e
```

```
P btnNxtPnl1     b
D btnNxtPnl1     pi
/free

  uiActn(HIDE: 'pnl1');
  uiActn(SHOW: 'pnl2');

/end-free
P                e
```

RPG *endpgm*

The last thing that runs on the RPG side is the ***endpgm*** sub routine. It occupies the generic ***msg*** JSON variable with the contents of RPG variable ***gMsgTxt***.

It places the ***gJuiActnArr*** array into gJRsp with name ***uiactn***.

Then the response is sent back to the browser by first sending the ***Content-Type*** followed by two line feeds (an HTTP "rule").

And finally we clean up by invoking ***json_dispose()***.

```
begsr endpgm;  
  json_putString(gJRsp: 'msg': gMsgTxt);  
  json_putArray(gJRsp : 'uiactn' : gJuiActnArr);  
  http_outStr('Content-Type: text/plain' + x'1515');  
  http_outPtr(json_toString(gJRsp));  
  json_dispose(gJRsp);  
endsr;
```

pnl2 DISPLAYED

Panel 2 (**pnl2**) is now shown and Panel 1 (**pnl1**) is hidden.

Here we see the btnNxtPnl2 RPG sub procedure that will process the NEXT button from **pnl2**.

If the user selects the NEXT button it will hide pnl2, show **pnl3** and set pnl3's form field values, using **json_putString** and **json_putInt**, so that it will be pre-filled with information.

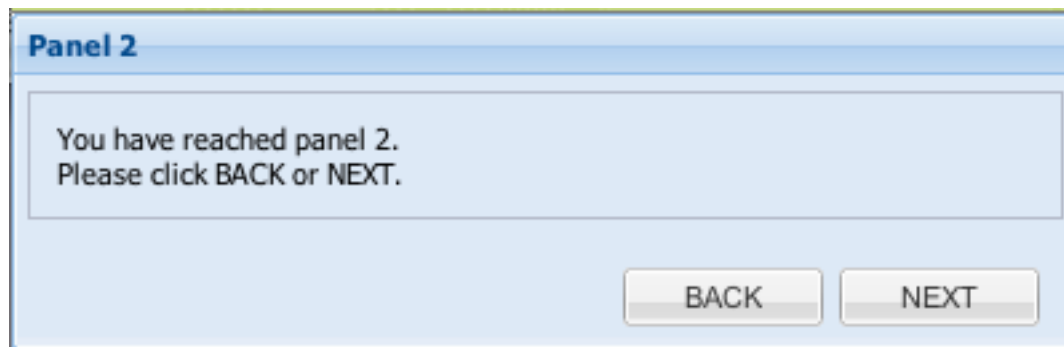
This would be similar to displaying a record from your database onto a green screen.

```
P btnNxtPnl2      b
D btnNxtPnl2      pi
/free

uiActn(HIDE: 'pnl2');
uiActn(SHOW: 'pnl3');

json_putString(gJRsp: 'NAM': 'Aaron Bartell');
json_putString(gJRsp: 'CRDCD': 'Pretty Good');
json_putString(gJRsp: 'DOB': '1979-04-22');
json_putInt(  gJRsp: 'AGE': 31);
json_putString(gJRsp: 'TIMESTAMP': %char(%timestamp()) );

/end-free
P                                  e
```



The screenshot shows a window titled "Panel 2" with a light blue background. Inside the window, there is a text box containing the message: "You have reached panel 2. Please click BACK or NEXT." Below the text box, there are two buttons: "BACK" and "NEXT".

pnl3 DISPLAYED

```
P btnStayPnl3      b
D btnStayPnl3      pi

D crdCd            s          20a
/free

  crdCd = http_getCgiVal('CRDCD': gIn.stdIn);

  if crdCd = 'Horrible';
    gMsgTxt = 'Hmm... maybe you should go elsewhere';
  endif;

/end-free
P                  e
```

Panel 3 (**pnl3**) is now shown and Panel 2 (**pnl2**) is hidden.

Here we see the **btnStayPnl3** RPG sub procedure. If the user presses the STAY button, then this sub procedure will check to see if the CRDCD is 'Horrible' and if so will set the gMsgTxt to convey a message.

The screenshot shows a dialog box titled "Enter Data" with a light blue header. It contains five input fields and two buttons. The fields are: "Name:" with the text "Aaron Bartell"; "Credit Code:" with a dropdown menu showing "Pretty Good"; "Date of Birth:" with the text "1979-04-22" and a calendar icon; "Age:" with the text "31"; and "IBM i timestamp:" with the text "2010-11-23-10.23.52.125". At the bottom right, there are two buttons: "CANCEL" and "STAY".

REQUIRED OBJECTS

OpenRPGUI is **very lightweight** in nature.

Below is a listing of the objects in the ORU11 library that are required for runtime. This is as-of OpenRPGUI v1.2

| Object | Type | Library | Attribute | Text |
|----------|---------|---------|-----------|------|
| ORUINTRO | *PGM | ORU12 | RPGLE | |
| HTTP | *SRVPGM | ORU12 | RPGLE | |
| JSON | *SRVPGM | ORU12 | | |



APACHE CONFIG

The following shows how Apache is configured for the OpenRPGUI installation. The **Directory** directives allow for both static content and program calls.

```
Listen *:81
DocumentRoot /www/oru12/htdocs
ScriptAliasMatch ^/oruweb12/(.*) /qsys.lib/oru12.lib/$1.pgm

<Directory /www/oru12/htdocs>
    order allow,deny
    allow from all
</Directory>
<Directory /qsys.lib/oru12.lib>
    allow from all
    order allow,deny
    options +ExecCGI
</Directory>
```



Apache

CURRENT NEEDS

OpenRPGUI is a community project that exists solely based on donated time and code. So far we have three contributors: **Aaron Bartell**, **Rory Hewitt**, and **Mihael Schmidt**. If you would like to *help with OpenRPGUI* then please head over the **Feature Request page** to see if you can fulfill any of the needs, or add your own idea/need.

Feature Request URL link: <http://OpenRPGUI.com/support>

Highest need right now: Post example code of your successes on sourceforge. Examples don't have to be big, they can show how to do a simple task.

Convert physical file data structure and contents to JSON. This requires calling system API's to decipher what a particular DB2 table layout is and subsequently "parse" the received in data structure pointer into a JSON string based on the meta-data received back from the system API's.



WE HAVE REACHED THE END!



Aaron Bartell
aaronbartell@mowyourlawn.com



lead developer of RPG-XML Suite (www.rpg-xml.com)

and owner of www.MowYourLawn.com

and check out his latest effort at www.SoftwareSavesLives.com

MOWYOURLAWN.COM



twitter.com/aaronbartell

