CGIDEV2

A tool allowing i5 developers to use HTML as an alternative to DDS

Presented by Gordon Larkins

Gordon is a seasoned developer with over 25 years of experience. He has been employed with the IT consulting firm ASK for the past 10 years. ASK is based in Lansing, Michigan and provides IT infrastructure, application development, and remote managed services to companies all over Michigan. Gordon's software development repertoire includes:

- RPG Free, ILE, 400, III, II
- CGIDEV2
- PHP
- SQL
- Javascript
- Ajax
- HTML
- XML – (Web services from the i5)
- Java
- LotusScript

Gordon has been developing web applications utilizing CGIDEV2 for over seven years. These applications were designed and created for multiple clients in various industries.

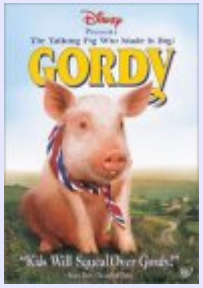
Email address: glarkins@justask.net

I have been known to respond to all of these variations of my name:

Gordon
Gordie
Gordo
G
G-Man
Gary – I work with Gary at ASK. Customers who have never met us face to face often get us confused.

I prefer the spelling of Gordie, over Gordy.

Gordie Howe



Gordy the Pig

ASK is an IBM Premier Business Partner who opened for business in 1993 as Application Specialist Kompany. ASK was created by some former IBM personnel who saw a need to support their AS/400 customers with their skill sets. In the beginning ASK was known by their customers as “that iSeries company”. ASK has now evolved from a Hardware concentric business into a Services based organization helping IT customers in many realms. Software, Networking, Managed Network Services, Help Desk, Storage Solutions, Disaster Recovery, Anti-Virus and Spam blocking and many others. With all of these service offerings, we do still sell hardware – and much more than just the iSeries. When your thinking IT, think ASK.

www.justask.net
877-ASK-4-ASK
877-275-4-275

CGIDEV2 is a tool set allowing programmers to write Common Gateway Interface (CGI) RPG programs through simple functions calls.

RPG programs execute CGIDEV2 function calls rather than directly invoke the HTTP API's directly handling the input/output from/to the remote browser. This makes your programs much easier to write/read/maintain/debug. The function calls to the service program are very efficient and fast.

HTML scripts are created and stored in either source physical file members or in IFS stream files. The HTML is used in the same manner as DDS. The HTML is used like a cross between a display file and a printer file. As the programmer you have much greater control on the presentation of the data. This technique is similar to that of an externally defined display and/or printer file, though HTML is can be easier than DDS to read and write. By using this technique, your programs are largely independent from the specific HTML presentation. You may change your HTML without changing your programs.

Mel Rothman, CEO of Mel Rothman, Inc., wrote these tools when he was an IBM employee in the IBM Custom Technology Center in Rochester, Minnesota.

Mel continues to enhance and support CGIDEV2 on a voluntary, as-available basis. Enhancements and fixes are contributed to IBM at no charge and are distributed by IBM.

IBM is CGIDEV2's owner and copyright holder.

Dr. Giovanni Perotti, now retired from IBM Italy, used CGIDEV2 to build the IBM Easy400 site, which contains CGIDEV2 and many additional tools and tutorials written by him.

All the materials on the Easy400 site are available at no charge.

CGIDEV2 is utilized by hundreds of companies and individuals around the world.

Disclaimer: This session on CGIDEV2 is not intended to teach HTML, Cascading Style Sheets (css), Javascript or Ajax. A full day course including some of these items is offered at Common. Parts of this demonstration were inspired by a book received from a course taught by Paul Touhy at the Autumn 2007 Common Conference.

Demonstration programs are shown at the end of this presentation. These programs use HTML to formulate the body of the web applications.

CSS is used to control font size, text color, back ground color, size control of different areas... much more.

Javascript is used for front end editing, cursor positioning, controlling what happens when buttons and/or areas are clicked... much more.

Ajax was not used in these demo applications. Ajax is a term referring to Asynchronous Javascript Processing. Ajax could be used to fill in areas. This process is very efficient when the programmer wants to only update a small area of the web page with data from the server.

CGIDEV2 in action

Today's demonstration includes green screen and web browser applications performing the following:

- Category Search and Select
- Category Maintenance
- Products Search and Select
- Products Maintenance

The Tables

Category Table

ASKMUMM.PARTCAT - Ask400i(Ask400)

Table | Columns | Key Constraints | Foreign Key Constraints | Check Constraints | Materialized Query | Partitioning

Column Name	Short Name	Data Type	Length	Text	Heading 1	Heading 2
CATEGORY	CATEGORY	CHARACTER	2	Category	Category	
DESC	DESC	CHARACTER	20	Category Description	Category Description	

Add... Remove Definition...

Products Table

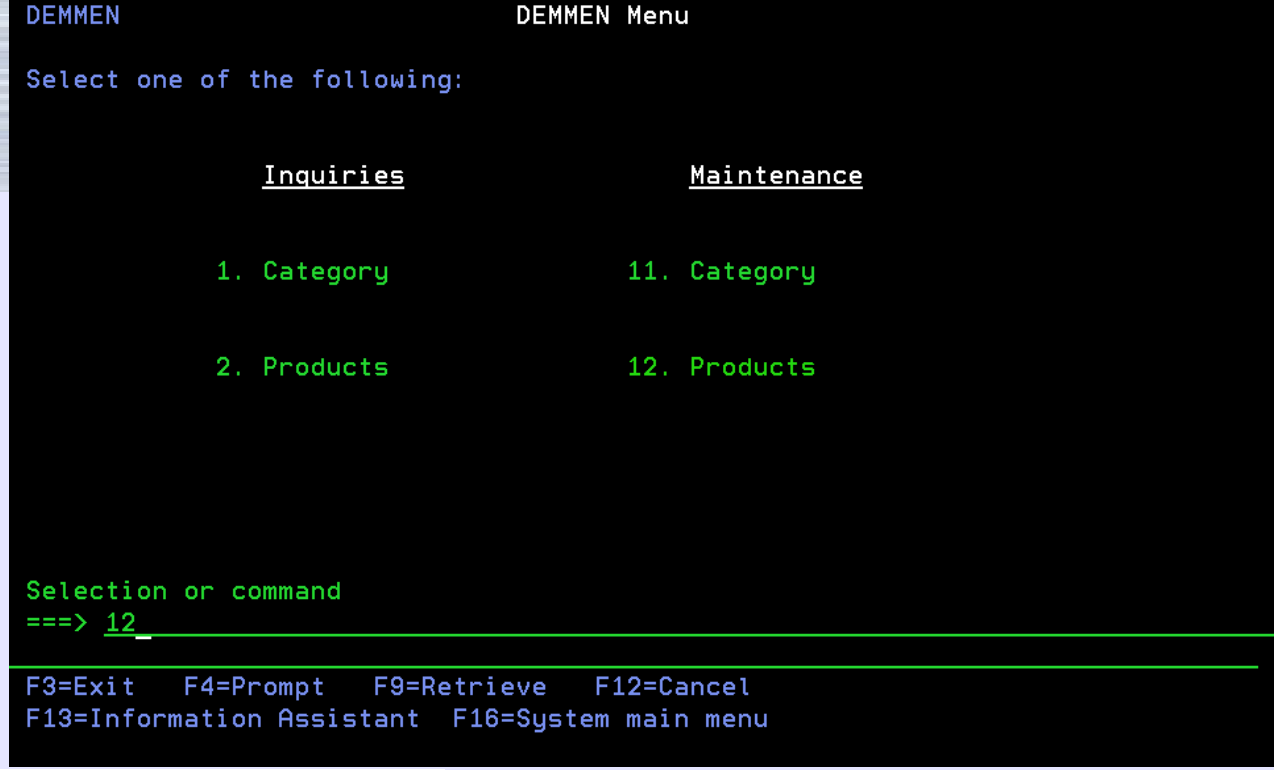
ASKMUMM.PRODUCTS - Ask400i(Ask400)

Table | Columns | Key Constraints | Foreign Key Constraints | Check Constraints | Materialized Query | Partitioning

Column Name	Short Name	Data Type	Length	Text	Heading 1	Heading 2
UNID	UNID	CHARACTER	10	Unique ID	Unique ID	
DESC	DESC	CHARACTER	20	Description	Description ...	
CAT	CAT	CHARACTER	2	Category	Category	
QTYONHAND	QTYONHAND	SMALLINT		Quantity on Hand	Quantity	On Hand ...
PACK	PACK	CHARACTER	2	Pack	Pack	
ITEMCOST	ITEMCOST	DECIMAL	9,2	Item Cost	Item Cost	
ITEMPRICE	ITEMPRICE	DECIMAL	9,2	Item Price	Item Price	
LASTMAINTDATE	LASTM00001	DATE		Date Last Maintained	Date Last	Maintained ..
LASTMAINTUSER	LASTM00002	CHARACTER	10	User Last maintained	User Last	Maintained ..

Add... Remove Definition... Move Up Move Down Browse...

This demonstration has similar components for Green Screen as well as web. The web applications are modeled after their green screen counterparts.



```
CATSSR      2/14/08      Category Search and Select      10:12:43  GLARKINS
CTLREC                        Maintenance Mode

                    Category     Description
                       01        Widgets
                       02        Spanners
                       03        Household Tools
                       27        Screwdrivers
                       30        Hammers and Widgets
                       35        Sandpaper
                       40        Other Implements
                       50        Category to go
                       60        Seasonal - Spring
                       61        Seasona - Summer
                       62        Seasonal - Autumn
                       63        Seasonal - Winter
                       71        Holiday - Valentines
                       72        Holiday - Easter
                                                              More...
          Place cursor on a category to change and press Enter.

                        F3=Exit    F5=Refresh    F6=Add
```

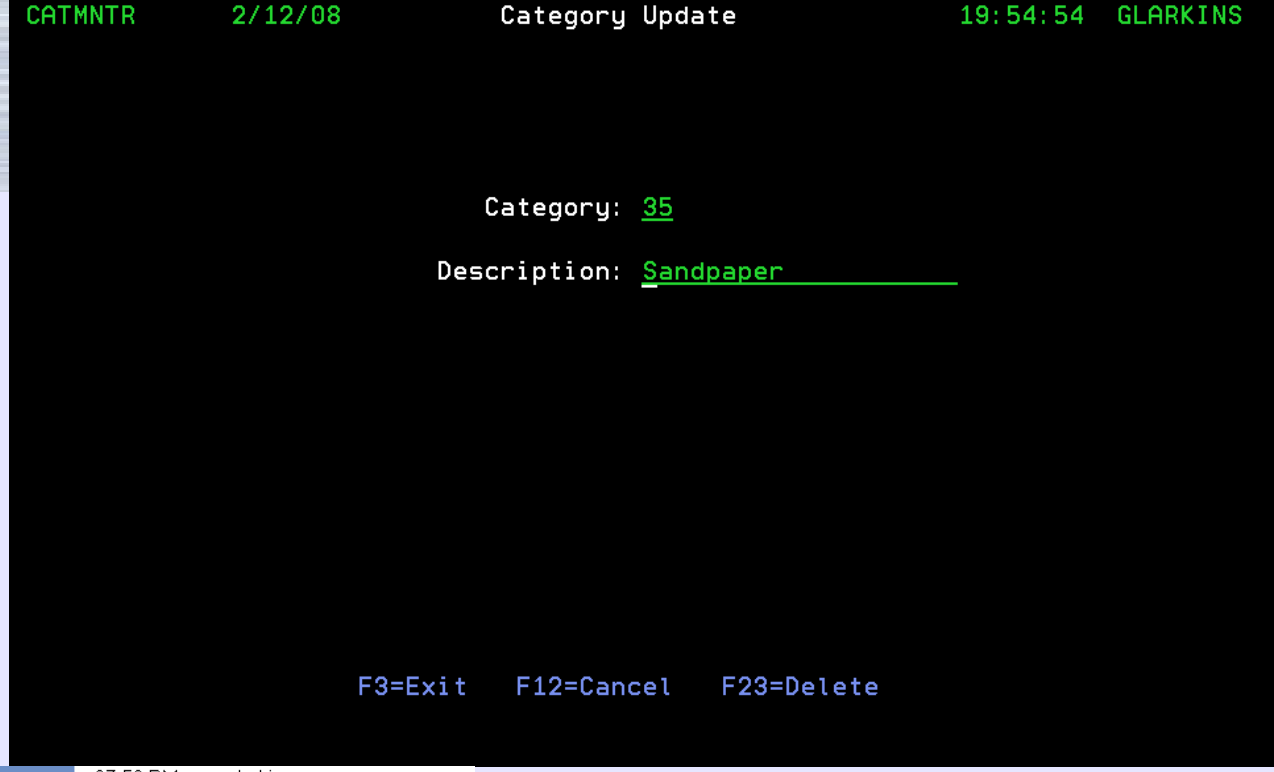
CATSSW 02/12/2008

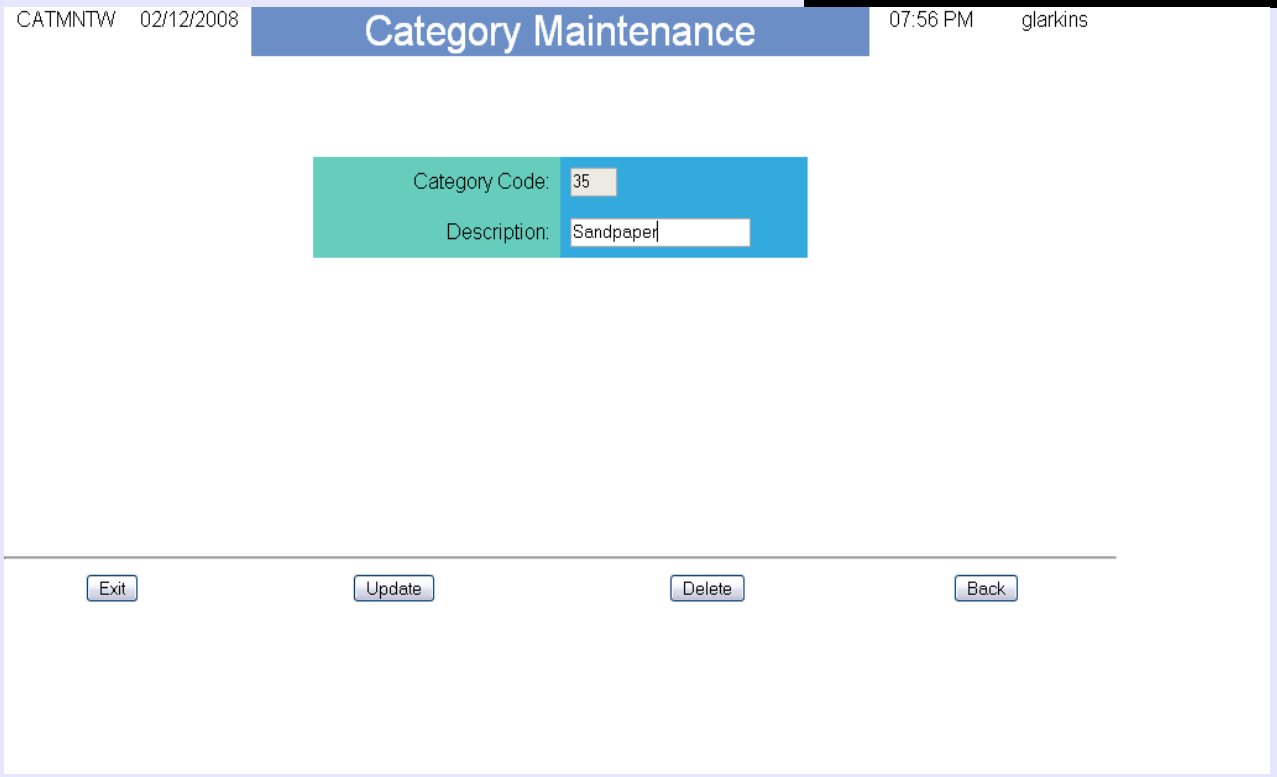
Category Search and Select
Maintenance Mode

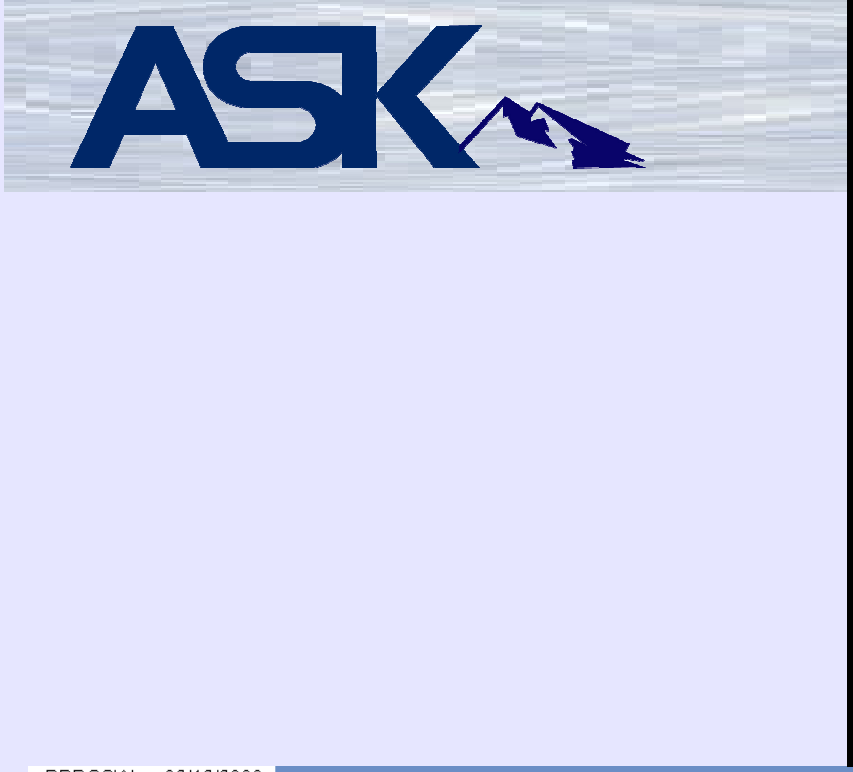
07:50 PM glarkins

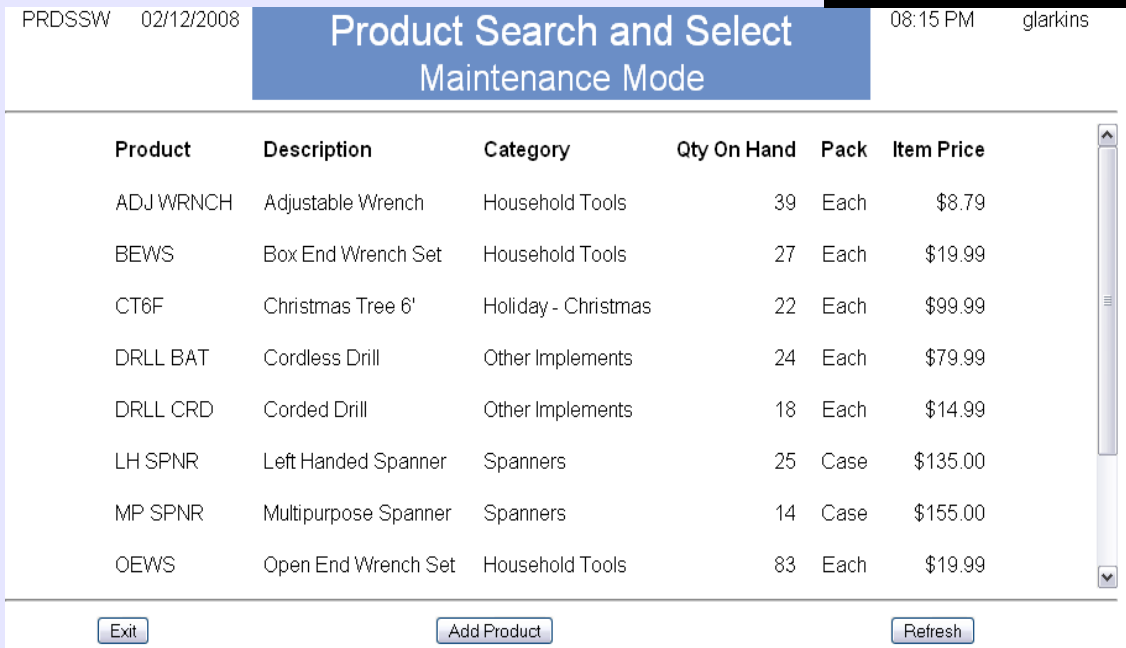
Category	Description
01	Widgets
02	Spanners
03	Household Tools
27	Screwdrivers
30	Hammers and Widgets
35	Sandpaper
40	Other Implements
50	Category to go
60	Seasonal - Spring
61	Seasona - Summer
62	Seasonal - Autumn

Exit | Add Category | Refresh

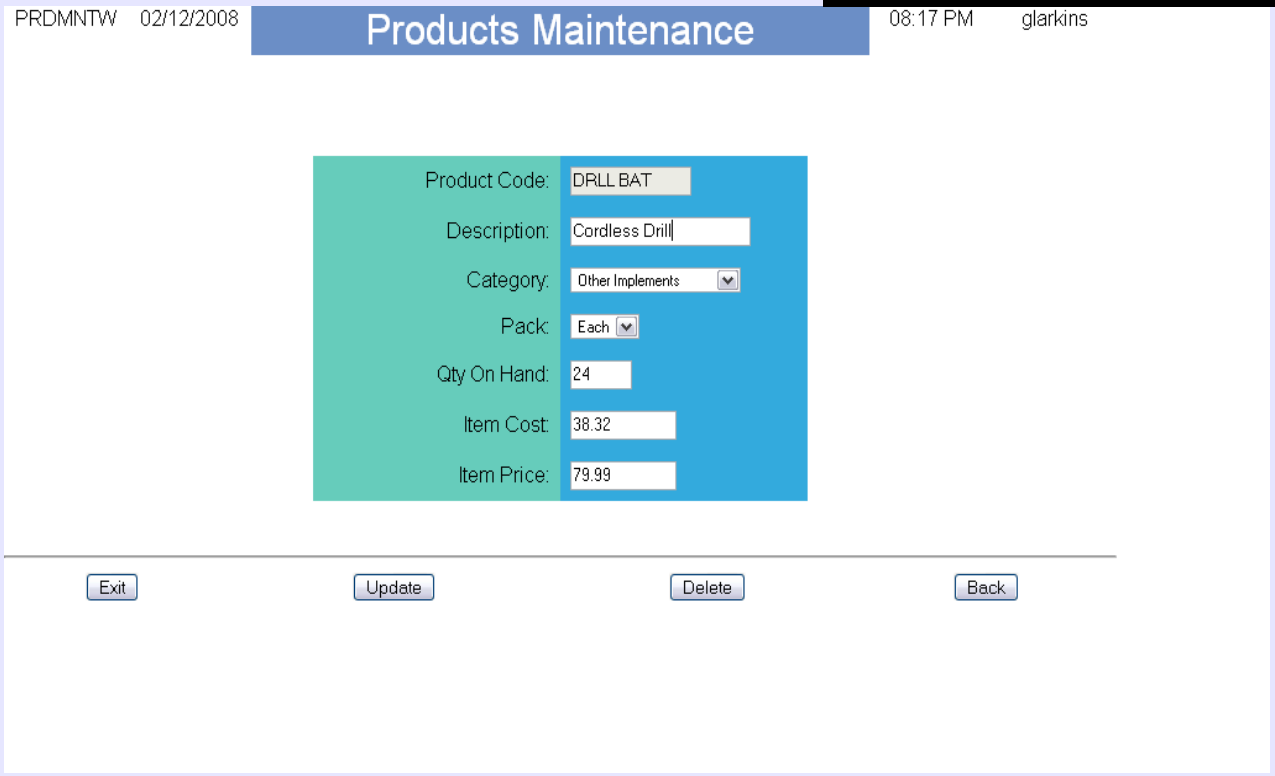

CATMNTR 2/12/08 Category Update 19:54:54 GLARKINS
Category: 35
Description: Sandpaper
F3=Exit F12=Cancel F23=Delete


CATMNTW 02/12/2008
Category Maintenance
07:56 PM glarkins
Category Code: 35
Description: Sandpaper
Exit
Update
Delete
Back


PRDSSR 2/14/08 Products Search and Select 09:52:23 GLARKINS
CTLREC Maintenance Mode
Unique ID Description Cat Pack Qty Item Cost Item Price
ADJ WRNCH Adjustable Wrench 03 EA 39 3.22 8.79
BEWS Box End Wrench Set 03 EA 27 8.74 19.99
CT6F Christmas Tree 6' 75 EA 22 39.43 99.99
DRLL BAT Cordless Drill 40 EA 24 38.32 79.99
DRLL CRD Corded Drill 40 EA 18 9.72 14.99
LH SPNR Left Handed Spanner 02 CS 25 95.00 135.00
MP SPNR Multipurpose Spanner 02 CS 14 125.00 155.00
OEWS Open End Wrench Set 03 EA 83 8.79 19.99
RH SPNR Right Handed Spanner 02 CS 34 95.00 135.00
SCSSRS Scissors 03 EA 83 1.74 3.29
SPADE Spade Shovel 03 EA 55 9.75 14.99
Bottom
Place cursor on a product to change and press Enter.
F3=Exit F5=Refresh F6=Add


PRDSSW 02/12/2008
Product Search and Select
Maintenance Mode
08:15 PM glarkins
Product Description Category Qty On Hand Pack Item Price
ADJ WRNCH Adjustable Wrench Household Tools 39 Each $8.79
BEWS Box End Wrench Set Household Tools 27 Each $19.99
CT6F Christmas Tree 6' Holiday - Christmas 22 Each $99.99
DRLL BAT Cordless Drill Other Implements 24 Each $79.99
DRLL CRD Corded Drill Other Implements 18 Each $14.99
LH SPNR Left Handed Spanner Spanners 25 Case $135.00
MP SPNR Multipurpose Spanner Spanners 14 Case $155.00
OEWS Open End Wrench Set Household Tools 83 Each $19.99
Exit
Add Product
Refresh


PRDMNTR 2/14/08 Products Update 09:58:31 GLARKINS
Product Code: DRLL BAT
Description: Cordless Drill
(F4) Category: 40 Other Implements
Pack: EA
Quantity on Hand: 24
Item Cost: 38.32
Item Price: 79.99
Last maintainted by and date: 0001-01-01
F3=Exit F12=Cancel F4=Prompt F23=Delete


PRDMNTW 02/12/2008
Products Maintenance
08:17 PM glarkins
Product Code: DRLL BAT
Description: Cordless Drill
Category: Other Implements
Pack: Each
Qty On Hand: 24
Item Cost: 38.32
Item Price: 79.99
Exit
Update
Delete
Back

This presentation contains sample code. The RPG sample code shown is in the style of /FREE.

This next slide contains most of the modules and functions included with CGIDEV2. Some of the older functions which are now discouraged were left out of this presentation. These older functions were very useful in CGIDEV and are still found in CGIDEV2 so older programs using them continue to work.

Modules and Functions

Module	Functions
XXXCGIPARS	ZhbGetInput, ZhbGetVarCnt, ZhbGetVar, ZhbGetVarUpper, ZhbGetVarPtr, ZhbCountAllVars, ZhbGetVarDetails
XXXCOOKIES	CrtCookie, GetCookieByName
XXXCOUNT	Countp
XXXDATA	Char2Hex, Hex2Char, C2N, C2N2, ChkNbr, xlatWCCSIDs, Uppify
XXXDEBUG	IsDebug, SetNoDebug, WrtDebug, WrtPSDS, WrtJobDbg
XXXDOMCD	DoCmd
XXXENVVARS	GetEnv, PutEnv, ContLen
XXXERRNO	Errno, Errnotxt
XXXHTMLMSG	AddMsg, CfgMsgs, ClrMsgs
XXXIFS	ChkIfsObj2
XXXUSRSPC	CrtUsrSpc, RtvUsrSpcPtr
XXXWRKHTML	ClrHtmlBuffer, CrtTagOpt, Encode, Encode2, EncodeBlanks, GetHtml, GetHtmlBytesBuffered, GetHtmlIfs, GetHtmlIfsMult, RtvHtmlRcd, RtvSubsVarInfo, UpdHtmlVar, WrtHtmlToStmf, WrtNoSection, WrtSection

Typical flow to send data to a web browser.

clrHTMLBuffer: clears any HTML output that has been buffered but has neither been sent to the browser nor written into a stream file.

It is a good idea to call clrHTMLBuffer in each CGI program before any uses of wrtSection or wrtNoSection. This practice prevents sending to the browser any buffered HTML from a previous, abnormally ended use of the program.

There are no parameters

RPG free example:

```
clrHTMLBuffer();
```

getHtmlIfsMult , **getHtmlIfs** , and **getHtml** read and process externally described HTML from multiple IFS stream files, a single IFS stream file, or a single source physical file member, respectively. The HTML records and information about any substitution variables are stored in dynamically allocated arrays inside the service program.

It is recommended that externally described HTML stream files be placed in directories not being served or cached by the HTTP server.

getHtmlIfsMult is newer, and can access one or more stream files. In addition, getHtmlIfsMult returns a data structure of indicators that tells you whether it was successful, and if not, why.

Storing externally described HTML in the IFS has the following advantages:

- There is no real limit to the length of a line
- By mapping the IFS to a PC's drive letter, you can easily use HTML editing and/or generating tools
- getHtmlIfsMult allows multiple IFS stream files to be read as if they are one. Therefore, common HTML can be stored in one or more files, and HTML for a particular file can be stored in its own file. At run time, all the files are read and processed as one.

Regardless of where you store the HTML, the Websphere Development Studio client (WDSc) is my recommended editing tool of choice.

getHtmlIfsMult is newer, and can access one or more stream files. In addition, getHtmlIFSMult returns a data structure containing an array of six indicators that can be checked to find out if any errors occurred. The indicators and their meanings are:

- no Errors: *on = no error occurred *off = one or more errors. Check other indicators.

- Name Too Long *on = one or more file's name exceeds 255 characters. File is ignored.

- Not Accessible *on = File or directory not found, authorization failure, etc. File is ignored.

- No Files Usable *on = All the files have been ignored.

- Dup Sections *on = One or more duplicate sections were found. Only the first occurrence is used.

- File Is Empty *on = File is empty and is ignored.

getHtmlIfsMult parameters

ifsFiles: The IfsFiles parameter contains the names of the files. A blank signifies the end of a file's name. The following limitations apply:

- Maximum length per file name: 255 bytes not counting the blank separator
- Maximum length of all the input file names, including any blank separators: 32767
- Maximum number of file names: 127

All of the files must use the same section name delimiters and substitution variable delimiters.

Section Delimiter Start:: (optional) used to override the default starting section delimiter of slash dollar sign /$

Section Delimiter End: (optional) used to override the default non-existent ending section delimiter of nothing

Variable Delimiter Start: (optional) used to override the default start variable delimiter of slash percent /%

getHtmlIfsMult parameters - (continued)

Variable Delimiter End: (optional) used to override the default end variable delimiter of percent slash %/

Example:

```
        errInds = getHtmlIfSMult('/www/'
                + %trim(psds.lib) + '/htdocs/'
                + %trim(psds.pgm) + '.html' : '<newsection>');
        updHtmlVar('title' : 'Sample Program');   // put the title on the page
        wrtSection('sectionName   *fini');  // send the section and send buffered data
...
```

HTML Example

```
<newSection>sectionName
<html>
<body>
<center><h3>/%title%/</h3></center>
<!--  The  /%title%/  will be replace with text from the RPG program  -->
...
```

updHtmlVar: Adds or updates variable names and values stored in dynamically allocated arrays in the service program. An optional parameter is used to initialize the arrays. These arrays are used to perform variable substitutions when HTML sections are written.

updHtmlVar2: Works like updHtmlVar but passes a pointer and a length to enable up to 16 MB of substitution data.

No data is received from these functions.

Parameters:

- variable name – name of the replacement variable in the HTML
- variable value – name of the variable in the program.
- action (optional)
  '1' = replace this variable in the arrays if it is already there. Otherwise add it to the arrays (default).
  '0' = clear arrays and write variable as the first element.
- trim instructions (optional)
  %trim - trim left and right (default)
  %triml - trim left only
  %trimr - trim right only
  %trim0 - don't trim

Example:

```
updHtmlVar('desc' : desc);
updHtmlVar2('paragragh' : reallyBigString);
```

crtTagOpt: Creates an drop down option tag.

Use this function to build selection options when the contents are not static, or the SELECTED option varies.

Do not use it for static selection boxes. Static, externally defined HTML is much more efficient.

No data is received from this function.

Parameters:

- String to be used for the value attribute
- String for the tag's associated text.
- Optional input containing the value of the option to have the SELECTED attribute. If this parameter matches the first parameter, the SELECTED attribute is output.

Example:

```
updHtmlVar('option' :
    crtTagOpt(%trim(userText) : %trim(tableCode) : %trim(lastSelectedCode));
```

encode: Sometimes it is necessary to display HTML tags as data, without having the browser process them as active HTML.

The encode function converts a varying length input field to a varying length return value, in which any occurrences of the special HTML characters " & < or > are converted to their corresponding HTML character entities, " & < and >.

encode2: The encode2 function, similar to encode, converts a varying length input field to a varying length return value, in which any occurrences of the special HTML characters are converted to their corresponding HTML character entities. encode2 retrieves the list of characters and related character entities from an IFS file. For more details, see the encode2 prototype in QRPGLESRC member PROTOTYPEB.

encodeBlanks: Sometimes it is necessary to display multiple consecutive blanks without having the browser convert them to a single blank.

The EncodeBlank function converts a varying length input field to a varying length return value, in which any occurrences of the blank character is converted to a non-breaking space, .

Example:

```
updHtmlVar('url':%trim(encode(urlValue)));
```

wrtSection: Writes one or more sections of HTML to a high performance, dynamically allocated buffer in the service program. The buffer is emptied and sent to the browser when the special section named *fini is written. Alternatively, the wrtHtmlToStmf function can be used instead of wrtSection(*fini) to empty the buffer's and write its contents into a user-designated stream file. wrtSection's nonewline parameter can be specified to force it not to append a newline character (x'15') to each HTML record.

No data is received from this function.

Parameters:

sections: One or more section names. If more than one, separate them with one or more blanks.

noNewLine: (optional) If not passed, wrtsection assumes *off
*off: wrtsection inserts (a newline character, x'15'), at the end of each html output line.
*on: causes each output html line to be written without a newline character being inserted. This is useful when binary data are being sent to the browser.

noDataString: (optional) What to do when a substitution variable is encountered and no value has been set up with UpdHtmlVar. If not passed, uses the default value **Missing Data**. Otherwise, uses the value passed.

Exemples:

```
wrtSection('sectionName');  // write one section
wrtSection('sectionName1 sectionName2 *fini'); // write 2 sections, send to browser.
wrtSection('*fini');  // no more sections to write, send buffered data to browser
```

wrtNoSection: Writes data, without using sections or performing variable substitution, to a high performance, dynamically allocated buffer in the service program. The buffer is emptied and sent to the browser when the special section named *fini is written. Alternatively, the wrtHtmlToStmf function can be used instead of WrtSection(*fini) to empty the buffer's and write its contents into a user-designated stream file.

This function would work great with a web services program, where the HMTL is returned from another source and you want to display the returned data in your program.

No data is returned with this function.

Parameters:

DataP is a pointer to the storage containing the data to be written.
DataLength is the number of bytes to be written, starting at that location.

Examples:

```
wrtNoSection(%addr(MyBuffer):MyBufferLength);
wrtNoSection(MyPointer:MyBufferLength);
```

wrtHtmlToStmf: Writes, then clears, the contents of the service program's high performance, dynamically allocated HTML buffer into a user-designated stream file.

To minimize the time during which an existing file would be unavailable, wrtHtmlToStmf

* Creates a temporary file and writes the HTML (or other data) into it
* Unlinks the target file if it exists
* Links the target name to the temporary file
* Unlinks the temporary file name.

If anyone is using the original file, the system defers the unlink activity until the last user finishes using it. Then, since the file has no links, the system deletes it.

The return value is the C errno if an error was detected. Otherwise, it is 0 .

Parameters:

stmf: path and name of the stream file to create or replace.
CodePage: (optional) the stream file code page.

Example:

```
errNo = wrtHtmltoStmf('/www/path/streamFile.txt' : 819);
```

chkIfsObj2: Checks IFS object's existence and optionally returns its type, size, and error information.

This functions name is chkIfsObj2 in order to distinguish it from Giovanni Perotti's IFSTOOL's ChkIfsObj function. No authority to the object is required to use this functions. *X authority is required for all subdirectories in the object's path.

If you don't care about the object's type or size or error details, all parameters except the first are optional.

This can be useful if you want to do an existence test prior to a wrtHtmlToStmf function.

Return parameter:

If the object is found and is accessible (good authority), ChkIfsObj2 returns a *on indicator. Otherwise, it returns *off.

chkIfsObj2: (continued)

Required Parameter

ObjPath: - Null terminated string of complete path to the object

Optional (*nopass) Parameters

ObjType If successful, contains the object's type.

ObjSize: If successful, contains the object's size in bytes. Otherwise, contains zero.

C_Errno: Contains C's errno value. If successful, contains zero. Otherwise, contains the C's errno value.

C_ErrText: The text associated with C's errno value. If successful, contains a zero length string. Otherwise, contains the C message text associated with C's errno value.

chkIfsObj2: (continued)

Examples:

if you only want to find out if the object is accessible

```
if chkIfsObj2('/home/joe/x.y');
```

if also want the object's type

```
if chkIfsObj2('/home/joe/x.y':objType);
```

if you also want the object's size

```
if chkIfsObj2('/home/joe/x.y':objType: objSize);
```

if you also want C's errno when a failure occurs

```
if chkIfsObj2('/home/joe/x.y':objType: objSize: C_Errno);
```

if you also want C's errno text when a failure occurs

```
if chkIfsObj2('/home/joe/x.y':objType: objSize: C_Errno: C_ErrText);
```

addMsg: adds a message's text and formatting level (1, 2, or 3) to the service program's arrays.

Return value:

0 No problems
1 Array full
2 Invalid level. Message is added with level = 1.

Inputs:

Message Text
Level (value 1, 2, or 3). Optional. Default is 1.

getMsgCnt: returns the number of messages currently stored in the arrays and can be used to condition calling wrtMsgs.

Return value

number of messages currently in the array.

clrMsgs: clears the messages stored in the arrays and sets their count to zero.

No parameters

cfgMsgs: Sets up section names, variable names for use by the wrtMsgs functions. All parameters are optional.

No return parameter.

Inputs:

Message Field Name: name of the externally described HTML's field to receive the message text. The default is msgtext.

MessageStart Section: name of the HTML section used to start the error message output. The default is msgstart.

Message End Section: name of the HTML section used to end the error message output. The default is msgend.

Message Level One Section: name of the HTML section used to output level 1 error messages. It should include the MsgFieldName substitution variable. The default is msgl1.

Message Level Two Section: name of the HTML section used to output level 2 error messages. It should include the MsgFieldName

rtvHtmlRcd: retrieves a record's contents from the externally described HTML.

Inputs are the section name (or *NONE) and a relative record number.

If a section name is specified, the relative record number is its position within the section.

If the section name is *NONE, the relative record number is its position among all the records, regardless of its section.

A return code indicates whether the record was found, and if not, why not.

"Why not" reasons include record not found (out of range), section not found, and record part of a duplicate, and therefore ignored, section.

rtvHtmlRcd can be used to analyze your HTML, to produce reports about it, and to implement various automated tools.

Returns

If not found (less than 1 or greater than number of records read), or part of a duplicate section: a null field, otherwise, the record's contents.

rtvHtmlRcd: (continued)

Outbound parameter:

Section name. If *NONE, gets record by relative record number regardless of section.

Relative record number (absolute or by section) (input)

Return code (output)

0 = record found and returned
-1 = section not found
-2 = record not found
-3 = record part of a duplicate section

rtvSubsVarInfo: retrieves information about substitution variables in your externally described HTML.

Inputs are the section name (or *NONE) and a sequence number.

If a section name is specified, the sequence number is the substitution variable's position within the section.

If the section name is *NONE, the sequence number is its position among all sections.

A return code indicates whether the record was found, and if not, why not.

"Why not" reasons include sequence number out of range and section not found.

An output data structure parameter is used to return the substitution variable's name, length of its name, its section's name, and the starting position in the HTML record where it was found.

rtvSubsVarInfo: (continued)

Returns

10-digit signed integer

- 0 = Substitution variable found and returned
- -1 = section not found
- -2 = Sequence number out of range for section or for list of all substitution variables

rtvSubsVarInfo: (continued)

Parameters

Section name (input). If *NONE, gets information by sequence number regardless of section.

Relative sequence number (absolute or by section) (input)

Data structure containing the following (output)

Section name
Character 50 varying
Null if return code not 0
Variable name
Character 30 varying
null if return code not 0
Variable's starting position in the html record
10 digit unsigned
0 if return code not 0
Length of variable's name (output)
10 digit unsigned
0 if return code not 0

getHtmlBytesBuffered: returns the number of bytes that have been buffered but have neither been sent to the browser nor written into a stream file.

This number is incremented each time output is written with wrtSection or wrtNoSection.

It is reset to 0 when either wrtSection('*fini') or wrtHtmlToStmf is run.

If this number is allowed to grow to more than 16 MB, the CGI program will fail.

Sampe:

```
bytesInBuffer = getHtmlBytesBuffered();
```

Externally described HTML function maximums

Description	Maximum
Source record length	For getHtmlIfs and getHtmlIfsMult: About 32765 bytes per line; for getHtml: 240 bytes (228 bytes for source data)
Total number of records. This is not a "per file" number. If getHtmlIfsMult is used, the sum of the records read from all files may not exceed this number.	32767
Number of unique substitution variables (each may appear multiple times in the source)	32767

Description	Maximum
Number of occurrences of substitution variables in the externally described HTML)	As many as will fit in 32767 HTML records as long as there are no more than 32767 unique substitution variable names.
Substitution variable name length	30 characters
Substitution variable value	1000 characters with updHtmlVar, updHtmlVar2 is 16 meg.
Substitution variables' delimiter's length	20 characters
Number of sections	1000
Section name length	50 characters
Section name delimiters' length	10 characters
Number of bytes of HTML that can be buffered before writing to the browser or a stream file	Approximately 2 terrabytes. This many bytes will take a very long time to process. It is recommended that you not try to test it.

Functions for creating and reading HTTP cookies

crtCookie: returns a cookie header in the form:
Set-Cookie: NAME=VALUE; expires=DATE; path=PATH; domain=DOMAIN_NAME; secure

See http://ww.netscape.com/newsref/std/cookie_spec.html for details abo how cookies work.

Parameters

Cookie name	(required input)
Cookie value	(required input)
Return code	(required output) 0 = no errors -1 blank cookie name -2 blank cookie value -4 invalid timestamp
Cookie domain	(required input, null or blank means none)
Cookie's path	(required input, null or blank means none)

crtCookie: (continued)

Whether secure	(optional input: *on=yes *off=no)
Cookie's expiration timestamp	(optional input) timestamp in format: YYYY-MM-DD-HH.MM.SS.MMMMM Use local date and time. CrtCookie converts to GMT. Note: the UTC offset is as of run time, not as the expiration date, so the expiration date/time could be off by the time between the UTC offset on standard versus daylight savings time.

Returns:

variable length character field, containing the generated Set-Cookie header. Blank if any errors occurred.

Example: mycookie = crtCookie('TEST':'The Data': RC:'ibm.com':'/':*off:
'2001-10-11-12.12.12.000000');

getCookieByName: retrieves a cookie's value given its name and occurrence.

Parameters

Cookie's name	(required input)
Cookie's occurrence	(optional input) 1 is assumed if not passed.

Returns:

value of the cookie having the name and occurrence. If not found, returns a null string.

Example: x = getCookieByName('TEST':1);

Environment Variable Functions

getEnv: returns the value of a user-specified environment variable. It uses the HTTP server's QtmhGetEnv API.

If an error occurs, returns blank and error information in qusec

Parameters

Environment variable name
qUsec

Examples:

```
domain = getenv('SERVER_NAME':qusec);
realUser = getenv('REMOTE_USER' : qusec);
```

For a list of environment variables, perform a google search on CGI Environment Variables.

putEnv: creates or changes an environment variable. It uses the HTTP server's QtmhPutEnv API.

If an error occurs, returns blank and error information in qusec

Parameters

Environment variable name
qUsec

contLen: Returns as a numeric variable, the CONTENT_LENGTH environment variable. Used by getinput. Should not be called directly.

Debugging Functions

The debugging functions use physical file CGIDEBUG, which has two fields: a time stamp and a 500 byte text field.

isDebug: returns a value to indicate whether debugging is on ('1') or off ('0').

setNoDebug: accepts an indicator variable. *on turns all debugging off; *off, allows debugging to operate normally, including the honoring of forced debugging output requests.

Normally, CGIDEV2 programs spend considerable time in the debugging routines. Running function setNoDebug *on can significantly reduce this overhead and improve performance at the cost of losing all debugging output.

Note that setNoDebug sets a global variable in the service program. If multiple CGI programs are running in the same named activation group, all those programs are affected in the same way by the most recent execution of setNoDebug.

wrtDebug: writes into the debugging physical file, CGIDEBUG, the text passed to it as a parameter. wrtDebug is used by several of the service program's functions. You can use it, as desired. No output is generated unless debugging output has been turned on by the CGIDEBUG *ON command, or the optional parameter, force, has been set to *on.

wrtPsds: receives the program status data area and unconditionally writes it in a formatted manner into the debugging file.

wrtJobDbg: writes the qualified job name, into the debugging file.

General Purpose Counter Function

countp: This function keeps track of counts (e.g. page hits) using a keyed physical file, CGICOUNT. A key (page name, program name, etc.) is passed to countp. It adds one to the existing count for that key and returns the updated counter. If the key doesn't exist, a record with its count initialized to 1 is added to the file and 1 is returned to the caller.

Example:

```
countp('myPage' : %trim(psds.pgm));
```

Data Handling Functions

char2Hex converts a character string to its hexadecimal representation.

Example: hexValue = char2Hex('ABC'); // hevValue will contain C1C2C3

hex2Char converts a character string in hexadecimal format to its character representation

Example: charValue = hex2Char('C1C2C3'); // charValue will contain ABC

chkNbr: accepts a character string and an optional parameter specifying the maximum number of digits to the left of the decimal point.

Additional optional parameters are available to request that any errors found should be formatted as messages and added to the service program's message arrays, the text that should be used to describe the field in the messages, and whether a message should be sent if the field's value is less than zero.

chkNbr: (continued) returns a structure containing seven indicators. The indicators and their meaning when *on are:

1. One or more errors occurred. This indicator is set on only if one or more of indicators 2 through 6 is set on. It is not set on when only indicator 7 (value is negative) is on.

2. Non-numeric characters (includes minus sign in wrong place)

3. Multiple decimal points

4. Multiple minus signs (both leading and trailing)

5. Zero length input or no numeric characters

6. Any of the following:
   * More than 21 digits to the left of the decimal point
   * More than 9 digits to the right of the decimal point
   * More digits to the left of the decimal point than specified in the maximum number of digits parameter

7. Value is less than 0. This indicator is used only to cause a message to be sent. It does not cause indicator 1 to be set on.

chkNbr: (continued)

Parameters:

Required
character field to test

Optional
maximum number of digits
add messages
field description
negative is error

Example:

```
myString = zhbGetVar('itemPrice');  // get item price from web page
chkNbrDS = chkNbr(myString);  // verify proper numeric value
// check indicators in chkNbrDS and process accordingly
```

c2n (old, not recommended for new development – replaced by c2n2): Converts a character string to a floating point variable. If non-zero, adds a small fuzz to the result in an attempt to ensure that subsequent rounding works as expected.

Input: variable length character field containing a valid decimal number in display format.

Output: floating point number which can then be converted to some other form either by assignment or via %DEC, %DECH, %INT, or %INTH built-in functions.

The only characters included in the conversion are digits, the minus sign, and the current RPG decimal point character. If multiple decimal points are found, only the first one is used.

Example: float = c2n('-123.34');

c2n2: Enhanced c2n. Converts a character string to a 30,9 packed variable.

Adapted from getnum function, written by Barbara Morris, IBM Toronto laboratory.

This function avoids precision problems with large floating point numbers by doing virtually all its work with characters. Performance is improved too.

Input: variable length character field containing a valid decimal number in display format.

Output: 30p 9 number

The only characters included in the conversion are digits, the minus sign, and the current RPG decimal point character. If multiple decimal points are found, only the first one is used.

Example: number = c2n2('-123.34');

uppify: Converts lowercase characters to uppercase

Examples:

charstring = uppify(charstring);
uppify uses XLATE on English language characters a through z only.
charstring = uppify(charstring:0)

uppify uses SysConvertCase API with the characters in the job's CCSID

charstring = uppify(charstring:n)

Uppify uses SysConvertCase API with the characters in CCSID n where n is a valid CCSID number.

If the optional parameter, CCSID, is not passed, the RPG XLATE operation code is used with standard English language characters.

CL command Function

doCmd: doCmd uses QCMDEXC to execute a user-specified CL command.

Used for such things as OVRDBF, ADDDLIBLE, CHGCURLIB. It is very nice having this available if you want to do an override database file to a different library or use a different member in a mult-member database.

Returns 0 if executed without error; otherwise 1.

Example:

```
// salesFile is defined as usrOpn in the format specification
// mbrVar = 'M200607', sales data for year 2006, year 07
cx  = docmd('ovrdbf file(salesFile) mbr(' + %trim(mbrVar) + ')');
if  cx = 1;
      exsr badOverride;
else;
      open salesFile;
// yada, yada, yada
```

User Space Functions

crtUsrSpc: creates a new user space:

The user space is created in the library specified by the caller.

Ether defaults or user specified values are used for public authority, text, initial size, and extended attribute

the user space's randomly generated name is returned as the functions value

a pointer to the user space is returned in a parameter,

a 7 character message ID is returned as a parameter. It contains blanks if no errors occurred; otherwise it contains a message ID.

crtUsrSpc: (continued)

Required Parameters
- User space library (input)
  - If the library not found, CrtUsrSpc sets the user space name to blanks and MsgId to CPF9810
  - If the requestor does not have change authority to the lib CrtUsrSpc sets the user space name to blanks and MsgId to
- Pointer to user space (output)
  - Set to null if the user space is not created
- Message ID (output)
  - blank if no errors
  - else, message id of error

Optional Parameters
- Public authority (input)
  - If not passed, it is set to *EXCLUDE
- Text (input)
  - If not passed, it is set to 'Created by CGIDEV2' plus time
- Initial size (input)
  - If not passed, it is set to 12288
- Extended attribute
  - If not passed, it is set to blanks

rtvUsrSpcPtr: retrieves a pointer to an existing user space.

Parameters
- User space name (input)
- User space library (input)

Returns
- If successful, pointer to the user space
- Otherwise, Null pointer

Getting and parsing browser input

zhbGetInput: this procedure returns a count of the number of variables received. Note: a variable with multiple instances counts as one variable.

zhbGetInput uses the server's QzhbCgiParse, QtmhGetEnv, and QtmhPutEnv APIs.

It stores the browser's input in dynamically allocated structures in the service program, enabling subsequent high speed retrieval by the CGIDEV2 functions.

This must be the first function called before any other INPUT functions.

zhbGetInput can handle up to 32767 input variable instances containing large quantities of data (it's internal buffer is dynamic and can grow up to 16 MB). Each variable can be up to 64000 bytes in length. For variables expected to be between 37268 and 64000 bytes in length, zhbGetVarPtr must be used. Otherwise, zhbGetVar, zhbGetVarUpper, or zhbGetVarPtr can be used.

It is recommended that you not write CGI programs that will transmit anywhere near the 16 MB limit! The response time will be awful.

zhbGetInput returns the number of unique variable names found (variables with multiple occurrences count as one).

Examine the QUSEC data structure for any errors.

Example:

```
nbrVars = zhbGetInput(savedquerystring:qusec);
```

zhbGetVarCnt: returns number of occurrences of a variable in the CGI input.

This is useful when the web page has multiple occurrences of the same variable name. This often happens with Selection fields that allow multiples to be chosen. It could also be used with Check Boxes.

Parameter:

nameOfVariable

Example:

```
ox = zhbGetVarCnt('checkBox');
for ix = 1 to ox;
      cbArray(ix) = zhbGetVar('checkBox' : ix);
endfor;
```

zhbGetVar: Returns the value of a CGI input variable.
If occurrence is omitted, the first occurrence is returned.
If the occurrence does not exist, a null string is returned.

Parameters:

name of variable
occurrence of variable (optional)

Example:

```
addrStreet = zhbGetVar('street');
```

zhbGetVarUpper: Returns the value of a CGI input variable with lower case characters converted to upper case.

If occurrence is omitted, the first occurrence is returned.

If the occurrence does not exist, a null string is returned.

Parameters:

name of variable
occurrence of variable (optional)

Example:

```
prdCode = zhbGetVarUpper(productCode);
```

zhbGetVarPtr: Provides a way to read browser inputs that exceed zhbGetVar's 32767 size limit.

Returns a pointer to the variable's data
If not found or length is 0, returns *null

Parameters
- Variable name, input
- occurrence, input
- length, output

Use the returned pointer with a based variable to read the data.

Example:

```
varPointer  = zhbGetVarPtr(lotsOfData : varLenthOut);
```

zhbCountAllVars: Returns number of occurrences of all variables in the CGI input. zhbGetInput must have been run before calling this function.

No parameters

Example:

```
varCount  = zhbCountAllVars;
```

zhbGetVarDetails: Returns detailed information for the nth variable of all variables counted by zhbCountAllVars

zhbGetInput must have been run before calling this function. If variables are requested in ordinal sequence, they will be returned in name, name's occurrence sequence. The names will be in upper case.

Parameters

ThisOccur	Input	Ordinal value of variable to return
ThisVarName	Output	The variable's name
ThisVarOccur	Output	Occurrence within this variable
FoundInd	Output	*on = found; *off = not found

Return value: the variable's value

No example.

VIP - Very Important Physical files and commands

CGIDEV2 includes two physical files:

CGIDEBUG – Will contain data to help you debug
CGICOUNT – Help keep track of how often pages are being accessed.

Command CGIDEBUG has one parameter:

- *ON – Turn debugging on.
- *OFF – Turn debugging off.
- *DSPDATA - Show the data
- *CLRDATA – Delete records from the file.

Performing a CGIDEBUG *CLRDATA tends to take a lot of time, especially if you have a lot of records. The records are deleted, the file is not cleared. A faster way is to end the HTTP server to remove any locks over file CGIDEBUG, issue a CLRPF command, restart the HTTP server.

What to Do Next?

Install CGIDEV2

Learn some HTML

Learn some javascript

Learn about Cascading Style Sheets

HTML, javascript, Cascading Style Sheets all have numerous books on them in the book stores. Web search engines such as Google are also extremely useful. Say you want to know more about an html input tag. Type in "HTML input tag" in google search and a wealth of knowledge will be made available to you.

References

Download CGIDEV2

http://www-03.ibm.com/systems/services/labservices/library/

The Easy400 site

More than just CGIDEV2. There are other open source offerings available here. This site is run by Giovanni B. Perotti. Giovanni and Mel Rothman are former IBM personnel who we have to thank for this terrific tool set.

www.easy400.net

Yahoo Groups has a group called Easy400. This group is dedicated to and very supportive of CGIDEV2.