Testing

Peter van Rijn

www.little-world.nl

Software Testing

- Why Testing?
- Testing Types?
- Testing Levels?

Demos

- Writing Tests
  - Testing Levels
    - With Different Libraries

Why Testing?

Meets the Requirements

Responds Correctly to Inputs

Performs within Acceptable Time

Can be Installed and Run

Achieves Results for the Stakeholders

Testing Types

Smoke and Sanity Testing

Continuous Testing

Regression Testing

Performance Testing

Acceptance Testing

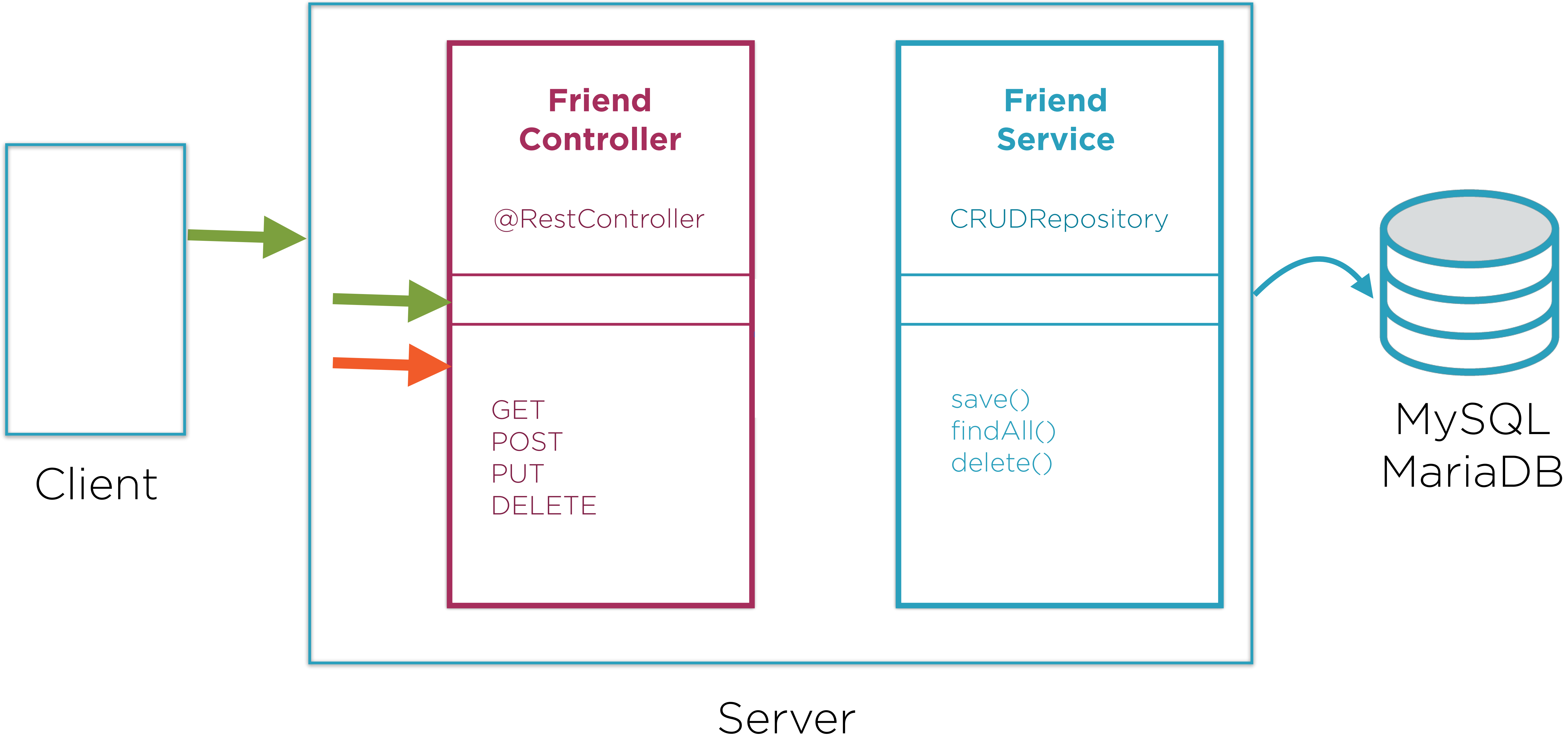
Testing Levels

Unit Testing

Integration Testing

System Testing

Test Architecture



Spring Boot Test Libraries

JUnit
De-facto Standard

Spring Test
Spring Boot Test Support

AssertJ
A Fluent Assertion Library

Hamcrest
Matcher Objects

JSONPath
XPath for JSON

Mockito
Mocking Framework

Demos

Test Pattern

1. Create new Friend
    - Assert if it is Created
2. List all Friends
    - Assert if the Friend is in the List
3. Delete the Friend
    - Assert if the List is Empty

Smoke and Sanity Testing

Assert if the Spring Context is Running

@SpringBootTest

- The Spring Context in JUnit Test

Demo

Spring Initializr includes a Test

Assert if the FriendController is Alive

- Using JUnit Assertions

System Testing

Test the Complete FriendApplication

Use RestTemplate as Client

- Has the Same Role as Postman

RestTemplate class

GET	getForObject()	getForEntity()
POST	postForObject()	postForLocation()
PUT	put()	
DELETE	delete()	
any	exchange()	execute()

Demo

Run the Server

Using Test Pattern

- With JUnit Assertions

RestTemplate

- postForEntity(url, friend, Friend.class);
- getForObject(url, Friend[].class)
- delete(url)

Integration Testing

Test FriendController and FriendService

Direct on the Java Code

Demo

Use @SpringBootTest

- @Autowired FriendController

Using Test Pattern

- With AssertJ Assertions

JPA Testing

Test the FriendService

And the Database

Demo

Use @DataJpaTest

- @Autowired FriendService
- @Autowired TestEntityManager

Using Test Pattern

- With AssertJ Assertions

Unit Testing

Test Standalone Controller

Mock the FriendService

Using Mockito

when(call**). thenReturn(**mock**)**

JSONPath is an XPath for JSON

- / has become $

Demo

Using @WebMvcTest()

- This is an other Spring Context!
- @Autowired MockMvc
- @MockBean FriendService

Using Test Pattern

MockMvc

- perform(), andExpect()
- status(), jsonPath()

Hamcrest Matcher

Exception Testing

Test Exception on FriendController

Assert if an Exception is Thrown

Demo

Using @SpringBootTest

- Create Method somethingIsWrong()
- That throws a ValidationException

Test if Exception is Thrown

- @Test(expected =)
- Using JUnit Assertions

Error Handling Testing

Test Error Handling on the Application

Use RestTemplate as Client

Assert HTTP Response

Demo

Run the Server

Using RestTemplate

- getForEntity()
- Returns ResponseEntity

Assert HTTP Response Status

- Use AssertJ Assertions

Spring REST Testing

System Testing

- RestTemplate
    - GET, POST, PUT, DELETE

Integration Testing

- @SpringBootTest
    - @Autowired

Unit Testing

- @WebMvcTest
    - Mockito, MockMvc, @MockBean

Assertions (Matchers)

JUnit

- assertEquals(b, a)
- assertEquals(3, list.size())
- assertThat(list, hasItem(1))

Hamcrest

- assertThat(a, is(equalTo(b)))
- assertThat(list, hasSize(3))
- assertThat(list, contains(1, 2, 3))

AssertJ

- assertThat(a).isEqualTo(b)
- assertThat(list).hasSize(3)
- assertThat(list).contains(1, 2, 3)

Tests should be Readable.

Non-programmers should be able to read or change a test.

Summary

Spring Boot Testing

- Libraries for all Testing Levels
- Assertions Libraries
  - To make them Readable