Workflow Tips

Here are some tips for working with Julia efficiently.

REPL-based workflow

As already elaborated in The Julia REPL, Julia's REPL provides rich functionality that facilitates an efficient interactive workflow. Here are some tips that might further enhance your experience at the command line.

A basic editor/REPL workflow

The most basic Julia workflows involve using a text editor in conjunction with the `julia` command line. A common pattern includes the following elements:

- Put code under development in a temporary module. Create a file, say `Tmp.jl`, and include within it

```
module Tmp
export say_hello

say_hello() = println("Hello!")

# your other definitions here

end
```

- Put your test code in another file. Create another file, say `tst.jl`, which looks like

```
include("Tmp.jl")
import .Tmp
# using .Tmp # we can use `using` to bring the exported symbols in `Tmp` into ou

Tmp.say_hello()
# say_hello()

# your other test code here
```

and includes tests for the contents of `Tmp`. Alternatively, you can wrap the contents of your test file in a module, as

```
module Tst
    include("Tmp.jl")
    import .Tmp
    #using .Tmp

    Tmp.say_hello()
    # say_hello()

    # your other test code here
end
```

The advantage is that your testing code is now contained in a module and does not use the global scope in `Main` for definitions, which is a bit more tidy.

- `include` the `tst.jl` file in the Julia REPL with `include("tst.jl")`.
- Lather. Rinse. Repeat. Explore ideas at the `julia` command prompt. Save good ideas in `tst.jl`. To execute `tst.jl` after it has been changed, just `include` it again.

Browser-based workflow

It is also possible to interact with a Julia REPL in the browser via IJulia. See the package home for details.

Revise-based workflows

Whether you're at the REPL or in IJulia, you can typically improve your development experience with Revise. It is common to configure Revise to start whenever julia is started, as per the instructions in the Revise documentation. Once configured, Revise will track changes to files in any loaded modules, and to any files loaded in to the REPL with `includet` (but not with plain `include`); you can then edit the files and the changes take effect without restarting your julia session. A standard workflow is similar to the REPL-based workflow above, with the following modifications:

1. Put your code in a module somewhere on your load path. There are several options for achieving this, of which two recommended choices are:

   a. For long-term projects, use PkgTemplates:

   `julia using PkgTemplates t = Template() generate("MyPkg", t)` This will create a blank package, `"MyPkg"`, in your `.julia/dev` directory. Note that PkgTemplates allows you to control many different options through its `Template` constructor.

In step 2 below, edit `MyPkg/src/MyPkg.jl` to change the source code, and `MyPkg/test/runtests.jl` for the tests.

b. For "throw-away" projects, you can avoid any need for cleanup by doing your work in your temporary directory (e.g., `/tmp`).

Navigate to your temporary directory and launch Julia, then do the following:

`julia pkg> generate MyPkg # type ] to enter pkg mode julia> push!(LOAD_PATH, pwd()) # hit backspace to exit pkg mode` If you restart your Julia session you'll have to re-issue that command modifying `LOAD_PATH`.

In step 2 below, edit `MyPkg/src/MyPkg.jl` to change the source code, and create any test file of your choosing.

2. Develop your package

Before loading any code, make sure you're running Revise: say `using Revise` or follow its documentation on configuring it to run automatically.

Then navigate to the directory containing your test file (here assumed to be `"runtests.jl"`) and do the following:

```
julia> using MyPkg

julia> include("runtests.jl")
```

You can iteratively modify the code in MyPkg in your editor and re-run the tests with `include("runtests.jl")`. You generally should not need to restart your Julia session to see the changes take effect (subject to a few limitations, see https://timholy.github.io/Revise.jl/stable/limitations/).

« Performance Tips Style Guide »