

# **The PrimeCheck App Case Study: Implementing Server Components (Part 2)**

**Douglas C. Schmidt**

**[d.schmidt@vanderbilt.edu](mailto:d.schmidt@vanderbilt.edu)**

**[www.dre.vanderbilt.edu/~schmidt](http://www.dre.vanderbilt.edu/~schmidt)**

**Professor of Computer Science**

**Institute for Software  
Integrated Systems**

**Vanderbilt University  
Nashville, Tennessee, USA**

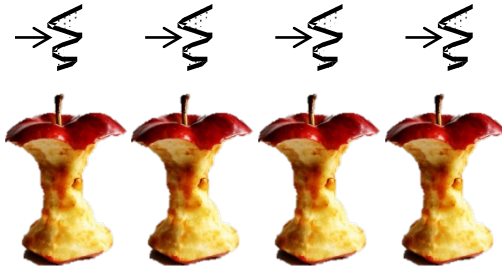


# Learning Objectives in this Part of the Lesson

- Understand the implementation of the PCServerController & PCServerService classes that run in the PrimeCheckApplication microservice

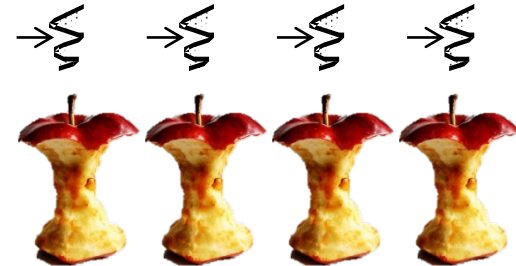
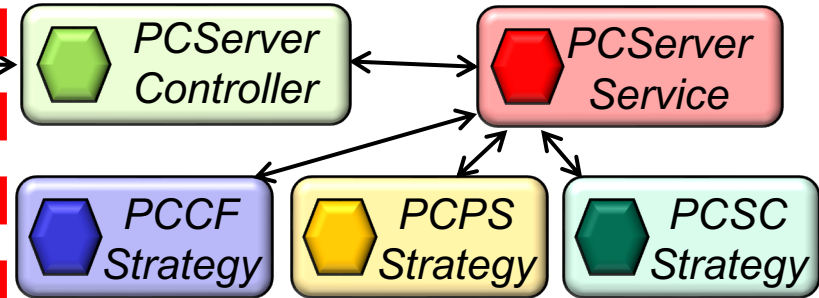
## PrimeCheckTest

```
20 /**  
21  * This program tests the PrimeCheckClient and its ability to  
22  * communicate with the PrimeCheckServerController.  
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      * This object connects to the TestClient. The @code @Autowired  
37      * annotation ensures this field is initialized via Spring  
38      * dependency injection, where an object receives another object  
39      * it depends on (e.g., by creating a @Link PrimeCheckClient).  
40      */
```



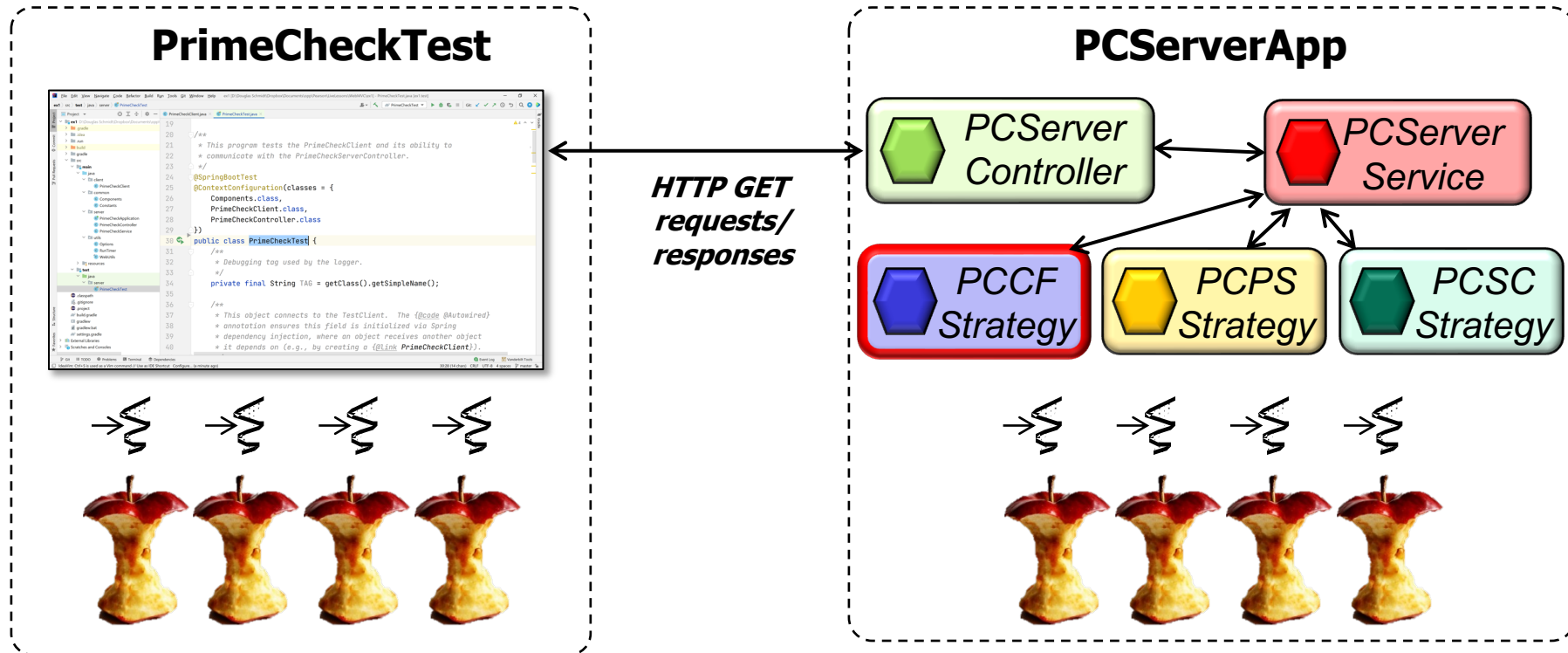
***HTTP GET  
requests/  
responses***

## PCServerApp



# Learning Objectives in this Part of the Lesson

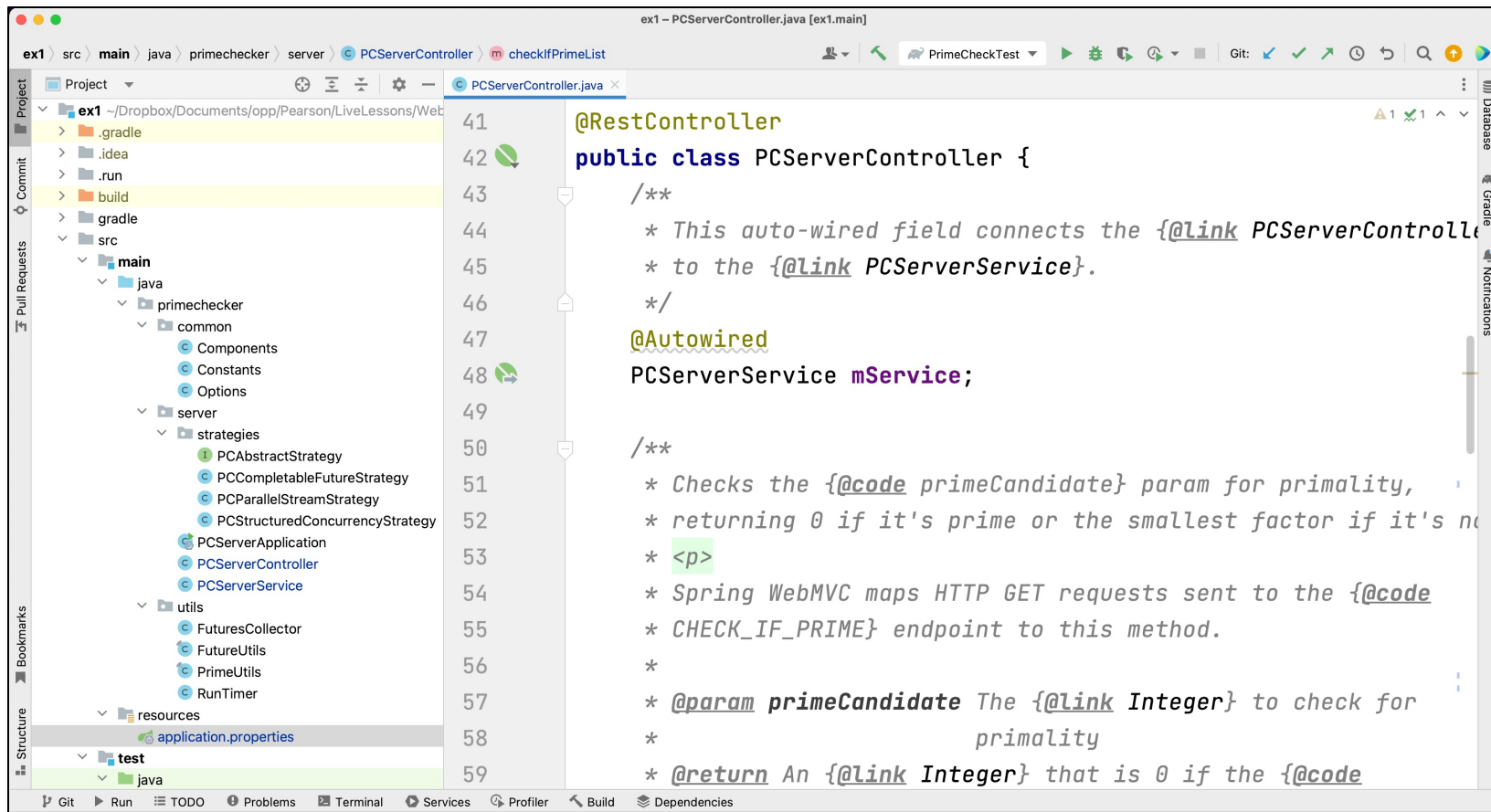
- Understand the implementation of the PCServerController & PCServerService classes that run in the PrimeCheckApplication microservice



---

# Implementing the PrimeCheck App Server

# Implementing the PrimeCheck App Server



```
ex1 - PCServerController.java [ex1.main]
ex1 | src | main | java | primechecker | server | PCServerController | checkIfPrimeList
Project | Commit | Pull Requests | Database | Gradle | Notifications
ex1 | src | main | java | primechecker | server | strategies | PCServerController
41 | @RestController
42 | public class PCServerController {
43 |     /**
44 |      * This auto-wired field connects the {@link PCServerController}
45 |      * to the {@link PCServerService}.
46 |      */
47 |     @Autowired
48 |     PCServerService mService;
49 |
50 |     /**
51 |      * Checks the {@code primeCandidate} param for primality,
52 |      * returning 0 if it's prime or the smallest factor if it's not.
53 |      * <p>
54 |      * Spring WebMVC maps HTTP GET requests sent to the {@code
55 |      * CHECK_IF_PRIME} endpoint to this method.
56 |      *
57 |      * @param primeCandidate The {@link Integer} to check for
58 |      * primality
59 |      * @return An {@link Integer} that is 0 if the {@code
```

See [github.com/douglasraigschmidt/LiveLessons/tree/master/WebMVC/ex1](https://github.com/douglasraigschmidt/LiveLessons/tree/master/WebMVC/ex1)

---

# End of the PrimeCheck App Case Study: Implementing Server Components (Part 2)