Getting Started with Pry in 5 minutes

Created by Stefanni Brasil and Thiago Araujo from hexdevs. Script and commands are from Justin Gordon's Rails Conf 2021 talk Implicit to Explicit: Decoding Ruby's Magical Syntax.

This post's examples are for a *Ruby on Rails* application. However, Pry works on any Ruby application.

Install pry

1. Add the following lines to your Gemfile. To install the auxiliary gems, remove the comments after `# Pry Auxiliary Gems`. You can also `gem install` each one of them if you don't want to add them to the Gemfile.

```
group :development, :test do
  # Basic Pry Setup
  gem 'awesome_print' # pretty print ruby objects
  gem 'pry', '~> 0.13.0' # Console with powerful introspection capabilities
  gem 'pry-byebug' # Integrates pry with byebug
  gem 'pry-doc' # Provide MRI Core documentation
  gem 'pry-rails' # Causes rails console to open pry. `DISABLE_PRY_RAILS=1 rails c` can still open with IRB

  # Auxiliary Gems
  # gem 'pry-rescue' # Start a pry session whenever something goes wrong
  # gem 'pry-theme' # An easy way to customize Pry colors via theme files
  # gem 'pry-stack_explorer' # Allows navigating Pry call stack
  # gem 'binding_of_caller' # To evaluate code from a higher up call stack context
end
```

1. Run `bundle install`
2. On your terminal, run `pry` to check you have everything setup. If you've got a Pry REPL running, you're good to go.

Configure pry

To get the `.pryrc` aliases working, you need to run the following command:

```
$ curl https://gist.githubusercontent.com/justin808/1fe1dfbecc00a18e7f2a/raw/e0ea4dd77d34724ee3bbc8345c244e7e78a21d7b/.pryrc > $HOME/.
```

It will copy this .pryrc file that configures and styles the pry console. If you want to do it manually, save the linked `~/.pryrc` file in your home folder.

How to use Pry

Add `binding.pry` to any ruby file to start the debugger. For example:

```
class Account < ApplicationRecord
  def self.active
    binding.pry # -< add this where you want to debug
    where(archived: false)
  end
end
```

When this line of code gets executed, a `pry` REPL will open.

Pry commands

Now, to the fun part!

Pry has tons of commands and features but the ones below are enough for you to get started:

```
!!!                     # run it anytime you want to exit the program.
help                    # overview of pry features
help alias              # list of commands aliases
help whereami           # see the docs for *any* pry command, on this case, whereami
#### code browsing
w                       # whereami
@                       # alias to whereami - describes the current location on the source code
$                       # displays the code's location line, the object's owner, method visibility and length
@ 5                     # displays the current location and the 5 previous and posterior lines of code
h                       # displays the last 20 commands
$ some_method           # display the some_method implementation
show-source method_name # shows the source code for a given method. Example: show-source User.create
find-method to_a        # finds method to_a
play -l line_number     # executes line_number in the current context
self                    # make Ruby's magical syntax more explicit
pp(obj)                 # pretty-print the object passed in
#### state navigation
cd SomeModule           # changes context to module or class
ls -m                   # lists methods in context
cd ..                   # move context back up
#### stepping
b                       # break
s                       # step
c                       # continue
n                       # next line
```

Note: If you're using Puma, you'll get multiple threads running. Check out Justin's thread about running Puma for Debugging with Pry to learn how to handle that.

We tested the commands twice. If you have any issues with this guide, or have any questions, send us a message: team@hexdevs.com

Have fun debugging with pry! 👋🏻