Package ‘future.tests’

May 22, 2023

Title  Test Suite for 'Future API' Backends
Version  0.7.0
Description  Backends implementing the 'Future' API, as defined by the 'future' package, should use the tests provided by this package to validate that they meet the minimal requirements of the 'Future' API. The tests can be performed easily from within R or from outside of R from the command line making it straightforward to include them in package tests and in Continuous Integration (CI) pipelines.
Imports  future (>= 1.22.1), cli, crayon, prettyunits, sessioninfo
Suggests  commonmark, base64enc, utils, tools, data.table, ff
VignetteBuilder  future.tests
License  LGPL (>= 2.1)
Encoding  UTF-8
URL  https://future.tests.futureverse.org,
     https://github.com/HenrikBengtsson/future.tests
BugReports  https://github.com/HenrikBengtsson/future.tests/issues
RoxygenNote  7.2.3
NeedsCompilation  no
Author  Henrik Bengtsson [aut, cre, cph],
        The R Consortium [fnd] (Project was awarded an Infrastructure Steering Committee (ISC) grant in 2017)
Maintainer  Henrik Bengtsson <henrikb@braju.com>
Repository  CRAN
Date/Publication  2023-05-22 05:30:19 UTC

R topics documented:

  add_test_plan ........................................... 2
  along_test_plans .................................... 2
  check .................................................. 3
  check_plan ............................................. 4
along_test_plans

evaluate_expr ................................................. 5
load_tests ..................................................... 6
make_test ...................................................... 6
register_test .................................................. 7
run_test ....................................................... 8
run_tests ...................................................... 8
skip_test ...................................................... 9
subset_tests .................................................. 10

Index 11

add_test_plan  Add a Future Plan to Test Against

Description
Add a Future Plan to Test Against

Usage
add_test_plan(expr, substitute = TRUE)

Arguments
expr ...
substitute ...

Value
(invisibly) returns current list of test plans.

along_test_plans  Evaluate an Expression Across A Set of Future Plans

Description
Evaluate an Expression Across A Set of Future Plans

Usage
along_test_plans(
  expr,
  substitute = TRUE,
  envir = parent.frame(),
  local = TRUE,
  plans = test_plans()
)
check

Arguments

expr An R expression.
substitute ...
envir The environment where tests are run.
local Should tests be evaluated in a local environment or not.
plans A list of future plans.

Value

A list of results, one for each future plan tested against.

check Run All or a Subset of the Tests Across Future Plans

Description

Run All or a Subset of the Tests Across Future Plans

Usage

check(  
  plan = NULL,  
  tags = character(),  
  timeout = NULL,  
  settings = TRUE,  
  session_info = FALSE,  
  envir = parent.frame(),  
  local = TRUE,  
  debug = FALSE,  
  exit_value = !interactive(),  
  .args = commandArgs()  
)

Arguments

plan (character vector) One or more future strategy plans to be validated.
tags (character vector; optional) Filter test by tags. If NULL, all tests are performed.
timeout (numeric; optional) Maximum time (in seconds) each test may run before a timeout error is produced.
settings (logical) If TRUE, details on the settings are outputted before the tests start.
session_info (logical) If TRUE, session information is outputted after the tests complete.
envir The environment where tests are run.
local Should tests be evaluated in a local environment or not.
check_plan

debug (logical) If TRUE, the raw test results are printed.
exit_value (logical) If TRUE, and in a non-interactive session, then use `base::quit()` to quit R with an exit code of 0 (zero) if all tests passed with all OKs and otherwise 1 (one) if one or more test failed.
.args (character vector; optional) Command-line arguments.

Value
(list; invisible) A list of test results.

Command-line interface (CLI)

This function can be called from the shell. To specify an argument, use the format `--test-<arg_name>=<value>`. For example, `--test-timeout=600` will set argument `timeout=600`, and `--tags=lazy,rng`, or equivalently, `--tags=lazy --tags=rng` will set argument `tags=c("lazy", "rng")`.

Here are some examples on how to call this function from the command line:

Rscript -e future.tests::check --args --test-plan=sequential
Rscript -e future.tests::check --args --test-plan=multicore,workers=2
Rscript -e future.tests::check --args --test-plan=sequential --test-plan=multicore,workers=2
Rscript -e future.tests::check --args --test-plan=future.callr::callr
Rscript -e future.tests::check --args --test-plan=future.batchtools::batchtools_local

The exit code will be 0 if all tests passed, otherwise 1. You can use for instance `exit_code=$?` to retrieve the exit code of the most recent call.

Examples

```r
## Not run:
results <- future.tests::check(plan = "sequential", tags = c("rng"))
exit_code <- attr(results, "exit_code")
if (exit_code != 0) stop("One or more tests failed")

## End(Not run)
```

---

check_plan  Run All Tests

Description

Run All Tests
**evaluate_expr**

**Usage**

```r
check_plan(
    tests = test_db(),
    defaults = list(),
    timeout = getOption("future.tests.timeout", 30),
    envir = parent.frame(),
    local = TRUE
)
```

**Arguments**

- **tests**: A list of tests to subset.
- **defaults**: (optional) Named list with default argument values.
- **timeout**: Maximum time allowed for evaluation before a timeout error is produced.
- **envir**: The environment where tests are run.
- **local**: Should tests be evaluated in a local environment or not.

**Value**

Nothing.

---

**evaluate_expr**  
**Evaluate an R Expression**

**Description**

Evaluate an R Expression

**Usage**

```r
evaluate_expr(
    expr,
    envir = parent.frame(),
    local = TRUE,
    output = c("stdout+stderr", "stdout", "none"),
    timeout = +Inf
)
```

**Arguments**

- **expr**: An expression
- **envir**: The environment where tests are run.
- **local**: Should tests be evaluated in a local environment or not.
- **output**: Specifies whether standard output, standard error, or both should be captured or not.
- **timeout**: Maximum time allowed for evaluation before a timeout error is produced.
Value

Value of test expression and benchmark information.

---

load_tests

*Loads Future Tests*

**Description**

Loads Future Tests

**Usage**

```r
load_tests(
  path = ".",
  recursive = TRUE,
  pattern = ".R$",
  root = getOption("future.tests.root", Sys.getenv("R_FUTURE_TESTS_ROOT",
    system.file("test-db", package = "future.tests", mustWork = TRUE)))
)
```

**Arguments**

- **path**
  A character string specifying a test script folder or file.
- **recursive**
  If TRUE, test-definition scripts are search recursively.
- **pattern**
  Regular expression matching filenames to include.
- **root**
  (internal) An alternative file directory from where `future.tests` tests are sourced.

**Value**

(invisible) the value of `test_db()`.

---

make_test

*Make a Test*

**Description**

Make a Test
register_test

Usage

make_test(
  expr,
  title = NA_character_,
  args = list(),
  tags = NULL,
  substitute = TRUE,
  reset_workers = FALSE,
  register = TRUE
)

Arguments

expr, substitute
  The expression to be tested and whether it is passed as a character already or not.
title (character) The title of the test.
args (optional) Named arguments.
tags (optional) Character vector of tags.
reset_workers (optional) Specifies whether background workers should futures.
register If TRUE, the test is registered in the test database, otherwise not.

Value

(invisibly) A Test.

register_test

Register a Test

Description

Register a Test

Usage

register_test(test)

Arguments

test A Test.

Value

(invisibly) The Test registered.
run_test  

---

### run_test

#### Run a Test

**Description**

Run a Test

**Usage**

```r
run_test(
  test,
  envir = parent.frame(),
  local = TRUE,
  args = list(),
  defaults = list(),
  output = "stdout+stderr",
  timeout = getOption("future.tests.timeout", 30)
)
```

**Arguments**

- **test**: A Test.
- **envir**: The environment where tests are run.
- **local**: Should tests be evaluated in a local environment or not.
- **args**: Arguments used in this test.
- **defaults**: (optional) Named list with default argument values.
- **output**: If TRUE, standard output is captured, otherwise not.
- **timeout**: Maximum time allowed for evaluation before a timeout error is produced.

**Value**

Value of test expression and benchmark information.

---

run_tests  

---

### run_tests

#### Run All Tests

**Description**

Run All Tests
**Usage**

```r
run_tests(
  tests = test_db(),
  envir = parent.frame(),
  local = TRUE,
  defaults = list(),
  output = "stdout+stderr"
)
```

**Arguments**

- `tests` A list of tests to subset.
- `envir` The environment where tests are run.
- `local` Should tests be evaluated in a local environment or not.
- `defaults` (optional) Named list with default argument values.
- `output` If TRUE, standard output is captured, otherwise not.

**Value**

List of test results.

**skip_test**

*Skip The Current Test*

**Description**

Signals a TestSkipped condition.

**Usage**

```r
skip_test(..., domain = NULL)
```

**Arguments**

- `...` zero or more objects which can be coerced to character (and which are pasted together with no separator) or (for message only) a single condition object.
- `domain` see `gettext`. If NA, messages will not be translated, see also the note in `stop`.

**Value**

(invisible) A `base::condition` of class `TestSkipped`. 
subset_tests

Identify Subset of Tests with Specified Tags and that Support Specified Argument Settings

Description

Identify Subset of Tests with Specified Tags and that Support Specified Argument Settings

Usage

subset_tests(tests = test_db(), tags = NULL, args = NULL, defaults = list())

Arguments

tests A list of tests to subset.
tag(s) (optional) A character vector of tags that tests must have.
args Named arguments with sets of values to test against.
defaults (optional) Named list with default argument values.

Value

A list of tests that support specified arguments.
Index

add_test_plan, 2
along_test_plans, 2

base::condition, 9
base::quit(), 4

check, 3
cHECK_plan, 4
evaluate_expr, 5
ggettext, 9
load_tests, 6
make_test, 6
register_test, 7
run_test, 8
run_tests, 8

skip_test, 9
stop, 9
subset_tests, 10