Programming Server Logic in Shiny: Takeaways

by Dataquest Labs, Inc. - All rights reserved © 2021

Syntax

- Take a column picked in varSelectInput() and use it within a tidyverse function with the !! operator:

```
output$plot <- renderPlot({
    heart %>%
      ggplot(aes(x = !!input$x_axis, y = cp))
  })
```

- You can use the renderDataTable() and dataTableOutput() functions together to display a dataset on the app:

```
output$user_data <- renderDataTable({ heart }) # In server.R
dataTableOutput("user_data") # In ui.R
```

- We can use the conditionalPanel() function to selectively display parts of the interface based on different conditions:

```
input.example == "matching value" # This is a Javascript condition
conditionalPanel(
  condition = "input.example == 'matching value'", # Notice the use of single and double quotes
  dataTableOutput("user_data")
)
```

Concepts

- When building a Shiny app, it's important to work slowly. Every time you implement a new feature in your code, make sure you close the app and rerun your code to reopen it. Writing a lot of code and then testing it later makes it harder to identify errors.
- The user experience is another facet of making a Shiny app.
- Here's a workflow for developing new items in your server file:
- Create another field in the `output` object that will hold the tibble that we want to show on the interface.
- Find the corresponding `render*` function that works specifically with a tibble (or a data table, equivalently) to assign to this new field. In our case, this will be the `renderDataTable()` function.
    - Alternatively, we could use `renderTable()`, but it has less functionality.
- Find the corresponding `*Output` function that works specifically with a tibble (or a data table, equivalently) and place it in the interface. In our case, this will be the `tableOutput()` function.

Further Reading

- More reading on the bang-bang operator

Takeaways by Dataquest Labs, Inc. - All rights reserved © 2021