Rscripts over the internet using FastRWeb and Rserve

Alex Fun: quickling@gmail.com

What is Rserve and FastRWeb

Brainchild of Simon Urbanek

Rserve lets you run R in a server and let other programs access your R scripts.

FastRWeb is a web server that communicates with Rserve

Rserve is integrated with...

Tableau

Ruby

C/C++

PHP

Java

FastRWeb ← this is what I will talk about

How did I find out about it?

Data scientist at Glasshat.com (2012 - 2015)

Started using Shiny to deliver reports to internal end users at start of 2014.

Moved on to FastRWeb in Nov 2014.

glasshat

PROJECT SETUP | DASHBOARD | ACTIONS | ACTION PLAN

Setup Project

Please tell us about your Project.

Project Name

SURF

Website URL

http://www.meetup.com/R-Users-Syd

Notifications

aardkat@gmail.com

Project Hours

40

Industry Sector

Web Stats & Internet Analytics

Ranking Data Source

Google Australia

Edit Landing Pages and Keywords

Let us know the Pages and Keywords you would like to rank better for. If you have a large number of URLs and Keywords, p
to import data into Glasshat.

Need help with choosing the right keywords? Find out more about Keyword Research here.

Priority Keywords

1

URL: HTTP:// www.meetup.com/r-users-sydney/

Keywords:
- sydney r users
- r meetup
- r user group

2

URL: HTTP:// www.meetup.com/r-users-sydney

Keywords:
- meet other r users
- learn r from other people
- sydney r meetup

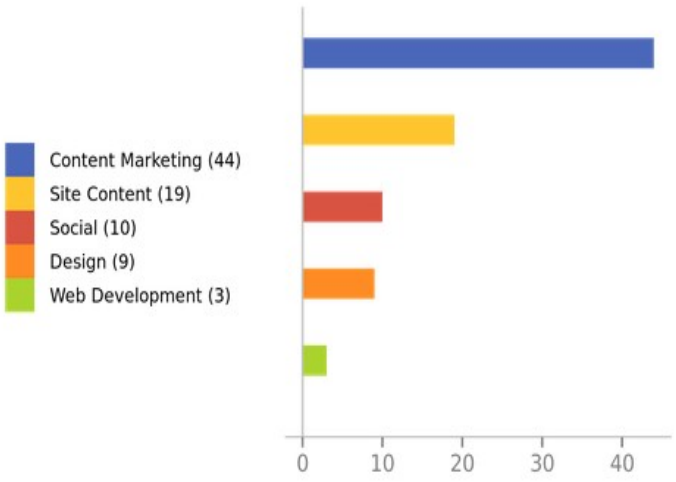
glasshat

Home / SURF / SURF

PROJECT SETUP | DASHBOARD | ACTIONS | ACTION PLAN

Glasshat has finished it's data collection and found the following areas of SEO most influential for your target keywords:

Influential SEO Factors for your industry



Recommended SEO Actions for your Website

Based on these factors Glasshat has identified 48 SEO Tasks for your website.

Action Collection is complete.

What about Shiny?

Shiny vs FastRWeb

Shiny	FastRWeb
Great for data exploration.	Great for delivering a well defined end product.

Shiny vs FastRWeb

Shiny	FastRWeb
Great for data exploration.	Great for delivering a well defined end product.
Need to understand reactive framework.	Easy to integrate existing R code.

Shiny vs FastRWeb

Shiny	FastRWeb
Great for data exploration.	Great for delivering a well defined end product.
Need to understand reactive framework.	Easy to integrate existing R code.
Leave framework to third party.	User in control of individual modules.

Shiny vs FastRWeb

Shiny	FastRWeb
Great for data exploration.	Great for delivering a well defined end product.
Need to understand reactive framework.	Easy to integrate existing R code.
Leave framework to third party.	User in control of individual modules.
$100/year basic paid plan @shinyapps $1100/year for authentication $9995 a year for Shiny Server Pro.	Free, excluding server costs. Can do HTTPs/Websockets/AJAX/ parallel connections

Syntax example

app.R

```r
library(shiny)
server <-
    shinyServer(function(input, output) {
    output$text <- renderText("Hello world!")
})


ui <- shinyUI(fluidPage(
    mainPanel(
        verbatimTextOutput("text")
    )
)
)

shinyApp(ui = ui, server = server)
```

7:1 (Top Level)

example0.txt.R

Source on Save

```r
run <- function(){
    print("hello world")
}
```

Implementation

Installation of FastRWeb (Ubuntu)

In R, run install.packages(c(“Rserve”, “FastRWeb”))

Find out where FastRWeb installed to using system.file(package="FastRWeb")

Navigate to this directory, type sudo ./install.sh

By default everything installs to /var/FastRWeb/

Helpful links:

https://cran.r-project.org/web/packages/FastRWeb/INSTALL

http://www.r-bloggers.com/setting-up-fastrweb-on-mac-os-x/

Configure your http server

Navigate to directory FastRWeb is installed in (/var/FastRWeb).

Edit code/rserve.R and add the lines

```
library(FastRWeb)
    .http.request <- FastRWeb:::.http.request
```

Edit code/rserve.conf and add the following line

```
http.port 80
```

Start your engine with sudo ./start!!!

*With the latest version of Rserve there is no need to set up Apache/RCGI.

Examples

Where are my scripts!?

Scripts live in the web.R folder:

```
rserve-surf
tmp.R
ubuntu@ip-172-31-47-198:/var/FastRWeb/web.R$ ls -1a
.
..
common.R
example1.png.R
example2.R
index.R
info.R
main.R
README
rserve-surf
tmp.R
ubuntu@ip-172-31-47-198:/var/FastRWeb/web.R$ ls rserve-surf -1a
.
..
example0.txt.R
example10_predict_best_text_colour.html.R
example1_print_argument.txt.R
example2.png.R
example2_print_argument.html.R
example3_print_argument.csv.R
example4_print_argument.json.R
example5_squaring.json.R
example6_squaring.json.R
```

Example 0

http://52.27.26.223/rserve-surf/example0.txt

example0.txt.R

Source on Save

Run

```
run <- function(){
    print("hello world")
}
```

(Script found at /var/FastRWeb/web.R/rserve-surf/example0.txt.R)

The run block

Running your R Script is as simple as putting them inside a “run” block:

sample_run_block.R

Source on Save Run

```
run <- function(...){

    # do your fancy coding here
    # :)
    # or just copy and paste existing code
}
```

Example 1

Arguments come after the file name in the browser. No spaces!

http://52.27.26.223/rserve-surf/example1_print_argument.txt?name=Alex

example1_print_argument.txt.R

Source on Save

Run

```
run <- function(name) {
    print(paste0("Hello ", name))
}
```

Example 2

WebResult gives you control over the content returned from the server.

http://52.27.26.223/rserve-surf/example2_print_argument.html?name=Mark

example2_print_argument.html.R

Source on Save Run

```
run <- function(name) {
    payload <- paste0("Hello ", name)
    payload <- paste0("<h1>", payload, "</h1>")
    WebResult(payload = payload, cmd = "html") # html
}
```

Four types of WebResult

html – This is default.

tmpfile – Good for sending csvs/graphics.

raw – Gives you full control over headers, good for json data output.

file – I don't usually use this.

We will see examples of the first three.

Example 3

Return a csv using WebResult(..., content.type = …) forces browser to acknowledge download

Save to /tmp/ folder and use file_name as payload.

http://52.27.26.223/rserve-surf/example3_print_argument.csv?name=Mark

example3_print_argument.csv.R

Source on Save

```r
run <- function(name) {
    df = data.frame(name = name);
    file_name = paste0("example3_output_", Sys.Date(), ".csv")
    write.csv(df, paste0("/var/FastRWeb/tmp/", file_name), row.names = FALSE)

    # Use cmd = "tmpfile" to find file_name in the tmp directory.
    # setting content.type forces browser to download as csv.
    WebResult(payload = file_name, cmd = "tmpfile",
              content.type = "text/csv; charset=utf-8")

}
```

Example 4

Use WebResult(..., cmd = “raw”) to control the response format. Content types are MIME codes:

http://52.27.26.223/rserve-surf/example4_print_argument.json?name=Mark

example4_print_argument.json.R

Source on Save

```r
run <- function(name) {
    payload <- jsonlite::toJSON(list(name = name), auto_unbox = TRUE)
    # use cmd = raw to send your own headers,
    # e.g. we want the browser to interpret the response as json
    WebResult(payload = payload, cmd = "raw",
              content.type = "application/json")
}
```

Example 5

Function arguments are interpreted as text:

http://52.27.26.223/rserve-surf/example5_squaring.json?x=3

example5_squaring.json.R

Source on Save

```
run <- function(x) {
    payload <- x^2
    WebResult(payload = payload, cmd = "raw",
              content.type = "application/json")
}
```

Example 6

Thus, it is important to process your input arguments:

http://52.27.26.223/rserve-surf/example6_squaring.json?x=6

example6_squaring.json.R

Source on Save

```r
run <- function(x) {
    # convert inputs
    x <- as.numeric(x)
    # end convert inputs

    payload <- jsonlite::toJSON(list(result = x^2),
                                auto_unbox = TRUE)
    WebResult(payload = payload, cmd = "raw",
              content.type = "application/json")
}
```

Example 7

Multiple arguments are separated with “&”

http://52.27.26.223/rserve-surf/example7_multiple_arguments.json?a=1&b=0&c=6

example7_multiple_arguments.jso...

Source on Save

```r
solve_quadratic_equation <- function(a, b, c) {
    # solve for roots of equation ax^2 + bx + c = 0 if real roots exist
    qdet = b^2 - 4*a*c
    if(qdet < 0){
        payload <- list(root = NA);
    } else if(qdet ==0) {
        payload <- list(root = (b + sqrt(qdet)) / (2*a))
    } else {
        payload <- list(root = (b + sqrt(qdet)) / (2*a), root2 = (b - sqrt(qdet)) / (2*a))
    }
    return(jsonlite::toJSON(payload, auto_unbox = TRUE, na = "string" ))
}

run <- function(a, b, c) {
    # process inputs
    a <- as.numeric(a)
    b <- as.numeric(b)
    c <- as.numeric(c)
    # end process inputs
    payload <- solve_quadratic_equation(a, b, c)
    WebResult(payload = payload, cmd = "raw", content.type = "application/json")
}
```

Example 8

Use ggsave to return graphics:

http://52.27.26.223/rserve-surf/example8_ggplot2.png?HEX=1b2c33&text_colour=FFFFFF

example8_ggplot2.png.R

Source on Save Run

```r
run <- function(HEX, text_colour){

    # very uninteresting plotting example :)
    p <- ggplot(df, aes(x = X, y = Y, label = labelnames)) +
        geom_rect(xmin = -Inf, xmax = Inf,   ymin = -Inf, ymax = Inf,   fill = HEX) +
        geom_text(size = 4, colour = text_colour) +

    # now use ggsave to put a temp file into the /var/FastRWeb/tmp/ directory
    # it is good idea to use either a guid or timestamp to save the file,
    # to avoid clashes from multiple sessions.
    file_name <- gsub("#", "", paste0(HEX,"_",text_colour, "_", gsub("( |:)", "-", Sys.time()), ".png"))
    ggsave(filename = paste0("/var/FastRWeb/tmp/", file_name), plot = p, height = 2.4, width = 3, dpi = 125)

    # after the file is loaded, it is deleted from the ../tmp directory
    WebResult(cmd = "tmpfile", payload = file_name, content.type = "image/png")

}
```

Machine learning example

With thanks to brain.js:

http://52.27.26.223/rserve-surf/find_best_text_colour.html

find_best_text_colour.html.R

Source on Save

```r
get_random_HEX <- function(){
    # this function produces a random colour in HEX

}

write_results_to_database <- function(choice){

    # write user choice to database
    make_con_to_database()
    postgresqlExecStatement(conn,
                            'insert into colour_choice
                            (r, g, b, text_colour)
                            VALUES ($1, $2, $3, $4)',
                            list(r, g, b, text_colour)
    )
    
}
```

find_best_text_colour.html.R

Source on Save

```r
run <- function(white, black, choice, ...) {
    HEX <- get_random_HEX()

    if(!missing(choice)) {
        # write results to database
        write_results_to_database(choice)
    }

    out()
    out("<form>")
    out("<title> Click the one which is easier to read </title>")
    out("Click the one which is easier to read<p>")
    oinput("white", type = "image",
           src = paste0("http://52.24.20.237/rserve-surf/example8_ggplot2.png?",
                        "text_colour=FFFFFF",
                        paste0("&HEX=", HEX)
           )
    )

    choice1 = gsub("#", "", paste0(HEX, "1"))
    oselection("choice", text = choice1, values = choice1, sel.value = 1, size = 0, st
    out("</form>")

    out("<form>")
    oinput("black", type = "image",
           src = paste0("http://52.24.20.237/rserve-surf/example8_ggplot2.png?",
                        "text_colour=000000",
                         paste0("&HEX=", HEX)
                        )
    )

```

Example 9

Retrieve data from the database and fit a model:

http://52.27.26.223/rserve-surf/example9_fit_model

example9_fit_model.R

Source on Save

```r
run <- function(){
    # Get data from database
    make_con_to_database()
    df <- dbGetQuery(conn, "select * from colour_choice")
    df$text_colour <- as.factor(df$text_colour)
    dbDisconnect(conn)

    # fit neural network
    nnmodel <- nnet(data = df, text_colour ~ r + g + b, size = 6, maxit = 1000)

    # save with appropriate filename
    file_list <- list.files('/var/FastRWeb/web')
    number_of_files = length(file_list)
    file_name = paste0("/var/FastRWeb/web/model_", (number_of_files + 1))
    save(nnmodel, file = file_name)
    print(paste0("Model saved to: ", file_name))
}
```

Example 10

Predict white or black text given RGB/HEX input:

http://52.27.26.223/rserve-surf/example10_predict_best_text_colour.html?hex=802080&model=6

example10_predict_best_text_col...

Source on Save

```r
run <- function(hex, r, g, b, model_number = 0){

    # try and find model
    if(model_number != 0){

        if(length(grep(paste0('model_', number_of_files), file_list)) != 1) {
            stop('Model not found or not well specified, using default')
        } else {
            load(file_name)

        }
    }

    # load in new data for prediction
    newdf <- data.frame(r, g, b)
    text_colour <- predict(nnmodel, newdata = newdf, type = "class")

   # now use existing ggplot to construct output
    out(paste0("Based on your choice of R: ", r, ", G: ", g, ", B: ", b,
               ", the optimal text colour is ", text_colour_printed, "<p>"))
    out(paste0("<img src=", paste0("http://52.24.20.237/rserve-surf/example8_ggplot2.png?",
                        "text_colour=", text_colour,
                        paste0("&HEX=", HEX)), ">")
    )
```

Questions?

source code for examples +
presentation can be found at
https://github.com/alexfun/rserve-surf-public

Cheat sheet: preloading code

rserve.R

Load all common packages in here.
Load common data sets.
Load machine learning models.

Stuff loaded on startup.

Global in memory

common.R

Same as rserve.R except code is run everytime a script is sourced.

Useful for small functions + rapid testing.

Global on load

your_script.R

Load functions/data local to your script.

Local on load

```
## This is jsut a friendly way to load package and report success/failure
## You will definiteily need FastRWeb, others are optional
pkgs <- c("XML", "Cairo", "Matrix", "FastRWeb", "ggplot2", "nnet")
cat("Loading packages...\n")
for (pkg in pkgs) cat(pkg, ": ",require(pkg, quietly=TRUE, character.only=T

## fix font mappings in Cairo -- some machines require this
if (exists("CairoFonts")) CairoFonts("Arial:style=Regular","Arial:style=Bol

## Load any data you want
data.fn <- paste(root, "code", "data.RData", sep='/')
if (isTRUE(file.exists(data.fn))) {
  cat("Loading data...\n")
  load(data.fn)
}

## init() is a special function that will be called from
## each script. Do what you want here - it is usually a good idea
## to have a "common" script that is loaded on each request
## so you don't need re-start Rserve for global code changes
init <- function() {
    set.seed(Sys.getpid()) # we want different seeds so we get different fi

    ## get a temporary file name for this session
    tmpfile<<-paste('tmp-',paste(sprintf('%x',as.integer(runif(4)*65536)),c

    ## if there is a common script, source it first
    common <- paste(root,"/web.R/common.R",sep='')
    if (isTRUE(file.exists(common))) source(paste(root,"/web.R/common.R",se
}

library(FastRWeb)
     .http.request <- FastRWeb:::.http.request


# load default model
load("/var/FastRWeb/web/model_0")
```

```
## if you use the example configuration supplied in "code"
## then this script will be loaded for all requests
## before the run() function is evaluated. You can use it
## for global processing, for example, to handle cookies.

getCookies <- function() {
    ## raw.cookies is a variable populated by the FastRWeb engine
    ckv <- gsub("^ +","",strsplit(request$raw.cookies,";",fixed=TRUE)[
    cookies <<- if (length(ckv)) {
            keys = unlist(lapply(strsplit(ckv,"="),function(x) x[1]))
                vals = substr(ckv,nchar(keys)+2,99999)
                names(vals) = keys
                vals
          } else character(0)
}

## just a dummy example - this will create a "cookes" variable
## that can be used by all run() scripts to access cookie contents.
getCookies()

make_con_to_database <- function(){
    library(RPostgreSQL)
    conn <<- dbConnect(PostgreSQL(),
                       user = "rserve_test",
                       password = "rserve_password",
                       host = "rserve-test-db.ctc8rhxcnutw.us-west-2.r
                       port = "5432",
                       dbname = "rserve_test_db")
}

common.R (END)
```

Cheat Sheet
FastRWeb Directory Structure

/code

Global config
e.g. http/https

Initialise memory state with packages/data/functions /models

/web.R

Your R scripts

Can have subfolders for structure

/tmp

Store temp files here for access by WebResult()

When files are retrieved they are deleted automatically

/web

Store permanent files here

Cheat sheet: general script structure

general_script_structure.R

Source on Save

```r
func1 <- function(...){
    # do calculations here for main run() block
}

func2 <- function(...){
    # some different batch of calculations
}

# main run block

run <- function(arg1, arg2, ...){
    # load stuff locally
    library(RPostgreSQL)

    # process inputs
    arg1 <- as.numeric(arg1)
    arg2 <- as.Date(arg2)
    # etc
    # end process inputs

    # do clever stuff to produce output
    payload <- func1(arg1) + func2(arg2)

    # now return payload using appropriate WebResult
    WebResult(payload = payload, cmd = ...,
              content.type = ...)

}
```