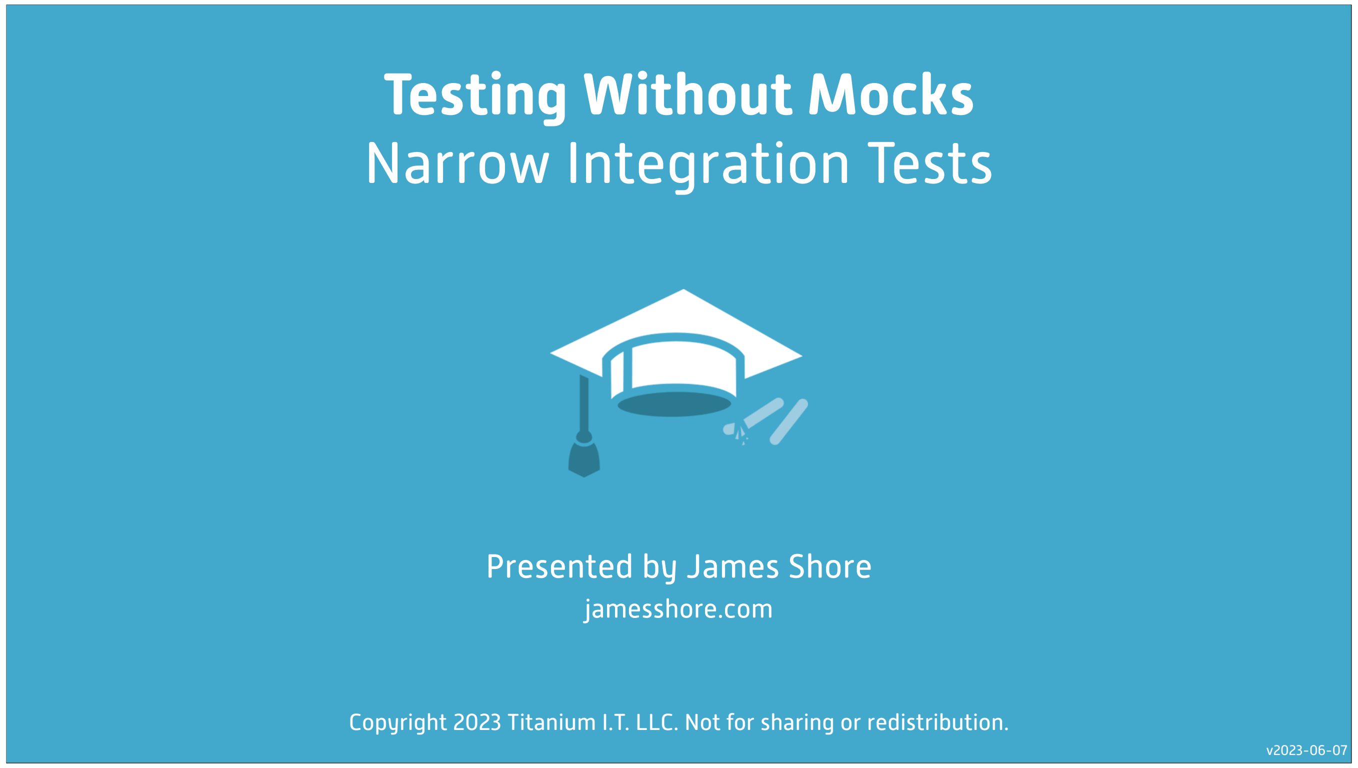


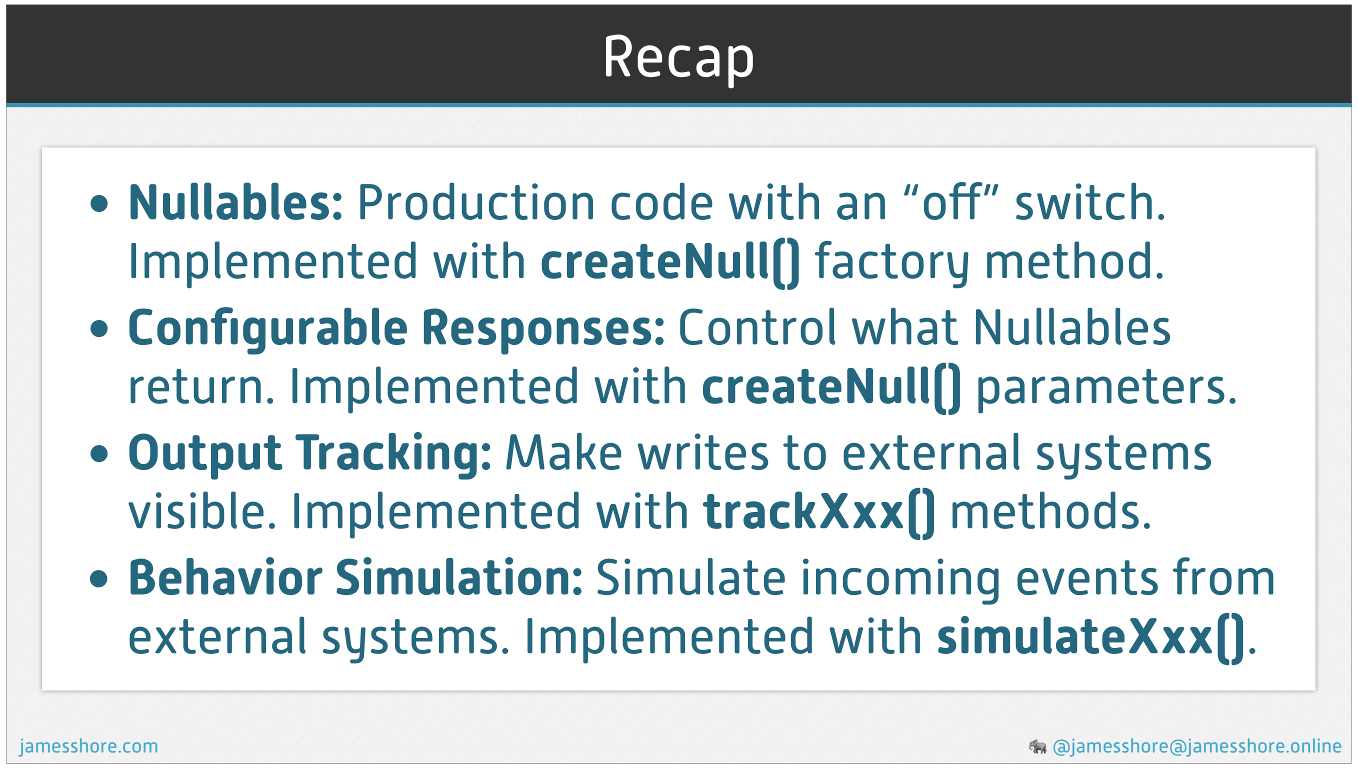
Testing Without Mocks
Narrow Integration Tests

Presented by James Shore
jamesshore.com

v2023-06-07

I'm James Shore. Today in “Testing Without Mocks,” we're talking about how to implement low-level infrastructure wrappers and test their communication with external systems.

As a reminder, watching this video is optional. I'll cover the same material during the course.



Recap

- **Nullables:** Production code with an "off" switch. Implemented with **createNull()** factory method.
- **Configurable Responses:** Control what Nullables return. Implemented with **createNull()** parameters.
- **Output Tracking:** Make writes to external systems visible. Implemented with **trackXxx()** methods.
- **Behavior Simulation:** Simulate incoming events from external systems. Implemented with **simulateXxx()**.

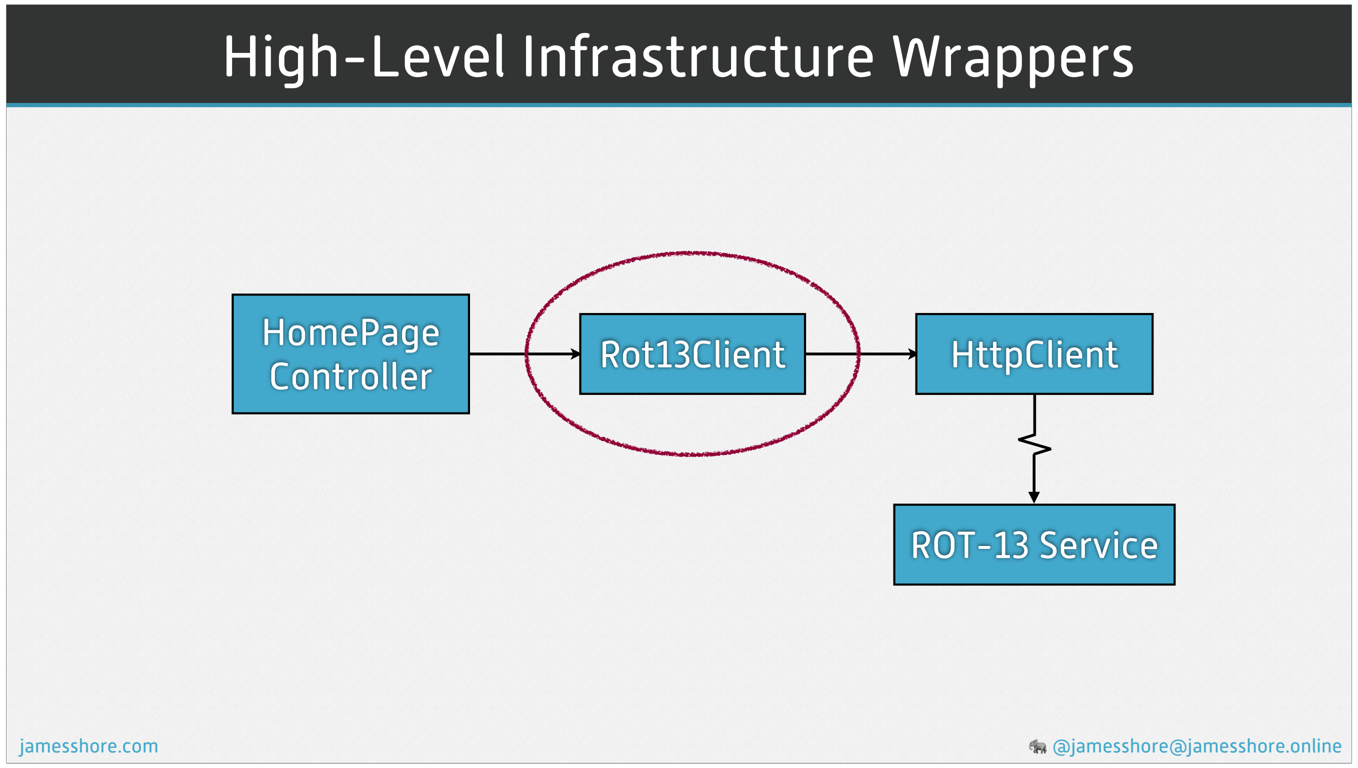
jamesshore.com

@jamesshore@jamesshore.online

To recap, we've been talking about a collection of patterns for writing **sociable, state-based tests** rather than mock-based tests, which are **solitary, interaction-based tests**. These patterns are useful because solitary tests pass when they should fail, and interaction-based tests fail when they should pass.

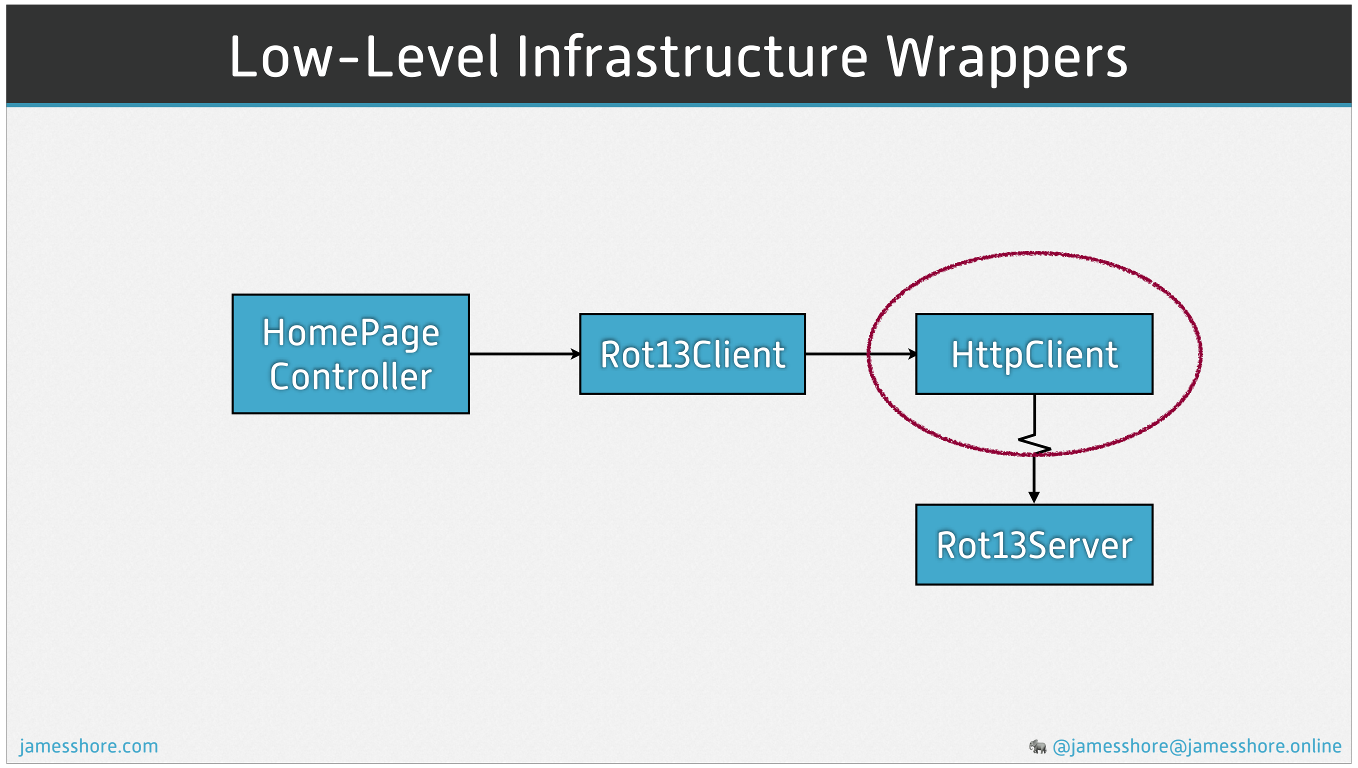
There are four core patterns:

1) **Nullables**, which are production code with an "off" switch. They can be configured to disable communication with the outside world by calling the "createNull()" factory method.
2) **Configurable Responses**, which is a way of controlling what Nullables return.
3) **Output Tracking**, which a way of tracking calls to external systems.
4) **Behavior Simulation**, which is a way of simulating events that come from external systems.

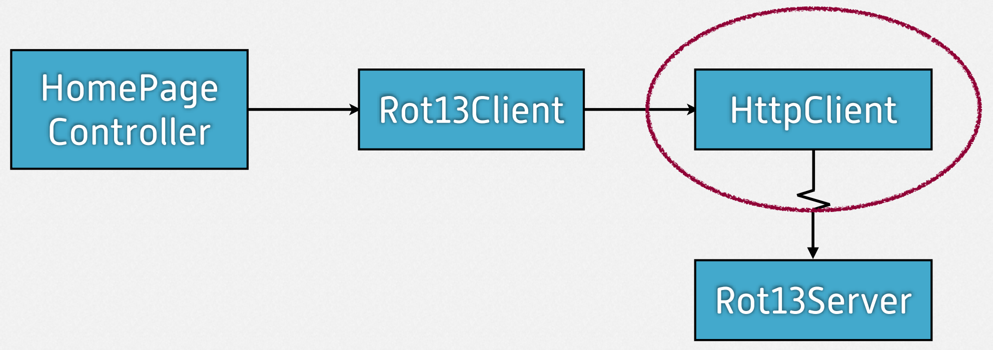


High-Level Infrastructure Wrappers

Last time, you learned how to implement those patterns in high-level infrastructure. Remember, high-level infrastructure wrappers are responsible for providing a clean interface to a specific piece of infrastructure. In our ROT-13 example, the Rot13Client is a high-level wrapper that's responsible for presenting a clean interface to the ROT-13 service.

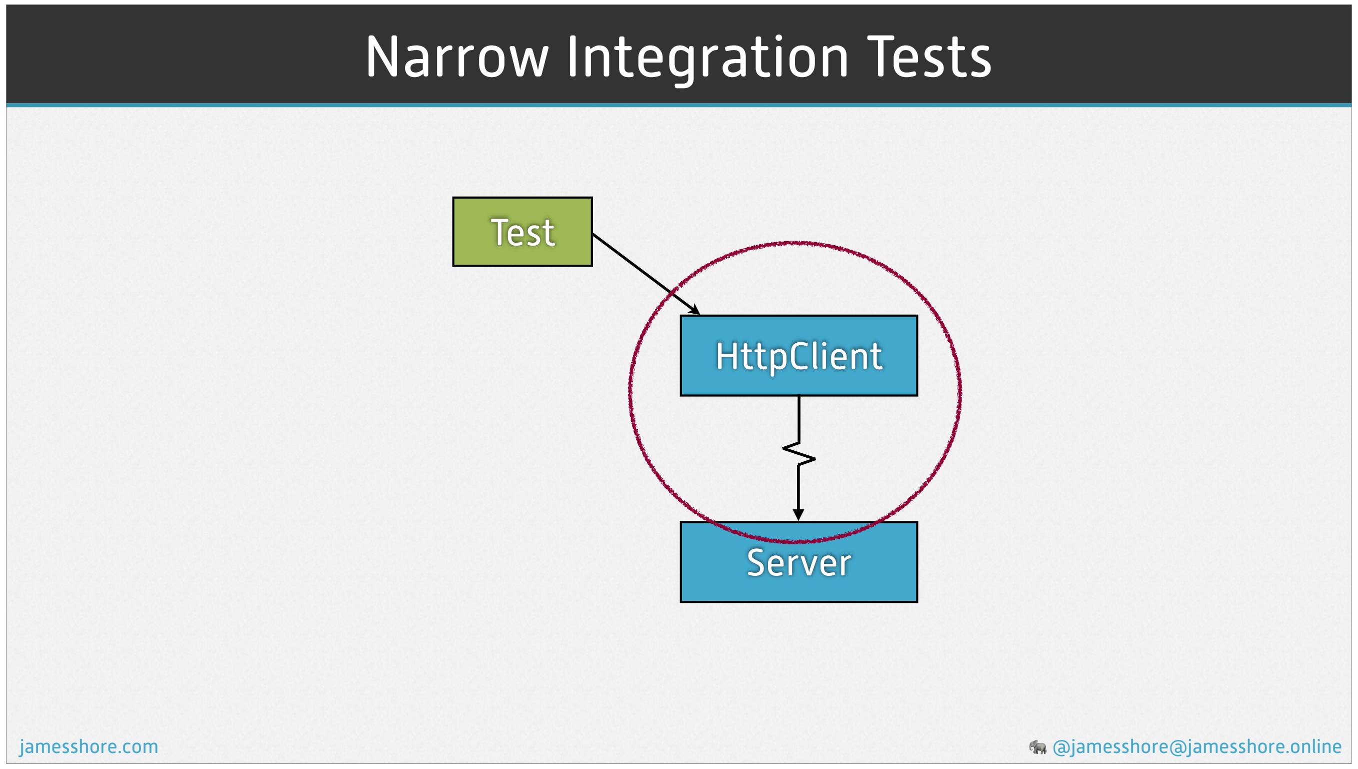


Low-Level Infrastructure Wrappers

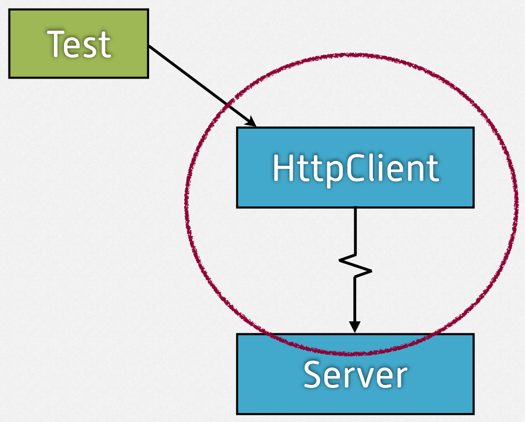


Today, you're going to learn how to implement low-level infrastructure wrappers. Low-level wrappers provide a generic interface for some communication technology. For example, you might have an HttpClient for HTTP, a PostgresDatabase for your database, or a WebSocketClient for real-time communication.

Low-level infrastructure wrappers are complicated, so the lesson is split up into two parts. Today, you'll learn how to test-drive a low-level wrapper. Next time, you'll learn how to make it nullable.



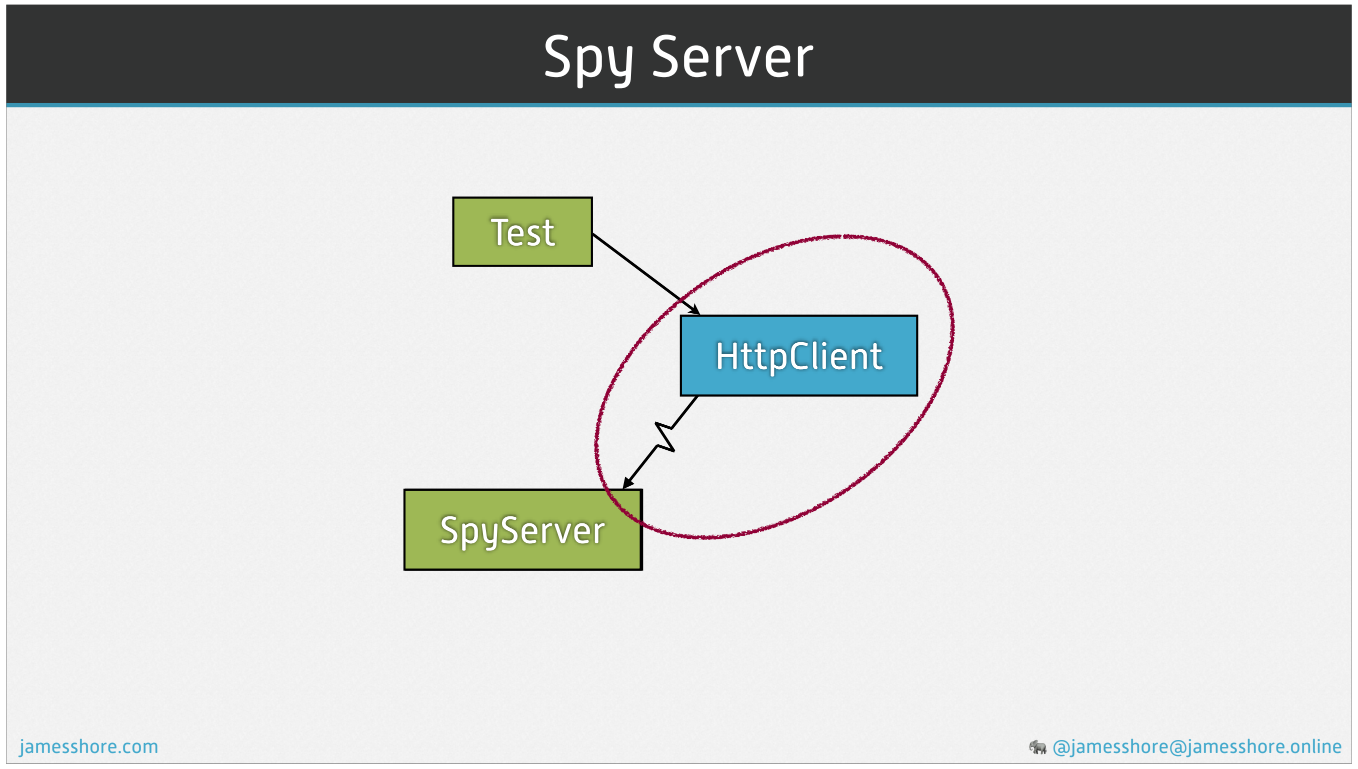
Narrow Integration Tests



Because low-level infrastructure wrappers abstract a communication protocol, we need tests that actually talk to a real server. Otherwise, how do we know the communication works? But we can write tests that almost as good as in-memory unit tests. They're called **narrow integration tests**. They're **narrow** because they're focused on a specific behavior—the communication protocol. They're **integration tests** because they talk to an external system.

Narrow integration tests need to talk to a real server. To prevent network issues and contention for test environments, run the server on your local development machine whenever possible.

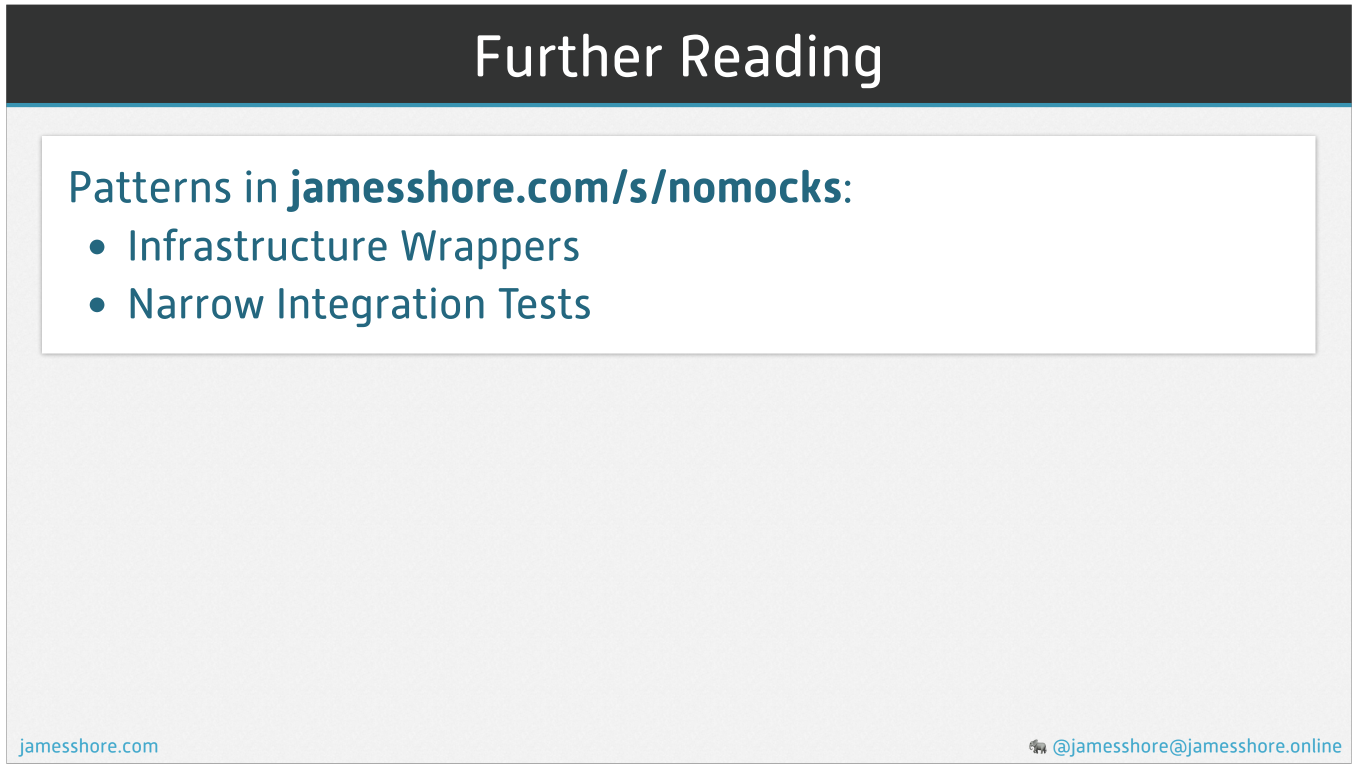
If you're creating a generic low-level wrapper, like HttpClient, the specific server won't be important. In that case, it's often best to create a **test** server that runs inside your tests and reports low-level details about how it's called. This is called a **Spy Server.**



Spy Server

If you're creating a generic low-level wrapper, like HttpClient, the specific server won't be important. In that case, it's often best to create a **test** server that runs inside your tests and reports low-level details about how it's called. This is called a **Spy Server.**

That's what you're going to do today. You're going to create a Spy Server and use it to write tests for HttpClient.



Further Reading

Patterns in **jamesshore.com/s/nomocks**:

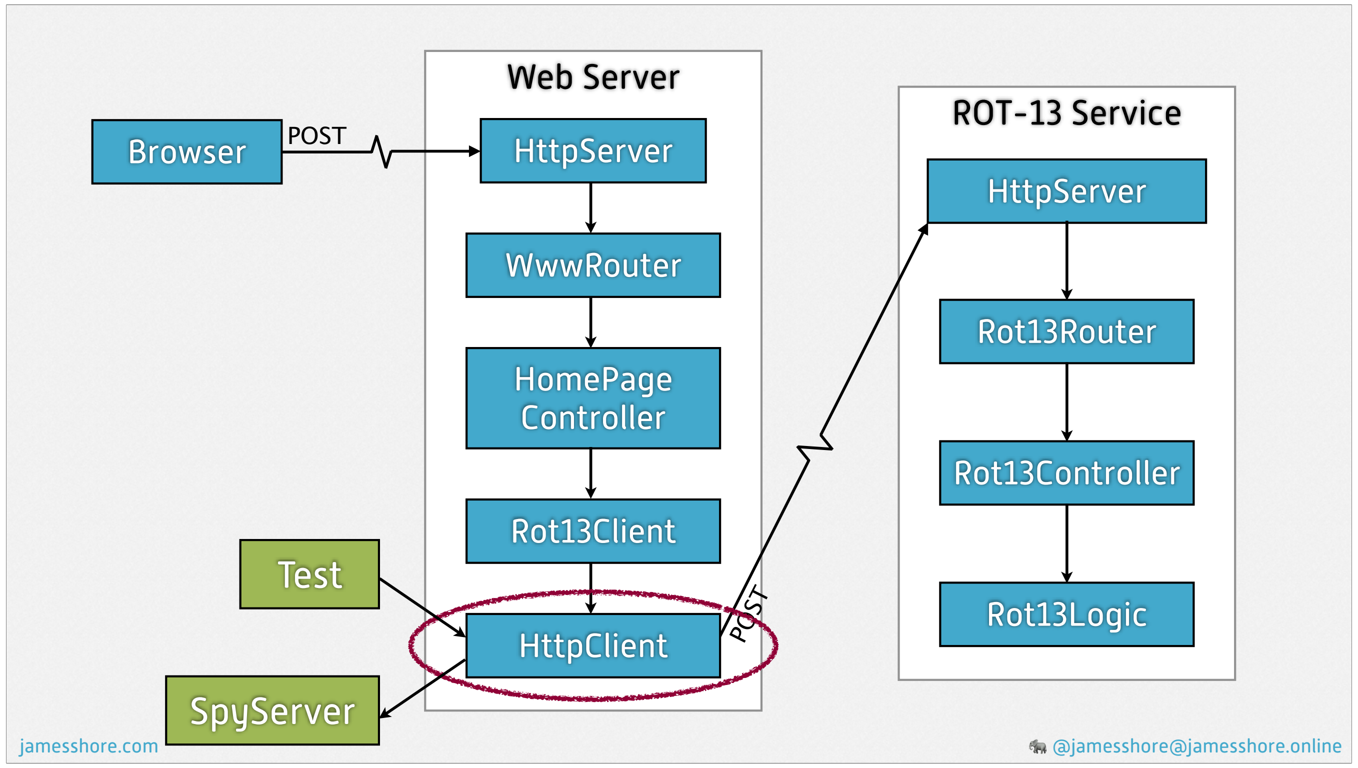
- Infrastructure Wrappers
- Narrow Integration Tests

jamesshore.com

@jamesshore@jamesshore.online

As usual, you can find more about these ideas in the “Testing Without Mocks” paper. The concepts are simple, but the implementation is complicated, so we’ll be spending nearly all our time on the exercises today.

Exercises

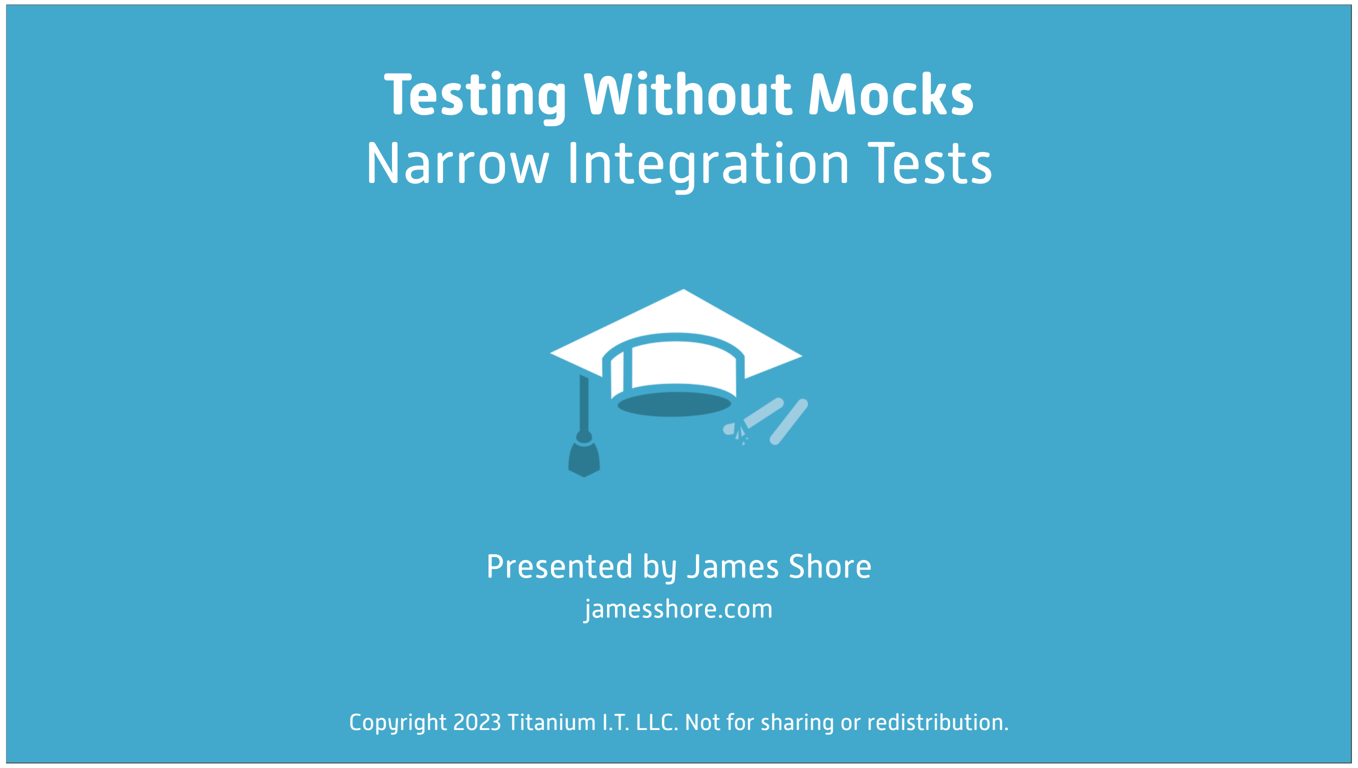


As I said, today you’ll be implementing the HttpClient and its Spy Server from scratch. This gets into low-level Node.js APIs, so be sure to use the hints. You can also read the “Concurrency” primer to get a better understanding of how Node.js events work.

(Walk through exercise setup.)

(Reminder about API docs, primers, and hints.)

(Reminder to use "Ask for help" button.)



Testing Without Mocks
Narrow Integration Tests

Presented by James Shore
jamesshore.com

And that’s a brief introduction to the concepts around narrow integration tests. As usual, the majority of the learning will come from doing the exercises. Remember to use those hints, and good luck.