Shiny R Tips

**Useful libraries:**

1. library(shinythemes) 🡪 provide themes to select
2. library(shinyWidgets) 🡪 provides different cool widgets
3. library(shinycssloaders) 🡪 provides some css functions: like spinner for progress

**UI Layout:**

Navbarpage( title = , theme = ,

TabPanel(

sidebarLayout(

sidebarPanel(

tabsetPanel( type = “tabs” ,

tabPanel()

)

),

mainPanel(

tabsetPanel( type = ,

tabPanel()

)

)

)

)

)

**UI Widgets**

1. **fileInput ( get file externally)**

fileInput(inputId = "newleasefile",

label = "Choose 1 file to upload (only accept xlsx file):",

multiple = FALSE,

accept = c(".xlsx")

)

1. **action button**

actionButton("newlse\_submibutton", "Submit!", icon = icon("thumbs-up"))

1. **download button**

downloadButton(outputId ="download\_new\_lease\_template", "Download New Lease Template File")

1. **ShinyWidgets: pickerInput**

pickerInput( inputId = "newselectbreakdownby", label = "Breakdown by:", choices = c('Lease ID',

'Legal Entity'

),

options = list(`actions-box` = TRUE), multiple = FALSE)

1. **dateRangeInput**

dateRangeInput(inputId = "Lease\_end\_dt",

label = "Select Lease End Date between:",

start = as.Date(NA), # make the filed blank

end = as.Date(NA),

min = min\_end\_dt, #cannot change to NA

max = max\_end\_dt, #cannot change to NA

format = "yyyy-mm-dd",

startview = "year"), # provide calendar for use to choose date

1. **selectInput**

selectInput(inputId = "Legal\_Entity",

label = "Legal Entity ID",

choices = c(sort(unique(as.character(lease$Legal\_Entity)))),

multiple = TRUE,

selectize = FALSE,

selected = NULL)

1. **textInput**

textInput( inputId = "LP\_Date",

label = "Lease Payment Day is Nth day of each month:",

placeholder = "Enter a number like 1",

value = '1')

1. **NumericInput**

numericInput( inputId = "change\_year",

label = "Lease Change Year:",

min = lubridate::year(Sys.Date()),

max = NA,

value = lubridate::year(Sys.Date()))

**Progress Indicator**

1. **Spinner (from library(shinycssloaders))**

withSpinner(DT::dataTableOutput(outputId = "New\_Lease\_VS\_Table"))

1. **Progress bar**

new\_lease\_table <- reactive({

withProgress(message = 'Creating Table', value = 0,{

n <- length(lease\_info\_df\_final)

for (i in 1:n) {

print(paste("i is",i))

datalist\_new[[i]] <- new\_lease\_forecast(lease\_info\_df = lease\_info\_df\_final[[i]],

recurring\_cost\_df = recurring\_cost\_df\_final[[i]]

)

incProgress(1/n, detail = paste("Running Part", i, "out of", n))

Sys.sleep(0.1)

}

})

**Add image/logo**

1. **Can convert image to an URI**

b65 <- **base64enc::dataURI**(file="C:/Users/yawu/OneDrive - George Weston Limited-6469347-MTCAD/2019 Data Scientist Finance/IFRS 16/Shiny App/ShinyApp/www/Loblaw.PNG", mime="image/png")

navbarPage(

title =div(img(src=b64, width = 40, height = 38, align = "middle", style = "position: relative; top: -3px;"),

"Lease Forecasting Web App", img(src=b65, width = 70, height = 38, align = "middle",

style = "position: relative; top: -3px; right: -1000px;"))

1. **Use URL directly**

title =div( "Lease Forecasting Web App", img(src="https://lcl-flip-cdn.azureedge.net/img/global/logo-footer-Loblaw\_Companies\_Ltd.png",

width = 90, height = 38, align = "middle",

style = "position: relative; top: -3px; right: -1000px;"))

---------------------------------------------------------------------------------------------------------------------------

**UI Output**

DT::dataTableOutput(outputId = )

**Attach file to Shiny as Template file (in server part)**

|  |
| --- |
| # new lease template  output$download\_new\_lease\_template <- downloadHandler(  filename = function() {  paste0("New Lease Template File", ".", "xlsx")  },  content = function(file) {  myfile <- srcpath <- 'C:/Users/yawu/Desktop/new\_lease\_template.xlsx'  file.copy(myfile, file)  }  ) |

**Check if data exists (not empty) or value not equal to NA, Blank, 0, missing ects**

A: **IsTruthy()**

**DT::datatable Good Tips**

DT::datatable(data = summary\_df,

container = htmltools::withTags(

table(tableHeader(colnames(summary\_df)),

tableFooter(sapply(summary\_df, function(x)

if(is.numeric(x)) currency(sum(x, na.rm = TRUE), digits = 0L)))

)),

rownames = FALSE,

options = list(paging = TRUE,

searching = TRUE,

autoWidth = TRUE,

ordering = TRUE,

dom = 'Bfrtip',

scrollY = 400,

scrollX = TRUE,

scroller = TRUE,

pageLength = 800,

footerCallback = JS(jsCode))

)

Key points: 1) can use container to contain table header and table footer (you can customize it)

2 ) searching in options() can allow user to search something in the table

1. SCROLLER 🡪 allow user to scroll table
2. FootCallback 🡪 allow user to use customized javascript code

**Caution when using DT:datatable**

1. If you use extended button in datatable, like download. It can only download the current page data NOT the whole data

Want to download all data 🡪 1 possible way is let server = F

But if your data is too larger, this will take long time to process

1. Same issue happened if you want to get **summarized row** in datatable

Solution would be: you calculate the *summarized column total* then append the values by using container

**Javascript code used to put column total into the table footer:**

|  |
| --- |
| jsCode <- "function(row, data, start, end, display) {var api = this.api(), data;$( api.column(0).footer() ).html('Total');}"  jsCode2 <- "function(row, data, start, end, display) {var api = this.api(), data;  $( api.column(0).footer()).html('Total');  $( api.column(4).footer()).html(LeaseNum)  }"  jsCode2 <- sub("LeaseNum", unique(Overall\_Lease\_VS\_Table\_df()$ttl\_unique\_lse),jsCode2)   * Replace LeaseNum by unique(Overall\_Lease\_VS\_Table\_df()$ttl\_unique\_lse) because LeaseNum value needs to be updated by case |
| **Q: what if need use some data frames outputs from reactive function in Shiny?**  A: # create reactiveValues to store data created from reactive in the previous steps  data\_created <- reactiveValues()  observe({ if(is.null(new.lease.table.update2()) == FALSE){  data\_created$dat1 <- new.lease.table.update2()  }  })  observe({ if(is.null(multi\_change\_lease2()) == FALSE){  data\_created$dat2 <- multi\_change\_lease2()  }  })  observe({ if(is.null(uni\_change\_lease2()) == FALSE){  data\_created$dat3 <- uni\_change\_lease2()  }  }) |
| Q: How to add modal error message to user?  A: use showModal()  observeEvent( any(unique(data\_created$dat2$Lease\_ID) %in% unique(data\_created$dat3$Lease\_ID)) == T |  any(unique(data\_created$dat1$Lease\_ID) %in% unique(data\_created$dat2$Lease\_ID)) == T |  any(unique(data\_created$dat1$Lease\_ID) %in% unique(data\_created$dat3$Lease\_ID)) == T,{    showModal(modalDialog(  title = "Error",  "Please ensure no lease ID(s) double selected!",  easyClose = TRUE  ))  }  ) |

**UI Functions**

|  |  |
| --- | --- |
| **Functions** | **Usage** |
| Reactive() | Update immediately once input values changing |
| eventReactive() | Delay reactive depends on something 1st |
| Observe() | Update value when something happened 1st |

|  |
| --- |
| newlease\_recurr\_cost <- eventReactive(input$newlse\_submibutton, {  req(lease\_recur\_cost())  new\_recurr\_cost <- lease\_recur\_cost()  # update fields type  new\_recurr\_cost$lse\_term\_start\_dt <- as.Date(new\_recurr\_cost$lse\_term\_start\_dt, format = "%Y-%m-%d")  new\_recurr\_cost$lse\_term\_end\_dt <- as.Date(new\_recurr\_cost$lse\_term\_end\_dt, format = "%Y-%m-%d")  # group by PRPTY\_NUM, lse\_term\_start\_dt, lse\_term\_end\_dt, bse\_rnt\_amt  new\_recurr\_cost2 <- new\_recurr\_cost %>% select(PRPTY\_NUM, lse\_term\_start\_dt, lse\_term\_end\_dt, bse\_rnt\_amt) %>%  dplyr::group\_by(PRPTY\_NUM, lse\_term\_start\_dt, lse\_term\_end\_dt) %>%  dplyr::summarise(monthly\_pay = sum(bse\_rnt\_amt, na.rm=TRUE)) %>%  dplyr::arrange(PRPTY\_NUM, lse\_term\_start\_dt, lse\_term\_end\_dt)  # Map with the lease\_pd\_cal table  lease\_recur\_cost2 <- sqldf("select \* from new\_recurr\_cost2 left join lease\_pd\_cal  on new\_recurr\_cost2.lse\_term\_start\_dt <= lease\_pd\_cal.CORP\_PD\_END\_DT and  new\_recurr\_cost2.lse\_term\_end\_dt >= lease\_pd\_cal.CORP\_PD\_STRT\_DT")  return(lease\_recur\_cost2)  })  **Delay calculation 🡪 need user to click the submit button** |