

In this lecture, we will discuss...

- ✧ Rspec Matchers



Rspec Matchers

- ✧ RSpec “hangs” **to** and **not_to** methods on **all outcome of expectations**
- ✧ **to()/not_to()** methods take **one parameter** – a **matcher**
- ✧ Matcher examples:
 - `be_true / be_false`
 - `eq 3`
 - `raise_error(SomeError)`



Be_predicate – boolean

- ✧ If the **object** on which the **test is operating** has a **predicate (boolean) method** – you **automatically** get a **be_predicate** matcher
- ✧ So, for example **be_nil** is a **valid matcher** since every Ruby object has a `:nil?` method

```
it "should sum two odd numbers and become even" do
  expect(@calculator.add(3, 3)).to be_even
  expect(@calculator.add(3, 3)).not_to be_odd
end
```



Be_predicate – boolean

```
~/coursera/code-module2/Lecture16-RSpec$ rspec
```

```
...
```

```
Finished in 0.00169 seconds (files took 0.08464 seconds to load)
```

```
3 examples, 0 failures
```

```
~/coursera/code-module2/Lecture16-RSpec$ rspec --format documentation
```

```
Calculator
```

```
  should add 2 numbers correctly
```

```
  should subtract 2 numbers correctly
```

```
  should sum two odd numbers and become even
```

```
Finished in 0.00187 seconds (files took 0.08772 seconds to load)
```

```
3 examples, 0 failures
```



More Matchers

https://relishapp.com/rspec/rspec-expectations/docs/built-in-matchers

Relish Public projects Plans & pricing Sign up Sign in

Publisher: RSpec
Project: RSpec Expectations 3.3

Change version

Browse documentation

feedback

Built in matchers

- > Equality matchers
- > Comparison matchers
- > Predicate matchers
- > Type matchers
- > `all` matcher
- > `be` matchers
- > `be_within` matcher
- > `change` matcher
- > `contain_exactly` matcher
- > `cover` matcher

Built in matchers

rspec-expectations ships with a number of built-in matchers. Each matcher can be used with `expect(...).to` or `expect(...).not_to` to define positive and negative expectations respectively on an object. Most matchers can also be accessed using the `(...).should` and `(...).should_not` syntax; see [using should syntax](#) for why we recommend using `expect`.

e.g.

```
expect(result).to eq(3)
expect(list).not_to be_empty
pi.should be > 3
```



Summary

- ✧ RSpec has a lot of built-in matchers readily available for simplifying writing tests

What's next?

- ✧ Module 3: Introduction to Ruby on Rails

