The PrimeCheck App Case Study: Implementing Server Components

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

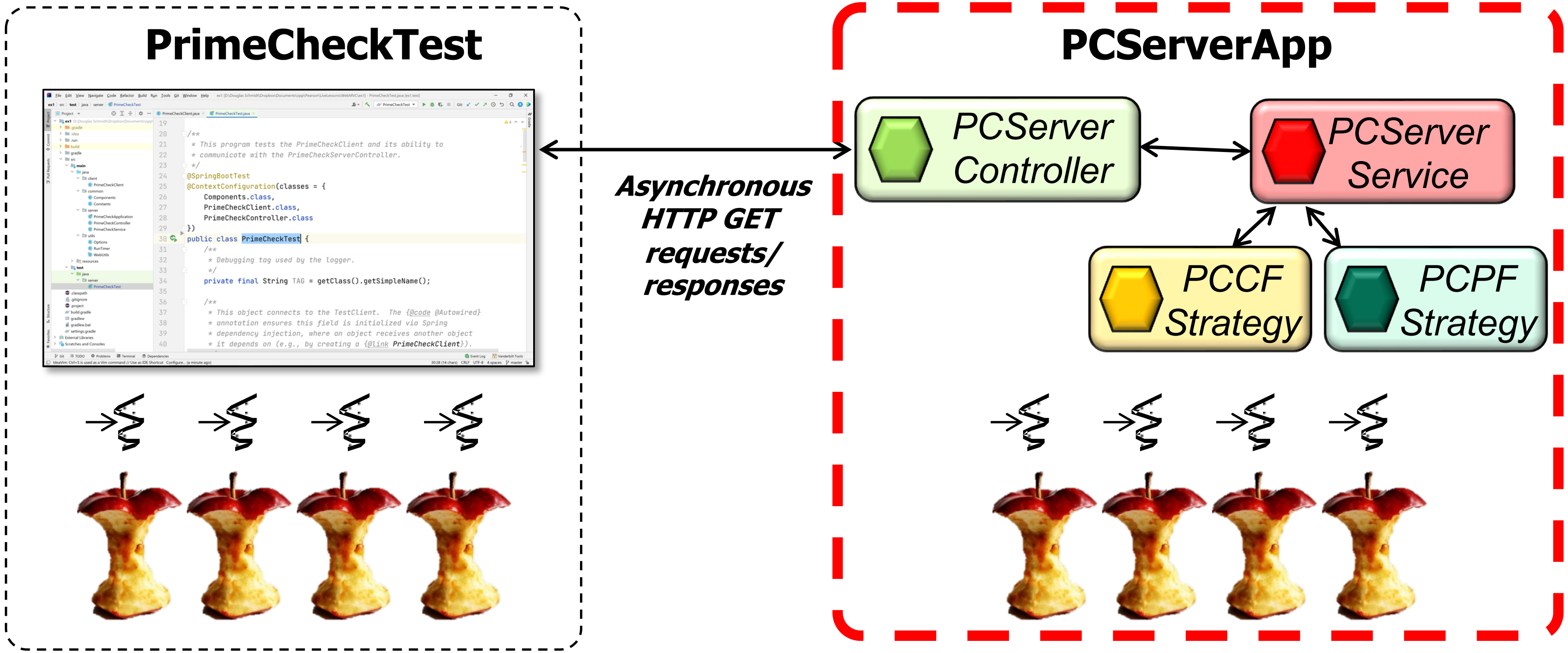
Professor of Computer Science

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

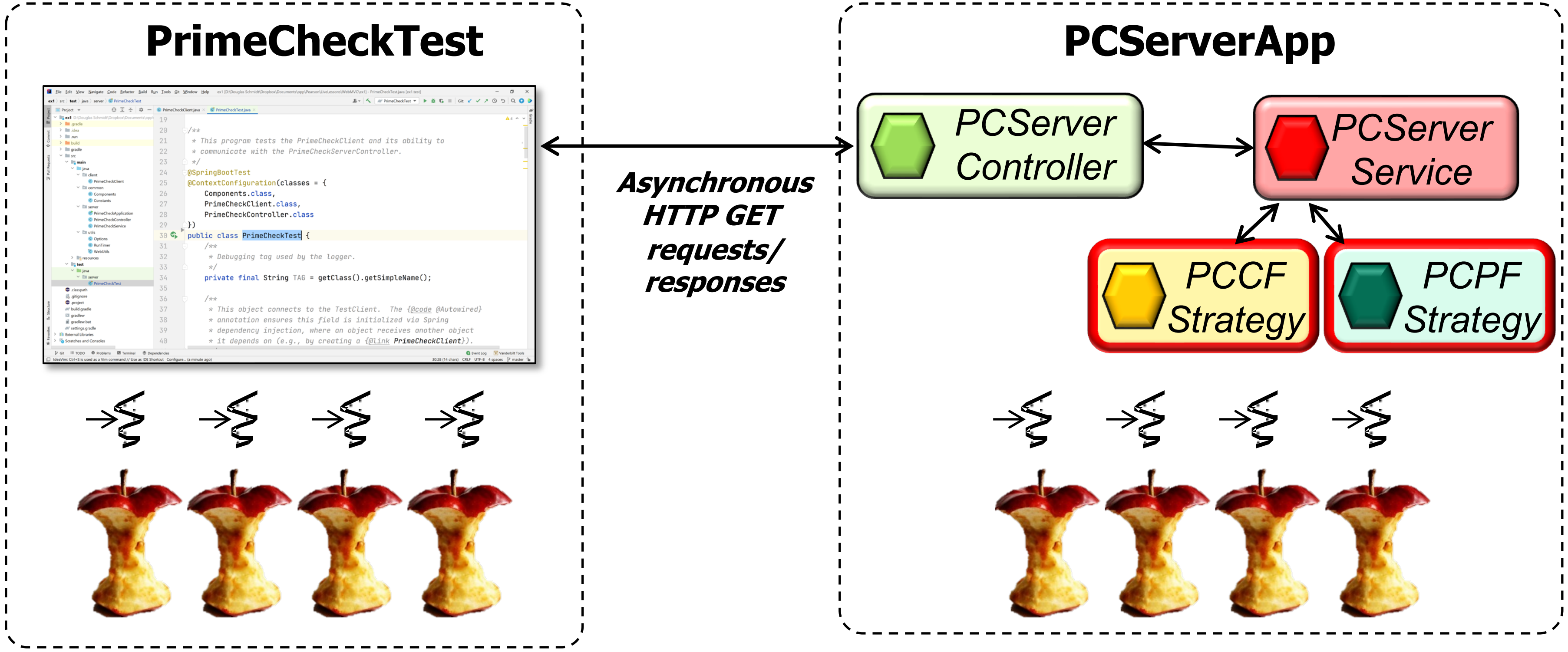


```
/**
 * This program tests the PrimeCheckClient and its ability to
 * communicate with the PrimeCheckServerController.
 */
@SpringBootTest
@ContextConfiguration(classes = {
    Components.class,
    PrimeCheckClient.class,
    PrimeCheckController.class
})
public class PrimeCheckTest {
    /**
     * Debugging tag used by the logger.
     */
    private final String TAG = getClass().getSimpleName();

    /**
     * This object connects to the TestClient.  The {@code @Autowired}
     * annotation ensures this field is initialized via Spring
     * dependency injection, where an object receives another object
     * it depends on (e.g., by creating a {@link PrimeCheckClient}).
```

Learning Objectives in this Part of the Lesson

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



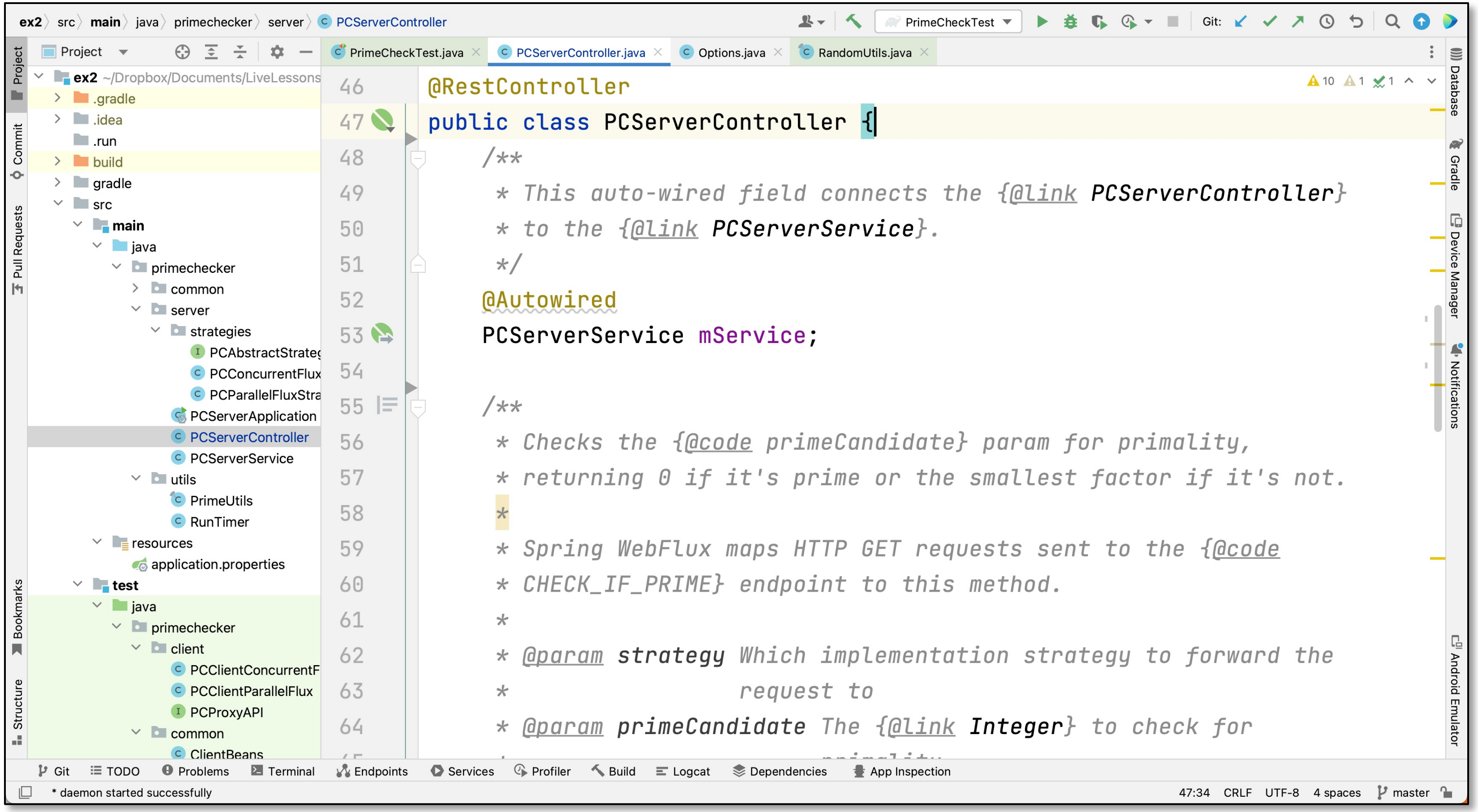
```
/**
 * This program tests the PrimeCheckClient and its ability to
 * communicate with the PrimeCheckServerController.
 */
@SpringBootTest
@ContextConfiguration(classes = {
    Components.class,
    PrimeCheckClient.class,
    PrimeCheckController.class
})
public class PrimeCheckTest {
    /**
     * Debugging tag used by the logger.
     */
    private final String TAG = getClass().getSimpleName();

    /**
     * This object connects to the TestClient.  The {@code @Autowired}
     * annotation ensures this field is initialized via Spring
     * dependency injection, where an object receives another object
     * it depends on (e.g., by creating a {@link PrimeCheckClient}).
```

The focus is on the Project Reactor concurrent & parallel strategies

Implementing the PrimeCheck App Server

Implementing the PrimeCheck App Server



```java
@RestController
public class PCServerController {
    /**
     * This auto-wired field connects the {@link PCServerController}
     * to the {@link PCServerService}.
     */
    @Autowired
    PCServerService mService;

    /**
     * Checks the {@code primeCandidate} param for primality,
     * returning 0 if it's prime or the smallest factor if it's not.
     *
     * Spring WebFlux maps HTTP GET requests sent to the {@code
     * CHECK_IF_PRIME} endpoint to this method.
     *
     * @param strategy Which implementation strategy to forward the
     *                 request to
     * @param primeCandidate The {@link Integer} to check for
```

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

End of the PrimeCheck App Case Study: Implementing Server Components