

The MathServices App Case Study: Implementing the GCD Microservice

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

**Institute for Software
Integrated Systems**

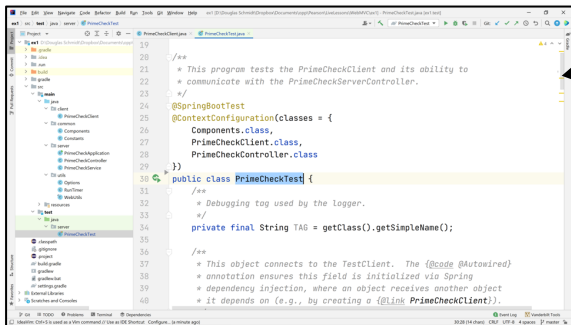
**Vanderbilt University
Nashville, Tennessee, USA**



Learning Objectives in this Part of the Lesson

- Understand the concurrent implementation of the GCDController & GCDService classes that run in the GCDApplication microservice

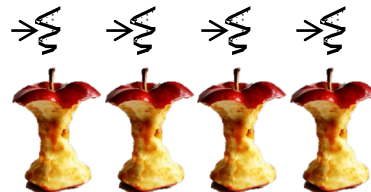
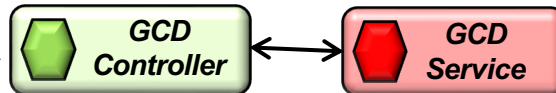
MathServicesDriver



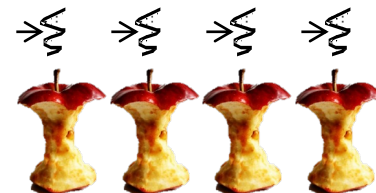
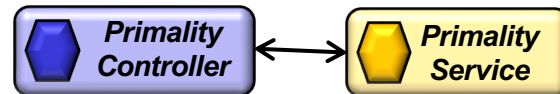
```
19 //**
20 // * This program tests the PrimeCheckClient and its ability to
21 // * communicate with the PrimeCheckServerController.
22 // */
23 //**
24 @SpringBootTest
25 @ContextConfiguration(classes = {
26     Components.class,
27     PrimeCheckClient.class,
28     PrimeCheckController.class
29 })
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      */
42 }
```

*HTTP GET
requests/
responses*

GCDApplication



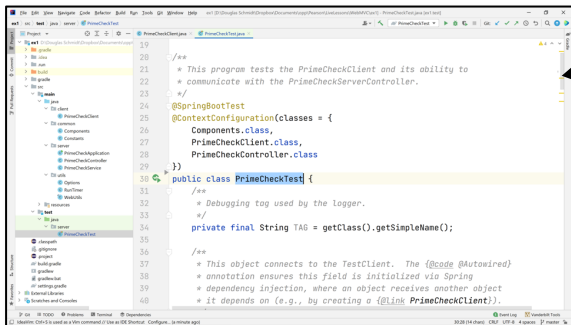
PrimalityApplication



Learning Objectives in this Part of the Lesson

- Understand the implementation of the GCDController & GCDService classes that run in the GCDApplication microservice

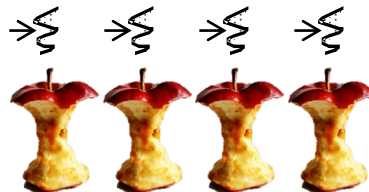
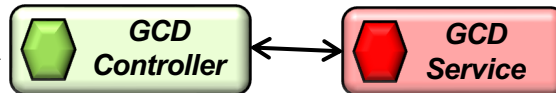
MathServicesDriver



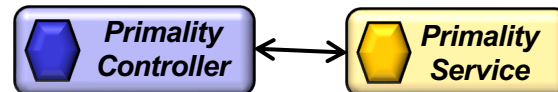
```
19 //
20 * This program tests the PrimeCheckClient and its ability to
21 * communicate with the PrimeCheckServerController.
22
23 //
24 @SpringBootTest
25 @ContextConfiguration(classes = {
26     Components.class,
27     PrimeCheckClient.class,
28     PrimeCheckController.class
29 })
30 public class 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
42     //
43 }
```

*HTTP GET
requests/
responses*

GCDApplication



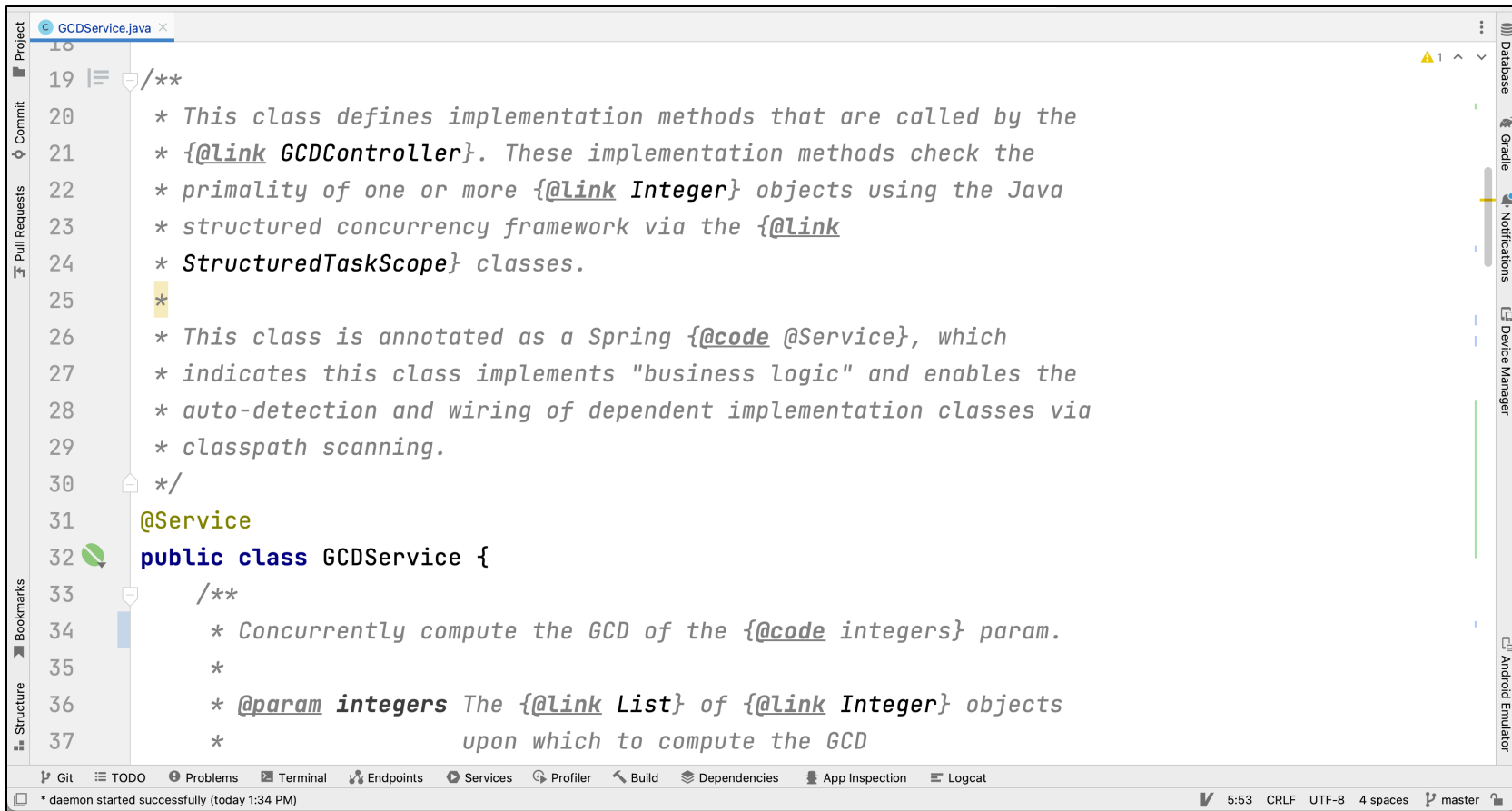
PrimalityApplication



The focus is on the Java parallel streams framework

Implementing the GCDApplication Microservice

Implementing the GCDApplication Microservice



```
18
19 /**
20  * This class defines implementation methods that are called by the
21  * {@link GCDController}. These implementation methods check the
22  * primality of one or more {@link Integer} objects using the Java
23  * structured concurrency framework via the {@link
24  * StructuredTaskScope} classes.
25  *
26  * This class is annotated as a Spring {@code @Service}, which
27  * indicates this class implements "business logic" and enables the
28  * auto-detection and wiring of dependent implementation classes via
29  * classpath scanning.
30  */
31 @Service
32 public class GCDService {
33     /**
34      * Concurrently compute the GCD of the {@code integers} param.
35      *
36      * @param integers The {@link List} of {@link Integer} objects
37      *                  upon which to compute the GCD
38     */
39 }
```

Git | TODO | Problems | Terminal | Endpoints | Services | Profiler | Build | Dependencies | App Inspection | Logcat

* daemon started successfully (today 1:34 PM)

5:53 CRLF UTF-8 4 spaces master

See github.com/douglasraigschmidt/LiveLessons/tree/master/WebMVC/ex3

End of the MathServices App Case Study: Implementing the GCD Microservices