Package 'crunchy'

July 21, 2020

Type Package

Title Shiny Apps on Crunch

Description To facilitate building custom dashboards on the Crunch data platform <https://crunch.io/>, the 'crunchy' package provides tools for working with 'shiny'. These tools include utilities to manage authentication and authorization automatically and custom stylesheets to help match the look and feel of the Crunch web application. The package also includes several gadgets for use in 'RStudio'.

Version 0.3.2

URL https://crunch.io/r/crunchy/, https://github.com/Crunch-io/crunchy

BugReports https://github.com/Crunch-io/crunchy/issues

License LGPL (>= 3)

Depends R (>= 3.0.0), crunch, shiny

Imports httpcache, miniUI, rstudioapi (>= 0.4)

Suggests covr, htmltools, httptest (>= 3.0.0), knitr, rmarkdown, shinytest, spelling, testthat, withr

RoxygenNote 6.1.1

VignetteBuilder knitr

Language en-US

NeedsCompilation no

Author Greg Freedman Ellis [aut, cre], Neal Richardson [aut], Jonathan Keane [aut], Gordon Shotwell [aut], Mike Malecki [aut]

Maintainer Greg Freedman Ellis <greg@crunch.io>

Repository CRAN

Date/Publication 2020-07-20 23:40:24 UTC

R topics documented:

`crunchPage` *Build a Crunchy UI*

Description

These are no longer necessary. Just use the `shiny` ones and it just works. These functions are left here for backwards compatibility.

Usage

```
crunchPage(...)

crunchFluidPage(...)

crunchFillPage(...)

crunchNavbarPage(...)
```

Arguments

`...`	arguments passed to `fluidPage`, `fillPage` or `navbarPage`

Value

The result of `fluidPage`, `fillPage` or `navbarPage`

Examples

```
## Not run:
crunchPage(
    fluidRow(
        column(6,
            selectInput("filter",
                label="Filter",
```

```
                choices=filterList,
                selected="All"),
            br(),
            plotOutput("funnel1", height="300"),
        ),
        column(6,
            selectInput("brand",
                label="Competitor",
                choices=brands,
                selected="Nike"),
            br(),
            plotOutput("funnel2", height="300"),
        )
    )
)

## End(Not run)
```

---

crunchyBody *A Shiny UI with Crunch auth*

Description

When using `crunchyServer()` to wrap your app in Crunch authentication and authorization, you need to wrap your UI body content inside `crunchyBody()`.

Usage

```
crunchyBody(...)
```

Arguments

...	UI elements for your app

Details

This is the part that is conditionally rendered if the user is allowed. Any UI elements you want always to show, including <head> tags, should go outside this function.

Value

A `uiOutput()` container into which `crunchyServer()` will conditionally render output.

See Also

`crunchyPublicBody()` `crunchyServer()`

Examples

```
## Not run:
shinyUI(fluidPage(
    tags$head(
        # This is content that will always be rendered
        tags$title("My secure app")
    ),
    crunchyBody(
        # This is content that only is rendered if the user is authorized
        fluidRow(
            column(6, h1("Column 1")),
            column(6, h1("Column 2"))
        )
    )
))

## End(Not run)
```

`crunchyPublicBody` *Alternate UIs for unauthenticated and unauthorized users*

Description

`crunchyServer()` and `crunchyBody()` allow you to protect your app with Crunch authentication and authorization. Add these UI contents to your `shiny::shinyUI()` body to display different content for visitors who are not authenticated with Crunch (`crunchyPublicBody()`) or who are authenticated but not authorized to access your app (`crunchyUnauthorizedBody`).

Usage

```
crunchyPublicBody(...)

crunchyUnauthorizedBody(...)
```

Arguments

`...`	UI elements for your app, to be conditionally rendered

Value

An empty string; these functions are called for their side effects of registering the UI elements so that `crunchyServer()` can render them as appropriate.

See Also

`crunchyBody()`; `setCrunchyAuthorization()` for governing who is authorized to view your app.

Examples

```
## Not run:
# This is the example from crunchyBody(), adding these alternate bodies:
shinyUI(fluidPage(
    tags$head(
        # This is content that will always be rendered
        tags$title("My secure app")
    ),
    crunchyBody(
        # This is content that only is rendered if the user is authorized
        fluidRow(
            column(6, h1("Column 1")),
            column(6, h1("Column 2"))
        )
    ),
    crunchyPublicBody(
        # This is shown to visitors who are not logged into Crunch at all
        h1("Please log into Crunch.")
    ),
    crunchyUnauthorizedBody(
        # This is for Crunch users who don't meet your authorization criteria
        # Perhaps they don't have access to a particular dataset
        h1("You don't have access to this app."),
        tags$div("Contact your_admin@example.com.")
    )
))

## End(Not run)
```

`crunchyServer` *A Shiny server with Crunch auth*

Description

To make sure that users who access your shiny.crunch.io app are allowed to access it, use `crunchyServer()` instead of `shiny::shinyServer()`, and wrap your UI content inside `crunchyBody()`. This will prevent anyone who is not logged into Crunch in their browser from accessing your app.

Usage

```
crunchyServer(func, authz = getOption("crunchy.authorization"))
```

Arguments

`func`	A `function(input,output,session)`, as you'd normally give to `shinyServer()`. If the user is not authenticated or authorized, this function will not be evaluated.
`authz`	A `function(input,output,session)` to evaluate to determine if the current user is authorized to enter the app. Since it will be called repeatedly, it should be cheap to execute.

Details

To restrict access further to only certain Crunch users, you can set an authorization method, either by passing a server function to the `authz` argument of this function, or by calling `setCrunchyAuthorization()`.

For a simple example app using `crunchyServer()`, copy `system.file("example_apps/crunchy_server/app.R",packag` supply your dataset id on line 14, and run it.

Value

Invisibly, a Shiny server function with the auth logic wrapped around `func`.

See Also

`crunchyBody()` for wrapping the UI contents, `crunchyPublicBody()` for specifying an alternate UI for when the user is not authenticated, `crunchyUnauthorizedBody()` for giving an alternate UI for users who are authenticated with Crunch but not authorized to view this app, and `setCrunchyAuthorization()` for governing who is authorized to view your app.

`listDatasetGadget` *Open dataset selector*

Description

Open dataset selector

Usage

```
listDatasetGadget(...)
```

Arguments

`...`	Further arguments passed to `crunch::listDatasets()`

Value

A `loadDataset()` call is pasted into your RStudio session

makeArrayGadget *Launch array builder gadget*

Description

Categorical Array and Multiple Response variables can be difficult to construct without being able to investigate the available variables, and their categories. This shiny gadget lets you select subvariables from the dataset list, and ensures that those variables have consistent categories. To use the gadget you must have at least one CrunchDataset loaded into the global environment.

Usage

```
makeArrayGadget(env = globalenv())
```

Arguments

env	the environment to run the gadget

Value

a valid call to `makeArray()` or `makeMR()`

setCrunchyAuthorization *Register authorization logic for your Crunchy app*

Description

Call this to set an expression or server function to evaluate to determine whether the current user is authorized to access your app. Ideally, this is cheap to execute because it will be called repeatedly.

Usage

```
setCrunchyAuthorization(func)
```

Arguments

func	A `function(input,output,session)` to call inside `crunchyServer()`

Value

Invisibly, the server function. This function is called for the side effect of setting the authorization function globally. The function should return `TRUE` if the current user is authorized.

Examples

```
setCrunchyAuthorization(function (input, output, session) {
    # Restrict to users who have crunch.io emails
    endsWith(email(shinyUser()()), "@crunch.io")
})
```

`shinyDataset` *Load a dataset for a Shiny session*

Description

This function wraps `crunch::loadDataset()` in a `shiny::reactive()` object for use in a Shiny app. It also ensures that the current user is authenticated with Crunch before proceeding.

Usage

```
shinyDataset(...)
```

Arguments

`...`	Arguments passed to `loadDataset`

Value

A Shiny `reactive` object.

Examples

```
## Not run:
shinyServer(function(input, output, session) {
    ds <- shinyDataset("Your dataset name")

    freqs <- reactive({
        fmla <- as.formula(paste("~", input$varname))
        crtabs(fmla, data=ds())
    })
})

## End(Not run)
```

shinyUser *Return Crunch User Information*

Description

This function returns information about the current user of the shiny app. This is useful if you want to change the behavior of the app depending on who is viewing the app. You can access elements of this record with crunch functions like `name()` or `email()`.

Usage

```
shinyUser()
```

Value

A user record if the user is logged in, otherwise a character vector or error

withCrunchyProgress *Display progress from Crunch API processes*

Description

Some potentially large operations, such as imports and exports, report progress in the Crunch API. In an interactive R session, they print a text progress bar. This context, which wraps `shiny::withProgress()`, reports that Crunch API progress up to the Shiny web app.

Usage

```
withCrunchyProgress(expr, ...)
```

Arguments

`expr`	Code to evaluate
`...`	Additional arguments passed to `shiny::withProgress()`

Value

The result of `expr`

Examples

```
## Not run:
withCrunchyProgress(
    ds <- newDataset(df),
    message = "Importing..."
)

## End(Not run)
```

Index