Building RESTful web applications

Building a web site in Multivalue or Pick.

I have been attempting to build a business web site for more than 15 years, experimenting with various technologies that have been available within the multivalue marketplace.

I saw glowing reports on how this or that technology is fantastic and yet I did not feel comfortable.

My goal was to build a web site without using locked in proprietary solutions. Proprietary things tend to come and go and it makes your whole business proposition dependent on a third party. Exactly why Apple doesn't support flash in IOS. I want to do what everyone else in the world is doing and use the language and framework that they are using without any lock-in.

And, in 2010, I read about HTML5 and REST and, wow, my goal is achievable.

Representational state transfer (REST) is a style of software architecture for distributed hypermedia systems such as the World Wide Web. The thing about REST is that it is not tied to any particular technology or platform. It cannot be taken away if a supplier decides to drop it. The term is often used in a looser sense to describe any simple interface which transmits domain-specific data over HTTP without an additional messaging layer such as SOAP or session tracking via HTTP cookies. A REST web service is a simple web service implemented using HTTP and the principles of REST. A web service supports JSON which is an easy option for multivalue because it is just text.

You can do everything that others do with PHP. All of a sudden, your 40 year old database friend is up there in the bleeding edge of computing again. In truth, HTML5 is not required, but the things that you can do with HTML5 are so 'interesting' that you would not think of doing this in a lesser specification.

Secondly, the User Interface (UI) or User eXperience (UX) is completely abstracted from the database and any database related stuff. The database and business logic stay on the server and the UI is on the client. I find this is a hard to grasp because, in MV-Land, database and UI are inextricably linked.

The examples I show here are, strictly speaking, not REST but they show the ease with which you can adopt a REST technology. They are derived from php+mysql examples and I show you how to replace that with basic+mv and plug this into openqm.

The package of components described will build a webserver, install an openqm account and install jquery library {http://www.jeasyui.com/} so that you can do web stuff.

Basic Components

My environment consists of a virtual machine with 1GB memory and 10GB of disk. A 32 bit ubuntu11.10 and qm_2-12-8. I have no MASTER.LOGIN or LOGIN paragraphs which is important. I use 'joe' as my unix editor and most of the 'editable' bits I have created as directories for this reason. It also means that I can easily copy from a unix directory. Within the code package are some utilities to facilitate some database stuff. They are primitive but serve the purpose for the sandbox.

Once I have openqm up and running, I do everything from the openqm prompt. Notice that !cmd executes a shell command.

This is a development account so I set permissions as everything (777). You will probably fall over permissions at least once.

The logfiles you need are at /etc/apache2/log, webdev/logfile and /usr/qmsys/errlog.

After you have installed your server, you will need some additional bits for your Linux server

```
sudo apt-get install openssh apache2 joe unzip
```

Not necessary, but you may like

```
Sudo apt-get install subversion gnome-terminal mc
```

Head off to http://www.openqm.com and download openqm for Linux, get a 30 day licence and install.

Create a data account for the website

```
sudo bash
mkdir -p /usr/qmaccounts/webdev
chmod -R 777 /usr/qmaccounts/webdev
cd /usr/qmaccounts/webdev
qm
```

You may need to register the account. In qm_2-12-8 it is no longer necessary.

```
ED QM.ACCOUNTS WEBDEV
001 /usr/qmaccounts/webdev
```

Create a user login point

```
ED VOC PSTART
001 PA
002 PTERM CASE NOINVERT
003 DISPLAY Welcome to WEBDEV central
```

More housekeeping

```
!mkdir datastore
!chmod -R 777 /usr/qmaccounts/webdev
OFF
```

Login again

```
cd ~
qm -awebdev -quiet PSTART
```

At this point, we have an account and the database is up and running

Create some tables

```
CREATE-FILE WEB.BP DIRECTORY
CREATE-FILE WEBPAGES DIRECTORY
CREATE-FILE CSS DIRECTORY
```

Install some code in WEB.BP

```
!wget http://newdevvm/code/WEBBP.tar
!tar xvf WEBBP.tar
basic WEB.BP *
catalog WEB.BP *
```

Create some helper VOC entries - WS as a shortcut to display the logfile

```
ED VOC WS
0001: PA
0002: sh tail -150 logfile/webpages
```

Create ON.ABORT to log the error

```
ED VOC ON.ABORT
0001: PA
0002: WEB.ABORT
```

Create a default web page

```
!joe WEBPAGES/index.htm
<html><body>
<h1>QM and web - It works!</h1>
<p>This is the default web page for this server for RESTful web applications</p>
</body>
</html>
```

Test the result – this is not using the web but determines if you have any errors.

```
!qm -awebdev -quiet webapp
Content-Type: text/html

Location:
http:///qmweb/webap
p/index.htm
```

```
(webdev)>ws
clean
0317T16:40_10---start--
0317T16:40_10-SHELL=/bin/bash
0317T16:40_10-TERM=xterm
0317T16:40_10-XDG_SESSION_COOKIE=267fca6b43c7f19fc6c4e04900000017-1332002102.528671-1324954732
0317T16:40_10-SSH_CLIENT=192.168.1.134 54074 22
0317T16:40_10-SSH_TTY=/dev/pts/2
0317T16:40_10-USER=sysadmin
0317T16:40_10-LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:...
0317T16:40_10-MAIL=/var/mail/sysadmin
0317T16:40_10-_=/usr/local/bin/qm
0317T16:40_10-LANG=en_US.UTF-8
0317T16:40_10-HOME=/home/sysadmin
0317T16:40_10-SHLVL=2
0317T16:40_10-LOGNAME=sysadmin
0317T16:40_10-SSH_CONNECTION=192.168.1.134 54074 192.168.1.139 22
0317T16:40_10-LESSOPEN=| /usr/bin/lesspipe %s
0317T16:40_10-DISPLAY=localhost:12.0
0317T16:40_10-LESSCLOSE=/usr/bin/lesspipe %s %s
0317T16:40_10---loaded--
```

Clear the log

```
!echo "cleared" > logfile/webpages
```

Go superuser and create an apache site for our website – all of this from inside openqm. This is not necessary and you may choose to do this from the unix shell.

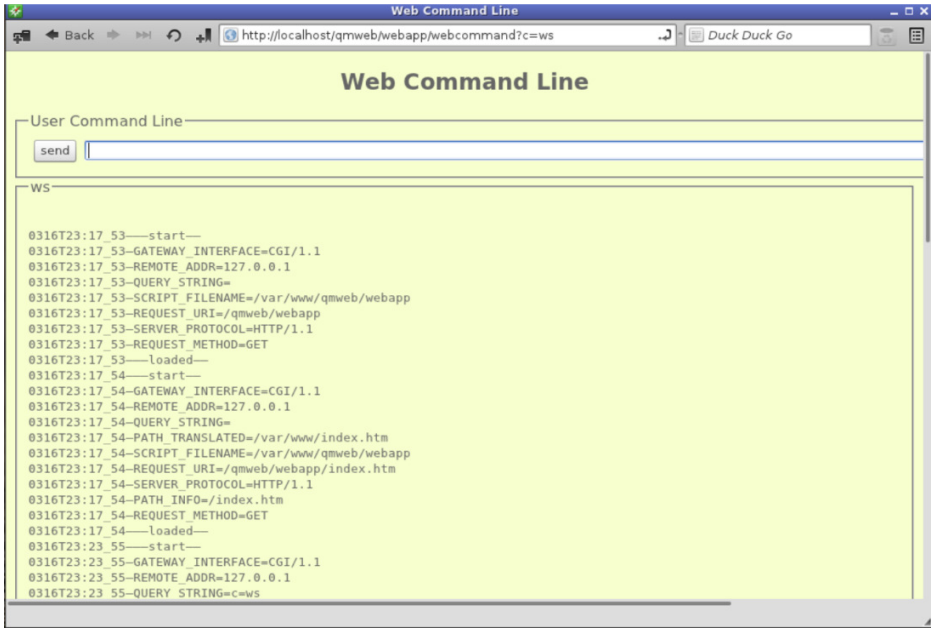
```
sudo qm -quiet PSTART
!mkdir /var/www/qmweb

!joe /var/www/qmweb/webapp
   #!/bin/sh
   cat - | qm -aWEBDEV -quiet WEBAPP

!joe /etc/apache2/sites-available/qmweb
   <Location /qmweb>

   SetHandler cgi-script
   Options +ExecCGI

   </Location>

!a2ensite qmweb
!chown -R www-data:www-data /var/www/qmweb
!chmod 744 /var/www/qmweb/webapp
!service apache2 reload
!chmod 666 logfile/webpages
off
```

Test your new website.

http://localhost/qmweb/webapp

Now go and check the logfile.

http://localhost/qmweb/webapp/webcommand?c=ws



It doesn't work for me

Check that you have followed each stage as quotes. Check the logfiles and check the permissions on all of the files accessed because that is the most likely cause of failure.

Benchmarking

```
!ab -n 100 -c 2 http://localhost/qmweb/webapp/webcommand?c=listu
Document Path:          /qmweb/webapp/webcommand?c=listu
Document Length:        897 bytes
Concurrency Level:      2
Time taken for tests:   11.618 seconds
Complete requests:      100

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.7      0       7
Processing:    93  230 250.8    127    1192
Waiting:       82  220 251.9    116    1192
Total:         93  230 250.8    127    1192
```

Things to note

For benchmarking, the WEB4LOG loses it's usefulness. The higher the concurrency, the more likely that the WEB4LOG process will fail to write all of the entries. This is expected as WEB4LOG was written to provide information for development not performance testing.

WEBCOMMAND is really dangerous. You can do anything on your server from the web application as long as qm recognises the command. Obviously, it will hang if the qm process is requesting keyboard input. In the code provided, Ubuntu shell commands are disabled. You can live dangerously and uncomment that bit of code.

Development with the jquery library and using jeasyui

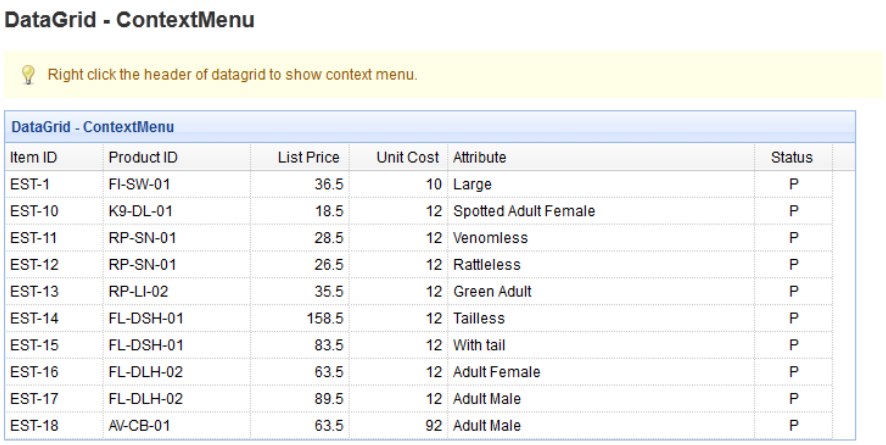
Get jeasyui

```
!wget http://www.jeasyui.com/download/downloads/jquery-easyui-1.2.5.zip
!mkdir jquery
!cd jquery;unzip ../jquery-easyui-1.2.5.zip
!sudo mv jquery /var/www
!sudo chown -R www-data:www-data /var/www/jquery
!cp /var/www/jquery/demo/* WEBPAGES
COPY FROM WEBPAGES TO CSS demo.css
FIX.WEBPAGES
```

The demo pages are now in our file WEBPAGES. Routine FIX.WEBPAGES removes the extraneous windows characters and fixes the path to the jquery library.

Example 1 – populating a datagrid

In your browser, dial up http://localhost/qmweb/webapp/datagrid3.html



If you examine the log, you will see that there are 3 calls to webapp. The first loads the webpage datagrid3.html, the second loads demo.css and the third, as shown in figure XX loads datagrid_data2.json. This is a text file with the required data in json format.

This is not so clever you may say. Create a program that outputs the data in the required format, replace the datagrid_data2.json name with your routine name in the html page and you are now talking to your database.

```
0318T09:49_272---start--
0318T09:49_272-GATEWAY_INTERFACE=CGI/1.1
0318T09:49_272-HTTP_X_REQUESTED_WITH=XMLHttpRequest
0318T09:49_272-REMOTE_ADDR=192.168.1.134
0318T09:49_272-QUERY_STRING=
0318T09:49_272-CONTENT_LENGTH=0
0318T09:49_272-PATH_TRANSLATED=/var/www/datagrid_data2.json
0318T09:49_272-SCRIPT_FILENAME=/var/www/qmweb/webapp
0318T09:49_272-HTTP_CACHE_CONTROL=no-cache
0318T09:49_272-REQUEST_URI=/qmweb/webapp/datagrid_data2.json
0318T09:49_272-HTTP_REFERER=http://192.168.1.139/qmweb/webapp/datagrid3.html
0318T09:49_272-SERVER_PROTOCOL=HTTP/1.1
0318T09:49_272-PATH_INFO=/datagrid_data2.json
0318T09:49_272-REQUEST_METHOD=POST
0318T09:49_272---loaded--
```

Create a database table.

```
CREATE.SCHEMA DATAGRID3(ITEMID VARCHAR(10),UNIT.COST FLOAT(10.2),STATUS VARCHAR(1),LIST.PRICE
FLOAT(10.2),ATTR1 VARCHAR(35),PRODUCT.ID VARCHAR(10))
```

Add the example data

```
JSON.IMPORT DATAGRID3 WEBPAGES/datagrid_data2.json
```

Create and compile a routine to deliver the data

```
0001: PROGRAM DATAGRID3.PGM
0002: * get json data for datagrid3.html
0003:
0004:
0005: $INSERT WEB.BP I_WEBCOMMON
0006:
0007:      COMMAND$LINE = "SORT DATAGRID3 PRODUCT.ID UNIT.COST STATUS LIST.PRICE ATTR1 ITEMID"
0008:      CALL WEB4LOG ('Loading data as json')
0009:      CALL JSON.FORMAT
0010:      PRINT "Content-Type: text/html"
0011:      PRINT
0012:      PRINT COMI
0013:
0014: END
```

Change WEBPAGES/datagrid3.htm to point to our routine

```
014 //    url: 'datagrid_data2.json',
015     url: 'datagrid3.pgm',
```

Dial up your web page http://localhost/qmweb/webapp/datagrid3.html. Check the log to see that we called our routine.

Example 2 – create and update data (CRUD)

This is a different example from jeasyui. It uses php routines to determine the method. We can replace the php and mysql with our own stuff.

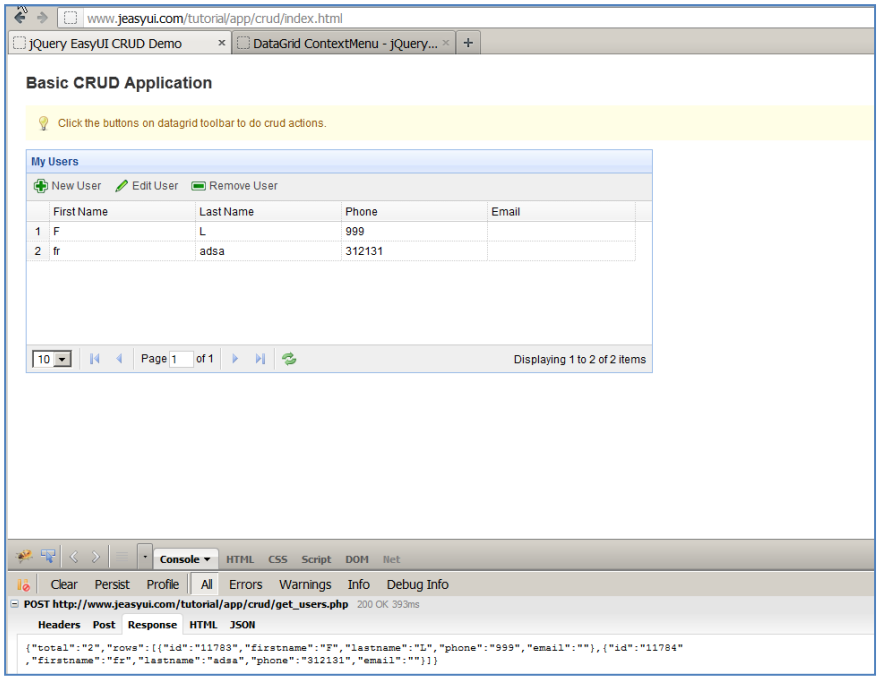
```
!wget http://www.jeasyui.com/tutorial/app/crud/downloads/easyui-crud-demo.
!mkdir cruddemo
!cd cruddemo;unzip ../easyui-crud-demo.zip
!cp cruddemo/index.html WEBPAGES/cruddemo.html
FIX.WEBPAGES cruddemo.html
```

We need to do a bit of investigating to see what we want.

```
(webdev)>!ls cruddemo
conn.php  get_users.php  index.html  remove_user.php  save_user.php  update_user.php  users.sql
```

When you study the cruddemo directory, you will see that there are 4 php routines. We will replace those with basic.

First lets find out what ‘get.users.php’ does. Go to the site using firefox with firebug turned on. In the console you can see the json that has been sent.



This says that we need a table with fields id, firstname, lastname, phone and email. Equally, you could read the php and work it out.

```
CREATE.SCHEMA WEBUSERS(ID INTEGER(10),FIRST.NAME VARCHAR(35),LAST.NAME VARCHAR(35),PHONE
VARCHAR(20),EMAIL VARCHAR(35))
```

A routine named get_users.php which looks remarkably like datagrid3.pgm with some more friendly exits.

```
001 PROGRAM GET_USERS.PHP
002 * get json data for CRUD
003
004
005 $INSERT I_WEBCOMMON
006
007       COMMAND$LINE = "SORT WEBUSERS ID FIRST.NAME LAST.NAME PHONE EMAIL"
008       CALL WEB4LOG ('Loading data as json')
009       CALL JSON.FORMAT
010       IF FATAL$ERROR THEN
011            CALL WEB4LOG(ERROR$VALUE)
012            COMI = '{"msg":"some errors occurred"}'
013       END
014       PRINT "Content-Type: text/html"
015       PRINT
016       PRINT COMI
017
018 END
```

We also need routines named REMOVE_USER.PHP, SAVE_USER.PHP, UPDATE_USER.PHP.