Package 'AnalysisPageServer'

October 17, 2017

Type Package

Title A framework for sharing interactive data and plots from R through the web.

Version 1.10.0

Date 2017-02-17

Author Brad Friedman <friedman.brad@gene.com>, Adrian Nowicki, Hunter Whitney <hunter@hunterwhitney.com>, Matthew Brauer <brauer.matthew@gene.com>

Maintainer Brad Friedman <friedmab@gene.com>

Description AnalysisPageServer is a modular system that enables sharing of customizable R analyses via the web.

License Artistic-2.0

LazyLoad yes

Imports methods, log4r, tools, rjson, Biobase, graph

Suggests RUnit, XML, SVGAnnotation, knitr

Enhances Rook (>= 1.1), fork, FastRWeb, ggplot2

VignetteBuilder knitr

biocViews GUI, Visualization, DataRepresentation

RoxygenNote 5.0.1

NeedsCompilation yes

R topics documented:

`add.event` *add.event*

Description

Add a new event to a registry

Usage

```
add.event(registry, event, overwrite = FALSE)
```

Arguments

`registry`	EventRegistry
`event`	String. Name for the Event.
`overwrite`	Logical. If the Event already exists, should I overwrite it? If TRUE then yes, without warning. If FALSE (default) then no, throw an error.

Details

Add a new event to a registry. If an Event of that name already exists then if `overwrite` is not set then an error is thrown, and if `overwrite` is set then the contents of the old Event are simply replaced. Use `add.handler` to add a handler to an existing Event.

Value

Nothing good.

Author(s)

Brad Friedman

Examples

```
r <- new.event.registry()
has.event(r, "mouseclick")
add.event(r, "mouseclick")
has.event(r, "mouseclick")
```

`add.event.handler` *add.event.handler*

Description

Add a Handler to an Event

Usage

```
add.event.handler(registry, event, handler)
```

Arguments

`registry`	EventRegistry
`event`	String. Name of the Event
`handler`	Function. The new Handler to add to the Event.

Details

A Handler is any function to be called when the event is triggered. If the return value of the Handler has a "CatchEvent" attribute which is TRUE then the event will be caught and not bubble to the next handler, and the "CatchEvent" attribute will be stripped before returning the value to the triggering context.

If the Event does not yet exist an error is thrown.

Value

Nothing good.

Author(s)

Brad Friedman

Examples

```
r <- new.event.registry()
add.event(r, "mouseclick")
add.event.handler(r, "mouseclick", function(x, y)  message("Mouse clicked at coordinates (", x, ", ", y, ")"))
trigger.event(r, "mouseclick", x = 30, y = 50)
```

analysis.link *analysis.link*

Description

Build a URL to run an analysis on the server

Usage

```
analysis.link(page, params = list(), app.base.url, width = 9, height = 7,
  device = "svg", include.plot.params = TRUE)
```

Arguments

page	Name of page
params	List of parameter values (as R objects—this function will encode them). Default: `list()` (no parameters).
app.base.url	Base URL for application. This is usually the prefix in which the app landing HTML page is found.
width	Width parameter for graphics devices. Normally this is in inches, although it depends on exactly how you set up your application. Default: 9. Ignored if `include.plot.params = FALSE`
height	Height parameter for graphics devices. Normally this is in inches, although it depends on exactly how you set up your application. Default: 7. Ignored if `include.plot.params = FALSE`
device	Device to use for plotting. Default: "svg". (This is going to be sent to the server, not run in the same process as this function. It would be ignored if the page was built with `no.plot = TRUE`.) Ignored if `include.plot.params = FALSE`
include.plot.params	Boolean, default TRUE. If TRUE then include the width, height, device and textarea_wrap parameters in the URL. in the URL. Otherwise omit them.

Details

Unlike `analysis.page.link` this is not a URL to open in the web browser, but rather the kind of URL used internally by the front end for its AJAX request to the server to perform an analysis, or retrieve the analyses as web services.

Value

URL

Author(s)

Brad Friedman

analysis.page.link *app.link*

Description

Link into the app in a particular state

Usage

```
analysis.page.link(page, params = list(), submit = TRUE, relative = "")
```

Arguments

page	AnalysisPage object. The name will be extracted from this object and its AnalysisPageParamSet will be used to validate the params argument. Or, you can provide a character string. Then you won't get the parameter value checking for free but you will get the page name.
params	Named list of parameter values. If page is an AnalysisPage then they will be validated using the parameter set of that page.
submit	If submit is true then turn on a flag that says that the analysis should be submitted. Otherwise the link will open the primary parameter area. It is an error to supply params if submit = FALSE (this may be allowed in the future, but right now it is not supported.). Default: TRUE
relative	String. This string will be prepended to the relative URL beginning with "?". Default is empty string, so you would get something like "#page/2way/...". If you gave, for example "http://research.gene.com/expressionplot/app.html" then you would get "http://research.gene.com/expressionplot/app.html#page/2way/...". In RApache context the Global variable SERVER$headers_in$Referer is very useful for this.

Details

It is possible to encode the application state into a URL which will then be executed by the front end. There are 2 parts to the state, and each is supplied as one argument to this function (which then becomes one URL-encoded parameter in the link)

"page" The name of the page within the app, such as "2way"

"params" A subset of parameters and their values, already filled out

Value

A relative URL beginning with "#", or a full URL if relative is provide

Author(s)

Brad Friedman

Examples

```
analysis.page.link("mypage", params=list(foo=1))
```

`analysis.page.of.current.app`

Retrieve an Analysis page from the current app

Description

If the current app (as returned by (`current.app`) has a page of the given name then it is returned. If the current app can't be found, or if it does not have such a page, then NULL is returned.

Usage

```
analysis.page.of.current.app(page)
```

Arguments

`page`	String, name of desired page.

Value

AnalysisPage, or NULl

Author(s)

Brad Friedman

`annotate.analysis.page.svg`

annotate.analysis.page.svg

Description

Annotate an AnalysisPage SVG plot

Usage

```
annotate.analysis.page.svg(svg.filename, x, y, ids, group.lengths = length(x),
  class.name = "plot-point", start = 0, uniquify.ids.suffix = NULL,
  verbose = FALSE)
```

Arguments

`svg.filename`	Path to SVG file
`x`	Vector of x values
`y`	Vector of y values (same length as x)
`ids`	Vector of ids (same length as x)
`group.lengths`	Positive integer vector summing to `x` (an ordered partition of `length(x)`) giving length of contiguous groups of elements. Each integer must be at least 3 or it won't be able to activate that group. The search will be for contiguous elements, then continue where the last one left off, possibly after a gap. Default: `length(x)`, look for a single contiguous block.

class.name	Class name to apply to points (default: "plot-point")
start	This is a 0-based integer. It is an offset of where to start looking for the plot elements. Default is 0, to start at the beginning. The meaning of the index after that is not exposed to the interface, however this function returns numbers that you can then use to "continue looking where you left off".
uniquify.ids.suffix	NULL or a string. If NULL then do not modify the identifiers in the SVG file. If a string, then call uniquify.ids.in.svg.files to modify the "glyph" and "clip" identifiers, using this word as the suffixes parameter).
verbose	Logical. Default, FALSE. Passed through (as integer) to C++

Details

The plot points are found by looking for a sequence of points whose x and y coordinates correlation >= 0.999 with the query x and y vectors.

Then each is tagged with class.name, and with the corresponding ID from the vector.

The file is overwritten.

Value

Integer, the "next start" position, or where to start looking to continue after this stretch. (Invisibly). If no match was found then returns NULL.

Author(s)

Brad Friedman

---

annotate.data.frame *annotate.data.frame*

Description

Clean up and annotate a data frame

Usage

```
annotate.data.frame(obj, required.fields = c("x", "y"), signif.digits = 3)
```

Arguments

obj	data.frame or AnnotatedDataFrame: the return value of a handler.
required.fields	Character vector of required fields. Default: c("x","y"). You could set to character(0), for example, if you don't want to force a check that "x" and "y" be present.
signif.digits	Integer, default 3, giving the number of significant digits to which "numeric" (but not "integer") columns should be rounded, using signif(). NULL means to not round at all.

Details

The obj argument should be a return value from a handler, either a data.frame or an annotated data.frame. If a data.frame then an AnnotatedDataFrame is built. Then three special fields in `varMetadata` are checked: "labelDescription" and "type"

If any is missing then they are built as follows:

labelDescription labelDescription always exists, but sometimes it has NA entries. In those cases it is set to the name of the variable (rowname of the `varMetadata`). This is the one that you most likely might want to set yourself.

type If not present, then it is calculated from the pData like this: `sapply(lapply(pData(obj), is), "[", 1)`. This will become one of "integer", "factor", "logical", "numeric" or "character", and the front end should know how to render these.

Columns that have type "numeric" (but not "integer") are rounded to the given number of significant digits.

Also, this throws an error if "x" or "y" field is missing

Value

AnnotatedDataFrame

Author(s)

Brad Friedman

`apache.httpd.conf` *apache.httpd.conf*

Description

Generate httpd.conf file for RApache deployment

Usage

```
apache.httpd.conf(driver.path, app.location, config.js.path,
  front.end.dir = system.file("htdocs/client/dist-aps", package =
  "AnalysisPageServer"), mod.R.path, skip.checks = FALSE)
```

Arguments

`driver.path`	Path to driver. Must contain call to add.handlers.to.global().
`app.location`	Location from which app will be deployed (e.g. "/myapp" to make the URL start "http://myserver.com/myapp").
`config.js.path`	Path to modified `config.js` file. This would have been generated with a call to `config.js`.
`front.end.dir`	Path to front end directory content. An alias will be set up to serve it directly from this location. Default is from the installed R package (found via `system.file`.
`mod.R.path`	Path to mod_R.so
`skip.checks`	Boolean, default FALSE. If TRUE then don't check for file existence or for the presence of "add.handlers.to.global" in the startup script.

Details

Generate httpd.conf file for RApache deployment. This returns a charvec of lines of the files. You still have to call `writeLines`. See the `ApacheDeployment` vignette for more information.

Value

Charvec

Author(s)

Brad Friedman

`aps.dataset.divs` *aps.dataset.divs*

Description

Generate HTML for multiple DIV elements corresponding to a paths list

Usage

```
aps.dataset.divs(paths.list, show.sidebar = rep(TRUE, length(paths.list)),
  show.table = rep(TRUE, length(paths.list)), num.table.rows = 10,
  extra.html.class = rep(list(character()), length(paths.list)),
  extra.div.attr = rep(list(NULL), length(paths.list)))
```

Arguments

`paths.list`	List of lists. The outer list corresponds to data sets and the inner lists have names in `$plot` and `$data`, giving relative paths to the SVG and JSON files (OK to omit one). Or a list with a `$paths.list` element, which would be used (this lets you pass the return value of `static.analysis.page` directly to this function).
`show.sidebar`	Logical vector of same length as `paths.list` to pass through corresponding elements to `aps.one.dataset.div`. Default: all TRUE.
`show.table`	Logical vector of same length as `paths.list` to pass through corresponding elements to `aps.one.dataset.div`. Default: all TRUE.
`num.table.rows`	Number of table rows to show. Default: 10. Recycled to `length(paths.list)`.
`extra.html.class`	List (of charvecs) of same length as `paths.list` to pass through corresponding elements to `aps.one.dataset.div`. Default: All empty charvec.
`extra.div.attr`	List (of named charvecs or NULLs) of same length as `paths.list` to pass through corresponding elements to `aps.one.dataset.div`. Default: all NULL.

Details

This function is meant to work with the return value of `static.analysis.page` That function returns an object with a `$paths.list` element which contains the relative paths to each of the plots and datasets. You pass that through as the first argument to this function and it will make divs corresponding to those plots. The other arguments are either vectors or lists of corresponding lengths to pass through to `aps.one.dataset.div`.

Value

Charvec of HTML divs corresponding to datasets in `paths.list`.

Author(s)

Brad Friedman

`aps.one.dataset.div` *aps.one.dataset.div*

Description

Create HTML for a div element to contain one AnalysisPageServer data set

Usage

```
aps.one.dataset.div(svg.path = NULL, data.path = NULL,
  show.sidebar = TRUE, show.table = TRUE, num.table.rows = 10,
  extra.html.class = character(), extra.div.attr = NULL)
```

Arguments

`svg.path`	Path (could be relative to index.html) to (annotated) SVG file. NULL to only have data table and no picture.
`data.path`	Path (could be relative to index.html) to JSON file containing data set. NULL to only have SVG and no table
`show.sidebar`	Boolean. If TRUE (default) then show sidebar. If FALSE then omit it.
`show.table`	Boolean. If TRUE (default) then show sidebar. If FALSE then omit it.
`num.table.rows`	Number of table rows to show. Default: 10
`extra.html.class`	Thesee are extra classes to add to the div. This could be used for whatever extended purpose you want, like extra styling or logic. Should be an unnamed charvec. Default is `character()`, just use the basic required for APS.
`extra.div.attr`	These are extra attributes to add to the div. For example you could add an `id` attribute. It should be a named charvec, or NULL (default) to not anything extra beyond that required for APS.

Details

Create HTML for a div element to contain one AnalysisPageServer data set. This function does not created, modify, or even check for existance of the SVG and JSON files. You provide paths and this function just includes those paths, however awful, in the HTML returned.

Value

HTML string

Author(s)

Brad Friedman

See Also

aps.dataset.divs, a convenience wrapper for this function to create multple divs at once.

---

aps.urlEncode *aps.urlEncode*

Description

urlEncode all the strings in a character vector

Usage

```
aps.urlEncode(vec)
```

Arguments

vec	Vector to encode. Will be coerced to character.

Details

The urlEncode function supplied by RApache has a few behaviors I don't expect. First, it requires only character arguments, so `urlEncode(1)` throws an exception. Second, it dies on the empty string.

This function is a wrapper for RApache's `urlEncode` that handles all these cases. It might later be replaced with another implementation, but the interface will stay. (The "aps" stands for `AnalysisPageServer`.)

Value

charvec of same length, with encoded strings

Author(s)

Brad Friedman - Regular

Examples

```
aps.urlEncode(1)
aps.urlEncode("")
aps.urlEncode("foo/bar")
```

array.param *array.param*

Description

Create an Array AnalysisPageParam

Usage

```
array.param(..., prototype, start = 1, min = 0, max = Inf)
```

Arguments

...	Passed through to simple.param. This includes at least "name", optionally "label" and "description". but not "type" (which is set to "array") or "value" (which is set to empty string, but ignored anyway, since the prototype parameter will have its own value).
prototype	A single AnalysisPageParam that is the prototype for each of the elements in the array. Note that while only one param is allowed, it could potentially be either a compound or another array parameter.
start	The starting length of the array that should be rendered
min	The minimum allowed length of the array. Buttons to remove elements should be de-activated below this level (default 0).
max	The maximum allowed length of the array. Buttons to add elements should be de-activated above this level (default Inf).

Details

An array AnalysisPageParam is a way of having a single parameter with multiple repetitions of some other (fixed) parameter type. The starting length, as well as minimum and maximum allowable lengths, are provided. If min != max then the front end should render some widget to add/remove elements.

By combining with compound.param a fairly complex data structure can now be specified.

Value

AnalysisPageParam of type "array"

Author(s)

Brad Friedman

Examples

```
one.gene <- simple.param(name="gene", label="Gene Symbol")
gene.set <- array.param(name="geneset", prototype=one.gene)
```

`autosignal.on.bloated.memory` *autosignal.on.bloated.memory*

Description

Send signal to self on BloatedMemory

Usage

```
autosignal.on.bloated.memory(events, signal = tools::SIGUSR1,
  pid = Sys.getpid(), logger = NULL, event.name = "BloatedMemory")
```

Arguments

`events`	EventRegistry object
`signal`	Signal to throw. Default is SIGUSR1 (defined in tools package). Normally this elicits similar behavior to SIGINT however when running inside Apache it lets the worker process finish the current request before killing itself.
`pid`	Process ID to which the signal should be sent. Default: `Sys.getpid()`, the current process. (If you provide something else it is no longer really an `autosignal`, just a `signal`).
`logger`	log4r object, optional
`event.name`	Name of event to listen for. Default: "BloatedMemory"

Details

Have the process send a signal to itself when BloatedMemory event is triggered.

Value

Nothing

Note

This attaches a listener—it does not actaully do anything until the BloatedMemory event is triggered, if ever.

Author(s)

Brad Friedman

`bind.memory.checker` *bind.*

Description

Bind the memory checker to the FinishAnalysis event

Usage

```
bind.memory.checker(app, max.mb, app.event = "FinishAnalysis",
  memory.event = "BloatedMemory")
```

Arguments

`app`	AnalysisPageRApacheApp
`max.mb`	Memory threshold for triggering BloatedMemory, in Megabytes. Required.
`app.event`	Name of existing event on which the new memory check listener should be registered. Default: "FinishAnalysis".
`memory.event`	Name of event to fire in case of memory usage above threshold. Default: "BloatedMemory".

Details

You supply an AnalysisPageRApacheApp. It also has a FinishAnalysis event. I add a new event (by default called BloatedMemory), and also a listener for FinishAnalysis, which calls `check.memory` each time `FinishAnalysis` is triggered. (this will then trigger BloatedMemory if memory usage is above threshold). Although the BloatedMemory event would now be triggered, unless a listener is attached to *it* nothing special will happen. (See `autosignal.on.bloated.memory` for this.)

Value

Nothing of note.

Author(s)

Brad Friedman

See Also

`check.memory`, `autosignal.on.bloated.memory`

bool.param *bool.param*

Description

Build a boolean AnalysisPageParam

Usage

```
bool.param(...)
```

Arguments

...	Passed through to simple.param. This includes at least "name", and optionally "label", "description" and "value" (which should be either TRUE or FALSE), but not "type".

Details

Build a boolean AnalysisPageParam. This is probably rendered as a checkbox. The value returned to the server should be either JSON "true" (corresponding to checked) or JSON "false".

If you do not provide a value (or if you provide value="", which is what happens in the parent constructor simple.param when you don't provide a value) then the default will be FALSE.

Value

An AnalysisPageParam

Author(s)

Brad Friedman

Examples

```
show.ids <- bool.param("show_ids", label="Show IDs", description="Show sample IDs on the plot", value=TRUE)
```

build.service *build.service*

Description

Build an AnalysisPage service

Usage

```
build.service(handler, param.set = default.service.paramset(handler),
  annotate.plot = FALSE, annotate.data.frame = FALSE, no.plot = TRUE,
  service = TRUE, skip.checks = TRUE, ...)
```

Arguments

handler	The handler function to convert
param.set	AnalysisPageParamSet for the handler. Default: default.service.paramset(handler)
annotate.plot	Default: FALSE
annotate.data.frame	Default: FALSE
no.plot	Default: TRUE
service	Default: TRUE
skip.checks	Default: TRUE. This is passed through to new.analysis.page and should normally not be modified. (we don't check services because they are not required to provide default arguments.)
...	Further arguments to pass to new.analysis.page

Details

Convert a functions into an AnalysisPage service. An AnalysisPage service is an AnalysisPage with the service flag set. That means that pages() will not return it, so it will not be directly available through the front end. The reason to do this would normally be so that it can be used to populate a combobox or other part of the website. In practice it also means that the return value will probably not use the AnalysisPageDataNode system—it is free to return some arbitrary JSON string or other text.

It is a wrapper for new.analysis.page, and passes all of its arguments through (but with different defaults now)

Value

AnalysisPage

Author(s)

Brad Friedman

See Also

new.analysis.page

Examples

```
poem.file <- system.file("examples/in-a-station-of-the-metro.html", package="AnalysisPageServer")
poem.html <- readLines(poem.file, warn = FALSE)
poem <- build.service(function()  {
  new.response(paste0(poem.html, "\n"),
  content.type = "text/html")
}, name = "poem")
```

check.memory *check.memory*

Description

Check memory usage and trigger an event if it exceeds some threshold.

Usage

```
check.memory(events, max.mb, logger = NULL, event.name = "BloatedMemory")
```

Arguments

events	EventRegistry object
max.mb	If the total "used" memory (Ncells + Vcells) is at least this much (in Mb) then trigger the next handler.
logger	log4r object. If non-NULL then memory usage is printed there with info(). Default: NULL, don't log.
event.name	Name of the event to trigger. Default: "BloatedMemory". It is passed used.mb and max.mb arguments

Details

Call gc() to check memory, possibly print the result to STDERR, then possibly call the next handler you supply if too much memory is being used.

Value

Nothing

Author(s)

Brad Friedman

check.same.svgs *check.same.svgs*

Description

Test that 2 SVG files have the same content

Usage

```
check.same.svgs(got.lines, exp.lines, ...)
```

Arguments

got.lines	Charvec of the lines of the SVG to test
exp.lines	Charvec of the lines of the reference SVG
...	Passed through to checkEquals (such as test name).

Details

Test that 2 SVG files have the same content. Most differences in whitespace are ignored, as are all "id", "class" and "type" tags.

Value

As checkEquals

Author(s)

Brad Friedman

`check.signal` *check.signal*

Description

Check if an expression results in a signal being delivered

Usage

```
check.signal(expr, signo, testname, no.signal = FALSE)
```

Arguments

`expr`	The expression to evaluate
`signo`	The signal number (consider using constants like SIGUSR1 from the tools package).
`testname`	Name for this test. Default is to build from signo argument.
`no.signal`	Logical, to invert the sense of the test. Default, FALSE, means to test that the signal was delivered. TRUE means to test that the signal was not delivered.

Details

Check if an expression results in a signal being delivered. The signal will be caught: you can safely deliver a signal such as SIGUSR1 that would normally cause the process to die.

Value

The value of the evaluated expression, invisibly, so you can do more testing if desired.

Author(s)

Brad Friedman - Regular

checkPackageInstalled *Checks if a given package is installed.*

Description

Checks if a given package is installed.

Usage

```
checkPackageInstalled(pkg, version = "0.0.0", required = FALSE)
```

Arguments

pkg	A character string containing a package name.
version	A minimum version number. Default: "0.0.0" (no version requirement)
required	A boolean. The functions stops if set to TRUE and if the required package is not present. Default is FALSE.

Value

A boolean.

Author(s)

Cory Barr

Examples

```
checkPackageInstalled("AnalysisPageServer")
```

checkRookForkForVignettes
checkRookForkForVignettes

Description

Check availability of Rook and fork for vignettes

Usage

```
checkRookForkForVignettes(rookforkOK)
```

Arguments

rookforkOK	Provide FALSE here if you want to simulate not having valid installed copies of rook/fork, without actually having to delete them. Normally you should not supply this argument

Details

Check availability of Rook and fork for vignettes. This function is only meant to be called at the top of the ExampleServers.Rmd and InteractiveApps.Rmd vignettes. It checks that the user has Rook >= 1.1 and fork installed. If not, then it emits a useful message about how to install it, and that the vignette will not build with all features. It also writes functions `kill.process`, `readLines` (yes! be careful!), `startRookAnalysisPageServer` and `new.rook.analysis.page.app` in the Global namespace to avoid calling the real functuons and instead just emit a short message that Rook/fork are not available. The message is also available in the global variable `no.rook.fork.msg`.

Really, you shouldn't use this function except if you are writing a new vignette.

Value

TRUE if Rook (>= 1.1) and fork are available, otherwise FALSE.

Note

Are you sure you really want to use this function? Probably not, unless you are calling it from the top of a new vignette. Otherwise your are really asking for trouble messing up your global namespace. See Details.

Why are you still here? Didn't I tell you not to call this function?

Author(s)

Brad Friedman

`clear.event.handlers` *clear.event.handlers*

Description

Clear the Handlers list for one Event

Usage

```
clear.event.handlers(registry, event)
```

Arguments

`registry`	EventRegistry
`event`	String. Name of the Event.

Details

Clear the Handlers list for one Event. Does not remove the Event from the EventRegistry.

Value

Nothing good.

Author(s)

Brad Friedman

Examples

```
r <- new.event.registry()
add.event(r, "mouseclick")
add.event.handler(r, "mouseclick", function(x, y)  message("Mouse clicked at coordinates (", x, ", ", y, ")"))
trigger.event(r, "mouseclick", x = 30, y = 50)
clear.event.handlers(r, "mouseclick")
trigger.event(r, "mouseclick", x = 30, y = 50)
```

`client.ip` *client.ip*

Description

Return client IP address

Usage

```
client.ip()
```

Details

This is the "X-Forwarded-For" header, if available, and otherwise the "remote_ip" component of the global SERVER variable

Value

`SERVER$remote_ip`

Author(s)

Brad Friedman

Examples

```
SERVER <<- list(remote_ip = "127.0.0.1")
client.ip()
```

`combobox.param` *combobox.param*

Description

Build a combobox AnalysisPageParam

Usage

```
combobox.param(name, ..., uri, dependent.params, prompt = "Enter search term",
  n.param = NULL, allow.multiple = FALSE, response.type = "simple",
  persistent = NULL, extra.persistent.dependencies = NULL, delay.ms = 0)
```

Arguments

name	Name of form element
...	Passed through to simple.param. This includes at least "name", optionally "label", "description" and "value", but not "type".
uri	URI, possibly with :-prefixed parameter names. For example "/get?x=:x;y=:y" has parameters "x" and "y". (See dependent.params next)
dependent.params	A character vector whose names are parameters from the uri, and whose values are the names of other form elements.
prompt	A prompt to display in disabled style before user starts typing (for self-dependent comboboxes only). Default is "Enter search term".
n.param	The name of a parameter that controls the maximum number of search hits, if the URL has such a parameter. Default is NULL, which means it does not have a parameter. If it does have such a parameter, then 0 means to return all hits, and otherwise a positive integer will limit the number of hits returned.
allow.multiple	If TRUE then render search hits as checkbox group and allow multiple selections. (The function will be provided a vector of all selected values.) If the search term changes then the old values are still accumulated. If other dependent parameter changes then they are reset. Default: FALSE
response.type	A string. "simple" means that the response will be a simple array of strings. The "id-long_name-reason" type is a special type for search hits, where an array of hashes is returned, each hash having "id", "long_name" and "reason" components.
persistent	Character or NULL. If non-NULL then it is passed to the front-end. It names a variable in persistent storage that should be used to initialize the value of the parameter. The front end will provide some mechanism to change the persistent value, but until the user does so the param will be initialized from the value in the persistent space.
extra.persistent.dependencies	Character vector specifying names of other parameters on which this one is conditionally persistent. See arg persistent.dependencies in simple.param. Whatever is provided in extra.persistent.dependencies is taken *in addition to* with unname(dependent.params). This is because if a combobox is persistent then it must be so conditional on its dependent parameters. It is an error to specify this for a non-persistent parameter.
delay.ms	Delay, in milliseconds, that the front-end should wait after keystrokes or paste before submitting queries. Default, 0, means no delay.

Details

Build a combobox AnalysisPageParam. This is a widget with both text and drop-down. However, the values in the drop-down depend on an AJAX query whose URI is built from current form element values, possibly including the current widget (namely the text typed so far). The drop-down values should be updated whenever one of the depenedent elements changes.

Value

An AnalysisPageParam

Author(s)

Brad Friedman

Examples

```
## Note the :query parameter is dependent on the same gene element. This makes it a type-ahead query.
gene <- combobox.param(name="gene", uri="/find_gene_id/:genome/:query/", dependent.params=c(genome="genome
```

`compound.param` *compound.param*

Description

Create a Compound AnalysisPageParam

Usage

```
compound.param(..., children)
```

Arguments

`...`	Passed through to `simple.param`. This includes at least "name", optionally "label" and "description". but not "type" (which is set to "compound") or "value" (which is set to empty string, but ignored anyway, since each of the contained parameters will have its own value).
`children`	AnalysisPageParamSet An AnalysisPageParamSet representing all of the nested parameters

Details

A compound AnalysisPageParam is a single parameter that has multiple parts. The parts are represented by an AnalysisPageParamSet, so could be arbitrarily nested. The front end is responsible for wrapping up all of the values in a JSON hash and passing in a single value.

This can be thought of as a way of building a hash out of other parameter types.

Value

AnalysisPageParam of type "compound"

Author(s)

Brad Friedman

Examples

```
plist <- list(simple.param(name="study"), simple.param(name="comp"), simple.param("feature.type", value="g
 comp <- compound.param(name="comp", label="Comparison", children=param.set(plist))
```

config.js *config.js*

Description

Build Javascript configuration

Usage

```
config.js(app.prefix = "/custom/RAPS", client.r.url = file.path(app.prefix,
  "R"), client.rest.url = "",
  template.file = system.file("config-template.js", package =
  "AnalysisPageServer"), static = FALSE, parameter.collection.url = if
  (static) "" else "params", page.collection.url = if (static) "" else
  "pages")
```

Arguments

app.prefix	Prefix for the path to your application. This will be used as the value for the "history.root" parameter in the Javascript file, and also to build the default client.r.url.
client.r.url	Location of R resources. Default: file.path(app.prefix, "R").
client.rest.url	Location of Sloth REST resources. Default: "". This is a poorly documented feature that most people should ignore.
template.file	Path to template file for config.js. Default is taken from "inst" directory of AnalysisPageServer package.
static	Boolean, default FALSE. Controls the default values for parameter.collection.url and page.collection.url.
parameter.collection.url	Default: If static = TRUE then "" else "params"
page.collection.url	Default: If static = TRUE then "" else "pages"

Details

Build Javascript configuration. This function returns Javascript which can be used as the config.js file for the front-end client.

The only reason to call this directly would be to set up specialized deployments.

Value

Charvec of Javascript

Author(s)

Brad Friedman

copy.front.end *copy.front.end*

Description

Copy the APS front end (HTML, CSS, JS, etc) to a web directory

Usage

```
copy.front.end(outdir, client.basedir = system.file("htdocs/client/dist-apss",
  package = "AnalysisPageServer"), include.landing.page = TRUE, ...)
```

Arguments

outdir	Target directory. This directory will contain your index.html file.
client.basedir	Path to client files. Default: system.file("htdocs/client/dist-apss", package = "AnalysisP Probably should not be modified except during development work on the client.
include.landing.page	Boolean. Should I include the landing page "analysis-page-server-static.html"? Default: TRUE
...	Passed through to file.copy, such as overwrite = TRUE

Details

This makes a copy of the complete APS *static* front end (HTML, CSS, JS, etc) to a web directory.

Value

Whatever file.copy returns.

Author(s)

Brad Friedman

Examples

```
message("See vignette embedding.html")
```

current.app *Return the currently running AnalysisPage app*

Description

Return the currently running AnalysisPage app. The way this is done is to first try to chase up the call stack and find the first environment which is an AnalysisPageRApacheApp, and return that. If that fails then it looks for app in the GlobalEnv and returns that. IF that also fails then it reutrns NULL.

Usage

```
current.app()
```

Value

Current AnalysisPageRApacheApp, or NULL if it can't be found.

Author(s)

Brad Friedman

---

custom.body.attr *custom.body.attr*

Description

Return custom attributes required for body element

Usage

```
custom.body.attr()
```

Details

This attribute must be included in the <body> element.

Value

Name charvec of attributes for body

Author(s)

Brad Friedman

---

custom.body.html *custom.body.html*

Description

Generate a <body> HTML line including attributes for APS

Usage

```
custom.body.html(extra.attr = NULL)
```

Arguments

extra.attr	Other attributes, provided in a named charvec.

Details

Generate a <body> HTML line including attributes for APS. Your <body> element must have the special attribute returned by the `custom.body.attr()`. This function makes a line of HTML code containing that, and any other attributes you want to include. It just opens the <body> element, but does not close it.

Value

One line of HTML with a <body> element opening tag.

Author(s)

Brad Friedman

`custom.html.headers` *custom.html.headers*

Description

Generate HTML for custom headers to load AnalysisPageServer CSS and viewport

Usage

```
custom.html.headers(libbase.prefix = "",
  viewport = "width=device-width, initial-scale=1.0",
  stylesheets = default.stylesheets(), ep.svg.styles = NULL)
```

Arguments

`libbase.prefix`	Prefix where your shared CSS files will be located. Default "" will be relative to the index.html file. Otherwise you'll need to end with a "/".
`viewport`	Default: "width=device-width, initial-scale=1.0". This will be used in a `<meta name="viewport">` tag.
`stylesheets`	Charvec of stylesheets to load. Default is `default.stylesheets()`.
`ep.svg.styles`	ep-svg-styles stylesheet. Default: NULL.

Details

Generate HTML for custom headers to load AnalysisPageServer CSS and viewport. To be honest I don't understand how all this works. The main thing is that it you put this stuff up top, in the header section. The only argument you should consider touching is `libbase.prefix`, if you are going to put your common libraries in a shared area instead making a copy next to each dataset.

Value

HTML string to be included in <head> section.

Author(s)

Brad Friedman

data.frame.to.json *data.frame.to.json*

Description

Create a JSON representation of a data.frame

Usage

```
data.frame.to.json(df)
```

Arguments

df	data.frame to represent as a JSON

Details

We represent a data.frame as an hash of hashes. Factors are first coerced into characters.

The outer hash is keyed by the rownames of your data.frame The inner hash is keyed by the colnames of your data.frame

Value

JSON string

Author(s)

Brad Friedman

data.frame.to.list *data.frame.to.json*

Description

Create a list representation of a data.frame

Usage

```
data.frame.to.list(df)
```

Arguments

df	data.frame to represent as a list

Details

We represent a data.frame as an hash of hashes. Factors are first coerced into characters.

The outer hash is keyed by the rownames of your data.frame The inner hash is keyed by the colnames of your data.frame

Value

list

Author(s)

Brad Friedman

Examples

```
df <- data.frame(A=1:3, B=3:1, C=factor(c("foo","bar","foo")), row.names = c("one", "two", "three"))
## Should give the following
## list(one=list(A=1, B=3, C="foo"),
##      two=list(A=2, B=2, C="bar"),
##      three=list(A=3, B=1, C="foo"))
data.frame.to.list(df)
```

---

default.param *default.param*

Description

Build a default AnalysisPageParam for one argument

Usage

```
default.param(name, prototype, ...)
```

Arguments

name	Name of the parameter
prototype	Default value on which the parameter should be built.
...	Further arguments passed to the constructor for the appropriate parameter type. For example, you can include label, description, advanced and show.if.

Details

You provide the name of the argument and default and I build an AnalysisPageParam. The magic here is as follows:

1. Named lists become compound.param, with default.param() then called recursively.
2. Unnamed lists of become array.param, with default.param() called on the first element of the list to build the prototype. The length of the list is taken as the start value. min/max default to 0/Inf. (Advanced Note: the name is copied to the sub-element)
3. Vectors of length > 1 become select.params
4. Vectors of length 0 or 1 and NULLs become simple.params

On any other type of argument it throws an error.

Value

AnalysisPageParam

Author(s)

Brad Friedman

Examples

```
default.param(name = "word", prototype = c("foo", "bar", "baz"), label = "Choose a word")
```

`default.param.set` *default.param.set*

Description

Build a basic ParamSet for your handler

Usage

```
default.param.set(handler)
```

Arguments

`handler`	A function. (Typically one you are using as an AnalysisPage handler)

Details

Each argument to your handler is rendered into a simple AnalysisPageParam with the name of the argument and type "text". The idea is that you will then modify it as necessary to get more complicated widgets.

Value

AnalysisPageParamSet

Author(s)

Brad Friedman

Examples

```
f <- function(A=1, B=2) {}
# param set with 2 form elements rendered as text inputs; something like  A [__________]   B [__________]
pset <- default.param.set(f)
```

default.service.paramset *default.service.paramset*

Description

Create an AnalysisPageParamSet for a service handler

Usage

```
default.service.paramset(handler)
```

Arguments

handler	Handler function of service

Details

The services are not required to supply default values since they don't have to be rendered as parameters. AnalysisPageServer still wants to have a param set. It can be used, for example, to check arguments when building URLs. So we artifically provide default values of 0 for everything, so you get a bunch of simple params.

Value

AnalysisPageParamSet

Author(s)

Brad Friedman

default.stylesheets *default.stylesheets*

Description

Default stylesheets for HTML headers

Usage

```
default.stylesheets()
```

Details

Default stylesheets for HTML headers

Value

character vector

Author(s)

Brad Friedman

dieIfWindows *Throw an error if platform is windows*

Description

Throw an error if platform is windows

Usage

```
dieIfWindows(caller = as.character(sys.call(-1))[1],
  errMesg = paste0(caller, "() does not run under windows"),
  isWindows = platformIsWindows())
```

Arguments

caller	Name of calling function. Default: as.character(sys.call(-1))[1]
errMesg	Error message to throw under windows. Default: paste0(caller, "() does not run under windo
isWindows	Boolean, indicating if current platform is windows. Default: platformIsWindows().

Value

Nothing

dies.ok *dies.ok*

Description

Test that an expression throws an error

Usage

```
dies.ok(call, regex, testname)
```

Arguments

call	An expression to evaluate
regex	A regular expression to match the error against. If omitted then don't test the exception text.
testname	A name for the test. Defaults to deparsing the call.

Details

Test that an expression throws an error.

Value

Runs one or two tests (the second test to match the error message against regex, if it was provided and if an error was successfully thrown).

Author(s)

Brad Friedman

Examples

```
dies.ok(stop("foo"), "foo", "it stops")
```

`embed.APS.dataset` *embed.APS.dataset*

Description

Embed an APS dataset

Usage

```
embed.APS.dataset(plot, df, title, show.sidebar = TRUE, show.table = TRUE,
  num.table.rows = 10, extra.html.classes = character(),
  extra.div.attr = character(), svg.args = list(), eval.args = list(envir
  = parent.frame()), outdir = get.APS.outdir(), randomize.filename = TRUE,
  ...)
```

Arguments

`plot`	If present, then either an expression, a function, or a path to SVG file (not yet annotated). If an then the expression will be evaluated after opening a plotting device. The expression will be evaluated in the calling frame, so your local variables will be accessible, but this can be changed by modifying `eval.args`. If a function, then the function will be called with no arguments. In that case you would control the context yourself by setting the function's environment. If path to an SVG then you would have already made the plot, and that would be used. If missing then no plot is drawn—only the table is shown.
`df`	data.frame of data. If omitted, then the return value of evaluating the plotting expression or function is used (if `plot` is not a character).
`title`	Caption for plot
`show.sidebar`	Boolean, default TRUE. Set to FALSE to not show the sidebar (filtering, tagging). (This is passed through directly to `aps.dataset.divs`.)
`show.table`	Boolean, default TRUE. Set to FALSE to not show the data table (still available on download. (This is passed through directly to `aps.dataset.divs`.)
`num.table.rows`	Number of table rows to show. Default: 10 (This is passed through directly to `aps.dataset.divs`.)
`extra.html.classes`	Charvec of extra HTML classes to include in the div. (This is embedded in a list then passed through directly to `aps.dataset.divs`.)
`extra.div.attr`	Names charvec of extra attributes to include in the div. (This is embedded in a list then passed through directly to `aps.dataset.divs`.)
`svg.args`	Arguments (other than filename) to pass to the `svg` function. This should be a named list. In particular, consider something like `list(width = 8, height=5)` to change the aspect ratio.

eval.args	Arguments to pass to evalq when evaluating your plot code. Ignored if plot is character or a function. Otherwise it should be a named list. Default is list(envir = parent.frame()), which means the evaluation will happen in the calling frame.
outdir	Output directory. Default: get.APS.outdir(), which is either "." or the directory of your knit2html target .html file.
randomize.filename	Passed through to static.analysis.page (but here the default is TRUE).
...	Passed through to static.analysis.page. ot overwrite, outdir, or write.client

Details

This function is meant to be called in a knitr document that is being knit with knit2html. It makes a few assumptions that are valid in that context.

It makes a call to static.analysis.page for you to annotate and write the SVG and JSON files, then emits the element to STDOUT. outdir defaults to ".". It only does one plot/dataset at a time.

Value

Returns the div, invisibly.

Author(s)

Brad Friedman

Examples

```
message("See vignette embedding.html")
```

---

encode.datanode *JSON-Encode an AnalysisPageDataNode for the front end*

Description

JSON-Encode an AnalysisPageDataNode for the front end. This just calls toJSON, but before doing so it makes sure that $warnings will be sent as an array.

Usage

```
encode.datanode(datanode)
```

Arguments

datanode	AnalysisPageDataNode or other object

Details

Mostly this function is only called once, from the $analysis method of an AnalysisPageRApacheApp.

Value

JSON-encoded string

Author(s)

Brad Friedman

`eval.within.time` *eval.within.time*

Description

Evaluate an R expression in a fork within a given time frame

Usage

```
eval.within.time(expr, secs, dsecs = c(0.001, 0.1), time = as.difftime(secs,
  units = "secs"), verbose = FALSE, write.obj = saveRDS,
  read.obj = readRDS, make.con = tempfile, cleanup.con = function(con) if
  (file.exists(con)) unlink(con), touch.con = function(con)
  writeLines(character(), con), con.touched = file.exists,
  make.signal.con = make.con, cleanup.signal.con = cleanup.con)
```

Arguments

`expr`	Expression to evaluate
`secs`	Seconds to timeout. Example: 10. Ignored if `time` is provied.
`dsecs`	Seconds for parent process to sleep between checking the child process. It will be recycled to length 2. The first interval will be the first element, then each time it will wait twice as long until the interval is at least as long as the second element, then it will wait the second elemtn. This keep the ratio of time required to run and time actually taken to run close to 1 without having excessive checking for longer processes. Default: c(0.001, 0.1) (seconds).
`time`	`difftime` object giving the timeout interval. Default: `as.difftime(secs, units = "secs")`, which simply means to build a `difftime` object from the `secs` argument. If this argument is provided then `secs` is ignored.
`verbose`	Boolean, default FALSE. If TRUE then emit messages with process IDs, etc.
`write.obj`	Function to serialize and write the resulting R object to the connection. First argument is the object and second is the connection. Default: `saveRDS`.
`read.obj`	Function to read and deserialize the resulting R object from the connection. Argument is the connection. Default: `readRDS`.
`make.con`	A function to make the connection for communication between child and parent. The function will be called once with no arguments. The child will then write to it with saveRDS and the parent will read from it with readRDS. Default: `tempfile`.
`cleanup.con`	A function to clean up a conncetion. Default: `function(con) if(file.exists(con)) unlink(con`
`touch.con`	A function to "touch" a connection. Default: `function(con) writeLines(character(), con)`. This is used to signal through the signal file.

`con.touched` A predicate to check if the connection has been touched. Default: `file.exists`.

`make.signal.con`
Same as `make.con`, but for the signal file. Default: `make.con`.

`cleanup.signal.con`
Like `cleanup.con`, but for the signal file. Default: `cleanup.con`.

Details

The expression is evaluated in a child process while the parent process waits up to the given time interval. If the child process finishes quickly enough it will signal to the parent process to wake up and return a particular value. If the time interval elapses before the child process finishes then the parent wakes up anyway and kills the child, then throws an error. (You may want to wrap this function in a `tryCatch` block to handle the error gracefully.)

The implementation uses the `fork` package, which is loaded—an error is thrown if unavailable. In fact, the parent sleeps for short intervals (controlled by `dsecs` param), each time waking up to check if either the time has elapsed or the child has finished, then acting accordingly.

The way the child signals to the parent is via the filesystem. There are two such files: the result file and the signal file. The child writes the result of the calculation to disk as a serialized R object. Usually you should try to keep this small. Then the child touches a second file, called the "signal" file, which signals that it is finished. Both of these are temporary files. While in loop, the parent checks for existence of the signal file. After exiting the loop, the parent reads the result file. An attempt is made to delete both files before returning or throwing an error.

The child process evalutes your expression within a try block. If this evaluation results in an error, then the captured error object is passed to the parent, which then throws it again.

It is possible that the child would *start* writing the result but not finish before the time elapses. That would be considered a timeout. The thing which the parent checks is if the signal file exists.

Don't be intimidated by the large number of arguments. Typically usage involves only the first two.

Value

The result of evaluating `expr`

Author(s)

Brad Friedman

`event.names` *event.names*

Description

Get vector of names of all existing Events.

Usage

```
event.names(registry)
```

Arguments

`registry` EventRegistry

Details

Get vector of names of all existing Events.

Value

Charvec

Author(s)

Brad Friedman

Examples

```
r <- new.event.registry()
event.names(r)
add.event(r, "mouseclick")
event.names(r)
```

---

execute.handler *execute.handler*

Description

Execute the handler

Usage

```
execute.handler(analysis.page, params, plot.file, file.params = list(),
  device = svg, annotate.plot = analysis.page$annotate.plot,
  max.annotated.regions = 5000, logger = create.logger(stderr(),
  log4r:::FATAL + 1))
```

Arguments

analysis.page	AnalysisPage object
params	Named list of parameters. These can include arguments to svg and arguments to the handler function. If there are any extra arguments then an error is thrown.
plot.file	Path to file to create. Should not exist already.
file.params	Named list of parameters (but defaults to empty list). These will be passed through as-is and should correspond to FILE uploads (being length-2 lists with $name and $tmp_name elements).
device	The plotting device function to use. Default: svg. You might specify png instead (you are passing the actual function here, not its name).
annotate.plot	Logical, indicating whether I should try to annotate the SVG plot. (If you aren't using the SVG device then this should be set to FALSE to not waste time trying to annotate the plot.) Default: analysis.page$annotate.plot
max.annotated.regions	Integer. If the handler returns more than this many regions then do not try to annotate them in the plot. Default: 5000
logger	log4r object. Default: no logging (FATAL + 1)

Details

`execute.handler` executes the plot function in the handler based on the parameter list, checks that the output is valid, adds the SVG attributes to the plot, and returns an AnnotatedDataFrame.

All of the parameters in the parameter list are JSON decoded. Even though this is really just extra work for the scalar parameters, we do it because otherwise it is confusing who needs to be de/encoded and who doesn't.

It is OK if your handler doesn't turn off the device when it's done. This wrapper will check if the current device hasn't changed. If so, it will call `dev.off`. This is useful because then you can use the same function in an interactive session, and also saves you one line of code. It's also OK if your handler *does* turn off the device. Then the current device will have decreased and the wrapper will known not to call dev.off again.

It is also OK if your handler returns a data.frame instead of an AnnotatedDataFrame. It just has to have `x`, `y`. An AnnotatedDataFrame will be built The interpretation of the fields in the AnnotatedDataFrame depend on your front end, but the guidelines are like this:

`type` "text", "numeric" or "none", to set sorting and filtering options.

`labelDescription` A display name for the column, instead of showing the actual name.

If `$no.plot` is true then the plotting device won't be opened or closed, and of course the plot won't be annotated.

If annotate.data.frame is set then your data.frame is converted to an AnnotatedDataFrame and your AnnotatedDataFrame is converted to an AnalysisPageDataNode of "table" type automatically.

Value

AnnotatedDataFrame, but throws error if the handler is not making a plot, or is returning invalid data.

Author(s)

Brad Friedman

See Also

`new.analysis.page`

Examples

```
page <- new.analysis.page(AnalysisPageServer:::sine.handler)
plot.file <- tempfile(fileext = ".svg")
plist <- lapply(list(xmin=-2*pi, xmax=2*pi, n= 50), rjson::toJSON)
sine.data <- AnalysisPageServer:::execute.handler(page, plist, plot.file=plot.file)
# now sine.data is an AnnotatedDataFrame
```

file.param *file.param*

Description

Build a file upload AnalysisPageParam

Usage

```
file.param(..., template.uri = NULL, dependent.params = NULL)
```

Arguments

...	Pass through to simple.param, including at least "name" but not including "type".
template.uri	URI, possibly with :-prefixed parameter names. For example "/get?x=:x;y=:y" has parameters "x" and "y". (See dependent.params next). (Note: this follows the way of doing it in combobox.param). This is optional. If provided, then the front-end can use this callback to allow the user to download a template. This is a template in two senses: the URI itself may be a template whose parameter values should be filled in, and the return value of the request is an excel file which is a template for the user to fill in.
dependent.params	A character vector whose names are parameters from the uri, and whose values are the names of other form elements. Error to provide this without providing template.uri

Details

Build a file upload AnalysisPageParam. This is rendered as a file upload element, to be uploaded along with the submission. (Currently there is no server-side mechanism for storing uploaded files, so asynchronous upload is not possible.) The description field should describe what type of file is expected.

On the server side your handler will get a list with "name", "tmp_name" elements giving the filename, and path to a local file (usually in /tmp)

Value

An AnalysisPageParam

Author(s)

Brad Friedman

Examples

```
cov.param <- file.param("cov", label="Covariate Data", description="A two-column Excel file, first column b
```

get.APS.outdir *Get current AnalysisPageServer output directory*

Description

Get current AnalysisPageServer output directory

Usage

```
get.APS.outdir()
```

Value

Path

Author(s)

Brad Friedman

See Also

set.APS.outdir, reset.APS.outdir

Examples

```
set.APS.outdir("/some/path")
get.APS.outdir()
reset.APS.outdir()
```

get.page *get.page*

Description

Return a registered function

Usage

```
get.page(registry, page.name)

## S3 method for class 'AnalysisPageRegistry'
get.page(registry, page.name)
```

Arguments

registry	AnalysisPageRegistry object
page.name	Name of the registered function

Details

Return a registered function

Value

The registered function. Stops if no such function is registerd

Author(s)

Brad Friedman

See Also

new.registry, register.page, has.page, pages

Examples

```
example(register.page, ask=FALSE)  # register the sine page
get.page(registry, "sine")         # should return the sine.handler function
```

`getCustomContent` *Functions to manage custom content other aspects of the request-specific environment*

Description

Custom content are HTML rendered as additional accordion sections. From the data structure point of view these are represented as a named of list of character vectors. The names are the section headers. Use appendCustomContent to add more content.

appendCustomContent adds custom content to be rendered in separate accordion section

clearRequestEnv clears the environment associated with the last request.

Usage

```
getCustomContent()

appendCustomContent(sectionName, content)

clearRequestEnv()
```

Arguments

sectionName	Name of section (string)
content	Character vector of HTML content to append

Value

getCustomContent returns named list of character vectors

appendCustomContent does not return anything good.

clearRequestEnv does not return anything useful

Author(s)

Brad Friedman

Examples

```
appendCustomContent(sectionName = "foo", content = c("<i>bar</i><br>","<b>baz</b>"))
getCustomContent()
clearRequestEnv()
```

`getTraceback` *getTraceback*

Description

Get traceback from tryKeepTraceback()

Usage

```
getTraceback(mto)
```

Arguments

`mto`	An object of the try-error class

Value

Traceback as a string

Examples

```
x <- tryKeepTraceback(stop("no way"))
if(is(x, "try-error"))  cat(getTraceback(x))
```

`has.event` *has.event*

Description

Predicate to test if an EventRegistry has an Event of a given name

Usage

```
has.event(registry, event)
```

Arguments

`registry`	EventRegistry
`event`	String. Name of the Event.

Details

Predicate to test if an EventRegistry has an Event of a given name

Value

Logical

Author(s)

Brad Friedman

Examples

```
r <- new.event.registry()
has.event(r, "mouseclick")
add.event(r, "mouseclick")
has.event(r, "mouseclick")
```

`has.page` *has.page*

Description

Predicate to test if some page is already registered under a name

Usage

```
has.page(registry, page.name)

## S3 method for class 'AnalysisPageRegistry'
has.page(registry, page.name)
```

Arguments

`registry`	Registry
`page.name`	An AnalysisPageRegistry

Details

Predicate to test if some page is already registered under a name

Value

Logical, indicating if a page is already registered under than name

Author(s)

Brad Friedman

See Also

`new.registry`, `register.page`, `get.page`, `pages`

Examples

```
example(register.page, ask=FALSE)  # register the sine page
has.page(registry, "sine")         # should return TRUE now.
```

ignore.lots.of.stuff *ignore.lots.of.stuff*

Description

Transformer for ignoring id, class, type and some whitespace

Usage

```
ignore.lots.of.stuff(lines)
```

Arguments

lines	Character vector of lines from the SVG file.

Details

This transformer strips all id, class and type tags, with one preceding space, from the SVG lines.

It also ignores what it thinks is space between tags, namely >\s+<

This is meant primary as an argument for `transformer` in `check.same.svgs`.

Not exported—you should fully qualify it with `AnalysisPageServer:::ignore.lots.of.stuff` if you want to use it.

All the lines will be concatenated, too, into a single character string.

Value

Character vector. Same lines, with id and class tags transformed.

Author(s)

Brad Friedman

See Also

`check.same.svgs`

is.registry *is.registry*

Description

Test if an argument is an AnalysisPageRegistry

Usage

```
is.registry(registry)
```

Arguments

registry	A candidate object

Value

Logical, indicating that the object is an "AnalysisPageRegistry"

`kill.process` *kill.process*

Description

Kill a process and wait for it.

Usage

```
kill.process(pid)
```

Arguments

`pid`	Process ID, or list with `$pid` component

Details

Kill a process and wait for it. Nothing more than `kill(pid); wait(pid)`, but handy to have a single function so you don't forget the `wait()` call.

Value

Same as `wait` in the `fork` package.

Author(s)

Brad Friedman

`known.param.sizes` *known.param.sizes*

Description

Get the controlled vocabulary of parameter size words

Usage

```
known.param.sizes()
```

Details

Returns the controlled vocabulary of parameter size words

Value

Character vector

Author(s)

Brad Friedman

lives.ok *lives.ok*

Description

Test that an expression lives OK

Usage

```
lives.ok(call, testname)
```

Arguments

call	An expression to evaluate
testname	A name for the test. Defaults to deparsing the call.

Details

Test that evaluating an expression lives OK (does not throw an exception)

Value

Runs one test. Returns the value of the evaluated expression

Author(s)

Brad Friedman

Examples

```
lives.ok(3+5, "addition lives OK")
```

make.standard.ids *make.standard.ids*

Description

Make a vector of standardized IDs

Usage

```
make.standard.ids(n, prefix = "Reg")
```

Arguments

n	Desired length of vector
prefix	String, default "Reg".

Details

Make a vector of standardized IDs.

Value

Character vector. Currently just "Reg1", "Reg2", ..., "Regn" (or starting with whatever prefix is).

Author(s)

Brad Friedman - Regular

`messageSectionName` *Get/set section name for "messages" section*

Description

Any `messages` thrown during execution of a page handler are display in a new section of the accordion. This controls the name. This is reset to "Messages" for each page, but the page can call this function to get or set the name.

Usage

```
messageSectionName(sectionName)
```

Arguments

`sectionName`	If present, new section name (e.g. "Your Messages").

Details

Note that all messages thrown will be collected at the end and made into this single section. Therefore, if the message section is renamed after throwing a message then both the old and any newer messages will appear under the new name.

If a section of the same name is created using `appendCustomContent` then these messages will just be appended to the end.

Value

A string, the section name for the messages section

Author(s)

Brad Friedman

new.analysis.page *new.analysis.page*

Description

Validate and prepare a handler for installation

Usage

```
new.analysis.page(handler, param.set = NULL, annotate.plot = TRUE,
  class.name = "plot-point", standard.ids = TRUE, skip.checks = FALSE,
  plot.pars.transformer = NULL, annotate.data.frame = TRUE,
  numeric.sig.digs = 3, no.plot = FALSE, name = NULL, label = name,
  description = label, advanced = 0, thumbnail = NULL, service = FALSE,
  in.menu = !service, paramset.transformer = NULL)
```

Arguments

handler	A handler function, as described above.
param.set	An AnalysisPageParamSet to use for the function. Or NULL, to call default.param.set. Note that it is not a requirement that all of the function arguments be included in the param set—they just won't be provided.
annotate.plot	Logical. Should plots generated by this handler be automatically annotated? Default: TRUE.
class.name	Character. What class label should be applied to automatically annotated points? Default: "plot-point". (Ignored if annotate.plot is FALSE.)
standard.ids	Logical. By default (TRUE), the rownames of your return value are ignored, and new ones are created like "Reg1","Reg2". The advantage of this is that these IDs are guaranteed to be standard-compliant HTML and SVG id tags. If you want to force using your real rownames as IDs (for example, to help in debugging), then set this to FALSE (FALSE is implemented but not tested). Or you can provide a function with the same signature as AnalysisPageServer:::make.standard.ids that will generate IDs for you (this is also implemented but not tested). When annotate.plot is FALSE (for example, when a PNG is requested) the rownames are always left alone and make.standard.ids is not called.
skip.checks	Logical. By default (FALSE) your handler is run once on its default arguments, and it is checked that it makes an SVG and that the SVG can be annotated (if annotate.plot was set). This is important to get right, but doens't really need to be done during production—it just slows down the server start up.
plot.pars.transformer	A function to transform plot parameters. It should have the signature function(plot=list(), other and return a list. The first argument is the plot parameters extracted from the user request (these are the parameters like "width" and "height" that are not related to the business of the request but are simply passed through to the device function), and the second is all the other parameters from the user request. The functinon returns a (named) list of further arguments to pass to the device function. The main use for this is to set the image dimensions based on the user request. In that case your function would return a list with "width" and "height" elements. The units would be inches for svg plot. png plot uses pixel

units, but if you add the parameter units="in" then you can use inches units. You can do this if "units" %in% names(formals(device)). The default plot.pars.transformer=NULL is to not transform the parameters at all.

annotate.data.frame
: Logical, indicating whether your return value should be passed throuhg annotate.data.frame. Default: TRUE. several checks appropriate for the standard case of data associated with plotted regions.

numeric.sig.digs
: Number of significant digits to which numeric columns in your data should be rounded. Default: 3. Set to NULL to not round (you could still round within your function if you wanted tighter control). "numeric" here means either that you set the a varMetadata "type" of the column to the string "numeric", or, if that is not available that is(column)[1] is "numeric". This means, in particular that integer columns will not be rounded.

no.plot
: This page is meant to return data but no plot. Default: FALSE (it *is* expected to return a plot).

name
: A name for the analysis page. Defaults to deparsing the handler argument. This meant to be an internal identifier for the page, only displayed to the user if label and description are unavailable.

label
: A display label for the page. This should be 1-3 words, to fit in the navbar. Default: name.

description
: A longer description for the page. This should be 1-2 sentences, to appear on rollover or in a summary page. Default: label

advanced
: An integer. 0 means not advanced (always display the page). 1, 2, 3 are increasing levels of advanced (only display the page in advanced mode). Default: 0

thumbnail
: A URL for a thumbnail to use when listing the page. NULL means to not store any thumbnail.

service
: A logical, default FALSE. TRUE means that this page should only be called as a service and should not be rendered as a user page. This also means that the return value will not be processed at all except for JSON-encoding (unless of course you return an AnalysisPageResponse).

in.menu
: A logical, default !service. TRUE means that the front-end should display this page in the menu. FALSE means that the front-end should not display the page in the menu, but should still be ready to render it, for example by app state link (contrast with service which the front end can't do anything with except provide a download link or use (as a service) to populate an input widget). The special condition service = FALSE, in.menu = TRUE builds a Page that the front end can use but doesn't show up in the menu. The combination of service = TRUE, in.menu = TRUE, doesn't make any sense and leads to an error.

paramset.transformer
: A function which accepts a named list of parameter values as its first argument and possibly the AnalysisPage object as its second argument, and returns a named list of parameter values. This transformation is applied last, *after* the individual parameters have been transformed, if applicable, but (of course) before the handler is called. Or NULL (default) to not do this transformation. The purpose of this is to be able to encode some reusable logic here for groups of parameters which would often be used together but whose transformation is inter-dependent. If both this argument and plot.pars.transformer are supplied then this transformation is applied first.

Details

An AnalysisPage handler is a function that satistifies the following properties:

1. Can be called with no arguments and return a valid value (to be used for testing in the next steps; although this can be relaxed with `skip.checks`).
2. It creates a plot but does not open the device (although this can be relaxed with `do.plot`)
3. It returns a data.frame with `x` and `y` fields. Alternatively it may return an `AnnotatedDataFrame`. (although this can be relaxed with `annotate.data.frame`)
4. `x` and `y` fields are numeric.
5. The points in test plot can be successfully found (based on the x and y coordinates) and labeled.

This function throws an error if the argument does not satistfy one of these. Otherwise it returns void.

The function will be called once at the time of running this function (typically during registration) with all of its defaults to verify the second and third requirements.

The return value is a list of class "AnalysisPage" with the following components:

`$handler` The handler function

`$params` An `AnalysisPageParamSet` (see `param.set`)

`$annotate.plot` An logical indicating whether the plots generated by the handler should be automatically annotated

`$class.name` A character giving the class to be applied to the annotated SVG elements

A list will be built with the information necessary to render the page. It will contain the handler function in the `$function` slot, as well as a `$params` slot listing all of the parameters and their relevant information. The class name of "AnalysisPage" will be slapped on this object for good measure.

Value

See above

Author(s)

Brad Friedman

See Also

`register.page`, `execute.handler`, `AnnotatedDataFrame`

Examples

```
page <- new.analysis.page(AnalysisPageServer:::sine.handler)
registry <- register.page(new.registry(), "sine", page)
## Note: above is equivalent to the following:
## registry <- register.page(registry, "sine", AnalysisPageServer:::sine.handler)
```

new.datanode.array *new.datanode.array*

Description

Construct a new array-type data node

Usage

```
new.datanode.array(name, children, ...)
```

Arguments

name	Name of the node
children	List of AnalysisPageDataNodes
...	Passed through to new.datanode, in particular label and description

Details

Construct a new array-type data node from a list of AnalysisPageDataNodes

Value

AnalysisPageDataNode

Author(s)

Brad Friedman

Examples

```
html.node <- example(new.datanode.html)$value
simple.node <- example(new.datanode.simple)$value
new.datanode.array(name = "arr", children = list(html.node, simple.node))
```

new.datanode.html *new.datanode.html*

Description

Construct a new HTML-type data node

Usage

```
new.datanode.html(name, value, ...)
```

Arguments

name	Name of the node
value	The value, an HTML string or charvec.
...	Passed through to new.datanode, in particular label and description

Details

Construct a new HTML-type data node. An HTML-type data node is like a simple data node but it has an HTML string or character vector as its value. It should be rendered as-is, but with activated analysis-page-data-set containers.

Value

AnalysisPageDataNode

Author(s)

Brad Friedman

See Also

`is.atomic`

Examples

```
new.datanode.html(name = "shakespeare",
                  value = "<i>Shall I compute thee to a summer's data?</i>")
```

`new.datanode.plot` *new.datanode.plot*

Description

Construct a new plot-type data node

Usage

```
new.datanode.plot(name, plot.file, table, warnings = character(),
  filter.widget = NULL, ...)
```

Arguments

`name`	Name of the node
`plot.file`	Path to plot file, relative to server tempdir.
`table`	A table-type `AnalysisPageDataNode`
`warnings`	Character vector of warnings, to be passed through to `new.datanode`, possibly after appending the warning described above for `filter.widget`. Default: `character()`
`filter.widget`	If provided, then an AnalysisPageFilterWidget. See `new.filter.widget` and `setFilterWidget`. A check is made that `filter.widget$data_field` is an actual field from the data table. If not, then the filter widget is omitted, with a warning. NULL (default) means to not include this.
`...`	Passed through to `new.datanode`, in particular `label` and `description`

Details

Construct a new plot-type data node from a plot.file and an AnnotatedDataFrame. Note: caption is included implicitly in the table object.

Value

AnalysisPageDataNode

Author(s)

Brad Friedman

`new.datanode.simple` *new.datanode.simple*

Description

Construct a new simple-type data node

Usage

```
new.datanode.simple(name, value, ...)
```

Arguments

`name`	Name of the node
`value`	The value
`...`	Passed through to `new.datanode`, in particular `label` and `description`

Details

Construct a new simple-type data node. A simple data node must have either a NULL value or a "scalar" value, which means length 1 atomic

Value

AnalysisPageDataNode

Author(s)

Brad Friedman

See Also

`is.atomic`

Examples

```
new.datanode.simple(name = "x", value = 100)
```

new.datanode.table *new.datanode.table*

Description

Construct a new table-type data node

Usage

```
new.datanode.table(name, data, caption = "", ...)
```

Arguments

name	Name of the node
data	An AnnotatedDataFrame. Unless "label" is already available, "labelDescription" in the varMetadata will be changed to "label" to agree with the syntax we use in the rest of the AnalysisPage interface.
caption	Caption for the table. Default: ""
...	Passed through to new.datanode, in particular label and description

Details

Construct a new table-type data node either from an AnnotatedDataFrame.

Value

AnalysisPageDataNode

Note

Captions for plots are included implicitly in the table component of their responses.

Author(s)

Brad Friedman

new.event.registry *new.event.registry*

Description

EventRegistry constructor

Usage

```
new.event.registry()
```

Details

An EventRegistry is a collection of Events. Each Event has a name and a list of functions, each known as a handler. Events can be modified or triggered. When triggered, each function is called in turn, and the final return value is returned to the triggering context.

Value

EventRegistry

Author(s)

Brad Friedman

Examples

```
r <- new.event.registry()
```

---

new.FastRWeb.analysis.page.run

new.FastRWeb.analysis.page.run

Description

Given an RApacheAnalysisPageServer object, create a FastRWeb-compatible handler

Usage

```
new.FastRWeb.analysis.page.run(app, FastRWeb.scriptname,
  FastRWeb.prefix = "/cgi-bin/R", APS.resources.location = "/R",
  front.end.location = "/dist-aps",
  front.end.dir = system.file("htdocs/client/dist-aps", package =
  "AnalysisPageServer"), tmpdir = tempdir(), FastRWeb.tmpdir = getwd(),
  referer = "", EP = NULL, REST.location = "/REST", verbose = FALSE,
  logger = create.logger(stderr(), if (verbose) log4r:::INFO else
  log4r:::FATAL), ...)
```

Arguments

app	AnalysisPageRApacheApp. Or an AnalysisPageRegistry from which to build an app (see `...`).
FastRWeb.scriptname	Name for the app within FastRWeb. A script called `FastRWeb.R` would be created, and the URL would be `FastRWeb.prefix/FastRWeb.scriptname`.
FastRWeb.prefix	Prefix for all the FastRWeb resources on your server. For example, if you are using a typical CGI deployment, as described in the FastRWeb INSTALL document, it would be "/cgi-bin/R", and your AnalysisPageServer app would be a group of URLs like /cgi-bin/R/APS/client/analysis-page-server.html etc (but see `FastRWeb.scriptname` for `APS` and `front.end.location` for `client`).

`APS.resources.location`
: Location relative to App base URL from which to serve dynamic AnalysisPageServer resources (like `analysis`, `pages`, and other details that normal users don't have to worry about). Default, "R", is probably fine.

`front.end.location`
: Location relative to App base URL from which to serve front end files. Default, "/client", is probably fine.

`front.end.dir`
: Path (in filesystem) to front end files. Default: `system.file("htdocs", package = 'AnalysisPag`

`tmpdir`
: Path to temporary directory to store files needed while the server is running. Default: `tempdir()`. This is ignored if `app` is an `AnalysisPageRApacheApp`. This is a directory private (in the OOP sense, not necessarily in the filesystem sense of the word "private") to the AnalysisPageServer system—FastRWeb never sees it directly. This means in particular that it doesn't have to be within the AnalysisPageServer hierarchy.

`FastRWeb.tmpdir`
: This is the path to FastRWeb's tmpdir. FastRWeb will only serve temporary files out of that directory.

`referer`
: FastRWeb does not (currently) parse the Referer from the headers, but you can put a string here which will be interpreted as such. I only put this here if you have an app which fails catastrophically if referer is unavaiable.

`EP`
: ExpressionPlotClient object, if needed for your app. Deprecated, and to be removed in a future version.

`REST.location`
: If `EP` is non-NULL, then the location from which to serve REST requests (relative to app base URL). Default: "/REST". Deprecated, and to be removed in a future version.

`verbose`
: Boolean, default FALSE. If TRUE then send progress messages.

`logger`
: log4r object, optional

`...`
: If `app` is actually an AnalysisPageRegistry then `...` is passed through h it to `rapache.app.from.registry` to build the `AnalysisPageRApacheApp`. just a convenient

Details

FastRWeb is another alternative for deployment of AnalysisPageServer applications. The benefit relative to Apache/RApache is that it may be easier to deploy. The benefit relative to Rook/Rhttpd is that it actually works (Rhttpd cannot handle concurrent connections). See `http://rforge.net/FastRWeb/` (and `http://rforge.net/Rserve/`) for information about the FastRWeb/Rserve system. This function assumes that you've already got a working installation of Rserve and fastRWeb, as described there. As with that example, FastRWeb is a layer between Rserve and either CGI or PHP, so you would also have to have a CGI or PHP server running. Or, you could try the experimental http server that comes starting in Rserve 1.7.

This function, `new.FastRWeb.analysis.page.run` returns a function which can be used as the `run` function for a FastRWeb script. So, typically, your FastRWeb script would do whatever necessary to create your AnalysisPageServer Registry (or App object). Then, the last line of the script would be to pass that object to this function, and assign the return value to `run`. In other words, this function would not normally be called interactively, but only within the FastRWeb framework.

For development purposes it is quite convenient to build your AnalysisPageServer application within the FastRWeb script. For example, in the default configuration, you could put this into "/var/FastRWeb/web.R/APS.R":

```
library(AnalysisPageServer)  reg <- trig.registry()  run <- new.FastRWeb.analysis.page.run(reg
```

This allows you to make changes to code and reload the page without restarting the server. In fact, any libraries will be reloaded, so you can change your package and re-install without restarting the server.

Once you want to move more into testing or production you'll want to do all the heavy lifting in the startup script. You would have to save the `run` object somewhere in the Global namespace and then simply return in in the FastRWeb script. Under default FastRWeb configuration you could add this code to the end of the startup script "/var/FastRWeb/code/rserve.R":

```
library(AnalysisPageServer)  myRun <- new.FastRWeb.analysis.page.run(trig.registry(), FastRWeb
```

Then, in "/var/FastRWeb/web.R/APS.R", you could just have nothing more than this:

```
run <- myRun
```

Note that the name of the FastRWeb script must be the same as `FastRWeb.scriptname`, but with a ".R" suffix appended.

It would be possible to extend this system to server multiple AnalysisPageServer apps from the same FastRWeb setup. Each one would get its own FastRWeb script, and I leave it as an exercise for the reader to build them all in the Rserve startup and assign the correct handler to `run` in each script.

In this example I point my browser to `http://localhost/cgi-bin/R/APS/dist-aps/analysis-page-server.html` to open the page.

Value

Not sure yet....

Author(s)

Brad Friedman

Examples

```
## Not run:
  library(AnalysisPageServer)
  reg <- trig.registry()
  run <- new.FastRWeb.analysis.page.run(reg)

## End(Not run)
message("See vignette FastRWebDeployment.html")
```

`new.registry` *new.registry*

Description

Make a new (empty) registry.

Usage

```
new.registry(...)
```

Arguments

...	AnalysisPages with which to initially populate the registry

Value

A new registry, which is just an empty list with class AnalysisPageRegistry.

Author(s)

Brad Friedman

See Also

register.page, has.page, get.page, pages

Examples

```
r <- new.registry()
```

---

new.response *new.response*

Description

Build a new AnalysisPageResponse object

Usage

```
new.response(body, content.type, status = 200, headers = character())
```

Arguments

body	Either a raw vector or a character vector that constitutes the response body.
content.type	A string giving the content-type, such as "text/plain"
status	Integer. An HTTP response status. Default, 200, means HTTP_OK
headers	Named charvec of extra HTTP headers. Default: character() (none)

Details

A handler may return an AnalysisPageResponse object, which is basically a complete response, if it doesn't want the framework to do any extra processing. This allows complete control over the response.

Value

AnalysisPageResponse object

Author(s)

Brad Friedman

Examples

```
poem.file <- system.file("examples/in-a-station-of-the-metro.html", package="AnalysisPageServer")
poem.html <- readLines(poem.file, warn = FALSE)
new.response(paste0(poem.html, "\n"), content.type = "text/html")
```

`new.rook.analysis.page.app` *new.rook.analysis.page.app*

Description

Given an RApacheAnalysisPageServer object, return a Rook app that can run it.

Usage

```
new.rook.analysis.page.app(app, EP = NULL, front.end.location = "/dist-aps",
  front.end.dir = system.file("htdocs/client", package =
  "AnalysisPageServer"), app.name = "RAPS",
  app.prefix = file.path("/custom", app.name), tmpdir = tempdir(), ...)
```

Arguments

`app`	AnalysisPageRApacheApp. Or an AnalysisPageRegistry from which to build an app
`EP`	ExpressionPlotClient object, if needed for your app.
`front.end.location`	Location relative to App base directory from which to serve front end files. Default: "/dist-aps".
`front.end.dir`	Path (in filesystem) to front end files. Default: `system.file("htdocs", package = 'AnalysisPag`
`app.name`	The name of the app you are going to use within Rook. This is used to build the prefix `/custom/${app.name}` from where the app will be served—the Javascript front end has to be notified of this. Ignored if `app.prefix` is supplied.
`app.prefix`	The prefix from which the app will be served. Default: `/custom/${app.name}`.
`tmpdir`	Path to temporary directory to store files needed while the server is running. Default: `tempdir()`.
`...`	If app is actually an AnalysisPageRegistry then `...` is passed through along with it to `rapache.app.from.registry` to build the `AnalysisPageRApacheApp`.

Details

new.rook.analysis.page.app

Value

Your app, as a Rook App

Author(s)

Brad Friedman

See Also

startRookAnalysisPageServer, kill.process

Examples

```
message("See vignette ExamplesServers.html")
```

---

pages *pages*

Description

Get names of all pages in registry

Usage

```
pages(registry, include.services = FALSE)

## S3 method for class 'AnalysisPageRegistry'
pages(registry, include.services = FALSE)
```

Arguments

registry	AnalysisPageRegistry object
include.services	Logical. Should I include services in my list of all pages? Default: FALSE, do not include services.

Details

Get names of all pages in registry

Value

Character vector of names of pages in registry

Note

Service pages are identified as those having their service flag set, which is done at page build time using the service parameter of the new.analysis.page constructor.

Author(s)

Brad Friedman

See Also

new.registry, register.page, has.page, get.page

Examples

```
empty.pages <- pages(new.registry())   # should be empty character vector
example(register.page, ask=FALSE)     # see register.page example---registers the sine handler
pages(registry)                        # should now be the character vector "sine"
```

param.set *param.set*

Description

Build a new AnalysisPageParamSet from a list of AnalysisPageParam's.

Usage

```
param.set(...)
```

Arguments

...	AnalysisPageParam objects. Or a single argument, being a list of AnalysisPageParam objects.

Details

Build a new AnalysisPageParamSet from a list of AnalysisPageParam's: Check for non-AnalysisPageParam elements and for duplicate names. Apply $name elements to names of param set. Set class to AnalysisPageParamSet

Value

AnalysisPageParamSet

Author(s)

Brad Friedman

Examples

```
par1 <- simple.param(name = "par1")
par2 <- bool.param(name = "par2")
pset <- param.set(par1, par2)
```

paramSetToJSON *Convert an AnalysisPageParamSet to a JSON string*

Description

This is almost just calling toJSON but it knows to first remove $transformer components, since functions can't be JSON encoded, and anyway that is really server-side information.

Usage

```
paramSetToJSON(ps)
```

Arguments

ps	AnalysisPageParamSet

Value

JSON string

Author(s)

Brad Friedman

`persistent.param.dependencies`
Return persistent parameter dependencies for an object

Description

Return persistent parameters dependencies for an object.

For an `AnalysisPageParam` the return value is either a charvec of the object's persistent dependencies. It is important to note that The namespace for persistent dependencies of an `AnalysisPageParam` is the same as the namespace of the page Parameters. This is because when calling a constructor for an `AnalysisPageParam` at most we might specify the names of other Parameters (for example for combobox dependencies). But the other Parameters are not available at that time, so we can't look up their corresponding names in the persistent namespace.

For an `AnalysisPageParamSet` or an `AnalysisPage`, the return value is a list named after all of the persistent params, the values being charvecs of their dependencies. Note that The namespace for persistent dependencies of an `AnalysisPageParamSet` is the persistent namespace, not the Page Parameter namespace.

For an `AnalysisPageRegistry` the return value is a list named after all of the persistent params of any of the pages, the values being charvecs of their dependencies. If there is an discrepancy in the dependencies for a given persistent param, then the union of all dependencies is taken. (This will be checked for acyclicity by `.validate.registry`.) The namespace for persistent dependencies of an `AnalysisPageRegistry` is the persistent namespace, not the Page Parameter namespace.

Usage

```
persistent.param.dependencies(x)

## S3 method for class 'AnalysisPageParam'
persistent.param.dependencies(x)

## S3 method for class 'AnalysisPageParamSet'
persistent.param.dependencies(x)

## S3 method for class 'AnalysisPage'
persistent.param.dependencies(x)

## S3 method for class 'AnalysisPageRegistry'
persistent.param.dependencies(x)
```

Arguments

`x`	An object for which there is a `persistent.param.dependencies` method.

Details

This is a named list. The names are the names of persistent parameters, and the values are the other persistent parameters on which they depend.

Value

Named list of charvecs, see Details.

Author(s)

Brad Friedman

---

persistent.params *Return names of persistent parameters for an object*

Description

Return names of persistent parameters for an object. The exact meaning depends on the type of object. For an AnalysisPage it would be the persistent params for that page. For an AnalysisPageRegistry it would be the persistent params for any of its pages.

Usage

```
persistent.params(x)

## S3 method for class 'AnalysisPageParam'
persistent.params(x)

## S3 method for class 'AnalysisPageParamSet'
persistent.params(x)

## S3 method for class 'AnalysisPage'
persistent.params(x)

## S3 method for class 'AnalysisPageRegistry'
persistent.params(x)
```

Arguments

x	An object for which there is a `persistent.params` method.

Details

Note that the names are from the shared namespace, which are the values passed as the `persistent` argument to the `AnalysisPageParam` constructor functions such as `simple.param`. Although this is usually the same as actual names of the `AnalysisPageParams` themselves there is no requirement that they be the same.

Value

Charvec of persistent params

Author(s)

Brad Friedman

`platformIsWindows` *Predicate to test if running on a windows platform*

Description

Predicate to test if running on a windows platform

Usage

```
platformIsWindows()
```

Value

Boolean: `TRUE` on windows, `FALSE` on other platforms

Author(s)

Brad Friedman

`protect.rapache.memory` *protect.rapache.memory*

Description

Set up events and handler to turn over memory-bloated worker processes

Usage

```
protect.rapache.memory(app, max.mb, app.event = "FinishAnalysis",
  memory.event = "BloatedMemory")
```

Arguments

`app`	AnalysisPageRApacheApp
`max.mb`	Maximum allowed memory usage before triggering turnover.
`app.event`	Name of event which should trigger this memory check. Default: "FinishAnalysis".
`memory.event`	Name of event which excess memory usage should trigger. Default: "BloatedMemory".

Details

Set up events and handler to turn over memory-bloated worker processes. When Rapache processes process requests that require large amounts of memory they don't return the memory to the OS. Eventually it can build up, slowing down the server when then has to turn to cache. Calling this function will add a check at each FinishAnalysis which, if the process is using more memory than the threshold specified by `max.mb`, delivers a SIGUSR1 signal to itself. This is a signal to Apache that the process should be turned over after finish the current request, thus pruning bloated workers.

Value

Nothing

Author(s)

Brad Friedman

`rapache.app.from.registry`
rapache.app.from.registry

Description

Build rapache app from an AnalysisPageRegistry

Usage

```
rapache.app.from.registry(registry, page.param = "page",
  textarea.wrap.param = "textarea_wrap", device.param = "device",
  decoder.param = "decoder", max.regions.param = "max_annotated_regions",
  default.max.regions = 10000, force = FALSE,
  tmpdir = Sys.getenv("WEB_TMPDIR"), tmpdir.timeout.seconds = 600,
  devices = .default.device.list, other.mime.types = c(json =
  "application/json"), mime.types = c(sapply(devices, "[[", "mime.type"),
  other.mime.types),
  query.param.decoders = .build.default.query.param.decoders(),
  brand.builder = .default.brand.builder, logger = create.logger(stderr(),
  log4r:::FATAL))
```

Arguments

`registry`	AnalysisPageRegistry from which to build you app.
`page.param`	Character. Name for the parameter which specifies the page. Default "page"
`textarea.wrap.param`	Character. Name for the parameter which specifies whether the response should be wrapped in a `<textarea>` tag. This is needed to support file uploads in browsers, like IE9, that don't support XMLHttpRequest2. Default "textarea_wrap". In addition to wrapping in textarea the response type will be set to "text/html", another hack that such browsers require.
`device.param`	Character. Name for the parameter which specifices the plotting device. Default: "device".

`decoder.param` Character. Name for the parameter which specifices the form query parameter decoding method. Default: "decoder".

`max.regions.param`
String. Name for the parameter which specifies the maximum number of regions for annotation. Default: "max_annotated_regions".

`default.max.regions`
Default maximum number of regions for annotation. If a plot has more than this many elements then it will not be annotated. Default: 10000.

`force` Logical. If set then an invalid registry is a warning instead of an error.

`tmpdir` Temporary directory into which plot files should be written. By default this is taken from the WEB_TMPDIR environment variable. It is checked upon opening. It is important under Apache that all of the processes use the same directory, otherwise they won't be able to find the plots made by other processes.

`tmpdir.timeout.seconds`
Temporary files will expire after this time. Default: 600 (10 minutes)

`devices` Named list. The names of the vector are the names of allowed plotting devices. The values are themselves lists, each having a `$mime.type` and `$function`, being the ploting function. Default is given by the private variable `.default.device.list`, and includes svg and png (but uses a modified png function so that it accepts units inches). Which one to be used (in the default scenario, a choice between `svg` and `png`) is controlled by the special `device.param` parameter. If that is not provided then the first device from this vector is used (default default is therefore "svg", which is the best since it can be annotated, but older browsers such as IE8 will need to do PNGs).

`other.mime.types`
Named charvec giving a mapping from file extensions other than those already in `devices` to MIME types that can be served out of the temporary storage/retrieval area. Default: `c(json = "application/json")`.

`mime.types` This is a named charvec giving a mapping from file extensions to mime types. Only files with extensions in this list can be served from the temporary storage/retrieval area. The default is to take the extensions/MIME-types defined in `devices` and add to them those in `other.mime.tpyes`.

`query.param.decoders`
Names list. The names are the names of allowed query param decoders (valid values for the `decoder.param` parameter). The values are functions which do the decoding. Default is just `list(url=urlDecode)`. `urlDecode` is a function supplied by RApache (or by testing framework).

`brand.builder` This is a function that takes a single argument called "persistent" which is a list of key value pairs representing the internal "persistent" state of the app. The persistent state is a namespace that particular parameters of particular pages can draw from, with the possible option of locking those parameters to the values in the persistent namespace. The purpose of the `brand.builder` function is to return a string that should be used in the top-left corner of the web page to briefly summarize the current state. The default brand builder always returns the string "AnalysisPageServer".

`logger` log4r object, optional

Details

This does most but not all of the work. You should create an R script that builds your page registry, then builds an rapache.app with this function. Within that R script you call add.handlers.to.global

to install the 5 handlers (handle.pages/handle.params/handle.plot/handle.data/handle.meta.data).

Next you have to tell apache to source your script upon startup. The directive for your httpd.conf is as follows: `RSourceOnStartup "/gne/home/friedmab/scr/apache-test/R-startup.R"`

Finally, you have to register the five handlers. This is done as follows

Value

AnalysisPageRApacheApp

Author(s)

Brad Friedman

`rapache.trig.app` *rapache.trig.app*

Description

Build the AnalysisPageRApacheApp for the trig example

Usage

```
rapache.trig.app(...)
```

Arguments

`...`	Other parameters to pass through to `rapache.app.from.registry`, such as `con`

Details

The toy registry has a sine page, a cosine page and the scattergram tool.

Value

AnalysisPageRegistry

Author(s)

Brad Friedman

See Also

`trig.registry`

register.page *register.page*

Description

Register a page

Usage

```
register.page(registry, page.name, page, overwrite = FALSE)
```

Arguments

registry	AnalysisPageRegistry object
page.name	Character. Name of the page to register
page	AnalysisPage or function. If a function is supplied instead of an AnalysisPage object then it will be coerced into an AnalysisPage object calling new.analysis.page.
overwrite	Logical. If FALSE (default) then throw and error if a page is already registered under that name. If TRUE then just warn.

Details

Register a page

Value

void

Author(s)

Brad Friedman

See Also

new.registry, has.page, get.page, pages, new.analysis.page

Examples

```
# Make a new registry
registry <- new.registry()

# Now register it under the name "sine" (in the "example" registry)
# and keep the modified registry.
registry <- register.page(registry, "sine", AnalysisPageServer:::sine.handler)
```

`remove.event` *remove.event*

Description

Remove an Event entirely

Usage

```
remove.event(registry, event)
```

Arguments

`registry`	EventRegistry
`event`	String. Name of the Event to remove

Details

Remove an Event entirely from the EventRegistry. Contrast with `clear.event.handlers`, which only removes the handlers for that event.

Value

Nothing good.

Author(s)

Brad Friedman

Examples

```
r <- new.event.registry()
add.event(r, "mouseclick")
has.event(r, "mouseclick")
remove.event(r, "mouseclick")
has.event(r, "mouseclick")
```

`remove.old.files` *remove.old.files*

Description

Remove old files from a directory

Usage

```
remove.old.files(tmpdir, tmpdir.timeout.seconds)
```

Arguments

tmpdir	Path to directory whose old files you want to delete
tmpdir.timeout.seconds	Time in seconds. An attempt will be made to delete files with ctimes older than this many seconds before the current time.

Details

Remove old files from a directory

Value

see unlink

Author(s)

Brad Friedman

---

reset.APS.outdir *Reset AnalysisPageServer output directory*

Description

This directory is used by embed.APS.dataset to decide where to save the .svg and .json files. This function resets it to its default, ".".

Usage

```
reset.APS.outdir()
```

Value

Nothing of note

Author(s)

Brad Friedman

See Also

get.APS.outdir, set.APS.outdir

Examples

```
set.APS.outdir("/some/path")
get.APS.outdir()
reset.APS.outdir()
```

rook.analysis.page.server.landing.page

rook.analysis.page.server.landing.page

Description

Return URL for landing page of a Rook AnalysisPageServer

Usage

```
rook.analysis.page.server.landing.page(baseurl)
```

Arguments

baseurl	String. Base URL (typically ending in "/custom/RAPS"), or a list with $url element.

Details

Return URL for landing page of a Rook AnalysisPageServer

Value

String, full URL to landing page.

Author(s)

Brad Friedman

Examples

```
message("See vignette ExamplesServers.html")
```

search.replace

search.replace

Description

Search and replace strings in a file

Usage

```
search.replace(infile, outfile, replacements, overwrite = FALSE)
```

Arguments

infile	Path to input file
outfile	Path to output file (must be different)
replacements	Named charvec of length 1 (single replacement) or 2 (double replacement)
overwrite	Boolean. If FALSE then outfile must not yet exist. If TRUE and it already exists then it will be overwritten.

Details

This is a very limited interface and only meant for internal use.

It will replace all occurances of a string with another string. It may do 1 or 2 replacements.

The result is written to a second file.

This is done in C++ so very fast (I hope).

Value

Nothing, but might throw an error.

Author(s)

Brad Friedman

---

select.param *select.param*

Description

Build a select AnalysisPageParam

Usage

```
select.param(..., value, choices, allow.multiple = FALSE,
  style = "dropdown")
```

Arguments

`...`	Passed through to `simple.param`. This includes at least "name", optionally "label" and "description" t not "type".
`value`	Default value. If not specified then the first entry in choices is taken to be the default.
`choices`	A character vector giving the choices to display. If named, then the values are used for display and names are used for the actual form values. If unnamed, then the values are used for both display and names.
`allow.multiple`	If TRUE then render as checkbox group and allow multiple selections. (The function will be provided a vector of all selected values.) If `allow.multiple` is TRUE then `style` must be "dropdown". Default: FALSE
`style`	Either "dropdown" (default), to render as dropdown list, or "radio", to render as radio group. If `allow.multiple` is TRUE then `style` must be "dropdown".

Details

Build a select AnalysisPageParam. This is probably rendered either as a dropdown or radio group. It is a selection from a fixed list of possible values. The list is known before page load time

Value

An AnalysisPageParam

Author(s)

Brad Friedman

Examples

```
color <- select.param("color", label="Color", description="Color of your house", choices=c("red","green","
```

---

service.link *Build a URL to call a AnalysisPageServer webservice*

Description

This function is simply a specialization of `analysis.link` with a few conveniences for webservice-type pages. In particular, the parameters of that function about plotting are not available.

Usage

```
service.link(page, params = list(), app.base.url)
```

Arguments

`page`	Name of page
`params`	List of parameter values (as R objects—this function will encode them). Default: `list()` (no parameters).
`app.base.url`	Base URL for application. This is usually the prefix in which the app landing HTML page is found.

Value

URL

Author(s)

Brad Friedman

---

set.APS.outdir *Set current AnalysisPageServer output directory*

Description

This directory is used by `embed.APS.dataset` to decide where to save the .svg and .json files.

Usage

```
set.APS.outdir(outdir)
```

Arguments

`outdir`	New output directory

Value

Nothing important

Note

It seems like it would be a good idea to follow this call with an `on.exit(reset.APS.outdir())`. But `on.exit` within a knitr chunk it will just first at the end of the chunk. If you are using knitr then you should just call `setup.APS.knitr()` at the top of your document then each document will have its output directory correctly set and you don't really have to worry. If you want to be really anal you could call `reset.APS.outdir()` at the bottom of your knitr document.

Author(s)

Brad Friedman

See Also

`get.APS.outdir`, `reset.APS.outdir`

Examples

```
set.APS.outdir("/some/path")
get.APS.outdir()
reset.APS.outdir()
```

---

setFilterWidget — `setFilterWidget` *sets the filter widget for the current analysis. This is the function most commonly used.*

Description

`setFilterWidget` sets the filter widget for the current analysis. This is the function most commonly used.

`getFilterWidget` retrieve the filter widget for the current analysis. This is normally used internally, to construct the final response for the analysis.

The AnalysisPageFilterWidget object specifies a "filter widget" to be displayed to the user. This is a grid of colored squares, each of which controls the filtering of a subset of the samples based on the values of a pheno field. This object specifies the dimension of the grid, the colors of the squares, rollovers to appear for each square, and the subset of samples that each square filters.

Usage

```
setFilterWidget(data.field, color, cells, inactive.color = "gray",
  type = "filter_grid")

getFilterWidget()

new.filter.widget(data.field, color, cells, inactive.color = "gray",
  type = "filter_grid")
```

Arguments

data.field	Name of table field which should be used for filtering.
color	Named character vector. Names should be values (or possible values) that the data field could take on. Values are string specifying colors. These are passed through directly to javascript so they should be valid colors there, whatever that means.
cells	Character matrix. This gives the layout of the filter grid. The values should all be either taken from names(color) or else be NA values. The NA values will be inactive (no rollover or click listeners).
inactive.color	Color for inactive cells (that are the positions with is.na(cells)). Default: "gray"
type	Filter widget type. The only currently supported type is "filter_grid".

Details

new.filter.widget is the constructor for this object. This should be used when constructing an AnalysisPageReponse explicitly, with the return value then passed to new.datanode.plot.

Value

setFilterWidget returns the newly set AnalysisPageFilterWidget object

getFilterWidget returns the curretn AnalysisPageFilterWidget object, or NULL if it has not yet been set

new.filter.widget returns an AnalysisPageFilterWidget

Author(s)

Brad Friedman

---

setup.APS.knitr *Set up knitr documents to contain AnalysisPageServer data sets*

Description

If you want to embed APS data sets within a knitr document then this function should be called at the top of the document like this:

Usage

```
setup.APS.knitr(outdir, include.css = system.file("AnalysisPageServer.css",
  package = "AnalysisPageServer"), include.toc = TRUE, quiet = TRUE)
```

Arguments

outdir	Output directory to which front end files should be written. Default: see details.
include.css	Paths to CSS files to include. See details.
include.toc	Boolean, default TRUE. Should I include a table of contents?
quiet	Boolean, default TRUE. Set to FALSE to turn on some diagnostic messages

Details

```
```{r echo = FALSE}   AnalysisPageServer::setup.knitr()   ```
```

Calling this function has the following effects:

1. The first effect is a heinous crime. It looks up the call stack to see if you are in the middle of a buildVignettes call. If so then it sets clean = FALSE for that call. The reason for this is that for the document to work it will need a bunch of auxiliary files like .css and .js, and if clean = TRUE then these files won't be left and your data sets will not show at all. During R CMD build buildVignettes is explicitly called with clean = TRUE so this is the only way I could figure out how to turn it off. If it can't find buildVignettes in the call stack then nothing special happens. This would be the case if you are just calling knit2html yourself.
2. The next effect is to copy all the front end files to the output directory. The default output directory is also kind of heinous. The files need to be next to the output file. So the function again looks up the call stack to find the knit2html call, then grabs the name of the output file from there and uses its directory as outdir. If it can't find a knit2html call then it throws an error.

   The output directory is saved with a call to the private function set.APS.outdir. This is then read back by embed.APS.dataset so that the data sets get written to the write place. You could add a reset.APS.outdir() call in a chunk at the bottom of your knitr document.
3. Any files in include.css are copied to the output directory and included as CSS. The default is a default stylesheet that will make your reports look like the vignettes.

Finally it returns the html headers as a "knit_asis" object to be included in your document. See custom.html.headers.

Value

HTML headers as knit_asis objects.

Author(s)

Brad Friedman

Examples

```
message("See vignette embedding.html")
```

---

simple.param *simple.param*

Description

Build a simple AnalysisPageParam

Usage

```
simple.param(name, label = name, description = label, value = "",
  type = "text", advanced = 0, show.if = NULL, display.callback = NULL,
  size = "medium", required = TRUE, persistent = NULL,
  persistent.dependencies = NULL, transformer = NULL)
```
```

Arguments

name	Name of form element
label	Label for form element (typically rendered to the left of the element)
description	Description for form element (typically rendered as roll-over text)
value	The default, starting value, for the form (default: "")
type	Type of form element. This can be "text", "textarea", "checkbox", "password" or "file"
advanced	Integer. 0 means the option is not advanced, and increasing levels indicate the option is for more advanced users. Advanced > 0 should be hidden under default mode.
show.if	A list of two elements: $name, giving the name of some other param in the set and $values, a character vector of values. The parameter under construction should only be shown if the named parameter takes on any of the values in the $values vector. Default (NULL) is to always show the parameter under construction.
display.callback	NULL, to follow show.if logic in deciding when/if to display the element, or a list with the two elements $uri and $dependent, which follow the same format as the corresponding arguments to combobox.param and provide a templated uri and a mapping from template variables to form parameters. The service should return JSON true if the widget is to be shown and false if not. It is allowed to have both a display.callback and a show.if—both conditions must be met in order to display the element.
size	A word giving the size of the element. The interpretation of this size is up to the front-end. Must from a defined set of words, which you can see by calling known.param.sizes() (currently no way to change this). Default: medium
required	Logical. Is this a required param? Default: TRUE. If set, then the front-end will the front-end user is required to set this parameter before submitting the request. The meaning of "set this parameter" is not entirely clear.
persistent	Character or NULL. If non-NULL then it is passed to the front-end. It names a variable in persistent storage that should be used to initialize the value of the parameter. The front end will provide some mechanism to change the persistent value, but until the user does so the param will be initialized from the value in the persistent space.
persistent.dependencies	A character vector or NULL (default) specifying the names of other parameters on which this one "depends". It is an error to include the parameter itself (name). It is an error to provide this when persistent is NULL. When persistent is non-NULL, providing persistent.dependencies makes this parameter not just "persistent" but "conditionally persistent", which is to say that the persistent value for this parameter is actually a hash lookup based on the the other parameters specified in this vector. A typical example would be a pheno fields parameter which is dependent on the study parameter. The names are taken from the Page namespace, which means that a parameter's $name is used when this differs from its $persistent slot.
transformer	A function with signature function(value, self) or function(value) which accepts as first argument the JSON-decoded value of the parameter returned from the front end and then performs some sort of transformation. The return value of the function will be ultimately passed to AnalysisPage handler. For

example, for a field which is a simple text widget but which is supposed to be numeric you might use `transformer = as.numeric`. But you could also implement more complicated logic here. The reason to put the logic here instead of in the handler is that it makes it easier to re-use the widget in multiple handlers. If you have a complex (nested) parameter then the nested elements' transformations, if any, are applied first, then the parental transformation is applied. Or, NULL (default) to not do any transformation beyond the JSON decoding.

Details

Build a simple AnalysisPageParam. These include mainly parameters that can be rendered simply as HTML <input> tags.

Value

An AnalysisPageParam. This is just a list with class name slapped on.

Author(s)

Brad Friedman

Examples

```
x <- simple.param("xmin", label="X-min", description="Minimum x value", type="text")
# Please see the "Persistent Parameters" and "Conditionally Persistent Parameters"
# sections of the Interactive Apps vignette for demonstrations of these functionalities
```

`sine.handler` *sine.handler*

Description

An example handler just for testing and development

Usage

```
sine.handler(xmin = 0, xmax = 3 * pi, n = 100)
```

Arguments

`xmin`	Numeric. Minimum x value to plot
`xmax`	Numeric. Maximum x value to plot
`n`	Integer. Number of points to plot

Details

This handler takes three parameters, xmin, xmax and n, makes a plot of the sin curve from xmin to xmax (using n equally spaced points), and returns a data.frame with the x and y coordinates, with IDs A-Z, A.1-Z.1, ...

Value

data.frame

Author(s)

Brad Friedman

`slider.param` *slider.param*

Description

Build a slider AnalysisPageParam

Usage

```
slider.param(..., min, max, step, value = min)
```

Arguments

`...`	Passed through to `simple.param`. This includes at least "name", optionally "label" and "description" but not "type".
`min`	Minimum value (number)
`max`	Maximum value (number)
`step`	Size of one step (must be <= max-max)
`value`	Default value. If not specified then the minimum is taken to be the default

Details

Build a slider AnalysisPageParam. This is a numeric variable. It has a minimum value, a maximum value, and a step size

Value

An AnalysisPageParam

Author(s)

Brad Friedman

Examples

```
slider <- slider.param("children", label="No. Children", description="Number of Children", min = 0, max = 10
```

`startRookAnalysisPageServer` *Start a new Rook AnalysisPage server*

Description

Start a new Rook AnalysisPage server. This is a convenience wrapper around `new.rook.analysis.page.app` which builds the Rook App and then also makes a Rook server (`Rhttpd` object) which just contains the one App. It then starts the server in a fork and returns the PID of the child process.

Usage

```
startRookAnalysisPageServer(reg, tmpdir = tempdir(), ...,
  app = new.rook.analysis.page.app(reg, tmpdir = tmpdir, app.name = app.name,
  ...), app.name = "RAPS", port = 5000)
```

Arguments

`reg`	AnalysisPageRegistry from which to build application. Passed through to `new.rook.analysis.page`
`tmpdir`	Directory for temporary files. Passed through to `new.rook.analysis.page.app`. Default: `tempdir()`.
`...`	Passed through to `new.rook.analysis.page.app`.
`app`	Rook App to put into the server. Default: `new.rook.analysis.page.app(reg, tmpdir = tmpdir,` Normally you would omit this argument.
`app.name`	Name for App within server, default "RAPS" (for Rook AnalysisPageServer). This will determine the second part of the URL, for example "/custom/RAPS".
`port`	Port on which to start listening.

Value

list with two componenets:

`$url` URL to base of application

`$pid` Process ID of server

Note

This function used to be called `start.rook.analysis.page.server` but that led to an R CMD check warning about S3 method inconsistency.

Author(s)

Brad Friedman

See Also

`new.rook.analysis.page.app`, `kill.process`

Examples

```
## Not run:
  registry <- AnalysisPageServer:::trig.registry()
  server <- startRookAnalysisPageServer(registry, port = 5102)

  ## do some stuff
  ## For example
  landing.page.url <- rook.analysis.page.server.landing.page(server)
  ## now go to your web browser and open landing.page.url

  ## Or maybe something in this R process. See what the pages are
  pages.url <- file.path(server$url, "R", "pages")
  pages <- fromJSON(readLines(pages.url, warn = FALSE))
  sapply(pages, "[[", "name")

  ## Kill the server
  kill.process(server)

## End(Not run)
message("See vignette ExamplesServers.html")
```

---

static.analysis.page *static.analysis.page*

Description

Create interactive AnalysisPage plots from static data

Usage

```
static.analysis.page(outdir, svg.files, dfs, titles, show.xy = FALSE,
  use.rownames.for.ids = FALSE, check.rowname.case = TRUE,
  check.html4.ids = TRUE, group.length.vecs = NULL, signif.digits = 3,
  verbose = FALSE, overwrite = FALSE, write.client = TRUE,
  client.basedir = system.file("htdocs/client/dist-apss", package =
  "AnalysisPageServer"), app.html = "analysis-page-server-static.html",
  build.full.url = write.client, data.subdir = if (write.client) "data" else
  ".", randomize.filename = FALSE)
```

Arguments

outdir	Base directory for output files. Will be created if it does not already exist (however, its parent directory must already exist).
svg.files	Character vector of paths to SVG files. NAs can be used as placeholders for datasets that have data but no plot. Length must be at least 1. If omitted then all NAs are used, something like `rep(NA, length(dfs))`, but a bit more careful about corner cases and types. (So you have to provide at least one of `svg.files` and `dfs`).
dfs	List of data frames of the same length as `svg.files` or, if `length(svg.files) == 1`, a single `data.frame`. NULLs can be used as placeholders for datasets that have plot but no data, but an error is thrown if the corresponding entry in `svg.files`

is also NA. If omitted then all NULLss are used. (So you have to provide at least one of svg.files and dfs). Note that for dfs we use NULLs since it is a list but for svg.files we use NAs since it is a vector and you can't hold a place in a vector with NULL.

titles
: A character vector of titles of the same length as svg.files to display above each plot. Default is rep("", length(svg.files)).

show.xy
: Logical. If FALSE (default) then the first two columns of your data (the x and y coordinates) are used to annotate the plot but not actually exposed to the user in the table or on rollover. If TRUE then they are exposed. Recycled to length(svg.files).

use.rownames.for.ids
: Logical, default FALSE. The default behavior is to generate and assign unique IDs to each point. This makes it impossible to tag two elements in the same plot, or even in different plots with the same ID. If you set this to TRUE then your rownames are used. This means that if you are not careful you might accidentally couple between multiple datasets on the page! Recycled to length of svg.files, so you can set it for each data set independently if you so choose.

check.rowname.case
: Logical, default TRUE. For data frames with use.rownames.for.ids TRUE a check is made that there are not two rownames that are equal without case senstitivity but not with (such as "FirstRow" and "firstrow"). If any is found then an error is thrown. This could possibly be a problem with some browsers, which might treat them the same. FALSE means to skip this check.

check.html4.ids
: Logical, default TRUE. For data frames with use.rownames.for.ids TRUE a check is made that rownames are valid HTML4 IDs: begin with a letter ([A-Za-z]), then followed by any number of letters, digits ([0-9]), hyphens ("-"), underscores ("_"), colons (":"), and periods (":."). (Taken from http://www.w3.org/TR/html4/types.html#h-6.2) FALSE means to skip this check and try to use whatever IDs are there.

group.length.vecs
: List of integer vectors or NULLs of the same length as svg.files (or a single vector or NULL if length(svg.files) == 1). If non-NULL, each one is passed through to annotate.analysis.page.svg as the group.lengths argument, which allows you to specify that the elements might be organized into multiple non-contiguous groups, for example separate panels. A single NULL is recycled to length.

signif.digits
: Passed through to annotate.data.frame. The number of significant digits to which non-integer numeric fields should be rounded.

verbose
: Boolean, default FALSE. If TRUE then message() will be used for progress updates.

overwrite
: If FALSE (default) then an error is thrown if the base directory is not empty. If TRUE then files will be added to the directory, possibly overwriting existing files of the same name.

write.client
: Boolean, default TRUE. Should I write the HTML/Javascript/CSS files necessary for the client, or just write the data files. The default is to write everything necessary. Use FALSE if you want to have only a single instance of the client files and only write data and plots with this function.

client.basedir
: Path to client files. Default: system.file("htdocs/client/dist-apss", package = "AnalysisP
Probably should not be modified except during development work on the client.

`app.html` Path to application `.html` file, relative to `client.basedir`. Default: "analysis-page-server-static.html".

`build.full.url` Boolean, default is the same as `write.client`. For the return value build a full URL starting with "file://", using the full (normalized) path to output directory and index.html, then the full query string. If FALSE then just return the query string.

`data.subdir` Subdirectory of `outdir` which will hold the data files. Special value of "." means to put them in `outdir` itself and not create a subdirectory. Default: "data" if `write.client` is TRUE and "." if it is FALSE.

`randomize.filename` Boolean, default FALSE. Should I add some random characters to the names of the plot and dataset files? Sometimes web browsers do not refresh these files properly and so adding these random characters can overcome these stubborn cache issues.

Details

Create interactive AnalysisPage plots from static data. An index.html file will be created which, when opened, will have all the data and interactivity.

Also in that subdirectory there will be other HTML and Javascript files as necessary.

Finally, your SVGs and data will be stored in subdirectories.

The first two columns of the data frame should be x and y coordinates of the points (or regions) in the plot that you want to associate with the rows of the data frame.

Value

List with two components. First is `$URL`, which is the URL to index.html file, or, if `build.full.url = FALSE` then just the query string. and second is `$paths.list`, which lists the paths to all of the written plot and data files, in the format described in `link{static.analysis.page.query.string}` (and suitable for passing to that function as the parameter of the same name).

Author(s)

Brad Friedman

Examples

```
message("See vignette StaticContent.html")
```

---

static.analysis.page.query.string

static.analysis.page.query.string

Description

Build the query string for a static analysis page

Usage

```
static.analysis.page.query.string(paths.list)
```

Arguments

paths.list	paths.list is (for example) the return value from .write.plots.and.data.for.static.analysi It is a list whose entries correspond to the datasets on your page. Each entry is in turn a list with a $plot and/or $data element, each of which is a URL (but could be relative to the application .html file) to the encoded SVG and JSON data files.

Details

All static analysis pages are deployed on top of the same HTML/Javascript/CSS stack. To point the client to the correct plots and data, their paths are encoded into the query part of the URL. This function performs that encoding.

The query string will begin with "#".

To form a URL to view your data, simply append this query string to the URL for the application .html file.

Value

Query string, starting with "#"

Author(s)

Brad Friedman

---

test.package *test.package*

Description

Run the RUnit test harness for this package

Usage

```
test.package(pattern = "^test.*R$", package = "AnalysisPageServer")
```

Arguments

pattern	String. Regular expression. Only filenames matching
package	Name of package to test. Default: "AnalysisPageServer" this expression will be included in the test harness. Default: "^test.*R$".

Details

Run the RUnit test harness for this package

Value

RUnitTestData

Author(s)

Brad Friedman, Cory Barr

See Also

runTestSuite, require

`trig.registry` *trig.registry*

Description

Build a toy registry for examples and testing

Usage

```
trig.registry()
```

Details

The toy registry has a sine and a cosine page and the scattergram page.

Value

AnalysisPageRegistry

Author(s)

Brad Friedman

Examples

```
tr <- trig.registry()
pages(tr)
```

`trigger.event` *trigger.event*

Description

Trigger a registered Event

Usage

```
trigger.event(registry, event, ...)
```

Arguments

`registry`	EventRegistry
`event`	Name of event to trigger
`...`	Further parameters are passed to each handler in turn.

Details

Trigger a registered Event.

Every handler is called in turn. If any handler returns a value with a "CatchEvent" attribute set to TRUE then no further handlers are called. That attribute is removed from the return value and the value is returned to the triggering context. Otherwise the return value of only the last function is called.

If no handlers are registered then NULL is returned.

If no Event exists of that name then an error is thrown.

Value

See Details

Author(s)

Brad Friedman

Examples

```
r <- new.event.registry()
add.event(r, "mouseclick")
add.event.handler(r, "mouseclick", function(x, y)  message("Mouse clicked at coordinates (", x, ", ", y, ")"))
trigger.event(r, "mouseclick", x = 30, y = 50)
```

---

tryKeepConditions *tryKeepConditions*

Description

Try-catch wrapper, keeping error traceback and conditions

Usage

```
tryKeepConditions(expr)
```

Arguments

expr	Expression to evaluate

Details

This is a try-catch wrapper. It returns a list with four elements:

`$value` The value of the evaluated expression, or NULL if an error was thrown and execution did not complete

`$messages` A list of `message` objects, each being a message thrown during the execution, in order

`$warnings` A list of `condition` objects, each being a message thrown during the execution, in order

`$error` NULL if there was no error, otherwise the `error` object, which can then be passed to `getTraceback` to retrieve the error

The elements of the $messages, $warnings and $error are all actually two-element lists, the first being the condition object itself (named $message, $warning or $error) and the second begin the call stack as returned by sys.calls() and named $calls.

The class of this object is set as "AnalysisPageValueWithConditions"

Value

AnalysisPageValueWithConditions

Author(s)

Brad Friedman

See Also

vwc.is.error

vwc.conditions vwc.error vwc.error.condition vwc.error.traceback vwc.is.error vwc.messages vwc.messages.conditions vwc.messages.tracebacks vwc.n vwc.n.messages vwc.n.warnings vwc.tracebacks vwc.value vwc.warnings vwc.warnings.conditions vwc.warnings.tracebacks

Examples

```
value.with.warning <- tryKeepConditions({warning("warning message"); 3})
value.with.error <- tryKeepConditions({stop("err message")})
```

---

tryKeepTraceback *tryKeepTraceback*

Description

Wrapper around try-catch

Usage

```
tryKeepTraceback(expr)
```

Arguments

expr	Expression to evaluate

Value

Result of expression or error if thrown

Examples

```
x <- tryKeepTraceback(stop("no way"))
if(is(x, "try-error"))  cat(getTraceback(x))
```

`uniquify.ids.in.svg.files` *uniquify.ids.in.svg.files*

Description

Uniquify IDs in a set of SVG filename

Usage

```
uniquify.ids.in.svg.files(svg.filenames,
  suffixes = unique.words(length(svg.filenames)),
  prefixes = .default.uniquify.ids.prefixes)
```

Arguments

`svg.filenames`	Paths to SVG files
`suffixes`	Charvec. Suffixes to add to IDs, corresponding to `svg.filenames`. These is added after the word "glyph" and before the next character. An underscore character is added on both sides, too, to separate your suffix visually from the word "glyph" and the numbers after it. Ignored if `new.glyph.word` is provided.
`prefixes`	Named list. The names are the tokens that need to be replaced, such as "glyph" and "path". The values are charvec of prefixes. Only when those words appear after one of their prefixes is it substituted. Default is taken from `AnalysisPageServer:::.default.`

Details

The SVG files made by R use identifiers like "glyph1-3", "glyph1-4" etc. In particular these are used to define paths for different characters in order to render text. Also there are "clip1", "clip2" etc which represent clip paths, which I think limits the viewable area in a layer of a plot, but whatever it is looks awful if it goes wrong.

If multiple SVG files are embedded in the same page then this is a big problem because they will all share the same namespace and may grab the paths defined in a different file.

This function will process a set of SVG files replacing each word "glyph" with a file-specfic suffix like "glyph_123_", and each word "clip" with "clip_123_" You can provide the suffixes explicitly or let this function generate some random words, one for each file.

This function does search-and-replace with these two cleverness-es:

1. It uses C++ so it is faster (I hope) than calling `gsub`.
2. It checks the context of the words "glyph" and "clip", so if you had an SVG containing that word other than identifier it should be preserved. This is not 100% bulletproof since it doesn't actually parse the SVG file but it should be 99.99% bulletproof, unless you go out of your way to break it.

Value

Nothing, modifies SVG file in place.

Note

Typical (and recommended) usage is to only provide the `svg.filenames` argument and leave the rest as defaults.

Author(s)

Brad Friedman

Examples

```
svg.filenames <- sapply(1:2, function(i)  {
  fn <- tempfile(fileext = ".svg")
  svg(fn)
  plot(1:10, main = paste("Plot", i), col = i)
  dev.off()
  fn
})
grep("glyph", readLines(svg.filenames[1]), value = TRUE)
uniquify.ids.in.svg.files(svg.filenames)
grep("glyph", readLines(svg.filenames[1]), value = TRUE)
```

---

urlDecode *urlDecode a string*

Description

When a function of this name is available from the global environment (such as when running under RApache) then that function is used. Otherwise a pure R implementation is provided.

Arguments

x	Character vector of strings to urlDecode

Value

Character vector of same length as x containing decoded strings

Author(s)

Brad Friedman

urlEncode *urlEncode a string*

Description

When a function of this name is available from the global environment (such as when running under RApache) then that function is used. Otherwise a pure R implementation is provided.

Arguments

x	Character vector of strings to urlEncode

Value

Character vector of same length as x containing encoded strings

Author(s)

Brad Friedman

validate.array.param.value *validate.compound.param.value*

Description

Validate an array-type AnalysisPageParam value

Usage

```
validate.array.param.value(app, val, transform.labeled = FALSE)
```

Arguments

app	AnalysisPageParam
val	Candidate value
transform.labeled	Passed through to validate.param.value. Default: FALSE.

Details

Validate an array-type AnalysisPageParam value:

1. val must be a list.
2. length(val) must be in the acceptable range (between app$min and app$max inclusive).
3. Each element of val must be validate by app$prototype. transform.labeled is passed on.

Value

value, possibly with elements transformed

Author(s)

Brad Friedman

`validate.bool.param.value`
validate.bool.param.value

Description

Validate a boolean-type AnalysisPageParam value

Usage

```
validate.bool.param.value(app, val)
```

Arguments

`app`	AnalysisPageParam
`val`	Candidate value

Details

Validate a boolean-type AnalysisPageParam value:

1. `val` must be length 1
2. `val` must be a logical

Value

`val` unmodified

Author(s)

Brad Friedman

`validate.combobox.param.value`
Validate a combobox-type AnalysisPageParam value

Description

Alias for `validate.labeled.param.value`

Usage

```
validate.combobox.param.value(app, val, transform.labeled = FALSE)
```

Arguments

app	AnalysisPageParam
val	Candidate value
transform.labeled	Logical. See details.

Value

Candidate value, possibly transformed into list.

`validate.compound.param.value`
validate.compound.param.value

Description

Validate a compound-type AnalysisPageParam value

Usage

```
validate.compound.param.value(app, val, transform.labeled = FALSE)
```

Arguments

app	AnalysisPageParam
val	Candidate value
transform.labeled	Passed through to `validate.param.value` of children. Default: FALSE.

Details

Validate a compound-type AnalysisPageParam value:

1. `val` must be a list. If length 0 then it is valid with no further checks.
2. `names(val)` must not have any duplicates.
3. `names(val)` must be a subset of `names(app$children)`.
4. Each of the values in the list must be validated by the corresponding child. `transform.labeled` is passed on.

Value

value, possibly with labeled children transformed

Author(s)

Brad Friedman

`validate.file.param.value` *validate.file.param.value*

Description

Validate a file-type AnalysisPageParam value

Usage

```
validate.file.param.value(app, val)
```

Arguments

`app`	AnalysisPageParam
`val`	Candidate value

Details

Validate a file-type AnalysisPageParam value:

Current all values are invalid and result in an error being thrown. The reason for this is that the use case I have in mind is to check values when constructing a URL, and I don't think file-uploads will be allowed to be URL-encoded. So I can't think what values will be valid. Once I have another use case where they ought be valid then I will know what form they should take.

Value

Never returns

Author(s)

Brad Friedman

`validate.labeled.param.value` *validate.labeled.param.value*

Description

Validate a labeled AnalysisPageParam value

Usage

```
validate.labeled.param.value(app, val, transform.labeled = FALSE)
```

Arguments

`app`	AnalysisPageParam
`val`	Candidate value
`transform.labeled`	Logical. See details.

Details

Validate a labeled AnalysisPageParam value. Currently the labeled param types are "combobox" and "select".

Unnamed scalars are OK and named scalars are also OK.

If the scalar is unnamed then its own name will be applied: `names(val) <- val`.

If `transform.labeled` is set then instead of returning the candidate value as-is, it is transformed into `list(v=real.value, r=readable.value)`.

If `$allow.multiple == TRUE` then `length(val) > 1` is OK. The encoding is simply `list(v=real.values, r=readabl`
where `real.values` and `readable.values` are equal-length vectors.

Value

Candidate value, possibly transformed into list.

Author(s)

Brad Friedman

`validate.param.list` *validate.param.list*

Description

Validate a list of parameter values for an AnalysisPageParamSet

Usage

```
validate.param.list(pset, plist, transform.labeled = FALSE)
```

Arguments

`pset`	AnalysisPageParamSet
`plist`	List of parameter values for a subset of the parameters.
`transform.labeled`	If TRUE, then labeled parameters (combobox and select) will be transformed as necessary so that they have the `list(v=value, r=readable.value)` format.> If FALSE (default), then they will be left as-is.

Details

Validate a list of parameter values for an AnalysisPageParamSet.

Value

Copy of `plist`, possibly transformed.

Author(s)

Brad Friedman

validate.param.value *validate.param.value*

Description

Validate a parameter value for an AnalysisPageParam

Usage

```
validate.param.value(app, val, transform.labeled = FALSE)
```

Arguments

app	AnalysisPageParam
val	Candidate value
transform.labeled	Logical. Should the parameter value be transformed if necessary to have the list(v=value, r=readable.value) format?

Details

This function dispatches to the type-specific validator.

transform.labeled is passed on only if that validator accepts such an argument. This should be just combobox and select types, and indicates that the parameter value should be transformed to have the list(v=value, r=readable.value) format.

Value

The candidate value, possibly transformed. Throws error if the value is invalid.

Author(s)

Brad Friedman

Examples

```
sp <- simple.param("foo")
validate.param.value(sp, 3)
```

validate.select.param.value *validate.select.param.value*

Description

Validate a select-type AnalysisPageParam value

Usage

```
validate.select.param.value(app, val, transform.labeled = FALSE)
```

Arguments

app	AnalysisPageParam
val	Candidate value
transform.labeled	Logical. See details.

Details

Validate a select-type AnalysisPageParam value.

1. The value must be a real value among the choices.
2. The value must validate by `validate.labeled.param.value`

If `transform.labeled` is set then instead of returning the candidate value as-is, it is transformed into `list(v=real.value, r=readable.value)`.

Value

Candidate value, possibly transformed into list.

Author(s)

Brad Friedman

validate.text.param.value *validate.text.param.value*

Description

Validate a text-type AnalysisPageParam value

Usage

```
validate.text.param.value(app, val)
```

Arguments

app	AnalysisPageParam
val	Candidate value

Details

Validate a text-type AnalysisPageParam value:

1. val must be a scalar (length-1 atomic)
2. val must be unnamed

Value

val unmodified

Author(s)

Brad Friedman

---

vwc.conditions *vwc.conditions*

Description

Return condition object(s) for an AnalysisPageValueWithConditions

Usage

```
vwc.conditions(vwc, type = "messages")
```

Arguments

vwc	AnalysisPageValueWithConditions
type	"messages" "warnings" or "error"

Details

Return condition object(s) for an AnalysisPageValueWithConditions

Value

List of condition objects for "messages" or "warnings" or a single condition object or NULL for "error".

Author(s)

Brad Friedman

See Also

vwc.error.condition, vwc.messages.conditions, vwc.warnings.conditions

Examples

```
vwc <- tryKeepConditions({message("whatever"); warning("warning message"); 3})
vwc.conditions(vwc, "messages")
vwc.conditions(vwc, "warnings")
vwc.conditions(vwc, "error")
```

`vwc.error` *vwc.error*

Description

Access error message from AnalysisPageValueWithConditions

Usage

```
vwc.error(vwc)
```

Arguments

vwc	AnalysisPageValueWithConditions

Details

Access error message from AnalysisPageValueWithConditions

Value

Charvec of warning messages

Author(s)

Brad Friedman

Examples

```
vwc <- tryKeepConditions({stop("This isn't going to work")})
vwc.error(vwc)
```

`vwc.error.condition` *vwc.error.condition*

Description

Get condition object for error

Usage

```
vwc.error.condition(vwc)
```

Arguments

vwc	AnalysisPageValueWithCondition

Details

Get condition object for error

Value

condition object for error, or NULL

Author(s)

Brad Friedman

Examples

```
vwc <- tryKeepConditions({stop("error!")})
vwc.error.condition(vwc)
```

---

vwc.error.traceback *vwc.error.traceback*

Description

Get traceback for error

Usage

```
vwc.error.traceback(vwc)
```

Arguments

vwc	AnalysisPageValueWithConditions

Details

Get traceback for error

Value

Charvecs, or NULL if there was no error. (In this it differs from vwc.tracebacks, which throws an error).

Author(s)

Brad Friedman

See Also

vwc.tracebacks

Examples

```
f <- function(msg)  {
  stop(msg)
}
vwc <- tryKeepConditions({
  f("foo")
})
vwc.error.traceback(vwc)
```

---

vwc.is.error *vwc.is.error*

Description

Predicate to test if an AnalysisPageValueWithConditions had an error

Usage

```
vwc.is.error(vwc)
```

Arguments

vwc	AnalysisPageValueWithConditions, as returned by tryKeepConditions

Details

Predicate to test if an AnalysisPageValueWithConditions had an error

Value

Logical

Author(s)

Brad Friedman

Examples

```
vwc <- tryKeepConditions({3+5})
vwc.is.error(vwc)

vwc <- tryKeepConditions({stop("error!")})
vwc.is.error(vwc)
```

vwc.messages *vwc.messages*

Description

Return condition Messages for an AnalysisPageValueWithConditions

Usage

```
vwc.messages(vwc, type = "messages")
```

Arguments

vwc	AnalysisPageValueWithConditions
type	Type of conditions. Must be "messages" or "warnings". Default: messages.

Details

Return condition Messages for an AnalysisPageValueWithConditions.

Value

Charvec of message strings

Author(s)

Brad Friedman

Examples

```
vwc <- tryKeepConditions({message("I've got something to say.")})
vwc.messages(vwc)
```

vwc.messages.conditions *vwc.messages.conditions*

Description

Get condition object for messages

Usage

```
vwc.messages.conditions(vwc)
```

Arguments

vwc	AnalysisPageValueWithCondition

Details

Get condition object for messages

Value

List of condition objects for messages (might be of length 0)

Author(s)

Brad Friedman

Examples

```
vwc <- tryKeepConditions({message("I've got something to say.")})
vwc.messages.conditions(vwc)
```

---

vwc.messages.tracebacks

vwc.messages.tracebacks

Description

Get list of messages tracebacks

Usage

```
vwc.messages.tracebacks(vwc)
```

Arguments

vwc	AnalysisPageValueWithConditions

Details

Get list of tracebacks for messages.

Value

List of charvecs.

Author(s)

Brad Friedman

See Also

vwc.tracebacks

Examples

```
f <- function(msg)  {
  message(msg)
}
vwc <- tryKeepConditions({
  f("foo")
  f("bar")
})
vwc.messages.tracebacks(vwc)
```

---

vwc.n *vwc.n*

Description

Get number of conditions of a given type for an AnalysisPageValueWithConditions

Usage

```
vwc.n(vwc, type)
```

Arguments

vwc	AnalysisPageValueWithConditions
type	"messages", "warnings" or "error"

Details

Get number of conditions of a given type for an AnalysisPageValueWithConditions.

Value

Number of conditions. (For "error" it can only be 0 or 1, and is equivalent to calling `as.integer(vwc.is.error())`).

Author(s)

Brad Friedman

Examples

```
vwc <- tryKeepConditions({stop("This isn't going to work")})
vwc.n(vwc, "error")
```

vwc.n.messages *vwc.n.messages*

Description

Get number of messages for an AnalysisPageValueWithConditions

Usage

```
vwc.n.messages(vwc)
```

Arguments

vwc	AnalysisPageValueWithConditions

Details

Get number of messages for an AnalysisPageValueWithConditions

Value

Non-negative Integer

Author(s)

Brad Friedman

Examples

```
vwc <- tryKeepConditions({message("Message")})
vwc.n.messages(vwc)
```

vwc.n.warnings *vwc.n.warnings*

Description

Get number of warnings for an AnalysisPageValueWithConditions

Usage

```
vwc.n.warnings(vwc)
```

Arguments

vwc	AnalysisPageValueWithConditions

Details

Get number of warnings for an AnalysisPageValueWithConditions

Value

Non-negative Integer

Author(s)

Brad Friedman

Examples

```
vwc <- tryKeepConditions({warning("I'm warning you.")})
vwc.n.warnings(vwc)
```

`vwc.tracebacks` *vwc.tracebacks*

Description

Return traceback(s) for an AnalysisPageValueWithConditions

Usage

```
vwc.tracebacks(vwc, type = "messages")
```

Arguments

vwc	AnalysisPageValueWithConditions
type	Type of conditions. Must be "messages", "warnings" or "error". Default: "messages". A (new) error is thrown if type is "error" but the vwc is not an error (that is, does not have an error, or more specifically, `!vwc.is.error(vwc)`).

Details

Return traceback(s) for an AnalysisPageValueWithConditions

Value

For "messages" or "warnings" it gives a list of Charvecs of tracebacks, as built by `getTraceback`. For "error" it only gives a single charvec, since there is only one error.

Author(s)

Brad Friedman

Examples

```
f <- function(msg)  {
  warning(msg)
}
vwc <- tryKeepConditions({
  f("foo")
  f("bar")
})
vwc.tracebacks(vwc, "warnings")
```

vwc.value *vwc.value*

Description

Get value of any AnalysisPageValueWithConditions

Usage

```
vwc.value(vwc)
```

Arguments

vwc	AnalysisPageValueWithConditions

Details

Get value of any AnalysisPageValueWithConditions. If an error was thrown then the value will be NULL.

Value

Value of original evaluated expression, or NULL if an error was thrown.

Author(s)

Brad Friedman

Examples

```
vwc <- tryKeepConditions({3+5})
vwc.value(vwc)
```

vwc.warnings *vwc.warnings*

Description

Access warning messages from AnalysisPageValueWithConditions

Usage

```
vwc.warnings(vwc)
```

Arguments

vwc	AnalysisPageValueWithConditions

Details

Access warning messages from AnalysisPageValueWithConditions

Value

Charvec of warning messages

Author(s)

Brad Friedman

Examples

```
vwc <- tryKeepConditions({warning("I'm warning you!"); warning("Again")})
vwc.warnings(vwc)
```

`vwc.warnings.conditions`

vwc.warnings.conditions

Description

Get condition object for warnings

Usage

```
vwc.warnings.conditions(vwc)
```

Arguments

vwc	AnalysisPageValueWithCondition

Details

Get condition object for warnings

Value

List of condition object for warnings (might be of length 0 if no warnings were thrown).

Author(s)

Brad Friedman

Examples

```
vwc <- tryKeepConditions({warning("I'm warning you!")})
vwc.warnings.conditions(vwc)
```

vwc.warnings.tracebacks *vwc.warnings.tracebacks*

Description

Get list of warnings tracebacks

Usage

```
vwc.warnings.tracebacks(vwc)
```

Arguments

vwc	AnalysisPageValueWithConditions

Details

Get list of tracebacks for warnings.

Value

List of charvecs.

Author(s)

Brad Friedman

See Also

vwc.tracebacks

Examples

```
f <- function(msg)  {
  warning(msg)
}
vwc <- tryKeepConditions({
  f("foo")
  f("bar")
})
vwc.warnings.tracebacks(vwc)
```

Index