Mininim

Introduction

Mininim is *the fastest* way to get started building Nim-based applications using familiar **Object-Oriented** paradigms. Combining a pre-selection of Nim's standard library with a collection of macros and its own modular packages, Mininim provides a more worry-free and high level approach to programming in Nim enabling you to build fast and optimize later.

Commands

Your First Command

Commands are shaped classes that define a sub-command for `mininim`. One example that you may already be familiar with from previous chapters is the `mininim serve` command which is added by the `mininim/web` package. Adding your own sub-command is pretty simple. Let's begin by creating our file `local/commands/Welcome.nim`.

Imports

In order to be able to shape our command, we'll want to make sure we add the following imports:

```
import
    mininim,
    mininim/cli
```

Type, Methods, Properties

You can use the provided `Process` class and extend it as follows:

```
type
    Welcome = ref object of Process
```

This will add the base `execute` method for overloading. Default behavior will simply return `0`. Or, alternatively, you can define your own class directly, so long as it conforms to the `CommandConcept`:

```
CommandConcept* = concept Process
    Process.execute(Console) is int
```

`Process.execute(Console): int`

The `execute` method is the actual logic that is executed when your command is run. As with any CLI application, we want to make sure we return an integer `0` for success, anything else for failure.

```
begin Welcome:
    method execute(console: Console): int =
        echo "Hello Mininim!"
        result = 0
```

Shape

Now we just need to give our command shape:

```
shape Welcome: @[
    Command(
        name: "welcome",
        description: "Show the welcome message",
    )
]
```

Running

Recompile and run:

```
bin/mininim welcome
```