

The PrimeCheck App Case Study: Implementing Server Components (Part 1)

Douglas C. Schmidt

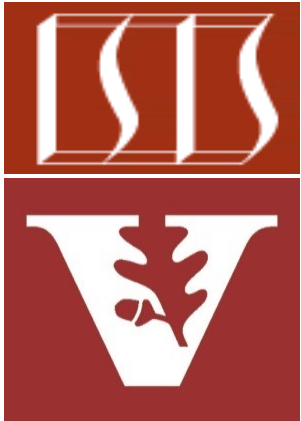
d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

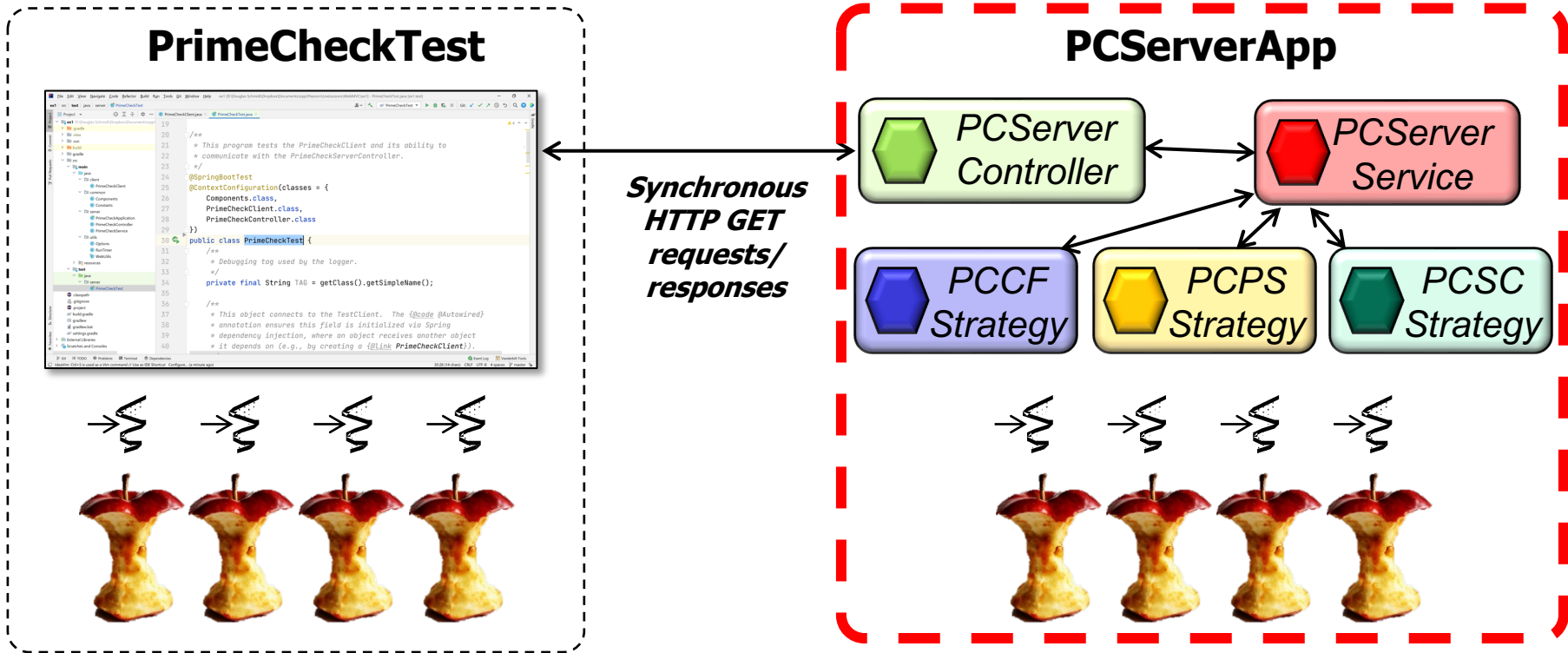
**Institute for Software
Integrated Systems**

**Vanderbilt University
Nashville, Tennessee, USA**



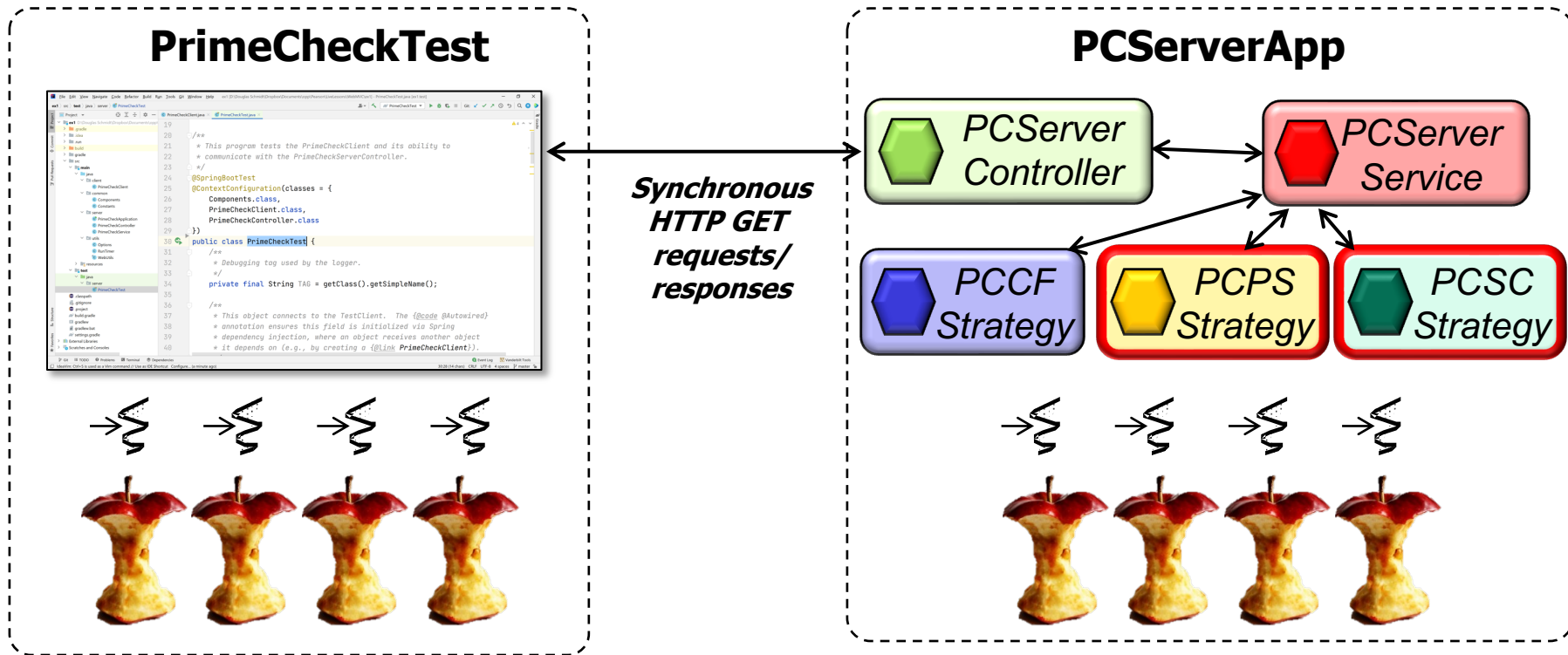
Learning Objectives in this Part of the Lesson

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



Learning Objectives in this Part of the Lesson

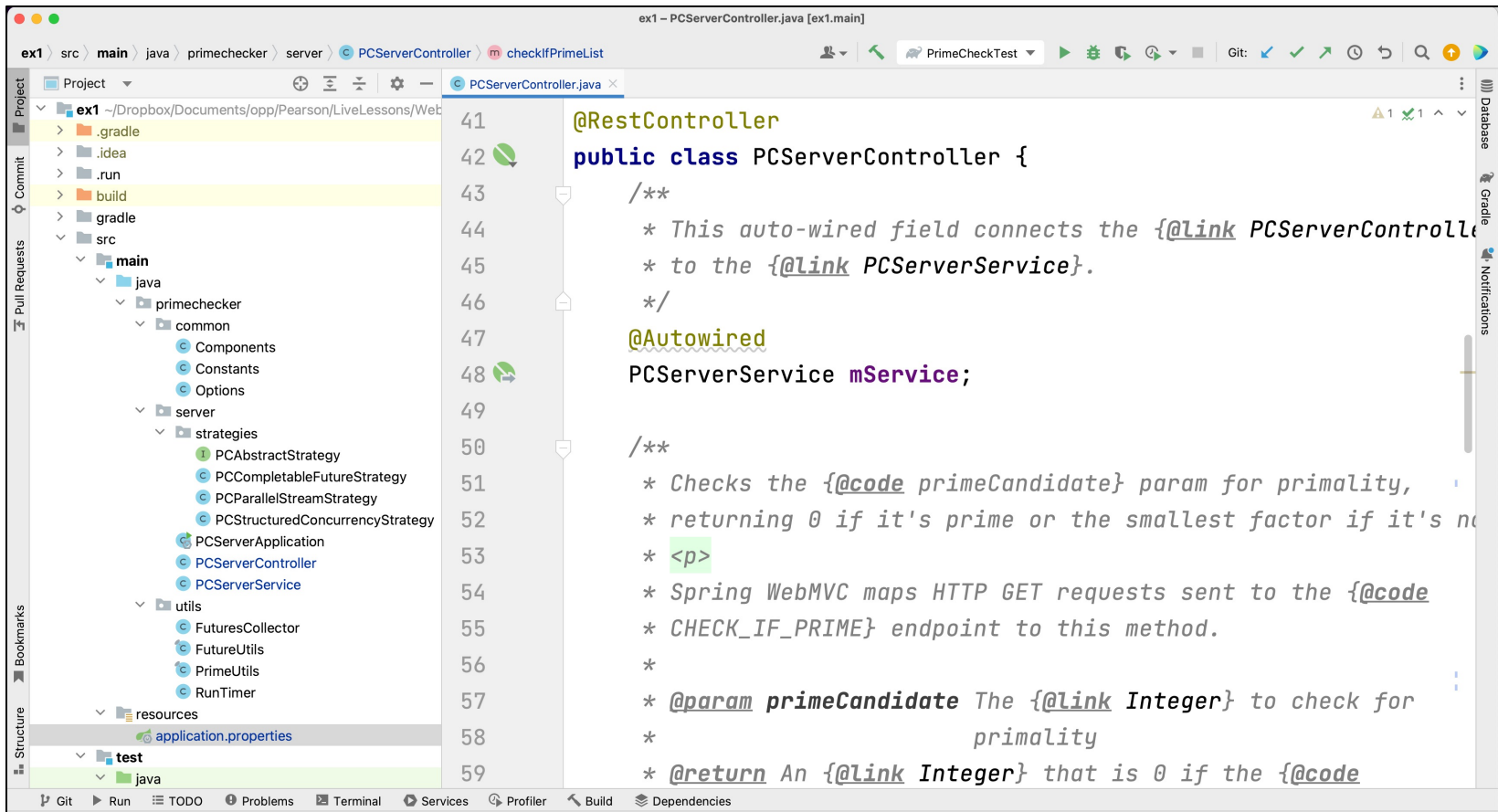
- Understand the implementation of the PCServerController & PCServerService classes & strategies that run in the PrimeCheckApplication microservice



The focus is on the Java parallel streams & structured concurrency strategies

Implementing the PrimeCheck App Server

Implementing the PrimeCheck App Server



```
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

End of the PrimeCheck App Case Study: Implementing Server Components (Part 1)