The LockManager App Case Study: Server Structure & Functionality

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

- This case study shows the structure & functionality of the controller & service classes for a LockManager microservice developed using Spring WebMVC.

See WebMVC/ex5/src/main/java/edu/vandy/lockmanager/server
Structure & Functionality of the LockManagerController
Structure & Functionality of the LockManagerController

- LockManagerController maps HTTP GET/POST requests to endpoint handlers

```java
@RestController
public class LockManagerController {
    @Autowired
    LockManagerService mService;

    ...
}
```

See WebMVC/ex5/src/main/java/edu/vandy/lockmanager/server/LockManagerController.java
Structure & Functionality of the LockManagerController

- LockManagerController maps HTTP GET/POST requests to endpoint handlers

```java
@RestController
public class LockManagerController {
    @Autowired
    LockManagerService mService;
    ...

    This annotation ensures request handling methods in the controller class automatically serialize return objects into HttpResponse objects
```

See [www.baeldung.com/spring-controller-vs-restcontroller](http://www.baeldung.com/spring-controller-vs-restcontroller)
Structure & Functionality of the LockManagerController

- LockManagerController maps HTTP GET/POST requests to endpoint handlers

```java
@RestController
public class LockManagerController {
  @Autowired
  LockManagerService mService;

  ...
```

See [www.baeldung.com/spring-autowire](http://www.baeldung.com/spring-autowire)

This field is auto-wired by Spring’s dependency injection framework
Structure & Functionality of the LockManagerController

- LockManagerController maps HTTP GET/POST requests to endpoint handlers

```java
@RestController
public class LockManagerController {
    ...
    @PostMapping(CREATE)
    public Boolean create(@RequestBody Integer permitCount)
    @GetMapping(ACQUIRE_LOCK)
    public DeferredResult<Lock> acquire()
    @GetMapping(ACQUIRE_LOCKS)
    public DeferredResult<List<Lock>> acquire(Integer permits)
    @PostMapping(RELEASE_LOCK)
    public Boolean release(@RequestBody Lock lock)
    @PostMapping(RELEASE_LOCKS)
    public Boolean release(@RequestBody List<Lock> locks)
    ...
}
```

These endpoint handler methods forward to the LockManagerService methods that fulfill the requests.
Structure & Functionality of the LockManagerController

- LockManagerController maps HTTP GET/POST requests to endpoint handlers

```java
@RestController
public class LockManagerController {
    ...
    @PostMapping(CREATE)
    public Boolean create(@RequestBody Integer permitCount)
    @GetMapping(ACQUIRE_LOCK)
    public DeferredResult<Lock> acquire()
    @GetMapping(ACQUIRE_LOCKS)
    public DeferredResult<List<Lock>> acquire(Integer permits)
    @PostMapping(RELEASE_LOCK)
    public Boolean release(@RequestBody Lock lock)
    @PostMapping(RELEASE_LOCKS)
    public Boolean release(@RequestBody List<Lock> locks)
    ...
}
```

See [www.baeldung.com/spring-new-requestmapping-shortcuts](http://www.baeldung.com/spring-new-requestmapping-shortcuts)
Structure & Functionality of the LockManagerController

- LockManagerController maps HTTP GET/POST requests to endpoint handlers

```java
@RestController
public class LockManagerController {
    ...
    @PostMapping(CREATE)
    public Boolean create(@RequestBody Integer permitCount) {...}
    @GetMapping(ACQUIRE_LOCK)
    public DeferredResult<Lock> acquire() {
        @GetMapping(ACQUIRE_LOCKS)
        public DeferredResult<List<Lock>> acquire(Integer permits) {
            @PostMapping(RELEASE_LOCK)
            public Boolean release(@RequestBody Lock lock) {
                @PostMapping(RELEASE_LOCKS)
                public Boolean release(@RequestBody List<Lock> locks) {
                    ...
        
See [www.baeldung.com/spring-new-requestmapping-shortcuts](http://www.baeldung.com/spring-new-requestmapping-shortcuts)
```
Structure & Functionality of the LockManagerController

- LockManagerController maps HTTP GET/POST requests to endpoint handlers

```java
@RestController
public class LockManagerController {
    ... 
    @PostMapping(CREATE)
    public Boolean create(@RequestBody Integer permitCount)
    @GetMapping(ACQUIRE_LOCK)
    public DeferredResult<Lock> acquire()
    @GetMapping(ACQUIRE_LOCKS)
    public DeferredResult<List<Lock>> acquire(Integer permits)
    @PostMapping(RELEASE_LOCK)
    public Boolean release(@RequestBody Lock lock)
    @PostMapping(RELEASE_LOCKS)
    public Boolean release(@RequestBody List<Lock> locks)
    ...
}
```

See [www.baeldung.com/spring-request-response-body](http://www.baeldung.com/spring-request-response-body)
Structure & Functionality of the LockManagerController

- LockManagerController maps HTTP GET/POST requests to endpoint handlers

```java
@RestController
public class LockManagerController {
  ...
  @PostMapping(CREATE)
  public Boolean create(@RequestBody Integer permitCount)
  @GetMapping(ACQUIRE_LOCK)
  public DeferredResult<Lock> acquire()
  @GetMapping(ACQUIRE_LOCKS)
  public DeferredResult<List<Lock>> acquire(Integer permits)
  @PostMapping(RELEASE_LOCK)
  public Boolean release(@RequestBody Lock lock)
  @PostMapping(RELEASE_LOCKS)
  public Boolean release(@RequestBody List<Lock> locks)
  ...
}
```

This Java class enables this microservice to produce the result from a thread of its choice.

See [springframework/web/context/request/async/DeferredResult.html](http://springframework/web/context/request/async/DeferredResult.html)
Structure & Functionality of the LockManagerService
Structure & Functionality of the LockManagerService

- LockManagerService defines methods called by LockManagerController, which implements a distributed semaphore using a Java ArrayBlockingQueue

```java
@Service
public class LockManagerService {
    private ExecutorService mExecutor = Executors
        .newVirtualThreadPerTaskExecutor();

    private ArrayBlockingQueue<Lock>
        mAvailableLocks;

    ...
```

See WebMVC/ex5/src/main/java/edu/vandy/lockmanager/server/LockManagerService.java
Structure & Functionality of the LockManagerService

- LockManagerService defines methods called by LockManagerController, which implements a distributed semaphore using a Java ArrayBlockingQueue

```java
@Service
public class LockManagerService {
    private ExecutorService mExecutor = Executors.newVirtualThreadPerTaskExecutor();

    private ArrayBlockingQueue<Lock> mAvailableLocks;
    ...
}
```

This annotation indicates the class implements "business logic" & enables auto-detection & wiring of dependent classes via classpath scanning

See [www.baeldung.com/spring-component-repository-service](http://www.baeldung.com/spring-component-repository-service)
LockManagerService defines methods called by LockManagerController, which implements a distributed semaphore using a Java ArrayBlockingQueue

@Service
class LockManagerService {
    private ExecutorService mExecutor = Executors.newVirtualThreadPerTaskExecutor();
    private ArrayBlockingQueue<Lock> mAvailableLocks;
    ...

See [java/util/concurrent/Executors.html#newVirtualThreadPerTaskExecutor](https://docs.oracle.com/javaee/8/api/javax/swing/swing-low-level/private/javase8reference/api/java/util/concurrent/Executors.html#newVirtualThreadPerTaskExecutor)

This Executor is used to run incoming HTTP requests off the servlet thread
Structure & Functionality of the LockManagerService

- LockManagerService defines methods called by LockManagerController, which implements a distributed semaphore using a Java ArrayBlockingQueue.

```java
@Service
public class LockManagerService {
    private ExecutorService mExecutor = Executors.
        .newVirtualThreadPerTaskExecutor();

    private ArrayBlockingQueue<Lock>
        mAvailableLocks;
    ...
```

Limits concurrent access to the fixed number of available locks managed by the LockManagerService.

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ArrayBlockingQueue.html
Structure & Functionality of the LockManagerService

• LockManagerService defines methods called by LockManagerController, which implements a distributed semaphore using a Java ArrayBlockingQueue

```java
@Service
public class LockManagerService {
    ...
    public Boolean create(Integer permitCount) {...}
    public DeferredResult<Lock> acquire() {...}
    public DeferredResult<List<Lock>> acquire(Integer permits) {.}
    public Boolean release(Lock lock) {...}
    public Boolean release(List<Lock> locks) {...}
    ...
```

These methods use the Java ArrayBlockingQueue to implement synchronous distributed semaphore semantics

See next part of the lesson on “Implementing the Server Components”
End of the LockManager App Case Study: Server Structure & Functionality