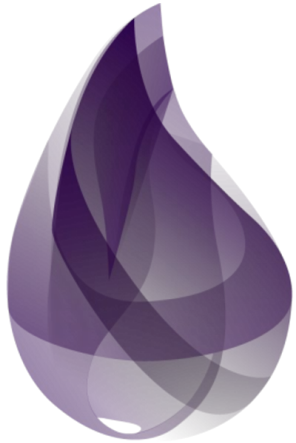


ExUnit

Testy Jednostkowe w Języku Elixir

ExUnit

- wbudowany framework do testów jednostkowych
- testy implementowane w skryptach Elixira
- rozszerzenie .exs
- `test` zamiast `def`
- nazwy w ” ”
- `ExUnit.start()` - najczęściej w osobnym pliku: `test/test_helper.exs`

Prosty przykład

```
ExUnit.start()

defmodule EasyExample do
    use ExUnit.Case

    test "addition" do
        assert 5 + 5 == 10
    end
end
```

Prosty przykład

```
$ elixir easy_example_test.exs

.

Finished in 0.04 seconds (0.04s on load, 0.00s on tests)
1 test, 0 failures

Randomized with seed 821000
```

Prosty przykład

```
ExUnit.start()

defmodule EasyExample do
    use ExUnit.Case

    test "addition" do
        assert 5 + 5 == 9
    end
end
```

```
$ elixir easy_example_test.exs
```

Prosty przykład

```
 1) test addition (EasyExample)
     easy_example_test.exs:6
     Assertion with == failed
     code:  assert 5 + 5 == 9
     left:  10
     right: 9
     stacktrace:
       easy_example_test.exs:7: (test)

Finished in 0.07 seconds (0.06s on load, 0.01s on tests)
1 test, 1 failure

Randomized with seed 889000
```

Funkcjonalność

między innymi:

- `ExUnit.Assertions`
- `ExUnit.Callbacks`
- `ExUnit.CaptureIO`
- `ExUnit.DocTest`

ExUnit.Assertions

- assert
- refute
- assert_raise

```
test "Assertions" do
    assert 5 > 4
    refute 4 > 5
    assert_raise ArithmeticError, fn -> 1/0 end
end
```

ExUnit.Assertions

- assert_received | assert_receive

```
defmodule Sender do
    def send_ping(pid) do
        send(pid, :ping)
    end
end
...

test "assert_received" do
    Sender.send_ping(self())
    assert_received :ping
end
```

ExUnit.CaptureIO

- capture_io

```
import ExUnit.CaptureIO
...
test "CaptureIO" do
    assert capture_io(fn -> IO.puts("Witaj") end) == "Witaj\n"
end
```

ExUnit.Callbacks

- setup i setup_all

```
setup_all do
    {:ok, ala: :ma_kota}
end
...
test "Callbacks", state do
    assert :ma_kota == state[:ala]
end
```

ExUnit.DocTest

```elixir
defmodule Add do
    @moduledoc false
    @doc """
    Adds two numbers.
    ## Examples
        iex> Add.add(5, 10)
        15
    """
    def add(a, b) do
        a + b
    end
end
```

```elixir
doctest Add
```

Mix

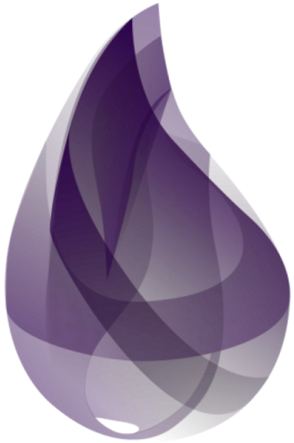
- Narzędzie do budowania projektów
- Utworzenie projektu
- Kompilacja projektu
- Testowanie projektu
- Dużo więcej...

Mix

```
$ mix new functionality_project
$ cd functionality_project
$ mix test

Compiling 1 file (.ex)
Generated functionality_project app

.....

Finished in 0.04 seconds
1 doctest, 4 tests, 0 failures

Randomized with seed 126000
```

Przykład

- kalkulator średniej ocen



Koniec

źródła:

https://hexdocs.pm/ex_unit

https://elixirschool.com/en/