Package: odin2 (via r-universe)

September 17, 2024

Title Next generation odin

Version 0.1.2

Description Temporary package for rewriting odin.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

URL https://mrc-ide.github.io/odin2, https://github.com/mrc-ide/odin2

BugReports https://github.com/mrc-ide/odin2/issues

Imports cli, dust2 (>= 0.1.5), monty, rlang

Suggests decor, knitr, rmarkdown, mockery, numDeriv, pkgload, testthat (>= 3.0.0), withr

Config/testthat/edition 3

Remotes mrc-ide/dust2, mrc-ide/monty

VignetteBuilder knitr

Language en-GB

Repository https://mrc-ide.r-universe.dev

RemoteUrl https://github.com/mrc-ide/odin2

RemoteRef main

RemoteSha b7b9857db4cba217baa4ec8ec3143110f930f863

Contents

odin *Compile an odin model*

Description

Compile an odin model, yielding a `dust_system_generator` object.

Usage

```
odin(
  expr,
  input_type = NULL,
  quiet = FALSE,
  workdir = NULL,
  debug = FALSE,
  skip_cache = FALSE,
  compatibility = "warning"
)
```

Arguments

`expr`	Odin code as the path to a file (a string), a character vector of code, or as an expression (typically within braces `{}`).
`input_type`	An optional string describing the type of input for `expr` - must be one of `file`, `text` or `expression`. If given, this skips the type detection logic and odin will throw an error if the wrong type of input is given. Using this may be beneficial in programmatic environments.
`quiet`	Logical, indicating if compilation messages from `pkgbuild` should be displayed. Error messages will be displayed on compilation failure regardless of the value used.
`workdir`	Optional working directory to use. If `NULL`, we work in the session-specific temporary directory. By using a different directory of your choosing you can see the generated code.
`debug`	Passed to pkgbuild::compile_dll, this will build a debug library.
`skip_cache`	Logical, indicating if the cache of previously compiled systems should be skipped. If `TRUE` then your system will not be looked for in the cache, nor will it be added to the cache after compilation.
`compatibility`	Compatibility mode to use. Valid options are "warning", which updates code that can be fixed, with warnings, and "error", which will error. The option "silent" will silently rewrite code, but this is not recommended for general use as eventually the compatibility mode will be removed (this option is primarily intended for comparing output of odin1 and odin2 models against old code).

Value

A `dust_system_generator` object, suitable for using with dust functions (starting from dust2::dust_system_create)

odin_error_explain *Explain odin error*

Description

Explain error codes produced by odin. This is a work in progress, and we would like feedback on what is useful as we improve it. The idea is that if you see an error you can link through to get more information on what it means and how to resolve it. The current implementation of this will send you to the rendered vignettes, but in future we will arrange for offline rendering too.

Usage

```
odin_error_explain(code, how = "pretty")
```

Arguments

code	The error code, as a string, in the form Exxxx (a capital "E" followed by four numbers)
how	How to explain the error. Options are pretty (render pretty text in the console), plain (display plain text in the console) and link (browse to the online help).

Value

Nothing, this is called for its side effect only

odin_package *Update odin code in package*

Description

Update generated code in a package that uses odin and dust to provide a model. This will generate new dust code in inst/dust and from that generate a full model in src, and an R interface in R/dust.R, along with the cpp11 attributes that are needed to use the model.

Usage

```
odin_package(path, quiet = FALSE, compatibility = "warning")
```

Arguments

path	Path to the package root (the directory that contains DESCRIPTION), or any path within that package.
quiet	Logical, indicating if compilation messages from pkgbuild should be displayed. Error messages will be displayed on compilation failure regardless of the value used.

`compatibility` Compatibility mode to use. Valid options are "warning", which updates code that can be fixed, with warnings, and "error", which will error. The option "silent" will silently rewrite code, but this is not recommended for general use as eventually the compatibility mode will be removed (this option is primarily intended for comparing output of odin1 and odin2 models against old code).

Details

This function is powered by dust2::dust_package, and the same pre-requisites apply here:

For your `DESCRIPTION` file:

- `dust2` must be in `Imports`
- `cpp11`, `dust2` and `monty` must be in `LinkingTo`

For your `NAMESPACE` file:

- you must have a suitable `useDynLib()` call with `.registration = TRUE`

If you do not satisfy these requirements, `dust2::dust_package` will fail with a message indicating actions you should take. Once set up, generally things will keep working.

If you want your packages to build on GitHub actions, or be installable via `remotes::install_github` you should add to your `DESCRIPTION`:

```
Remotes: mrc-ide/dust2, mrc-ide/monty
```

Note that you do not need to include odin2 itself as a dependency.

Value

Invisibly, the path to the package. However, this function is typically called for its side effect of updating files in `inst/dust` and `src` within this package after you have changed the odin code in `inst/odin`.

---

odin_show *Show generated odin code*

Description

Show generated code from compiling an odin model.

Usage

```
odin_show(expr, input_type = NULL, compatibility = "warning")
```

Arguments

`expr`	Odin code as the path to a file (a string), a character vector of code, or as an expression (typically within braces `{}`).
`input_type`	An optional string describing the type of input for `expr` - must be one of `file`, `text` or `expression`. If given, this skips the type detection logic and odin will throw an error if the wrong type of input is given. Using this may be beneficial in programmatic environments.
`compatibility`	Compatibility mode to use. Valid options are "warning", which updates code that can be fixed, with warnings, and "error", which will error. The option "silent" will silently rewrite code, but this is not recommended for general use as eventually the compatibility mode will be removed (this option is primarily intended for comparing output of odin1 and odin2 models against old code).

Value

A character vector, with class `odin_code` that has a pretty-print method defined.

Index