NetRexx Server Pages
24th International Rexx Language Symposium
Raleigh/Durham, NC
René Vincent Jansen, May 8th, 2013
What is it

- Java2EE, also called enterprise Java
- First there was Jeeves, servlets, JSP, JSF
- Consists of Application Servers, and Web Containers that are part of those
- Pages are code compiled on the fly to servlets
The goal

- The inspiration is not unlike Rails (for Ruby)
- To program an active website in NetRexx with as less setup as possible
- No setup would be ideal
- Use standards - we are not bound to one product
An implementation choice

- We chose a web container called Jetty for this purpose
- It is light, can be embedded and does not need a lot of configuration
- We can develop 'in place' and see results

dinsdag 7 mei 13
The new NetRexx.org site

- This site has been started fresh after the first version was made with a proprietary html5 product, that unfortunately never was too fast on one of the most ridiculed popular browsers

- Not using NetRexx for the NetRexx site was not compatible with our sense of justice

- It is slowly getting its form now; still much ideas unimplemented
Active parts of the site (currently)

- 'the Hursley time' using qtime, one of the first Rexx (and also NetRexx) programs
- the download and automatic build page
- the examples page, these are straight out of the Kenai source repository, and formatted as tables with comments read out of a file
- the left and right columns, and the page footer
- the response form
Downloading Jetty

Setting it up

- Unzip it to any directory you want
- set the JETTY_HOME environment variable to this directory, e.g. setenv JETTY_HOME=`pwd`
Delete the whole sample set

- delete anything under JETTY_HOME/webapps
- or save them somewhere for reference
Modify the /etc

- We do want our NetRexx Server Pages to have the .nsp filename extension - call it chauvinism
- in JETTY_HOME/etc/webdefaulrts.xml,
Add one line to recognize .nsp as .jsp

- `<servlet-mapping>`
- `<servlet-name>jsp</servlet-name>`
- `<url-pattern>*.nsp</url-pattern>`
- `<url-pattern>*.jsp</url-pattern>`
- `<url-pattern>*.jspf</url-pattern>`
- `<url-pattern>*.jspx</url-pattern>`
Add a ROOT app to webapps

- This ROOT app will be the application selected for the url of the website
- These are called domains in J2EE
- We can arrange for domains to be served by virtual servers
Running it

- Make sure the JETTY_HOME environment variable is set correctly
- In JETTY_HOME/bin, issue ./.jetty.sh start
- (stopping would be ./.jetty.sh stop)
Adding active content to pages

- Server side
- Edit html pages and add the tags
- e.g. for the qtime program
We use jsp tags

**Hursley Labs**

Hursley, located near Winchester in the UK, is the place where many famous products originate. Incidentally, in Hursley, according to the [qtime](http://netrexx.org/examples/ibm-historic/qtime.nrx) program, one of the first-ever Rexx programs, 1979. This is the NetRexx version from 1996, which is almost identical. Being British, NetRexx listens to both `center` and `centre` method spellings.
The `<jsp:usebean>` tag

- This instructs the J2EE processor to find a class corresponding to this on the classpath that is formed by ROOT/WEB-INF/classes
- In this case, the class is found in ROOT/WEB-INF/classes/com/rvjansen
- That is the package name I gave it
The `<jsp:getProperty>` tag

- this has a `name=` attribute which refers to the `<useBean>` tag and a `property=` attribute which refers to the name of the property
- if you understand this, the hardest part is in the past
Small mods to qtime

definitions

package com.rvjansen

class qtime

/* QTIME. This program displays the time in real English. */
/* If "?" is given as the first argument then the */
/* program displays a description of itself. */

properties indirect
out = Rexx ""

method qtime() protect

/* Nearestness phrases - using associative array lookup */
near:"" /* default */
near[0]="" /* exact */

/* Extract the hours, minutes, and seconds from the time. */
/* Use the Java Date class as Rexx.Date not yet implemented */
parse Date() . . . now . /* time is fourth word */
parse now hour:\"min\":\"sec\"

/* not needed for the current AMS host centre */
hour = hour + 1 /* quick zulu time fix - change soon */
if hour = 13 then hour = 1

if sec<30 then min=min+1 /* round up minutes */
mod=min//5 /* where we are in 5 minute bracket */
out=\"it\"s near[mod] /* start building the result */
if min>50 then hour=hour+1 /* we are 30 the hour... */
min=min+2 /* shift minutes to straddle a 5-minute point */

/* don't do this as West Virginia noon is zulu midnight */
/* Now special-case the result for Noon and Midnight */
/* if hour/12=0 & min/60<=4 then do */
/* if hour=12 then say out \"noon\" */
/* else say out \"Midnight\" */
/* return */
/* end */

min=min-(min//5)
if hour>12
then hour=hour-12 /* get rid of 24-hour */
else if hour=0 then hour=12 /* ... and allow for midnight */
/* Determine the phrase to use for each 5-minute seg */
select
when min=0 then nor /* add \"o\"clock */
when min=5 then out=\"five past\"
when min=10 then out=\"ten past\"
when min=15 then out=\"quarter past\"
when min=20 then out=\"twenty past\"
when min=25 then out=\"twenty-five past\"
when min=30 then out=\"half past\"
when min=35 then out=\"twenty-five to\"
when min=40 then out=\"twenty to\"
when min=45 then out=\"quarter to\"
when min=50 then out=\"ten to\"
when min=55 then out=\"five to\"
end

numbers="one two three four five six" /* (continues */
   "seven eight nine ten eleven twelve */
out=out numbers.word(hour) /* add the hour */
if min=0 then out=out \"o\"clock /* ... and o\"clock to */

/* Mike Cowlishaw, December 1978 - January 1985 */
/* NetRexx version March 1996 */
dinsdag 7 mei 13
The examples page

```java
method perDirectory(dirName_) protect signals IOException, FileNotFoundException
output.println("<table><tr class="row">" + "Example</td><th>Description</th></tr></table>")
-- get directory
f = File(dirName_)
do
  in = BufferedReader(FileHeader(dirName_,'/legenda.txt'))
  loop forever
    line = in.readLine()
    if line = null then leave
    parse line filename '|' explanation
    legendaMap.put(filename,explanation)
  end
  catch Exception
  end -- do

linkDir = dirName_.substr(13)

files = f.listFiles()
loop i=0 to files.length-1
  fileName = Regex(files[i].toString())
  if fileName.pos(' .svn') >0 then iterate
  if fileName.pos(' .makefile') >0 then iterate
  if fileName.pos(' .legenda.txt') >0 then iterate
  endDelim = fileName.lastIndexOf('/')
  fileName2 = fileName.substr(endDelim+1)
  if i % 2 = 0 then output.println("<tr class="rowA"><td>")
  else output.println("<tr class="row"><td>")
    link = '<a href=' + linkDir + '/filename2.toString()' + ' filename2.toString()' + '><a>
    output.println(link.toString())
    expl = this.legendaMap.get(fileName2)
    if expl = null then expl = ''
    output.println(expl)
  end
output.println('</table>')
```
The feedback form

Thank you, your message has been sent.
The feedback code

```java
options binary
package com.rv.jansen
import javax.servlet.jsp.
/
/*
 * Class message implements the message to send from the webpage
 */
/*
 * Created on: di, 12, mrt 2013 12:13:28 +0100
 */

class message
properties indirect
firstname = String
lastname = String
emailaddr = String
message = String
pc = PageContext
/
/*
 * Default constructor
 */
method message()

method doIt() protect
out = PrintWriter(BufferedWriter(FileWriter('messages.txt',1)))
out.println(Date())
out.println(pc.getRequest().getRemoteAddr())
out.println(this.getFirstName())
out.println(this.getLastName())
out.println(this.getEmailAddr())
out.println(this.getMessage())
out.println('')
out.close()
return ""
```
Solving problems

- top
- kill -3 <pid>
- jstack <pid>
- trace
Potential problems

- class not found
- threading issues
Multithreading

- Active web content programs are multithreaded by nature
- Even one user can have multiple windows open and/or press the submit buttons in a high tempo
Avoiding threading issues

- no static variables
- use the synchronized version of JVM collection classes
Future plans

- Bridging it to ooRexx using BSF4ooRexx
- Making an integrated component for this (JavaBean)
Conclusion

- Using NetRexx, you are able to put together an active website using standard J2EE concepts and facilities.
- There is only one line added to a standard config file.