# Package: ShinyTester (via r-universe)

September 8, 2024

**Type** Package

**Title** Functions to Minimize Bonehead Moves While Working with 'shiny'

**Version** 0.1.0

**Author** Amit Kohli

**Maintainer** Amit Kohli <amit@amitkohli.com>

**Description** It's my experience that working with 'shiny' is intuitive once you're into it, but can be quite daunting at first. Several common mistakes are fairly predictable, and therefore we can control for these. The functions in this package help match up the assets listed in the UI and the SERVER files, and Visualize the ad hoc structure of the 'shiny' App.

**License** GPL-2

**Imports** dplyr, purrr, readr, stringr, tidyr, visNetwork

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 5.0.1

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2017-02-01 16:46:15

## Contents

ShinyDummyCheck *ShinyDummyCheck*

## Description

This function takes Shiny files themselves as inputs and tries to combine the different assets presented in the ui and server files to see whether they match up.

## Usage

```
ShinyDummyCheck(directory = ".", ui = "ui.R", server = "server.R")
```

## Arguments

| | |
|---|---|
| directory | the directory or website containing the files for the Shiny App. Defaults to current working directory |
| ui | a character vector size 1 containing the name of the UI files. defaults to "ui.R" |
| server | a character vector size 1 containing the names of the SERVER file. defaults to "server.R" |

## Details

For now, it only works where the server and ui files are seperate (ie, it doesn't work for 'app.R' yet)

You can test with your own app, go to your shiny app, make that your working directory, and then type 'ShinyDummyCheck()'

## Value

Returns a dataframe with the matchings b/w ui and server files. Also spawns them in VIEW mode. The structure of the table is as follows: - Item - The name of the asset that maybe should be on both server.R and ui.R - SrvCall - the TYPE of object that you're saying this specific item is (in server.R) - isOutput - is a binary that will specify if in server.R you wrote just 'item' or 'output$item' - VisualCall - is the TYPE of thingie you're trying to push the item into (in ui.R). - Status - Compares the SrvCall to the VisualCall, also looks at isOutput and then applies some rules to figure out if it's probably ok or not.

The Status types that are currently being checked for are: The conditions being checked are: It's OK if: - the server calls 'render(.)' and the ui calls 'Output(.)' (where . is the same Item). I also make exceptions for print==text and textoutput==verbatimtextoutput - If the server calls a reactive block, the ui should not have that Item name

It's NOT ok if: - the server is calling a non-reactive and the UI doesn't have it. (this causes false positive errors for things like 'observe' etc...) - the server is calling a reactive block and there IS something showing up on the ui - you are trying to show a non-reactive block in the ui, but forgot to put 'Output$' before the item name in the server

## Examples

```
ShinyDummyCheck(directory = system.file("example", package = "ShinyTester"))
```

ShinyHierarchy *ShinyHierarchy*

## Description

Create a hierarchical network chart that shows the _ad hoc_ structure of your shiny Server.

## Usage

```
ShinyHierarchy(directory = getwd(), ui = "ui.R", server = "server.R",
  offsetReactives = T)
```

## Arguments

| | |
|---|---|
| directory | the directory or website containing the files for the Shiny App. Defaults to current working directory |
| ui | a character vector size 1 containing the name of the UI files. defaults to "ui.R" |
| server | a character vector size 1 containing the names of the SERVER file. defaults to "server.R" |
| offsetReactives | a boolean that specifies if the middle row (the reactives) should show up in one row or whether there should be a small offset. TRUE by default. |

## Details

You can test with your own app, go to your shiny app, make that your working directory, and then type ‘ShinyHierarchy()‘

## Value

It returns a very very nice network chart with BASICALLY three-ish ROWS of nodes. The first one is the UI Inputs, the middle row(s) are the reactives, and the last row are the outputs being visualized. The hesitation for the second row (the reactives) is because I have introduced a small offset to each node in the middle row in order to see reactive flows into each other (if they are all in the same row, you can’t really see them). You can avoid this behavior by setting the parameter offsetReactives = F.

## Examples

```
ShinyHierarchy(system.file("example", package = "ShinyTester"))
```

# Index