Package 'rameritrade'

September 21, 2020

Title TD Ameritrade API Interface for R

Version 0.0.0.9000

URL https://github.com/tonytrevisan/rameritrade/, https://developer.tdameritrade.com/

BugReports https://github.com/tonytrevisan/rameritrade/issues

Description R package to interface with the TD Ameritrade API that can be found at https://developer.tdameritrade.com/. This package contains a suite of functions that allow for authentication, trading, price requests, account information etc. In order to use this package you will need to register a TD Ameritrade developer app following the link above. You will also need a TD Ameritrade brokerage account to link to your app. See the README for more details. PLease note, the fate of the TD Ameritrade API is in question after the Schwab acquisition. Hopefully the API stays in tact and this package will be updated if needed.

License GPL-3

Encoding UTF-8

LazyData true

Imports httr,
urltools (>= 1.7.3),
lubridate,
dplyr,
jsonlite,
magrittr

RoxygenNote 7.1.1

R topics documented:

`act_data_df` *Get a dataframe of data for the Access Token TD Brokerage Accounts*

Description

The output will be a dataframe of requested details for TD Ameritrade accounts linked to the access token. Use `act_data_list` for an output in list form. Request can be for current positions, current balances, or current day orders.

Usage

```
act_data_df(
  dataType = c("balances", "positions", "orders"),
  accessToken = NULL
)
```

Arguments

`dataType`	'balances' for current cash balances, 'positions' for current account positions, 'orders' for orders entered on the current day
`accessToken`	A valid Access Token must be set using `auth_new_accessToken`. The most recent access token will be used by default unless one is manually passed into the function

Value

a dataframe of requested account details

Examples

```
## Not run:

### A valid refresh token can be fed into the function below for a new Access Token
positions = act_data_df('positions')


## End(Not run)
```

`act_data_list` *Get a list of data for the Access Token TD Brokerage Accounts*

Description

The output will be a list of requested details for TD Ameritrade accounts linked to the access token. Use `act_data_df` for a cleaner output in a dataframe. Request can be for current positions, current balances, or current day orders.

Usage

```
act_data_list(
  dataType = c("balances", "positions", "orders"),
  accessToken = NULL
)
```

Arguments

`dataType`	'balances' for current cash balances, 'positions' for current account positions, 'orders' for orders entered on the current day
`accessToken`	A valid Access Token must be set using `auth_new_accessToken`. The most recent access token will be used by default unless one is manually passed into the function

Value

a list of requested account details

Examples

```
## Not run:

### A valid refresh token can be fed into the function below for a new Access Token
positions = act_data_list('positions',accessToken)


## End(Not run)
```

`auth_init_loginURL` *Auth Step 1: Initial Log In URL*

Description

Create URL to grant App access to a TD Brokerage account

Usage

```
auth_init_loginURL(callbackURL, consumerKey)
```

Arguments

`callbackURL`	User generated Callback URL associated with registered TD app
`consumerKey`	TD generated Consumer key associated with registered TD app. Essentially an API key.

Details

To access the TD Ameritrade API the user will need to create an account on the TD Ameritrade Developer site. Note, this is separate and distinct from a TD Brokerage account. Once logged in to the developer site, use My Apps to register an application. The two inputs to this function come from the new application. A Consumer Key functions like an API key and is provided by TD Ameritrade. A Callback URL is created by the user. The output of this function will be a URL that leads to a landing page that allows a user to log into a TD Ameritrade Brokerage account.

The Callback URL can be anything. The example below assumes the Callback URL is https://myTDapp. The Consumer Key is auto generated and can be found under My Apps > Keys. This function will use these two inputs to generate a URL where the user can log in to their standard TD Ameritrade Brokerage Account and grant the application access to the brokerage account, enabling the API. The Authorization Code generated at the end of the log in process will feed into `auth_init_refreshToken`. For questions, please reference the TD Ameritrade Authentication FAQ

Value

login url to grant app permission to TD Brokerage account

See Also

`auth_init_loginURL` for login url, `auth_init_refreshToken` for initial Refresh Token, `auth_new_accessToken` for a new Access Token, `auth_new_refreshToken` to reset an existing Refresh Token before expiration

Other authentication functions: `auth_init_refreshToken()`, `auth_new_accessToken()`, `auth_new_refreshToken()`

Examples

```
## Not run:

### Visit the URL below to log in to a TD Brokerage account
### Once a successful log in is completed the landing page will be a blank page
### The full URL of the landing page is the Authorization Code for auth_init_refreshToken

loginURL = auth_init_loginURL('https://myTDapp',
                              'consumerKey')


## End(Not run)
```

`auth_init_refreshToken`

Auth Step 2: Obtain Initial Refresh Token

Description

Get an initial Refresh Token using the Authorization Code

Usage

```
auth_init_refreshToken(callbackURL, consumerKey, authcode_url)
```

Arguments

`callbackURL`	User generated Callback URL associated with registered TD app
`consumerKey`	TD generated Consumer key associated with registered TD app. Essentially an API key.
`authcode_url`	Authorization URL Code generated from a successful log in to the `auth_init_loginURL`

Details

Once a URL has been generated using `auth_init_loginURL`, a user can visit that URL to log into a TD brokerage account, granting the TD app access to the account. Once the button "Allow" is pressed, the user will be redirected, potentially to "This site can't be reached". This indicates a successful log in. The URL of this page contains the Authorization Code. Paste the entire URL, not just the Authorization Code, into this function. The code will be an extremely long alpha numeric string. The output of this function will be a Refresh Token which will be used to gain access to the TD Brokerage account(s) going forward, avoiding the auth_init functions going forward. The Refresh Token will last for 90 days, but the user can use `auth_new_refreshToken` to reset the token before expiration. If the Refresh Token expires, a new Authorization Code will need to be generated by logging into the URL from `auth_init_loginURL`

The Refresh Token output should be saved in a very safe location, but also accessible. It will be needed to generate an Access Token, which is used for general account access. The Access Token expire every 30 minutes.

Value

Refresh Token that is valid for 90 days

See Also

auth_init_loginURL for login url, auth_init_refreshToken for initial Refresh Token, auth_new_accessToken for a new Access Token, auth_new_refreshToken to reset an existing Refresh Token before expiration

Other authentication functions: auth_init_loginURL(), auth_new_accessToken(), auth_new_refreshToken()

Examples

```
## Not run:

### The URL after a successful login can be fed into the function below
refreshToken = auth_init_refreshToken('https://myTDapp','consumerKey',
                                      'https://myTDapp/?code=Auhtorizationcode')

saveRDS(refreshToken,'/secure/location/')


## End(Not run)
```

---

auth_new_accessToken *Auth Step 3: Get Access Token*

Description

Get a new Access Token using a valid Refresh Token

Usage

```
auth_new_accessToken(refreshToken, consumerKey)
```

Arguments

refreshToken	An existing Refresh Token generated using auth_init_refreshToken or auth_new_refreshToken
consumerKey	TD generated Consumer key associated with registered TD app. Essentially an API key.

Details

An Access Token is required for most functions within rameritrade. It serves as a user log in to a TD Brokerage account. The token is valid for 30 minutes and allows the user to place trades, get account information, get order history, pull historical stock prices, etc. A Refresh Token is required to generate an Access Token. Functions auth_init_refreshToken or auth_new_refreshToken can be used to generate Refresh Tokens which stay valid for 90 days. The Consumer Key is generated automatically when an App is registered on the TD Ameritrade Developer site. By default,

the Access Token is stored into options and will automatically be passed to downstream function. However, the user can also submit an Access Token manually if multiple tokens are in use (for example: when managing more than one log in.)

When running this function manually (i.e. through RStudio), the function will check for a default Access Token. If the default Access Token has not expired, the user will be prompted to verify a new Access Token is desired. This may be the case if more than one TD log in is being used. When running this function in a non-interactive environment (i.e. CRON Job), the default behavior will be to refresh the Access Token.

Value

Access Token that is valid for 30 minutes. By default it is stored in options.

See Also

auth_init_loginURL for login url, auth_init_refreshToken for initial Refresh Token, auth_new_accessToken for a new Access Token, auth_new_refreshToken to reset an existing Refresh Token before expiration

Other authentication functions: auth_init_loginURL(), auth_init_refreshToken(), auth_new_refreshToken()

Examples

```
## Not run:

### A valid Refresh Token can be fed into the function below for a new Access Token
refreshToken = readRDS('/secure/location/')
accessToken = auth_new_accessToken(refreshToken,'APPCONSUMERKEY')


## End(Not run)
```

---

auth_new_refreshToken *Auth Step 4: New Refresh Token before expiration*

Description

Get a new Refresh Token using an existing Refresh Token

Usage

```
auth_new_refreshToken(refreshToken, consumerKey)
```

Arguments

refreshToken	An existing Refresh Token generated using auth_init_refreshToken or auth_new_refreshToken
consumerKey	TD generated Consumer key associated with registered TD app. Essentially an API key.

Details

A Refresh Token is used to generate Access Tokens through the function auth_new_accessToken The initial Refresh Token must be generated manually using a URL specific to a registered app. Use auth_init_loginURL to generate an app specific URL and then use auth_init_refreshToken to process the Authorization Code and generate the initial Refresh Token. The Refresh Token will expire every 90 days. This function uses the current Refresh Token to generate a new Refresh Token, avoiding the manual process above. TD indicates they do look for frequent Refresh Token generation. This function should be used conservatively and as close to every 90 days as possible.

When running this function manually (i.e. through RStudio), the function will check the days left until expiration for the Refresh Token being passed. If the remaining time is greater than 15 days, the user will be prompted to verify that a new Refresh Token should be created. The user can select to request a new token, but there is no net benefit in doing so and TD encourages limiting new token generation. When running this function in a non-interactive environment (i.e. CRON Job), if the remaining time until expiration is greater than 15 days, the default behavior will be to NOT reset the Refresh Token because the new token will have the same access and capabilities as the existing token.

Value

Refresh Token that is valid for 90 days

See Also

auth_init_loginURL for login url, auth_init_refreshToken for initial Refresh Token, auth_new_accessToken for a new Access Token, auth_new_refreshToken to reset an existing Refresh Token before expiration

Other authentication functions: auth_init_loginURL(), auth_init_refreshToken(), auth_new_accessToken()

Examples

```
## Not run:

### A valid Refresh Token can be fed into the function below for a new Refresh Token
currefreshToken = readRDS('/secure/location/')
newrefreshToken = auth_new_refreshToken(currefreshToken,'APPCONSUMERKEY')
saveRDS(newrefreshToken,'/secure/location/')


## End(Not run)
```

---

market_hours *Get Market Hours*

Description

Returns a list output for a specified day and market that details the trading window for that day

Usage

```
market_hours(
  marketDate = Sys.Date(),
  marketType = c("EQUITY", "OPTION", "BOND", "FUTURE", "FOREX"),
  accessToken = NULL
)
```

Arguments

marketDate	The market date to pull details for
marketType	The asset class for hour: 'EQUITY','OPTION','BOND','FUTURE','FOREX'
accessToken	A valid Access Token must be set using auth_new_accessToken. The most recent access token will be used by default unless one is manually passed into the function

Value

List output of times and if the specified date is a trading day

Examples

```
## Not run:

### Market hours for the current date
market_hours()


## End(Not run)
```

---

option_chain *Get Options Chain*

Description

Search an Option Chain for a specific ticker

Usage

```
option_chain(
  ticker,
  strikes = 10,
  inclQuote = TRUE,
  startDate = Sys.Date(),
  endDate = Sys.Date() + months(12),
  accessToken = NULL
)
```

Arguments

ticker	underlying ticker for the options chain
strikes	the number of strikes above and below the current strike
inclQuote	include pricing details (will be delayed if account is set for delayed quotes)
startDate	the start date for expiration (should be greater than or equal to today). format yyyy-mm-dd
endDate	the end date for expiration. format yyyy-mm-dd
accessToken	A valid Access Token must be set using auth_new_accessToken. The most recent access token will be used by default unless one is manually passed into the function

Details

Return a list containing two data frames. The first is the underlying data for the symbol. The second item in the list is a data from that contains the options chain for the specified ticker

Value

a list of 2 data frames - underlying and options chain

Examples

```
## Not run:

### Pull all option contracts expiring over the next 6 months
### with 5 strikes above and below the at-the-money price
option_chain(ticker='SPY',strikes=5,endDate = Sys.Date() + months(6))


## End(Not run)
```

---

order_cancel *Cancel an Open Order*

Description

Pass an order ID and Account number to cancel an existing open order

Usage

```
order_cancel(orderId, accountNumber, accessToken = NULL)
```

Arguments

orderId	A valid TD Ameritrade Order ID
accountNumber	The TD brokerage account number associated with the Access Token
accessToken	A valid Access Token must be set using auth_new_accessToken. The most recent access token will be used by default unless one is manually passed into the function

Value

order API URL

Examples

```
 ## Not run:

order_cancel(orderId=123456789,accountNumber=987654321)


## End(Not run)
```

---

order_detail *Get Details for a Single Order*

Description

Pass an order ID and Account number to get details such as status, quantity, ticker, executions (if applicable), etc.

Usage

```
order_detail(orderId, accountNumber, accessToken = NULL)
```

Arguments

orderId	A valid TD Ameritrade Order ID
accountNumber	The TD brokerage account number associated with the Access Token
accessToken	A valid Access Token must be set using auth_new_accessToken. The most recent access token will be used by default unless one is manually passed into the function

Value

list of order details

Examples

```
## Not run:

order_detail(orderId=123456789,accountNumber=987654321)


## End(Not run)
```

`order_place` *Place Order for a specific account*

Description

Place trades through the TD Ameritrade API using a range of parameters.

Usage

```
order_place(
  accountNumber,
  ticker,
  quantity,
  instruction,
  orderType = "MARKET",
  limitPrice = NULL,
  stopPrice = NULL,
  assetType = c("EQUITY", "OPTION"),
  session = "NORMAL",
  duration = "DAY",
  stopPriceBasis = NULL,
  stopPriceType = NULL,
  stopPriceOffset = NULL,
  accessToken = NULL
)
```

Arguments

`accountNumber`	The TD brokerage account number associated with the Access Token
`ticker`	a valid Equity/ETF or option. Use symbol_detail to confirm
`quantity`	the number of shares to be bought or sold. Must be an integer. TD has indicated they have capabilities for fractional shares but have not 'yet' enabled the feature
`instruction`	Equity instructions include buy, sell, buy_to_cover, sell_short Option instructions include buy_to_open
`orderType`	MARKET, LIMIT (requiring limitPrice), STOP (requiring stopPrice), STOP_LIMIT, TRAILING_STOP (requiring stopPriceBasis, stopPriceType, stopPriceOffset)
`limitPrice`	the limit price for a limit or stop_limit order

stopPrice	the stop price for a STOP or STOP_LIMIT order
assetType	Equity or Option. No other asset types are available at this time
session	Normal for normal market hours, AM or PM for extended market hours
duration	how long will the trade stay open without a fill: DAY, GOOD_UNTIL_CANCEL, FILL_OR_KILL
stopPriceBasis	the basis for a STOP, STOP_LIMIT, or TRAILING_STOP which include LAST, BID, ASK
stopPriceType	the link to the stopPriceBasis. VALUE for dollar difference or PERCENT for a percentage offset from the price basis
stopPriceOffset	the offset used for the stopPriceType, 10 and PERCENT is a 10 from the current price basis
accessToken	A valid Access Token must be set using auth_new_accessToken. The most recent access token will be used by default unless one is manually passed into the function

Details

A valid account and access token must be passed. An access token will be passed by default when auth_new_accessToken is executed successfully and the token has not expired, which occurs after 30 minutes. Only equities and options can be traded at this time. This function is built to allow a single trade submission. More complex trades can be executed through the API, but a custom function or submission will need to be constructed. See details below. To build more custom trading strategies, reference the TD Ameritrade API Instructions or the order sample guide. A full list of the input parameters and details can be found at the links above. Please note that in rare cases, the documentation may not be accurate in the API section, so the Order Sample guide is a better reference. TEST ALL ORDERS FIRST WITH SMALL DOLLAR AMOUNTS!!!

Value

the trade id, account id, and other basic details

Warning

TRADES THAT ARE SUCCESSFULLY ENTERED WILL BE SUBMITTED IMMEDIATELY THERE IS NO REVIEW PROCESS. THIS FUNCTION HAS 100S OF POTENTIAL COMBINATIONS AND ONLY A HANDFUL HAVE BEEN TESTED. IT IS STRONGLY RECOMMENDED TO TEST THE DESIRED ORDER ON A VERY SMALL QUANTITY WITH LITTLE MONEY AT STAKE. ANOTHER OPTION IS TO USE LIMIT ORDERS FAR FROM THE CURRENT PRICE. TD AMERITRADE HAS THEIR OWN ERROR HANDLING BUT IF A SUCCESSFUL COMBINATION IS ENTERED IT COULD BE EXECUTED IMMEDIATELY. DOUBLE CHECK ALL ENTRIES BEFORE SUBMITTING.

Examples

```
## Not run:

accountNumber = 1234567890
```

```
### Standard market buy order
order_place(accountNumber = accountNumber,ticker='AAPL',quantity = 1,instruction='buy')

### Stop limit order
order_place(accountNumber = accountNumber,ticker='AAPL',
            quantity = 1,instruction='sell',duration='good_till_cancel',
            orderType = 'stop_limit',limitPrice=98,stopPrice=100)

### Trailing Stop Order
order_place(accountNumber = accountNumber,ticker='AAPL',quantity = 1,
            instruction='sell', orderType = 'trailing_stop',stopPriceBasis = 'BID',
            stopPriceType = 'percent',stopPriceOffset = 10)

### Option Order with a limit price
order_place(accountNumber = accountNumber, ticker='SLV_091820P24.5',
            quantity = 1, instruction='BUY_TO_OPEN', duration='Day',
            orderType = 'LIMIT', limitPrice = .02, assetType = 'OPTION')


## End(Not run)
```

`order_search` *Search for orders by date*

Description

Search for orders associated with a TD account over the previous 60 days. The result is a list of three objects:

1. jsonlite formatted extract of all orders
2. all entered orders with details
3. a data frame of all executed orders with the executions

Usage

```
order_search(
  accountNumber,
  startDate = Sys.Date() - months(1),
  endDate = Sys.Date(),
  maxResults = 50,
  orderStatus = "",
  accessToken = NULL
)
```

Arguments

accountNumber	The TD brokerage account number associated with the Access Token
startDate	Orders from a certain date. Will not pull back orders older than 60 days. format yyyy-mm-dd
endDate	Filter orders that occurred before a certain date. format yyyy-mm-dd
maxResults	the max results to return in the query
orderStatus	search by order status (ACCEPTED, FILLED, EXPIRED, CANCELED, REJECTED, etc)
accessToken	A valid Access Token must be set using auth_new_accessToken. The most recent access token will be used by default unless one is manually passed into the function

Value

a list of three objects: a jsonlite formatted extract of all orders, all entered orders with details, a data frame of all executed orders with the executions

Examples

```
## Not run:

order_search(accountNumber=987654321, startDate = Sys.Date()-days(5),
             maxResult = 100, orderStatus = 'FILLED')


## End(Not run)
```

---

price_hisotry_mult *Get price history for a multiple securities*

Description

Pulls price history for a list of security based on the parameters that include a date range and frequency of the interval. Depending on the frequency interval, data can only be pulled back to a certain date. For example, at a one minute interval, data can only be pulled for 30-35 days. PLEASE NOTE: Large data requests will take time to pull back because of the looping nature. TD Does not allow bulk ticker request, so this is simply running each ticker individually. For faster and better historical data pulls, try Tiingo or FMP Cloud

Usage

```
price_hisotry_mult(
  tickers = c("AAPL", "MSFT"),
  startDate = Sys.Date() - months(1),
  endDate = Sys.Date(),
  freq = c("daily", "1min", "5min", "10min", "15min", "30min"),
  accessToken = NULL
)
```

Arguments

tickers	a vector of tickers - no more than 15 will be pulled. for bigger requests, split up the request or use Tiingo, FMP Cloud, or other free data providers
startDate	the Starting point of the data
endDate	the Ending point of the data
freq	the frequency of the interval. Can be daily, 1min, 5min, 10min, 15min, or 30min
accessToken	A valid Access Token must be set using auth_new_accessToken. The most recent access token will be used by default unless one is manually passed into the function

Value

a tibble of price data

Examples

```
## Not run:

### Use a valid access token and a vector of one or more tickers
tickHistday = price_hisotry_mult(c('GOOG','TSLA'),freq='5min')


## End(Not run)
```

---

price_history_single *Get price history for a single security*

Description

Pulls price history for a single security based on the parameters that include a date range and frequency of the interval. Depending on the frequency interval, data can only be pulled back to a certain date. For example, at a one minute interval, data can only be pulled for 30-35 days

Usage

```
price_history_single(
  ticker = "AAPL",
  startDate = Sys.Date() - months(1),
  endDate = Sys.Date(),
  freq = c("daily", "1min", "5min", "10min", "15min", "30min"),
  accessToken = NULL
)
```

Arguments

ticker	a single ticker
startDate	the Starting point of the data
endDate	the Ending point of the data
freq	the frequency of the interval. Can be daily, 1min, 5min, 10min, 15min, or 30min
accessToken	A valid Access Token must be set using auth_new_accessToken. The most recent access token will be used by default unless one is manually passed into the function

Value

a tibble of price data

Examples

```
## Not run:

### Use a valid access token and a vector of one or more tickers
tickHist5min = price_history_single(c('GOOG','TSLA'),freq='5min')


## End(Not run)
```

---

price_quote_df *Get Quotes for specified tickers in data frame form*

Description

Quotes may be delayed depending on agreement with TD Ameritrade. If the account is set up for real-time quotes then this will return real-time. Otherwise the quotes will be delayed.

Usage

```
price_quote_df(tickers = c("AAPL", "MSFT"), accessToken = NULL)
```

Arguments

tickers	One or more tickers
accessToken	A valid Access Token must be set using auth_new_accessToken. The most recent access token will be used by default unless one is manually passed into the function

Value

a data frame with quote details for each valid ticker submitted

Examples

```
## Not run:

### Use a valid access token and a vector of one or more tickers
quoteList = price_quote_df(c('GOOG','TSLA'),accessToken)


## End(Not run)
```

`price_quote_list` *Get Quotes for specified tickers in List form*

Description

Quotes may be delayed depending on agreement with TD Ameritrade. If the account is set up for real-time quotes then this will return real-time. Otherwise the quotes will be delayed.

Usage

```
price_quote_list(tickers = c("AAPL", "MSFT"), accessToken = NULL)
```

Arguments

`tickers`	One or more tickers
`accessToken`	A valid Access Token must be set using `auth_new_accessToken`. The most recent access token will be used by default unless one is manually passed into the function

Value

a list of data with quote details for each valid ticker submitted

Examples

```
## Not run:

### Use a valid access token and a vector of one or more tickers
quoteList = price_quote_list(c('GOOG','TSLA'),accessToken)


## End(Not run)
```

`rameritrade` *rameritrade: A package for using the TD Ameritrade API*

Description

The rameritrade package provides four categories of important functions:

1. authenticating into a TD Brokerage account
2. working with orders
3. getting account information
4. pulling pricing/market data

Details

Details for the functions can be found within the function help pages with examples also available within the README.

`symbol_detail` *Get ticker details*

Description

Get identifiers and fundamental data for a specific ticker

Usage

```
symbol_detail(ticker, accessToken = NULL)
```

Arguments

`ticker`	a valid ticker or symbol
`accessToken`	A valid Access Token must be set using `auth_new_accessToken`. The most recent access token will be used by default unless one is manually passed into the function

Value

data frame of ticker details

Examples

```
## Not run:

### details for Apple
symbol_detail('AAPL')


## End(Not run)
```

`transact_search` *Search for all Transaction types*

Description

Search for all Transaction types

Usage

```
transact_search(
  accountNumber,
  startDate = Sys.Date() - months(1),
  endDate = Sys.Date(),
  transType = "All",
  accessToken = NULL
)
```

Arguments

`accountNumber`	The TD brokerage account number associated with the Access Token
`startDate`	Transactions after a certain date. Will not pull back transactions older than 1 year. format yyyy-mm-dd
`endDate`	Filter transactions that occurred before a certain date. format yyyy-mm-dd
`transType`	Filter for a specific Transaction type. No entry will return all types. For example: TRADE, CASH_IN_OR_CASH_OUT, CHECKING, DIVIDEND, INTEREST, OTHER
`accessToken`	A valid Access Token must be set using `auth_new_accessToken`. The most recent access token will be used by default unless one is manually passed into the function

Details

Can pull trades as well as transfers, dividend reinvestment, interest, etc. Any activity assoicated with the account.

Value

a jsonlite data frame of transactions

Examples

```
## Not run:

### Transactions for the last 5 days
transact_search(accountNumber=987654321, startDate = Sys.Date()-days(5))


## End(Not run)
```

Index