



University of
Sheffield

COM1001 SPRING SEMESTER

Professor Phil McMinn

p.mcminn@sheffield.ac.uk

Acceptance Testing with RSpec

Acceptance Testing

Acceptance criteria are a set of predefined requirements that must be met to make a story complete.

Acceptance criteria can be used to determine when implementation on a story has been completed – something we can verify by turning these criteria into **acceptance tests**.

Acceptance tests can be written in RSpec. They are a type of system test, because they involve interacting with the system.

From a web application perspective, this means that they **interact with the application's front end as if they were a user.**

User Steps with Capybara

Capybara is a gem that lets us simulate a user interacting with a web page with Ruby code.

The following simulates interaction with the Add Player page of the **football_players** example.

Simulates a user typing in a URL into their browser

This simulates the user filling in different form fields with data. The field name comes from the **name** attribute of the HTML input element

Finally, clicking the submit button. Again, the name here comes from the **name** attribute from the HTML tag. We must write it exactly as it appears, including putting the characters in the right case.

```
visit "/add"  
fill_in "first_name", with: "George"  
fill_in "surname", with: "Test"  
fill_in "gender", with: "M"  
fill_in "club", with: "Mantester Utd"  
fill_in "country", with: "Northern RSpec"  
fill_in "position", with: "Midfield"  
fill_in "date_of_birth", with: "1946-05-22"  
click_on "Submit"
```

Acceptance criteria:
Given-When-Then

```
RSpec.describe "Editing a player" do
  context "given a player has been added" do
    context "when that player is edited" do
      it "then shows the updated player on the search page" do
        # add the initial player
        visit "/add"
        fill_in "first_name", with: "George"
        fill_in "surname", with: "Test"
        fill_in "gender", with: "M"
        fill_in "club", with: "Mantester Utd"
        fill_in "country", with: "Northern RSpec"
        fill_in "position", with: "Midfield"
        fill_in "date_of_birth", with: "1946-05-22"
        click_on "Submit"

        # edit the initial player
        visit "/edit?id=1"
        fill_in "first_name", with: "Zinedine"
        click_on "Submit"

        # check the player is in the list
        expect(page).to have_content "Zinedine Test"
      end
    end
  end
end
```

Checking the page has
content is written
slightly differently with
Capybara compared to
how we saw it before.

The test can move
around the app
clicking links and filling
in text boxes like a real
user, except it's all
automated

football_players/spec/acceptance/edit_a_player_spec.rb

Running RSpec Acceptance Tests

Running RSpec acceptance tests is no different to any other RSpec test, we invoke the `rspec` command at the command line. We can run it specifically with the `spec/acceptance` directory, where acceptance tests are placed, or along with all RSpec tests.

Although Capybara can drive a real web browser to do the testing (e.g., through the use of Selenium), we've got it configured in “**headless**” mode. This is faster, and works on Codio.

However, because we cannot “see” the tests running, it's sometimes hard to understand why a test fails, because we cannot see what the test is seeing:

```
codio@todaylaser-zoomdarwin:~/workspace/com1001-2024/week3/football_players_example$ rspec spec/acceptance/add_a_player_spec.rb
F
```

Failures:

```
1) Adding a player when a new player is added then shows the player on the search page
   Failure/Error: expect(page).to have_content "George Best"
     expected `#<Capybara::Session>.has_content?("George Best")` to be truthy, got false
   # ./spec/acceptance/add_a_player_spec.rb:16:in `block (3 levels) in <top (required)>'
```

```
Finished in 1.86 seconds (files took 2.37 seconds to load)
```

```
1 example, 1 failure
```

Failed examples:

```
rspec ./spec/acceptance/add_a_player_spec.rb:3 # Adding a player when a new player is added then shows the player on the search page
```

Debugging Test with `save_page`

This will save the HTML page that Capybara “sees” to the filestore

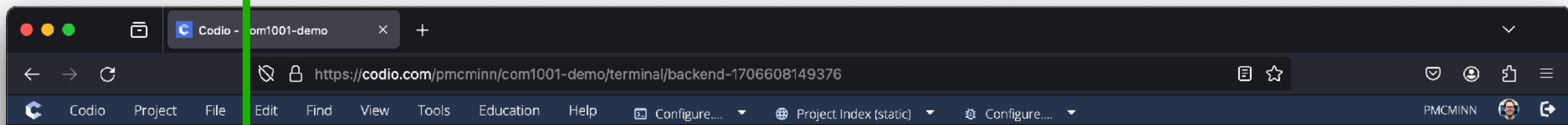
Note that `save_page` must come before the failing `expect` statement.

```
it "then shows the player on the search page" do
  # add the player
  visit "/add"
  fill_in "first_name", with: "George"
  fill_in "surname", with: "Test"
  # ... (more details added here) ...
  click_on "Submit"

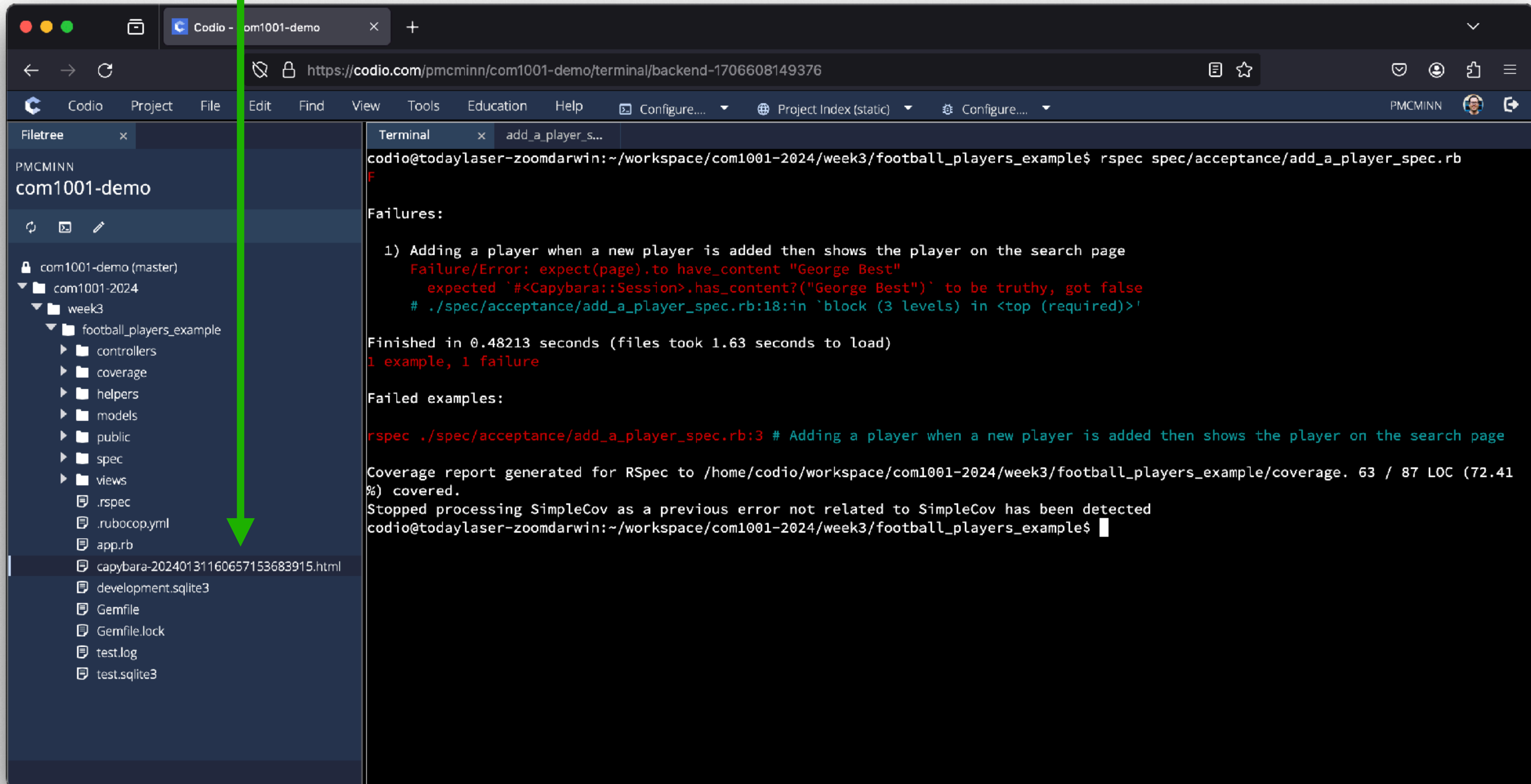
  save_page

  # check the player is listed on the search page
  expect(page).to have_content "George Best"
end
```

Deliberate mistake to demonstrate the point in this slide. The added test player is was a legend in his time, the one and only “George Test”.



Deliberate mistake to demonstrate the point in this slide. The added test player is was a legend in his time, the one and only “George Test”.



The screenshot displays the Codio IDE interface. On the left, the Filetree panel shows the project structure for 'com1001-demo'. The 'week3' directory is expanded, showing the 'football_players_example' directory. A green arrow points from the top of the slide to the file explorer. The terminal window on the right shows the output of an RSpec test run. The test fails with the message: 'Failure/Error: expect(page).to have_content \"George Best\"'. The failure message is repeated in red text. The terminal output also shows the test finished in 0.48213 seconds and generated a coverage report.

```
codio@todaylaser-zoomdarwin:~/workspace/com1001-2024/week3/football_players_example$ rspec spec/acceptance/add_a_player_spec.rb
F

Failures:

  1) Adding a player when a new player is added then shows the player on the search page
     Failure/Error: expect(page).to have_content "George Best"
      expected `#<Capybara::Session>.has_content?("George Best")` to be truthy, got false
     # ./spec/acceptance/add_a_player_spec.rb:18:in `block (3 levels) in <top (required)>'

Finished in 0.48213 seconds (files took 1.63 seconds to load)
1 example, 1 failure

Failed examples:

rspec ./spec/acceptance/add_a_player_spec.rb:3 # Adding a player when a new player is added then shows the player on the search page

Coverage report generated for RSpec to /home/codio/workspace/com1001-2024/week3/football_players_example/coverage. 63 / 87 LOC (72.41%) covered.
Stopped processing SimpleCov as a previous error not related to SimpleCov has been detected
codio@todaylaser-zoomdarwin:~/workspace/com1001-2024/week3/football_players_example$
```

Codio - com1001-demo

https://codio.com/pmcminn/com1001-demo/terminal/backend-1706608149376

CodioProjectFileEditFindViewToolsEducationHelpConfigure...Project Index (static)Configure...PMCMINN

Filetree

PMCMINN
com1001-demo

com1001-demo (master)
com1001-2024
week3
football_players_example
controllers
coverage
helpers
models
public
spec
views
.rspec
.rubocop.yml
app.rb
capybara-20240131160657153683
development.sqlite3
Gemfile
Gemfile.lock
test.log
test.sqlite3

Terminal

add_a_player_s...

```
codio@todaylaser-zoomdarwin:~/workspace/com1001-2024/week3/football_players_example$ rspec spec/acceptance/add_a_player_spec.rb
F

Failures:

  1) Adding a player when a new player is added then shows the player on the search page
     Failure/Error: expect(page).to have_content "George Best"
       expected `#<Capybara::Session>.has_content?("George Best")` to be truthy, got false
     # ./spec/acceptance/add_a_player_spec.rb:18:in `block (3 levels) in <top (required)>'

Finished in 0.48213 seconds (files took 1.63 seconds to load)
1 example, 1 failure

Failed examples:

rspec ./spec/acceptance/add_a_player_spec.rb:3 # Adding a player when a new player is added then shows the player on the search page

Coverage report generated for RSpec to /home/codio/workspace/com1001-2024/week3/football_players_example/coverage. 63 / 87 LOC (72.41%) covered.

Processing SimpleCov as a previous error not related to SimpleCov has been detected
laser-zoomdarwin:~/workspace/com1001-2024/week3/football_players_example$
```

Preview static

Download

Code Playback

Deploy...

Set as project index

Rename...

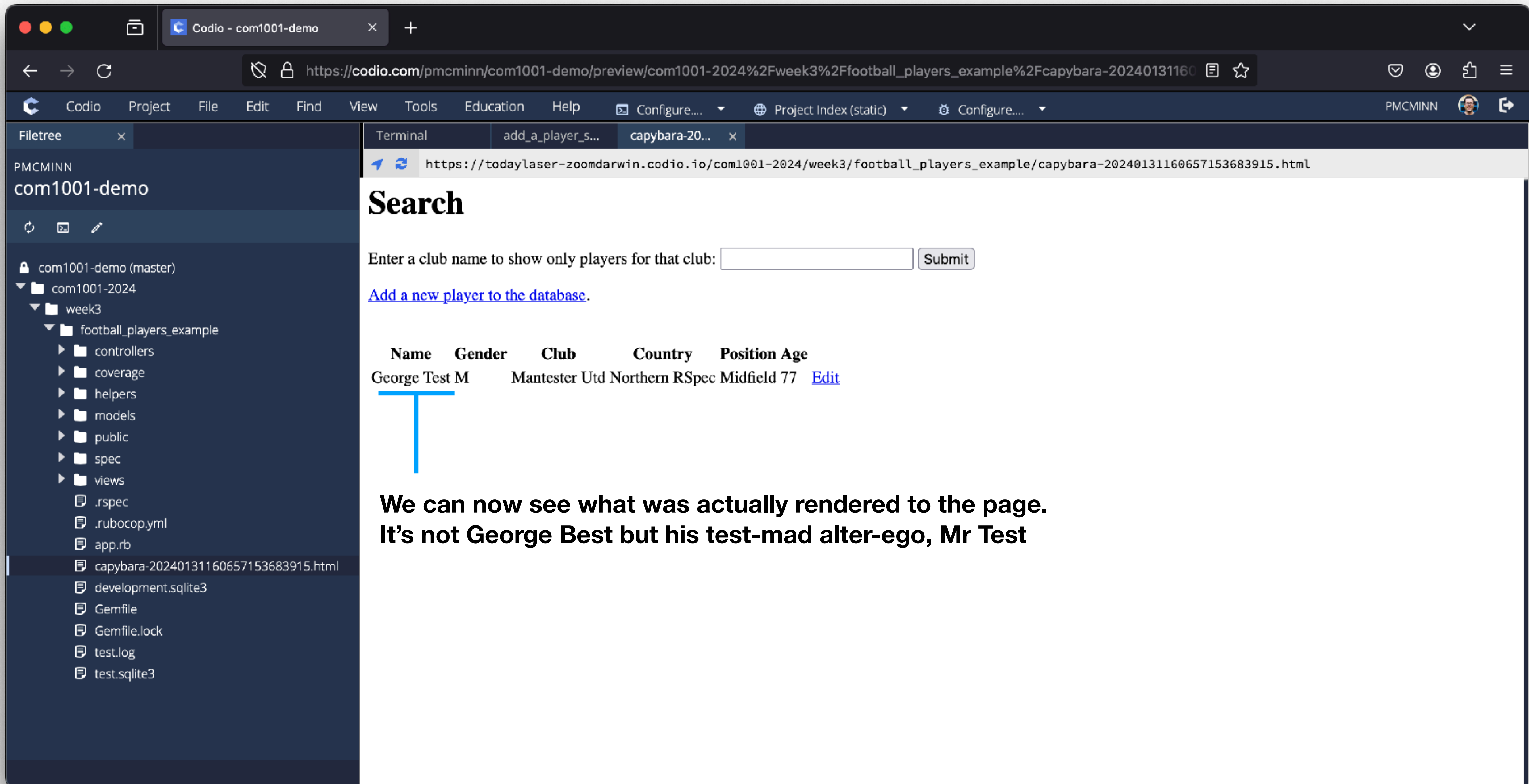
Delete...

Copy...

Duplicate...

Right click the file and
click "Preview static"
to view it in Codio

https://codio.com/pmcminn/com1001-demo/terminal/backend-1706608149376#



Search

Enter a club name to show only players for that club:

[Add a new player to the database.](#)

Name	Gender	Club	Country	Position	Age
George Test M		Mantester Utd Northern RSpec		Midfield	77

[Edit](#)

We can now see what was actually rendered to the page.
It's not George Best but his test-mad alter-ego, Mr Test

More Examples

Check out the [football_players/spec/acceptance](#) directory in the examples repository for more examples of using Capybara with RSpec to test other parts of the app, including the edit, search and delete pages.

For more on Capybara, see:

- Its web page: <https://teamcapybara.github.io/capybara/>
- A cheat sheet for using Capybara: <https://devhints.io/capybara>