

The PrimeCheck App Case Study: Implementing Server Components

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

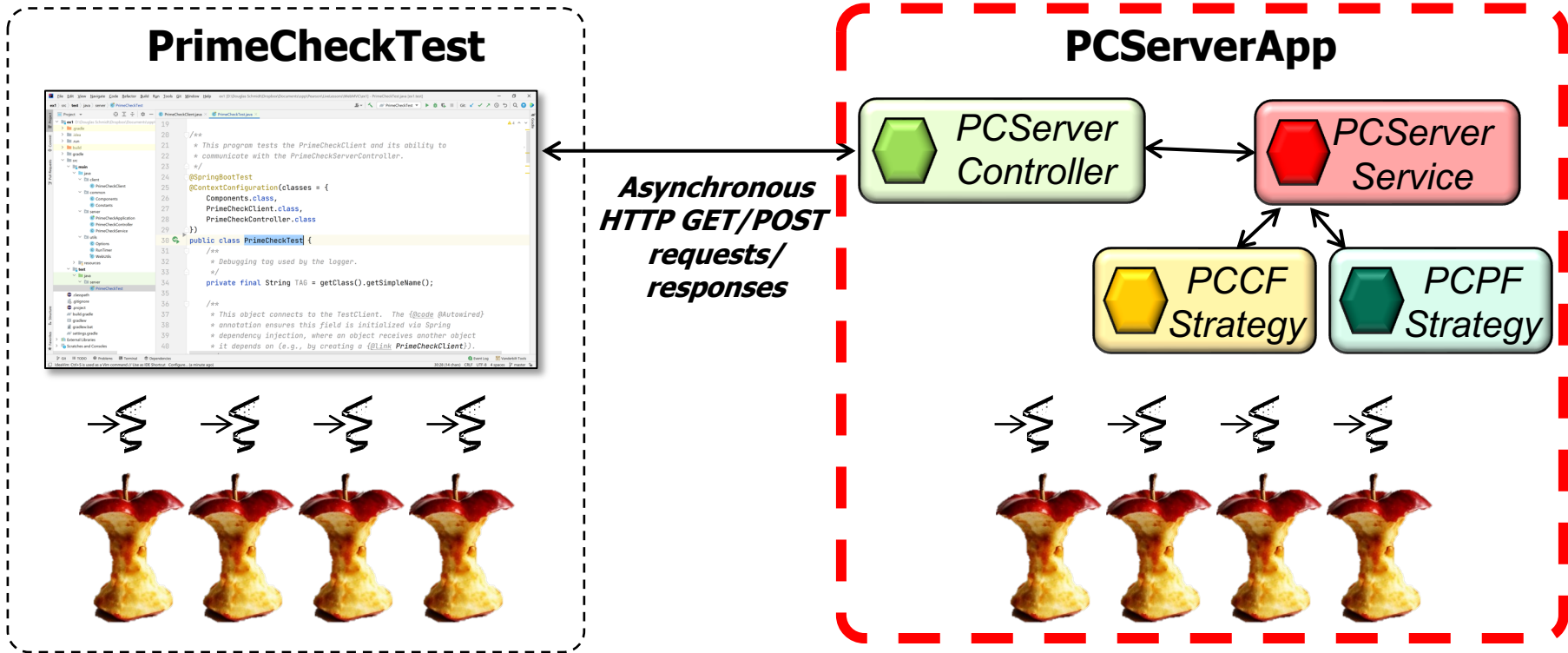
**Institute for Software
Integrated Systems**

**Vanderbilt University
Nashville, Tennessee, USA**



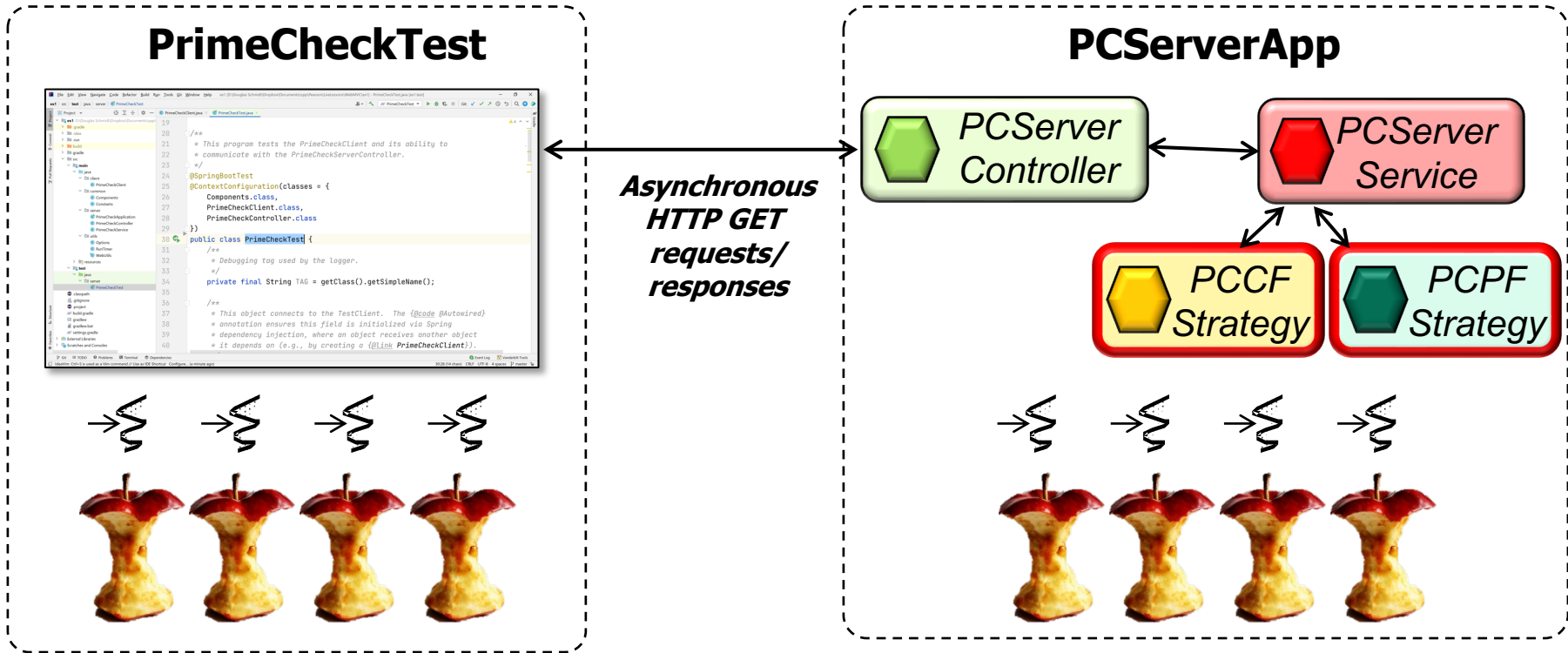
Learning Objectives in this Part of the Lesson

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



Learning Objectives in this Part of the Lesson

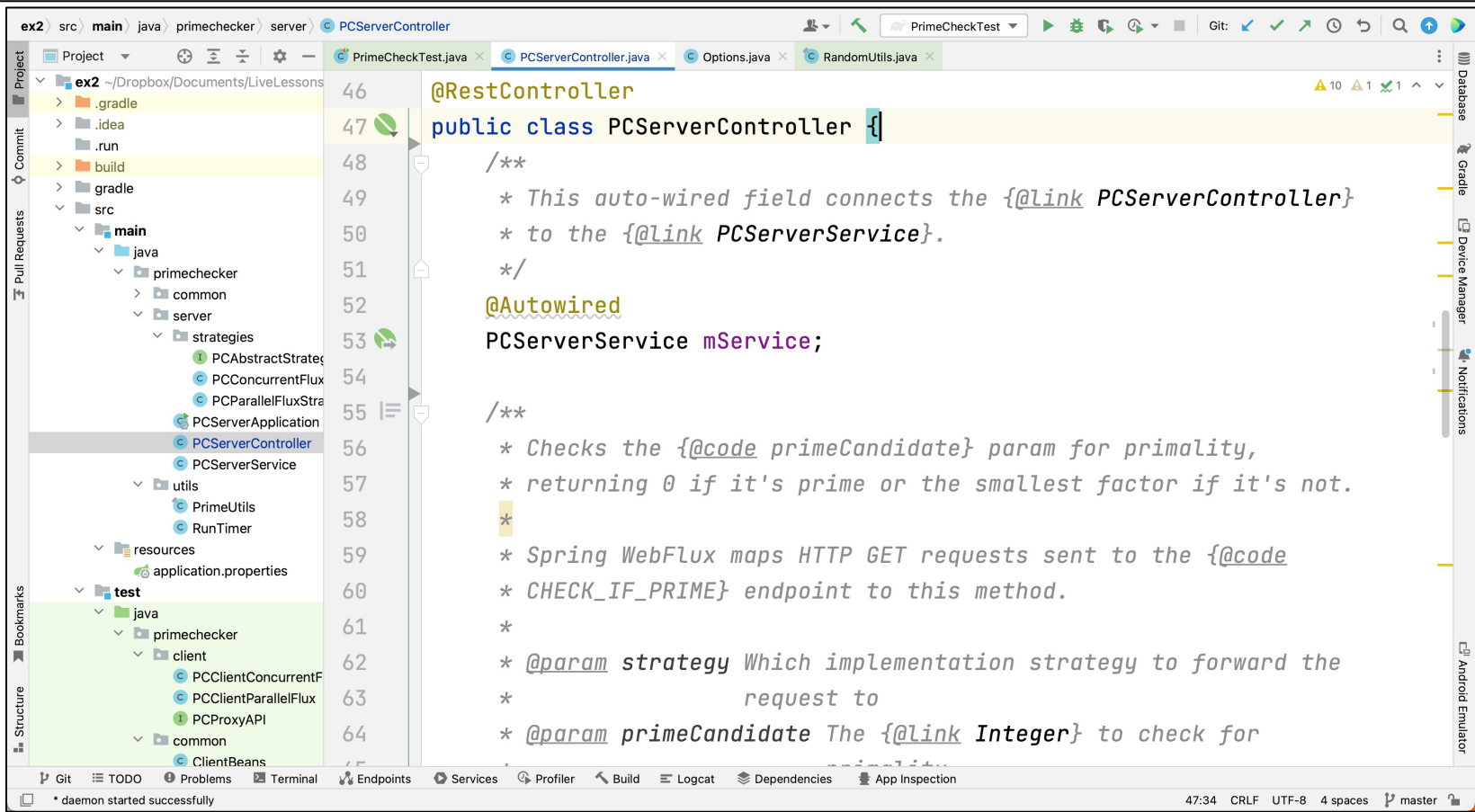
- Understand the implementation of the PCServerController & PCServerService classes & strategies that run in the PrimeCheckApplication microservice



The focus is on the Project Reactor concurrent & parallel strategies

Implementing the PrimeCheck App Server

Implementing the PrimeCheck App Server



```
46 @RestController
47 public class PCServerController {
48     /**
49      * This auto-wired field connects the {@link PCServerController}
50      * to the {@link PCServerService}.
51      */
52     @Autowired
53     PCServerService mService;
54
55     /**
56      * Checks the {@code primeCandidate} param for primality,
57      * returning 0 if it's prime or the smallest factor if it's not.
58      */
59     * Spring WebFlux maps HTTP GET requests sent to the {@code
60     * CHECK_IF_PRIME} endpoint to this method.
61     *
62     * @param strategy Which implementation strategy to forward the
63     * request to
64     * @param primeCandidate The {@link Integer} to check for
```

* daemon started successfully

47:34 CRLF UTF-8 4 spaces master

See github.com/douglasraigschmidt/LiveLessons/tree/master/WebFlux/ex2

End of the PrimeCheck App Case Study: Implementing Server Components