CGIDEV2-Easiest and Quickest way to the web

- What really is Web Programming?
- Web programming is nothing more than a program sending a string of text, via TCP/IP, to a browser.
- RPG is trusted to run your business, could it also be trusted to “push a few bytes” to a browser?
- Of course it can!

CGIDEV2-Easiest and Quickest way to the web

- There are several technologies used in Web development:
1. The markup language (HTML)
2. The formatting language, Cascading Style Sheets(CSS)
3. The scripting language (JavaScript)
4. Host Language (in CGIDEV2 = RPG)

- The Host Language is the most important and most difficult to learn
  - Does all the business logic
  - Does all the Database I/O
  - AKA all the heavy lifting.
- You don't have to be an expert in HTML, CSS or JavaScript to be an effective web developer, but you do have to be an expert in the Host Language.
- What if I were to tell you that you are already an expert in the most important and most difficult part of web programming?

CGIDEV2-Fastest way to the web

Using what you already know to create web applications

disclaimer:
All code examples contained herein are provided to you "as is". THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE EXPRESSLY DISCLAIMED.

This is a introduction to CGIDEV2 presentation is not a tutorial on RPG, HTML, CSS or JavaScript.

Unless otherwise noted, all features covered in this presentation apply to all releases of RPG greater than or equal to V5R4.

So what in CGIDEV2 is new to you?

1. I/O – Getting data from and writing data to different software– Browser
2. *Key concept* –Template file(s) – HTML
    1. 5 CGIDEV2 functions built around this concept.
        1. 1 – Read HTML Template(s) into your program
        2. 2 – To retrieve and parse incoming Data
        3. 2 – To update and write the HTML to the browser.

Template file(s) – HTML

IFS stored HTML file(s) that CGIDEV2 will be read into your RPG program.

Your RPG program will update and write the HTML to the buffer and then send the contents of the buffer to the browser.

1. HTTP Header
2. Sections
3. Substitution Variables

Template file(s)-HTML
1). HTTP header

The first line of the HTML being sent to the browser must contain the following HTTP header followed by 2 carriage returns(enter key twice):

Example:

<IBMi>customerheader
Content-type: text/html
{blank line}
{blank line}
<html>
**** followed by more html content ****

Browser receives a string of bytes, HTTP header tells browser how to render those bytes.

Template file(s)-HTML
2). Sections

1. Similar to DSPF/PRTF DDS record format
2. Must start in the first character of the HTML line
3. Can be a maximum of 50 characters long
4. Can be a variable in your RPG application
5. Delimiters can be user defined(default /$)
6. Section values are not written to the browser
7. Maximum 200 sections per web page.
8. Max number of bytes buffered approx. 2 terrabytes

Template file-HTML
2). Sections (continued)

Example:

```
</TR>
<IBMi>customerdetail
    <TR class="/%dclass%/">
      <TD align="left" nowrap><A href="../cgi-
bin/wcust002?wLSTNAM=/%LSTNAM%/&wCUSNUM=/%CUSNUM%
/&tstamp=/%tstamp%/&sesid=/%sesid%/&rtnpgm=/%pgmid%/"
>/%LSTNAM%/</TD>
      <TD align="left" nowrap>/%INIT%/</TD>
      <TD align="left" nowrap>/%STREET%/</TD>
      <TD align="left" nowrap>/%CITY%/</TD>
      <TD align="left" nowrap>/%STATE%/</TD>
      <TD align="left" nowrap>/%ZIPCOD%/</TD>
    </TR>
<IBMi>customerfooter
</TABLE>

wrtsection('customerdetail');
```

Template file(s) – HTML
3). Substitution Variables

Delimiters defined in *gethtmlifsmult* function

Defaults:
variable name start delimiter: /%
variable name end delimiter: %/
Example:

<TD nowrap><INPUT type="text" name="wSTATE" value="/**%wSTATE%**/" maxlength="2" size="2" style="/**%wSTATEsty%**/" ></TD>

To retrieve the values of these substitution variables into you RPG program variables:

wSTATE = zhbGetVar('wSTATE');

RPG code to update the values of these substitution variables in your HTML:

UpdHtmlVar('wSTATE':wSTATE);
UpdHtmlVar('wSTATEsty':wSTATEsty);

RPG Source Modifications

```
H bnddir('TEMPLATE2')

  /copy *libl/qrpglesrc,prototypeb
  /copy *libl/qrpglesrc,usec

   //Indicators for GetHtmlIfsMult subprocedure
D IfsMultIndicators...
D                         ds
D  NoErrors                       n
D  NameTooLong                    n
D  NotAccessible                  n
D  NoFilesUsable                  n
D  DupSections                    n
D  FileIsEmpty                    n
   // Saved query string
Dsavedquerystring...
D                                 s         32767      varying
DnbrVars                          s            10i  0
```

Minimum 5 CGIDEV2 functions

1. ZhbGetInput - **Read Input from browser**
2. zhbGetVar - **Map variables from query string**
3. getHtmlIfsMult - **Load Response HTML template**
4. updHtmlVar - **Update HTML substitution variables**
5. wrtSection - **Output the HTML**

zhbgetinput – Read Input from browser

Zhbgetinput(querystring: Environment_datastructure)

Example:

nbrVars = zhbgetinput(savedquerystring:qusec);

- Where QUSEC is the data structure that contains errors.
- nbrVars contains number of variables defined in the query string.
- Same function for GET or POST methods to retrieve the name/value pairs.

zhbGetVar – Map values from query string

zhbGetVar(variable_name)

Examples:

```
<TD align="left" nowrap><A href="../cgi-
bin/wcust002?wLSTNAM=/%LSTNAM%/&wCUSNUM=/%CUSNUM%
/&tstamp=/%tstamp%/&sesid=/%sesid%/&rtnpgm=/%pgmid%/"
>/%LSTNAM%/</TD>
```

```
rtnpgm  = zhbGetVar('rtnpgm');
```

Extract numeric fields from the querystring:

```
D BigDec         s              30p 9

BigDec = c2n2(zhbGetVar('wCUSNUM'));
wCUSNUM    = BigDec ;
```

gethtmlifsmult – Load template HTML

Gethtmlifsmult(IFS_path_and_HTML_file:
section_id_start:
section_id_end:
variable_id_start:
variable_id_end)

Example:

```
D IfsMultIndicators...
D                                ds
D  NoErrors                       n   *on = no error occurred *off = one or more errors. Check other
                                      indicators.
D  NameTooLong                    n   *on = one or more file's name exceeds 255 characters. File is ignored.
D  NotAccessible                  n    on = File or directory not found, authorization failure, etc. File is
                                      ignored.
D  NoFilesUsable                  n   *on = All the files have been ignored.
D  DupSections                    n   *on = One or more duplicate sections were found. Only the first
                                      occurrence is used.
D  FileIsEmpty                    n   *on = File is empty and is ignored.

IfsMultIndicators = gethtmlifsmult(ifspath:'<IBMi>');
```

gethtmlifsmult – Load HTML template (continued)

Allows the programmer to define their own Section and Variable delimiters, example:

gethtmlifsmult(ifspath: '<!-- Sec_': ' -->':'<var400>': </var400>')

ifspath = '/' + %trim(pgmlib) + '/html/' + %trim(pgmid) + '.html';

Defaults provided are:

Section name start delimiter: /$

Substitution Variable name start delimiter: /%

Substitution Variable name end delimiter: %/

1. Maximum 200 sections per web page.
2. Section names can be 50 characters long.
3. Substitution Variables name cane be up to 30 characters long.
4. Delimiters length can be 10 characters long.

updHtmlVar – Update HTML substitution variables

updHtmlVar(variable_name:variable_value:action)

action (optional)
'1' = replace this variable if it exists, otherwise add it (default)
'0' = clear arrays and write this one as the first

Examples:

<TD nowrap><INPUT type="text" name="wSTATE" value="/%wSTATE%/" maxlength="2" size="2" style="/%wSTATEsty%/" ></TD>

UpdHtmlVar('wSTATE':wSTATE);
UpdHtmlVar('wSTATEsty':wSTATEsty);

1. Substitution variables name cane be up to 30 characters long.
2. 32,767 unique variable names are supported per page.

wrtsection – output the HTML to the Buffer/Browser

Wrtsection('section1 section2 etc')

Works like DDS writes all following HTML until it reaches next Section name.

Example:
HTML:

```
//Write header section to Buffer:
  wrtsection('customerheader');
//write buttons according to Mode.
  if mode='ADD' or Mode='COPY';
  wrtsection('addbuttons');
  else;
  wrtsection('maintbuttons');
  endif;
//Write footer section and Flush Output Buffer:
  wrtsection('customerfooter *fini');
```

MVC support

1. Supports MVC through “loose” binding of external HTML with RPG
2. Changes to HTML have no effect on RPG program and visa versa.
3. *Everything* in the HTML Template source can be treated as a variable in your RPG:
    1. HTML file(s) names and paths to those files
    2. The Sections (record formats)of HTML you write
    3. The Substitution Variables (fields) you update in the HTML file.
        1. Not just database fields, HTML, CSS and JavaScript can be rendered dynamically
4. With great power, comes great responsibility (*Voltaire*). CGIDEV2 is very forgiving:
    1. Will not hard halt your RPG application if:
        1. HTML files do not exist
        2. Section you are witting does not exist
        3. Substitution variable you are trying to update does not exist
    2. Web page may not render correctly.

Closing Notes

Why does CGIDEV2 offer the best performance on the most modest of systems?

1. Compiled, optimized RPG (need I say more?)
2. Least system overhead, only Apache HTTP server.
3. Once called, RPG programs stay resident in memory (don't set on *INLR).
4. IFS HTML files are only read once and are shared within the same activation group.
5. Download MMAIL instead of CGIDEV2.
6. Did I mention that you get the CGIDEV2 source code!

Your RPG expertise + the performance + cost and ease of use of CGIDEV2, makes for a winning combination!