The LockManager App Case Study: Test Driver Implementation & Behavior

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

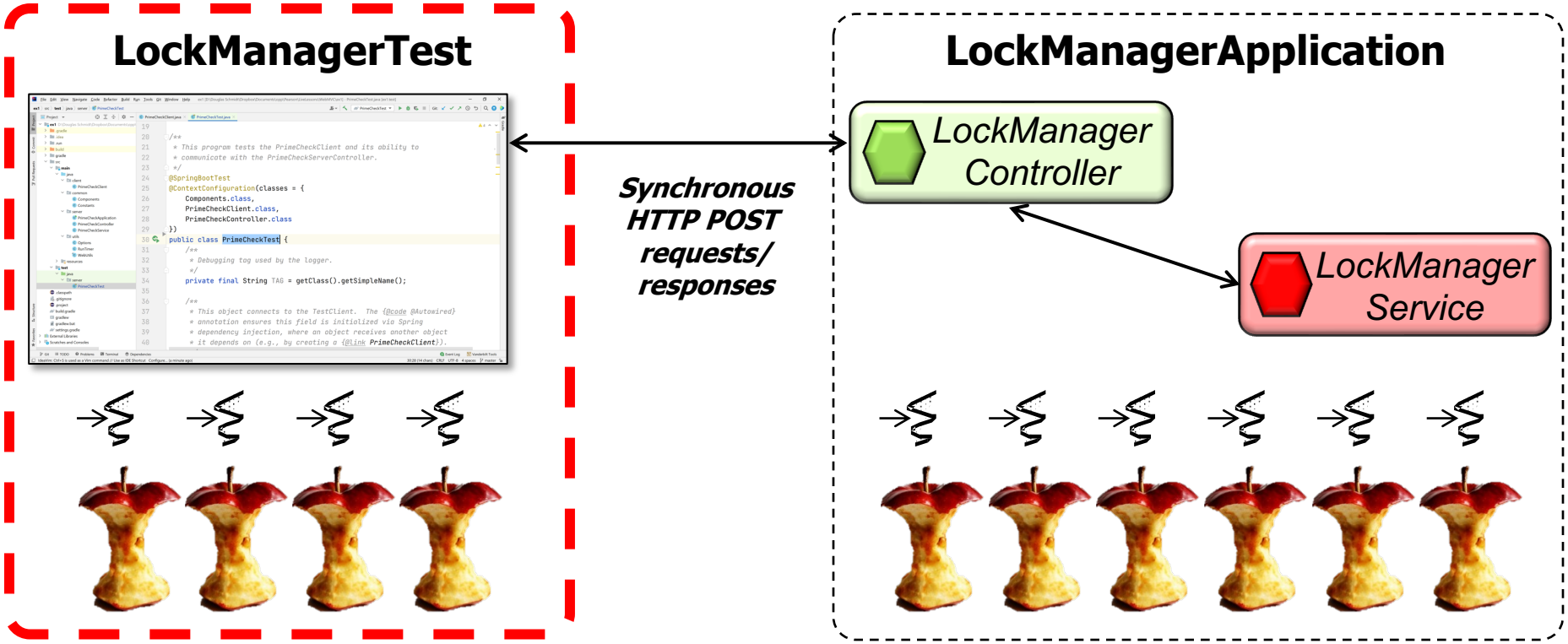
Professor of Computer Science

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 synchronous methods on the LockManagerController

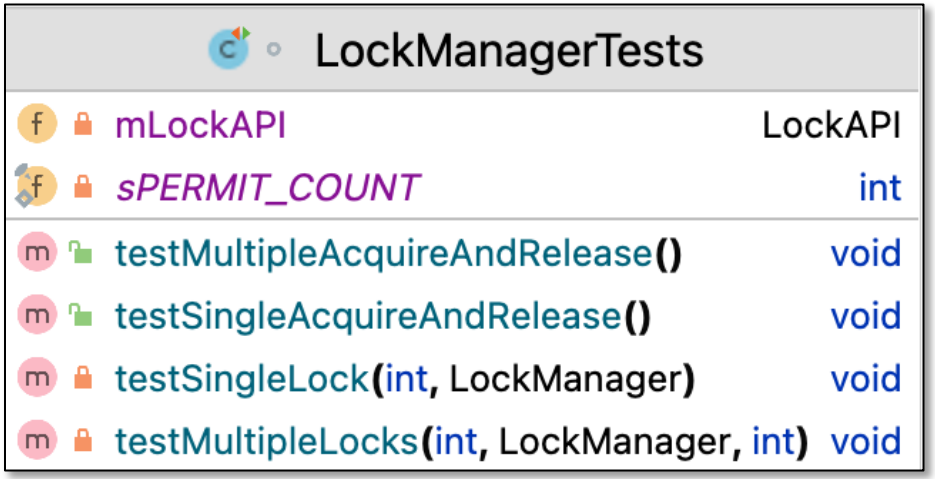


See github.com/douglascraigschmidt/LiveLessons/tree/master/WebMVC/ex5

Overview of the LockManagerTest Driver

Overview of the LockManagerTest Driver

- This SpringBootTest class exercises all the features of the LockManager Application microservice

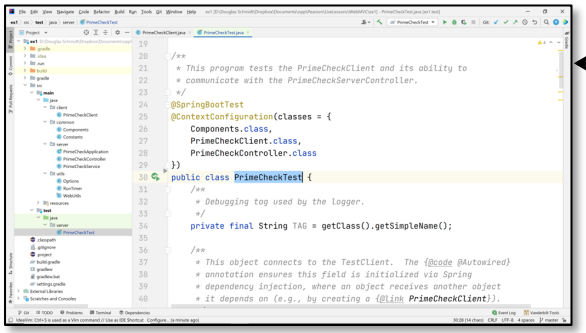


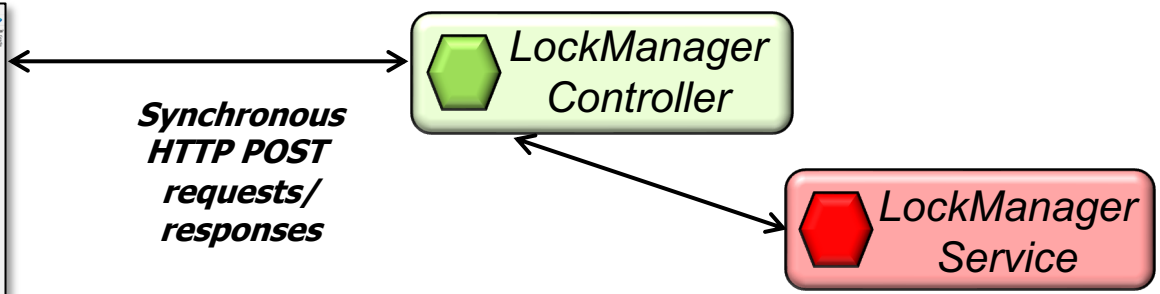
See www.baeldung.com/spring-boot-testing

Overview of the LockManagerTest Driver

- This SpringBootTest class exercises all the features of the LockManager Application microservice
  - It loads the application context & tests the interaction between client & server components

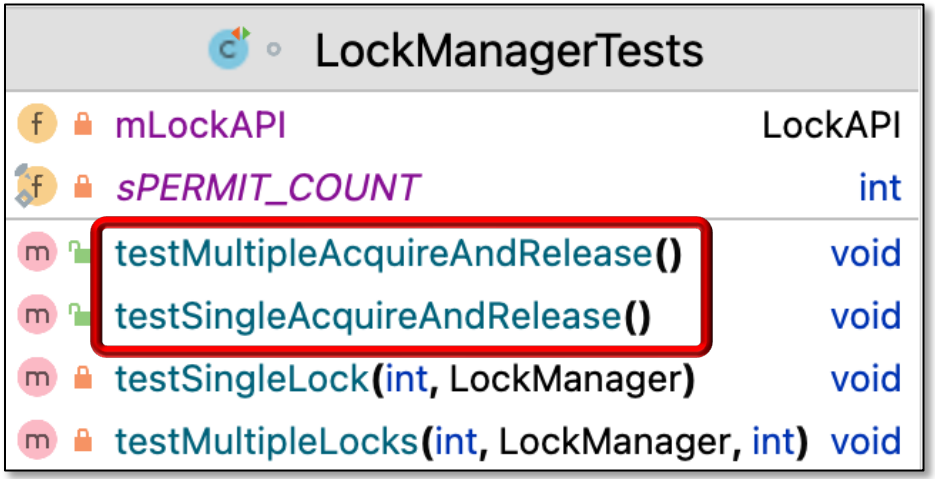
LockManagerTests	
mLockAPI	LockAPI
sPERMIT_COUNT	int
testMultipleAcquireAndRelease()	void
testSingleAcquireAndRelease()	void
testSingleLock(int, LockManager)	void
testMultipleLocks(int, LockManager, int)	void

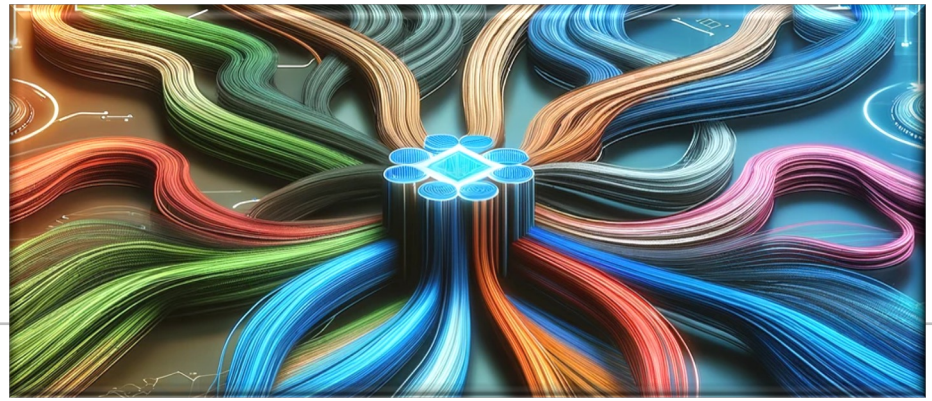




Overview of the LockManagerTest Driver

- This SpringBootTest class exercises all the features of the LockManager Application microservice
  - It loads the application context & tests the interaction between client & server components
  - Its @Test-annotated methods validate the behavior of the LockManagerApplication under concurrent scenarios where multiple processes/threads interact via shared resources





Overview of the LockManagerTest Driver

- The testSingleAcquireAndRelease() method
  - Ensure multiple client threads can acquire & release a single lock concurrently from a LockManager

```
int maxLocks = 2;
int maxClients = 4;

var lockManager = mLockAPI
   .create(maxLocks);

IntStream
   .range(0, maxClients)
   .parallel()
   .forEach(client ->
              testSingleLock
                 (client,
                  lockManager));
```

Overview of the LockManagerTest Driver

- The testSingleAcquireAndRelease() method
  - Ensure multiple client threads can acquire & release a single lock concurrently from a LockManager

Create a LockManager with a permit count of 2

```
int maxLocks = 2;
int maxClients = 4;

var lockManager = mLockAPI
   .create(maxLocks);

IntStream
   .range(0, maxClients)
   .parallel()
   .forEach(client ->
              testSingleLock
                 (client,
                  lockManager));
```

Overview of the LockManagerTest Driver

- The testSingleAcquireAndRelease() method
  - Ensure multiple client threads can acquire & release a single lock concurrently from a LockManager

```
int maxLocks = 2;
int maxClients = 4;

var lockManager = mLockAPI
   .create(maxLocks);

IntStream
   .range(0, maxClients)
   .parallel()
   .forEach(client ->
            testSingleLock
              (client,
               lockManager));
```

Execute test operations for 4 clients concurrently using parallel streams

Overview of the LockManagerTest Driver

- The testSingleAcquireAndRelease() method
  - Ensure multiple client threads can acquire & release a single lock concurrently from a LockManager

```
int maxLocks = 2;
int maxClients = 4;

var lockManager = mLockAPI
   .create(maxLocks);

IntStream
   .range(0, maxClients)
   .parallel()
   .forEach(client ->
              testSingleLock
                (client,
                 lockManager));
```

Each client calls the testSingleLock() method, which tests acquiring & releasing a single lock

Overview of the LockManagerTest Driver

- The testSingleLock() method
  - Acquire & release a single Lock object on a given client

```
void testSingleLock
   (int client,
    LockManager lockManager) {

     var lock = mLockAPI
        .acquire(lockManager);


     var result = mLockAPI
        .release(lockManager,
                 lock);

}
```

Overview of the LockManagerTest Driver

- The testSingleLock() method
  - Acquire & release a single Lock object on a given client

Invoke remote acquire() & release() methods on the LockManagerController via the generated LockAPI proxy

```
void testSingleLock
   (int client,
    LockManager lockManager)  {

    var lock = mLockAPI
      .acquire(lockManager);
    ...

    var result = mLockAPI
      .release(lockManager,
               lock);

}
```

Overview of the LockManagerTest Driver

- The testMultipleAcquireAndRelease() method
  - Ensure multiple client threads can acquire & release multiple locks concurrently from a LockManager

```
int maxLocks = 4;
int maxClients = 8;
int maxPermits = 2;

var lockManager = mLockAPI
   .create(maxLocks);

IntStream
   .range(0, maxClients)
   .parallel()
   .forEach(client ->
              testMultipleLocks
                 (client,
                  lockManager,
                  maxPermits));
```

Overview of the LockManagerTest Driver

- The testMultipleAcquireAndRelease() method
  - Ensure multiple client threads can acquire & release multiple locks concurrently from a LockManager

```
int maxLocks = 4;
int maxClients = 8;
int maxPermits = 2;

var lockManager = mLockAPI
   .create(maxLocks);

IntStream
   .range(0, maxClients)
   .parallel()
   .forEach(client ->
              testMultipleLocks
                 (client,
                  lockManager,
                  maxPermits));
```

Create a LockManager with a permit count of 4

Overview of the LockManagerTest Driver

- The testMultipleAcquireAndRelease() method
  - Ensure multiple client threads can acquire & release multiple locks concurrently from a LockManager

```
int maxLocks = 4;
int maxClients = 8;
int maxPermits = 2;

var lockManager = mLockAPI
   .create(maxLocks);

IntStream
   .range(0, maxClients)
   .parallel()
   .forEach(client ->
            testMultipleLocks
              (client,
               lockManager,
               maxPermits));
```

Execute test operations for 8 clients concurrently using parallel streams

Overview of the LockManagerTest Driver

- The testMultipleAcquireAndRelease() method
  - Ensure multiple client threads can acquire & release multiple locks concurrently from a LockManager

```
int maxLocks = 4;
int maxClients = 8;
int maxPermits = 2;

var lockManager = mLockAPI
   .create(maxLocks);

IntStream
   .range(0, maxClients)
   .parallel()
   .forEach(client ->
              testMultipleLocks
               (client,
                lockManager,
                maxPermits));
```

Each client calls the testMultipleLocks() method, which tests acquiring & releasing multiple permits

Overview of the LockManagerTest Driver

- The testMultipleLocks() method
  - Acquire & release multiple Lock objects on a given client

```
void testMultipleLocks
   (int client,
    LockManager lockManager,
    int maxPermits) {

     var locks = mLockAPI
       .acquire(lockManager,
                maxPermits);
     ...

     var result = mLockAPI
       .release(lockManager,
                locks);

}
```

Overview of the LockManagerTest Driver

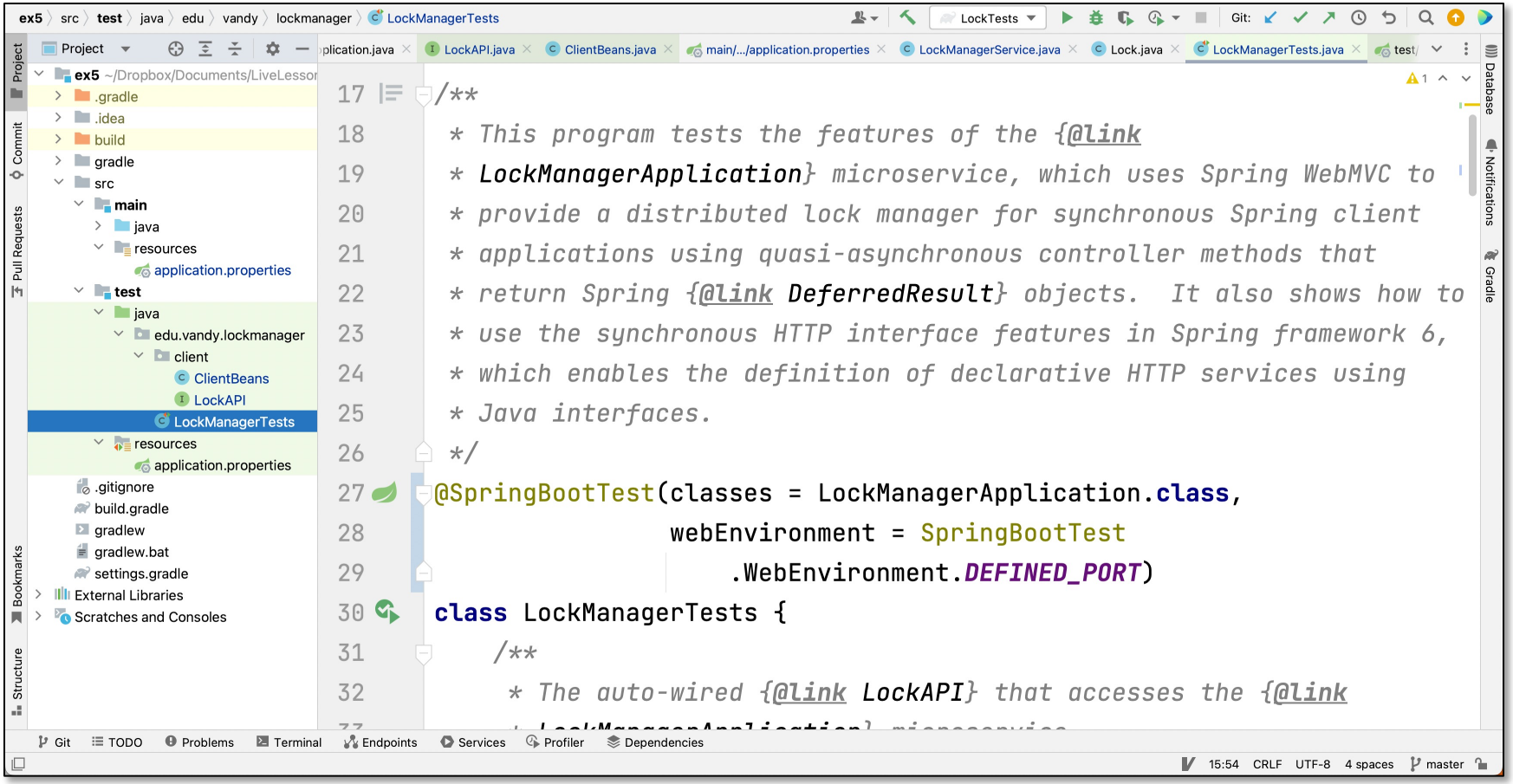
- The testMultipleLocks() method
  - Acquire & release multiple Lock objects on a given client

```
void testMultipleLocks
   (int client,
    LockManager lockManager,
    int maxPermits)  {

     var locks = mLockAPI
       .acquire(lockManager,
                maxPermits);

     ...

     var result = mLockAPI
       .release(lockManager,
                locks);

}
```

Invoke remote acquire() & release() methods on the LockManagerController via the generated LockAPI proxy

Implementing the LockManagerTest Driver

Implementing the LockManagerTest Driver



See WebMVC/ex5/src/test/java/edu/vandy/lockmanager/LockManagerTests.java

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