

The LockManager App Case Study: Test Driver & Client Implementation

Douglas C. Schmidt

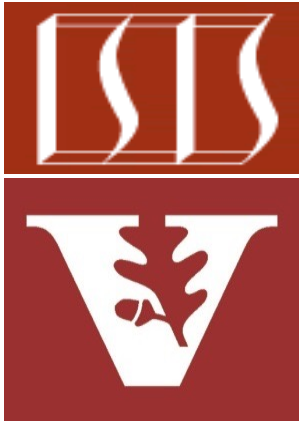
d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

**Institute for Software
Integrated Systems**

**Vanderbilt University
Nashville, Tennessee, USA**



Learning Objectives in this Part of the Lesson

- Understand the implementation of the LockManagerTest class & associated client code that invoke asynchronous methods on the LockManagerController

LockManagerTest

```
19 //
20 // This program tests the PrimeCheckClient and its ability to
21 // communicate with the PrimeCheckServerController.
22 //
23 //
24 @SpringBootTest
25 @ContextConfiguration(classes = {
26     Components.class,
27     PrimeCheckClient.class,
28     PrimeCheckController.class
29 })
30 public class PrimeCheckTest {
31     //
32     // Debugging tag used by the logger.
33     //
34     private final String TAG = getClass().getSimpleName();
35
36     //
37     // This object connects to the TestClient. The @Autowired
38     // annotation ensures this field is initialized via Spring
39     // dependency injection, where an object receives another object
40     // it depends on (e.g., by creating a @Link PrimeCheckClient?).
41 }
```



**Asynchronous
HTTP GET/POST
requests/
responses**

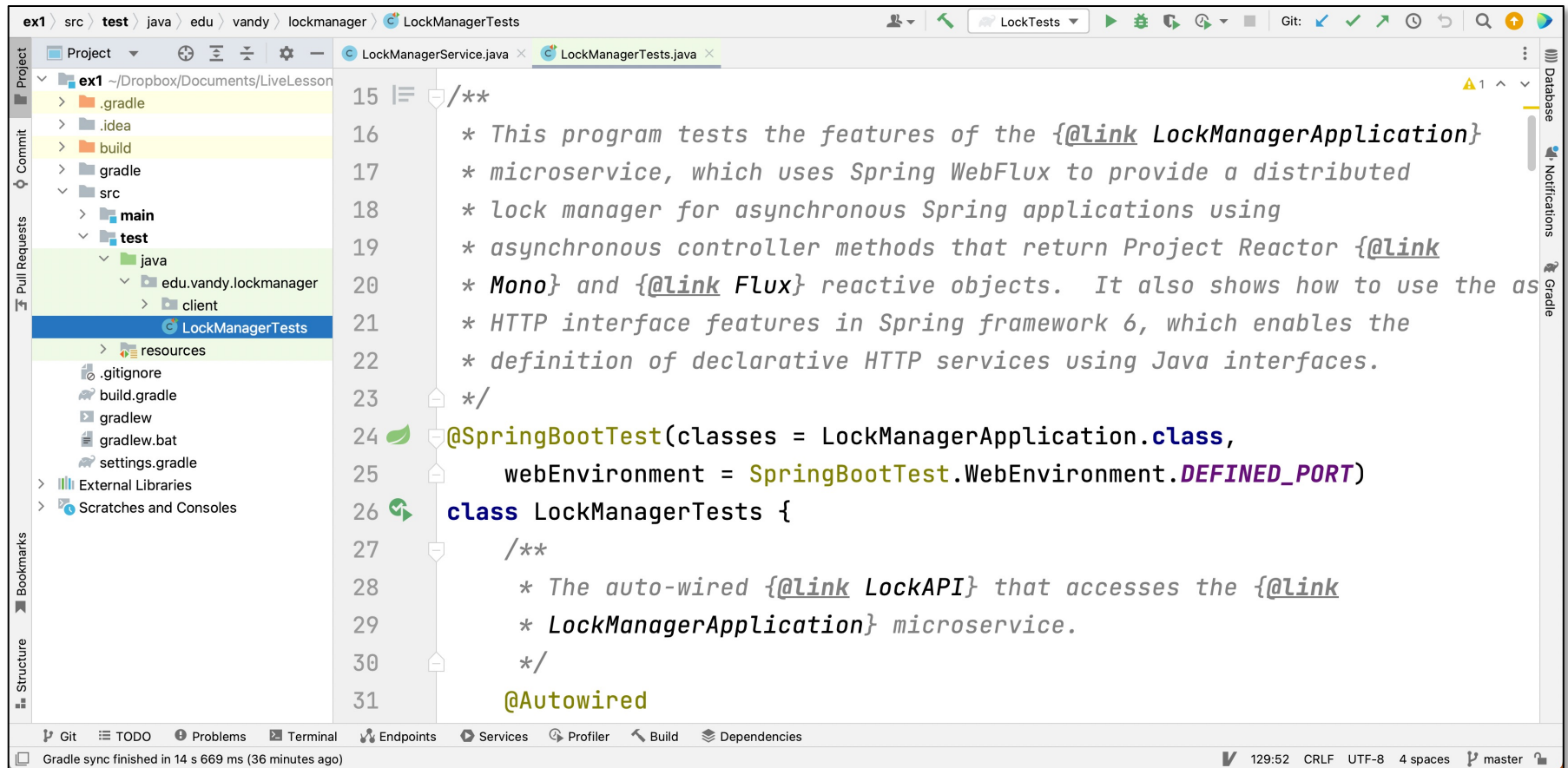
LockManagerApplication



See github.com/douglasraigschmidt/LiveLessons/tree/master/WebFlux/ex1

Implementing the LockManagerTest Driver

Implementing the LockManagerTest Driver



```
ex1 | src | test | java | edu | vandy | lockmanager | LockManagerTests
Project | Commit | Pull Requests | Bookmarks | Structure
ex1 ~/Dropbox/Documents/LiveLesson
> .gradle
> .idea
> build
> gradle
> src
> main
> test
> java
> edu.vandy.lockmanager
> client
LockManagerTests
> resources
.gitignore
build.gradle
gradlew
gradlew.bat
settings.gradle
External Libraries
Scratches and Consoles
Database
Notifications
Gradle

15 /**
16  * This program tests the features of the {@link LockManagerApplication}
17  * microservice, which uses Spring WebFlux to provide a distributed
18  * lock manager for asynchronous Spring applications using
19  * asynchronous controller methods that return Project Reactor {@link
20  * Mono} and {@link Flux} reactive objects. It also shows how to use the as
21  * HTTP interface features in Spring framework 6, which enables the
22  * definition of declarative HTTP services using Java interfaces.
23  */
24 @SpringBootTest(classes = LockManagerApplication.class,
25                 webEnvironment = SpringBootTest.WebEnvironment.DEFINED_PORT)
26 class LockManagerTests {
27     /**
28      * The auto-wired {@link LockAPI} that accesses the {@link
29      * LockManagerApplication} microservice.
30      */
31     @Autowired
```

Git | TODO | Problems | Terminal | Endpoints | Services | Profiler | Build | Dependencies

Gradle sync finished in 14 s 669 ms (36 minutes ago) | 129:52 | CRLF | UTF-8 | 4 spaces | master

See [WebFlux/ex1/src/test/java/edu/vandy/lockmanager](https://github.com/spring-projects/spring-webflux/tree/master/spring-webflux-examples/ex1/src/test/java/edu/vandy/lockmanager)

End of the LockManager App Case Study: Test Driver & Client Implementation