gWidgetsWWW

John Verzani, gWidgetsRGtk@gmail.com

March 2, 2009

Abstract:

The **gWidgets** package provides an API for interfacing with a few of the available GUI toolkits avaiilable through R. The **gWidgetsWWW** package provides an implementation of the **gWidgets** API for use with through the web. The package depends on the **Rpad** package to provide a pipe from a web browser back to an R process and again back to the web browser. **Rpad** can be run in both a local or server version. The server version allows GUIs written with **gWidgetsWWW** to be shared over the internet. The **gWIdgetsWWW** package uses the EXT JS javascript libraries (`www.extjs.com`). These need to be downloaded and installed.

The current status of the project is still experimental. The package does not have much testing as of yet. This document needs to be significantly enhanced. The author appreciates any feedback.

1 Overview

When a user opens a webpage with **gWidgetsWWW** code, the **Rpad** functions call back into an `R` session. The `R` commands produce javascript that is `cat`ted back to the browser which renders it on the screen. There is no way to initiate a request back to the browser from the R console, only in response to a request.

The **Rpad** package uses AJAX technology to process subsequent requests. Unlike traditional CGI style programming, each page is not reloaded after a new request. So the initial page may take a while to load (it loads in lots of javascript libraries), but once loaded the GUIs are fairly responsive.

For the server version, **Rpad** uses a PERL package to keep a separate session for each initial loading of a page. This allows multiple users to access the same GUIs at the same time.

To make a GUI in **gWidgets** can be as easy as loading the **gWidgets** package then calling

```
w <- gwindow("simple GUI with one button", visible=FALSE)
g <- ggroup(cont = w)
b <- gbutton("click me", cont = g, handler = function(h,...) {
  gmessage("hi", parent = b)
})
visible(w) <- TRUE
```

With **gWIdgetsWWW** there are two steps: The first is to create a web page that the user can call. A template is available in the Examples directory. Here is the basic form:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<!-- Call in Rpad  -->
<script type="text/javascript">
djConfig = {isDebug: true};
rpadConfig = {rpadHideSource: true,rpadRun: "all"};
</script>
<style type="text/css">
.Rpad_background-color: #eeeeee;font-family: georgia;}
</style>
<script type="text/javascript" src="gui/dojo.js"></script>
<script type="text/javascript" src="gui/Rpad_main.js"></script>


<!-- Call in processing -->
<script type="text/javascript" src="processing.js"></script>
<script type="text/javascript" src="processinginit.js"></script>

<!-- Call in Ext and its style sheet -->
<script type="text/javascript" src="ext/adapter/ext/ext-base.js"></script>
```

```
<script type="text/javascript" src="ext/ext-all.js"></script>
<link rel="stylesheet" type="text/css" href="ext/resources/css/ext-all.css">


<!-- Google Maps API requires one to register on the website http://code.google.com/a
<!-- Replace the key below with your own. This works for 127.0.0.1:8079 -->
<!-- <script -->
<!-- src="http://maps.google.com/maps?file=api&v=2.x&key=ABQIAAAAYpRTbDoR3NFWvhN4JrY1
<!-- -TpGz1e2ncErJceBS7FrNBqzV5DPxkpbheIzZ9nTJPsQ" -->
<!-- type="text/javascript"> -->
<!-- </script> -->


</head>
<body onunload="rpad.logoutR();">
<pre dojoType="Rpad" rpadType="R" rpadOutput="javascript" rpadRun="init">
Rfile = "ex-template.R"
basedir = ifelse(RpadIsLocal(), "","/var/www/Rpad/")
file = paste(basedir, Rfile,sep="")

require(gWidgetsWWW, quietly=TRUE)
source(file)
</pre>
</BODY>
</HTML>
```

The user changes the `Rfile` name and possibly the default base directory when run in server mode. This file does a few things

- It loads the necessary javascript files to use Rpad
- It loads the necessary extjs files to use the EXT JS libraries
- It loads the necessary javascript libraries to use processing.js (for drawing graphics)
- It optionally loads javascript for the ggooglemaps widget.

The function `rpad.logoutR()` will be called when the page is unloaded. The `R` code is contained in a `pre` environment. In this case, we make sure the R process will find the sourced in file by using absolute paths. (Do not use `setwd` here.)

The second part is to write a file with the **gWidgets** commands. The file `ex-template.R` above, could look like:

```
> w <- gwindow("simple GUI with one button", visible=FALSE)
> g <- ggroup(cont = w)
> b <- gbutton("click me", cont = g, handler = function(h,...) {
+   gmessage("hi", parent = b)
+ })
> visible=FALSE
```

This would produce a very basic GUI with a single button. Clicking the button will popup a message. Delaying the drawing of top-level window is options in **gWIdgets**, but required here as it creates the javascript code to draw the initial GUI.

Another difference is there can only be one top-level windows. All other `gwindow` instances should use the `parent` argument to specify an object that acts as the parent of a subwindow. (an animation will appear from there, say.)

The main webpage for **gWidgetsWWW** `http://www.math.csi.cuny.edu/gWidgetsWWW` contains examples of some basic GUIs. These are included in the Examples subdirectory of the package once installed.

2 The containers

The **gWidgetsWWW** package has all the following containers:the top-level container `gwindow`, subwindows also constructed through `gwindow` (use a parent object); the box containers`ggroup`, `gframe` and `gexpandgroup`; the tabular layout container `glayout`; the notebook container `gnotebook`, but no `gpanedgroup`. The demo `http://www.math.csi.cuny.edu/gWidgetsWWW/ex-containers.html` contains examples,

To make a component appear in response to some action – such as happens with gexpandgroup, one can add it to a box container dynamically. Or one can put it in a `ggroup` instance and toggle that containers visibility with the `visible` method, in a manner identical to how `gexpandgroup` is used.

3 The widgets

Most of the standard widgets work as expected. Examples are visible on`http://www.math.csi.cuny.edu/gWidgetsWWW/ex-widgets.html`. No attempt has been made

to include the compound widgets `gvarbrowser`, `ggenericwidget`, `gdfnotebook`, `gcommandline` and `ggraphicsnotebook`.

4 Different features

There are a few differences/additions for this package. This package is implemented independently of the **gWidgetsWWW** package, and so there may be some unintended inconsistencies in the arguments. The package uses the **proto** package for object-oriented support, not S3 or S4 classes. There are some advantages to this, and some drawbacks. One advantage is the user can modify objects or call their internal methods.

4.1 Data persistence

For the server version, each time a page is loaded a new R process is loaded. Any variables stored in memory are forgotten. To keep data persistent across pages, one can load and write data to a file. The example `http://www.math.csi.cuny.edu/gWidgetsWWW/ex-editdata.html` indicates how this might be done.

4.2 Comboboxes

The `http://www.math.csi.cuny.edu/gWidgetsWWW/ex-gcombobox.html` example shows how comboboxes can show a vector of items, or a data frame with a column indicating an icon, or even a third icon with a tooltip. As well, the `gedit` widget does not have a type-ahead feature, but the combobox can be used for this purpose.

The following will set this up.

```
> cb <- gcombobox(state.name, editable=TRUE, cont = w)
> cb$..hideTrigger <- TRUE ## set property before being rendered
```

4.3 graphics

There is no plot device avaiable. Rather, one uses the **Cairo** device driver to create graphic files whichare then shown using `gimage`. See `http://www.math.csi.cuny.edu/gWidgetsWWW/ex-image.html` for an example. The **Cairo** package is used, as it does not depend on X11, so works in server installations as well. An example is provided with the package. One caveat is that the `gimage` object can use relative paths, but the `Cairo` driver should use absolute paths when running under server mode.

4.4 gprocessingjs

The `gprocessingjs` widget provdes a javascript-like device for writing graphics to. See `http://www.math.csi.cuny.edu/gWidgetsWWW/ex-gprocessingjs.html` for an example. It is not an actual device though, it only provides some familar methods for drawing grpahics objects. The user needs to implement the basic graphs using these methods.

4.5 ggooglemaps

The `ggooglemaps` widget provides access to a sliver of the google maps API. The example at `http://www.math.csi.cuny.edu/gWidgetsWWW/ex-ggooglemaps.html` does not currently work in server mode. This sliver could be enlarged quite easily if desired. This work well with the local version, but may not go on the server version. To use this widget, a key must be obtained from google and the html file must call this in. An example is provided that works through the local web server provided by Rpad.

5 Installation

[NEEDS MORE!]

Installation can be as easy as:

- install Rpad following instructions at `www.rpad.org/Rpad`.
- Download and install the extjs java script libraries in a directory `ext/` that is references relatively by the browser.
- install the **gWidgetsWWW** package and write a GUI
- Start the local version of Rpad and then load the web page that calls in your GUI.

It is suggested that all this be done within a secure environement, as the R user has access to system calls. One method is to use a virtual appliance approach, say with vmware server (`vmware.com`). Then setup consists of

- install vmware

- Install a host operating system (say JEOS a small linux distrubution from the Ubunto project.)
- Install a web server and R (`sudo apt-get install` is convenient)