Package: microCRAN (via r-universe)

August 30, 2024

Type Package

Title Hosting an Independent CRAN Repository

Version 0.9.0-1

Description Stand-alone HTTP capable R-package repository, that fully supports R's install.packages() and available.packages(). It also contains API endpoints for end-users to add/update packages. This package can supplement 'miniCRAN', which has functions for maintaining a local (partial) copy of 'CRAN'. Current version is bare-minimum without any access-control or much security.

License GPL-3

Depends R (>= 4.0.0)

Imports rlang (>= 1.1.0), plumber (>= 1.2.0), assertthat (>= 0.2.1), mime, xtable

Suggests miniCRAN, testthat (>= 3.0.0), covr

Encoding UTF-8

RoxygenNote 7.2.3

Collate 'handlers-static.R' 'miniCRAN.R' 'packages.R' 'handlers-.R' 'api.R' 'microCRAN-package.R'

Config/testthat/edition 3

NeedsCompilation no

Author Stefan McKinnon Edwards [aut, cre, cph] (<https://orcid.org/0000-0002-4628-8148>), Kamstrup A/S [cph]

Maintainer Stefan McKinnon Edwards <sme@iysik.com>

Date/Publication 2023-11-03 22:00:02 UTC

Repository https://stefanedwards.r-universe.dev

RemoteUrl https://github.com/cran/microCRAN

RemoteRef HEAD

RemoteSha 915de01bc1c0dafa0b4510a7287cddc0a6dee2ee

Contents

`microCRAN-package` *microCRAN: Hosting an Independent CRAN Repository*

Description

Stand-alone HTTP capable R-package repository, that fully supports R's install.packages() and available.packages(). It also contains API endpoints for end-users to add/update packages. This package can supplement 'miniCRAN', which has functions for maintaining a local (partial) copy of 'CRAN'. Current version is bare-minimum without any access-control or much security.

CRAN directory structure

```
/root
/---/src/contrib/
/---/src/contrib/PACKAGES(?.rds|.gz)
/---/src/contrib/microCRAN_v1.0.0.tar.gz
/---/src/contrib/4.0/PACKAGES
/---/src/contrib/4.1/PACKAGES
/---/src/contrib/4.2/PACKAGES
/---/src/contrib/4.3/PACKAGES
/---/bin/windows/contrib/4.0/PACKAGES
/---/bin/windows/contrib/4.0/microCRAN_v1.0.0.zip
/---/bin/windows/contrib/4.1/PACKAGES
/---/bin/windows/contrib/4.2/PACKAGES
/---/bin/windows/contrib/4.3/PACKAGES
/---/bin/macosx/contrib/4.0/PACKAGES
/---/bin/macosx/contrib/4.0/microCRAN_v1.0.0.tgz
/---/bin/macosx/contrib/4.1/PACKAGES
/---/bin/macosx/contrib/4.2/PACKAGES
/---/bin/macosx/contrib/4.3/PACKAGES
```

Author(s)

Maintainer: Stefan McKinnon Edwards <sme@iysik.com> (ORCID) [copyright holder]

Other contributors:

- Kamstrup A/S [copyright holder]

---

addPackage *Adds a package-file to the repository*

Description

Methods for taking a file (".tar.gz", ".zip", ".tgz") and placing it in the repository, all locally.

Usage

```
addPackage(
  fn,
  type = c("source", "mac.binary", "win.binary"),
  repo_dir,
  is.new = TRUE
)
```

Arguments

fn	Path to package
type	Type of package, see explanation in section "Binary packages" in utils::install.packages().
repo_dir	Path to local directory, where the root of the repository is. The (source) packages will be stored locally at {repo_dir}/src/contrib/.
is.new	Logical, if TRUE, it causes an error if the file

Value

Invisibly returns the number of packages described in the resulting 'PACKAGES', 'PACKAGES.gz' and 'PACKAGES.rds' files. If 0, no packages were found and no files were written.

See Also

miniCRAN::addLocalPackage(), tools::write_PACKAGES()

Examples

```
f <- system.file('extdata/microCRAN_0.1.0.zip', package = 'microCRAN', mustWork = TRUE)
root <- tempdir()
addPackage(f, type = 'win.binary', repo_dir = root)
```

build *Build and start the microCRAN site*

Description

`build` creates the Plumber-router and `run` starts the service.

Usage

```
build(
  repo_dir,
  url_path,
  redirect_url,
  title,
  description,
  contact,
  license,
  tos
)

run(pr, host = "127.0.0.1", port = 1881, url_path, ...)
```

Arguments

`repo_dir`	Path to local directory, where the root of the repository is. The (source) packages will be stored locally at `{repo_dir}/src/contrib/`.
`url_path`	Optional prefix to endpoint. The CRAN repository will be available at e.g. `http://127.0.0.1:port/path_prefix/` with the "contrib.url" as `http://127.0.0.1:port/path_pre`
`redirect_url`	Url, if supplied, requests to static assets (package files, etc.) are redirected to another service instead of being handled by cran_static_path_handler. It can be beneficial to let e.g. an Apache httpd service handle those.
`title`, `description`, `contact`, `license`, `tos`	Descriptions of the API. Some defaults are used, see section below or `https://www.rplumber.io/articles/annotations.html`.
`pr`	A Plumber-router, e.g. as returned from `build`.
`host`	A string that is a valid IPv4 or IPv6 address that is owned by this server, which the application will listen on. "0.0.0.0" represents all IPv4 addresses and "::/0" represents all IPv6 addresses.
`port`	A number or integer that indicates the server port that should be listened on. Note that on most Unix-like systems including Linux and Mac OS X, port numbers smaller than 1025 require root privileges.
`...`	Additional arguments passed on to `plumber::pr_run()`.
`run`	Logical, should the method run the site immediately?

Details

Point `repo_dir` to you *local* filesystem. If the (sub-)directory does not exist, it will be created when an R-package is added through the corresponding endpoint.

Value

A new Plumber-router object.

API Descriptions

The fields `title`, `description`, `contact`, `license`, and `tos` are used for describing the API in the resulting Swagger-documents. These follow the OpenAPI type descriptions, see `https://spec.openapis.org/oas/v3.0.3#info-object`.

Field name	Type	Description
title	string	The title of the API.
description	string	A short description of the API.
tos	string	A URL to the Terms of Service for the API.
contact	list	A "Contact Object", i.e., list-object with fields "name", "url" and "email".
license	list	A "License Object", i.e., list-object with fields "name" and "url".
version	string	The version of the API.

`cran_static_path_handler` *Handler for package repository files*

Description

Creates handler for handling access to static files in the repository, i.e., the `src/contrib`, `bin/windows/contrib`, and `bin/macosx/contrib` subdirectories.

Use cran_static_redirect_handler to let another process (e.g. an Apache httpd or nginx server) handle the request, by redirecting it.

Usage

```
cran_static_path_handler(repo_dir, path_prefix = NULL)

cran_static_redirect_handler(dest_url)
```

Arguments

`repo_dir`	Path to local directory, where the root of the repository is. The (source) packages will be stored locally at `{repo_dir}/src/contrib/`.
`path_prefix`	Optional URL-component, if the repository exists at a subdirectory of the host (see run's `path`). Here, it is only for decorative purposes.
`dest_url`	The url requests are forwarded to, after being appened with the request; it should contain all path-components up to the '/src' or '/bin' parts.

Value

A handler (function) for use in a Plumber router "filter" .

Examples

```
require(plumber)
pr() |>
 pr_filter('static',
  cran_static_path_handler(repo_dir, path_prefix = 'cran'))
pr() |>
  pr_filter('redirect',
    cran_static_redirect_handler("http://my.local.cran:80/cran"))
```

---

directory_listing *Directory listing*

Description

Creates a simple HTML page with table of all files and subdirectories. May throw a 403 or 404 code.

Usage

```
directory_listing(req, res, path, path_prefix)

directory_listing_html(path, url_path, add_dir = TRUE)
```

Arguments

req, res	A "request"- and "response"-object, respectively
path	Path to directory to list
path_prefix	Optional URL-component, if the repository exists at a subdirectory of the host (see run's path). Here, it is only for decorative purposes.
url_path	The entire, relative path, that is displayed to end user in the URL.
add_dir	Logical, if the requested URL does not end with a '/', set this to TRUE, else all links will point to the parent directory.

Value

directory_listing returns the response-object. directory_listing_html returns a HTML-string with the entire page.

docker *Running microCRAN as a Docker container*

Description

A `Dockerfile` with an `init`-script is installed with this package, which can be used for running a Docker container with this package.

The path to the directory with the file can be found by running

```
system.file('docker/', package = 'microCRAN', mustWork = TRUE)
```

Running

To run, place microCRAN's source-tarball in the build directory with the `Dockerfile`.

Enable BuildKit and build the image:

```
DOCKER_BUILDKIT=1 docker build .
```

The build-process will automatically install any source-tarballed R-package that is located in the build-context (i.e. directory with the `Dockerfile`).

Start the container. Remember to map the port and the directory for the repository, else the repository is lost if the container is restarted. See the list below for variables to use.

```
docker run -d -v /data/my_local_cran:/var/cran -p 1881:1881 -e CRAN_DIR=/var/cran <microcran-image>
```

`CRAN_HOST` Host for the service to listen to. See also plumber::pr_run.

`CRAN_PORT` Port to listen on. Defaults to 1881.

`CRAN_DIR` Path to root of local directory where the repository's files are stored.

`CRAN_URL_PATH` Absolute path to the repository, as seen from the client.

`CRAN_REDIRECT_URL` If static files should be served by something else.

See Also

`build()`, plumber::pr_run

pr_add_package *Plumber route for adding package to repository*

Description

Creates a Plumber route that handles an incoming R-package. Use rather `build()` to build the entire API.

Usage

```
pr_add_package(pr, path = "/add", repo_dir)
```

Arguments

`pr`	A Plumber router-object
`path`	The path to the endpoint
`repo_dir`	Path to local directory, where the root of the repository is. The (source) packages will be stored locally at `{repo_dir}/src/contrib/`.

Value

`NULL` invisibly; called to modify the response.

See Also

`build()`, plumber::pr_post

read_DESCRIPTION_zip *Read DESCRIPTION file from package*

Description

Read DESCRIPTION file from package

Usage

```
read_DESCRIPTION_zip(fn)

read_DESCRIPTION_tar(fn)
```

Arguments

`fn`	Path to either zip, tar.gz or tgz file.

Value

List-object with contents of DESCRIPTION file.

Examples

```
package <- system.file('extdata/microCRAN_0.1.0.zip',
  package='microCRAN', mustWork=TRUE)
read_DESCRIPTION_zip(package)
```

Index