

Extracted from:

# The RSpec Book

---

## Behaviour Driven Development with RSpec, Cucumber, and Friends

This PDF file contains pages extracted from The RSpec Book, published by the Pragmatic Bookshelf. For more information or to purchase a paperback or PDF copy, please visit <http://www.pragprog.com>.

**Note:** This extract contains some colored text (particularly in code listing). This is available only in online versions of the books. The printed versions are black and white. Pagination might vary between the online and printer versions; the content is otherwise identical.

Copyright © 2009 The Pragmatic Programmers, LLC.

All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

The  
Pragmatic  
Programmers

# The RSpec Book

Behaviour Driven Development  
with RSpec, Cucumber,  
and Friends

*David Chelimsky*  
with *Dave Astels,*  
*Zach Dennis,*  
*Aslak Helleøy,*  
*Bryan Helmkamp,*  
and *Dan North*

*Edited by Jacquelyn Carter*

The Facets  of Ruby Series



Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and The Pragmatic Programmers, LLC was aware of a trademark claim, the designations have been printed in initial capital letters or in all capitals. The Pragmatic Starter Kit, The Pragmatic Programmer, Pragmatic Programming, Pragmatic Bookshelf and the linking *g* device are trademarks of The Pragmatic Programmers, LLC.

Every precaution was taken in the preparation of this book. However, the publisher assumes no responsibility for errors or omissions, or for damages that may result from the use of information (including program listings) contained herein.

Our Pragmatic courses, workshops, and other products can help you and your team create better software and have more fun. For more information, as well as the latest Pragmatic titles, please visit us at

<http://www.pragprog.com>

Copyright © 2010 The Pragmatic Programmers LLC.

All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

Printed in the United States of America.

ISBN-10: 1-934356-37-9

ISBN-13: 978-1-934356-37-1

Printed on acid-free paper.

B12.0 printing, December 3, 2009

Version: 2010-1-3

# Describing Code with RSpec

---

In the last chapter, we introduced and used Cucumber to describe the behaviour of our Codebreaker game from the outside, at the application level. We wrote a Cucumber feature with a scenario and step definitions that will handle the steps in the scenario, but we're getting an error. The code in a step definition is trying to interact with a `Codebreaker::Game` object, but there is no application code to support this yet.

In this chapter we're going to use RSpec to *describe* behaviour at a much more granular level: the expected behaviour of instances of the `Game` class.

## 5.1 Getting started with RSpec

To get going, create a file named `game_spec.rb` in `spec/codebreaker/` and add the following code:

[Download](#) `cb/06/spec/codebreaker/game_spec.rb`

```
module Codebreaker
  describe Game do
    end
  end
end
```

The `describe()` method hooks into RSpec's API, and it returns a `Spec::ExampleGroup`, which is, as it suggests, a group of examples—examples of the expected behaviour of an object. If you're accustomed to xUnit tools like `Test::Unit`, you can think of an `ExampleGroup` as being akin to a `TestCase`.

Open up a shell and cd to the codebreaker project root directory and run the `game_spec.rb` file with the `spec` command,<sup>1</sup> like this:

```
spec spec/codebreaker/game_spec.rb
```

The resulting output should include “uninitialized constant Codebreaker::Game (NameError)” To fix that we need to do a few things.

## Connect the specs to the code

First, add a file named `game.rb` with the following code in the `lib/codebreaker` directory:

[Download](#) `cb/07/lib/codebreaker/game.rb`

```
module Codebreaker
  class Game
    end
  end
end
```

As we progress, we’ll maintain a parallel structure like this in which each source file (e.g. `lib/codebreaker/game.rb`) has a parallel spec file (e.g. `spec/codebreaker/game_spec.rb`). See the *Joe Asks...* on the next page for more on this.

We’ll want to load `game.rb` and any other source files from a single location, so require the file from `lib/codebreaker.rb`:

[Download](#) `cb/07/lib/codebreaker.rb`

```
require 'codebreaker/game'
```

Next, the spec helper needs to require `codebreaker`:

[Download](#) `cb/07/spec/spec_helper.rb`

```
require 'codebreaker'
```

RSpec adds the `lib` directory to the load path, so there is no need to do so explicitly.

Lastly we need to require `spec_helper.rb` from `game_spec.rb`, which should then look like this:

[Download](#) `cb/07/spec/codebreaker/game_spec.rb`

```
require 'spec_helper'
```

```
module Codebreaker
  describe Game do
    end
  end
```

---

1. The `spec` command is installed when you install the `rspec` gem.



### Joe Asks...

#### Shouldn't we avoid a 1-to-1 mapping?

Perhaps you've heard that a 1-to-1 mapping between objects and their specs is a BDD no-no. There is some truth to this, but the devil is in the details.

We want to avoid a strict adherence to a structure in which every object has a single example group, and every method has a single code example. That sort of structure leads to long examples that take an object through many phases, setting expectations at several stopping points in each example. Examples like these are difficult to write to begin with, and much more difficult to understand and debug later.

A 1-to-1 mapping of spec-file to application-code-file, however, is not only perfectly fine, it is actually beneficial. It makes it easier to understand where to find the specs for code you might be looking at. It also makes it easier for tools to automate shortcuts like the one in The RSpec TextMate bundle, which switches between spec-file and application-code-file with CTRL-SHIFT-DOWN.

**end**

Now run `game_spec.rb` with the `spec` command again. You should see output like this:

```
Finished in 0.001545 seconds
```

```
0 examples, 0 failures
```

This tells us that everything is hooked up correctly and we can move on.

### Connect the features to the code

To see where we are in relation to our feature, add the `lib` directory to the load path and require `codebreaker` in `features/support/env.rb`:

Download `cb/07/features/support/env.rb`

```
$LOAD_PATH << File.expand_path('.././../lib', __FILE__)
require 'codebreaker'
```

Running `features/codebreaker_starts_game.feature` gives us a different error now:

Feature: code-breaker starts game

```
As a code-breaker
I want to start a game
So that I can break the code
```

Scenario: start game

```

# 07/features/codebreaker_starts_game.feature:10
Given I am not yet playing
# 07/features/step_definitions/codebreaker_steps.rb:1
When I start a new game
# 07/features/step_definitions/codebreaker_steps.rb:5
wrong number of arguments (1 for 0) (ArgumentError)
./07/features/step_definitions/codebreaker_steps.rb:7:in `initialize'
./07/features/step_definitions/codebreaker_steps.rb:7:in `new'
./07/features/step_definitions/codebreaker_steps.rb:7:in
`/^I start a new game$/'
07/features/codebreaker_starts_game.feature:12:in
`When I start a new game'
Then I should see "Welcome to Codebreaker!"
# 07/features/step_definitions/codebreaker_steps.rb:11
And I should see "Enter guess:"
# 07/features/step_definitions/codebreaker_steps.rb:11
```

```
1 scenario (1 failed)
4 steps (1 failed, 2 skipped, 1 passed)
0m0.001s
```

We're getting an `ArgumentError` instead of a `NameError`. This tells us two things: first, the feature is hooked up to the correct code; second, the `Game` needs to handle the messenger argument we pass to the `initialize` method.

The process we're about to go through is the Red-Green-Refactor cycle straight out of Test-Driven Development. The idea is that you write a failing example (red), write only enough code to make the example pass (green), and then remove any unwanted duplication (refactor).

This is a process not unlike music or dancing. You get into a groove and it moves very, very quickly. To strive for that feeling, we're going to go through these steps in rapid succession with very little discussion between each step.

# The Pragmatic Bookshelf

---

The Pragmatic Bookshelf features books written by developers for developers. The titles continue the well-known Pragmatic Programmer style and continue to garner awards and rave reviews. As development gets more and more difficult, the Pragmatic Programmers will be there with more titles and products to help you stay on top of your game.

## Visit Us Online

---

### The RSpec Book's Home Page

<http://pragprog.com/titles/achbd>

Source code from this book, errata, and other resources. Come give us feedback, too!

### Register for Updates

<http://pragprog.com/updates>

Be notified when updates and new books become available.

### Join the Community

<http://pragprog.com/community>

Read our weblogs, join our online discussions, participate in our mailing list, interact with our wiki, and benefit from the experience of other Pragmatic Programmers.

### New and Noteworthy

<http://pragprog.com/news>

Check out the latest pragmatic developments, new titles and other offerings.

## Buy the Book

---

If you liked this eBook, perhaps you'd like to have a paper copy of the book. It's available for purchase at our store: [pragprog.com/titles/achbd](http://pragprog.com/titles/achbd).

## Contact Us

---

Online Orders:	<a href="http://www.pragprog.com/catalog">www.pragprog.com/catalog</a>
Customer Service:	<a href="mailto:support@pragprog.com">support@pragprog.com</a>
Non-English Versions:	<a href="mailto:translations@pragprog.com">translations@pragprog.com</a>
Pragmatic Teaching:	<a href="mailto:academic@pragprog.com">academic@pragprog.com</a>
Author Proposals:	<a href="mailto:proposals@pragprog.com">proposals@pragprog.com</a>
Contact us:	1-800-699-PROG (+1 919 847 3884)