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

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 that run in the PrimeCheckApplication microservice
Learning Objectives in this Part of the Lesson

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

The focus is on several Java completable future concurrency 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.
     * <p>
     * Spring WebMVC maps HTTP GET requests sent to the {@code
     * CHECK_IF_PRIME} endpoint to this method.
     * <p>
     * @param primeCandidate The {@link Integer} to check for
     * primality
     * @return An {@link Integer} that is 0 if the
     * ```
```

See [github.com/douglascraigschmidt/LiveLessons/tree/master/WebMVC/ex1](github.com/douglascraigschmidt/LiveLessons/tree/master/WebMVC/ex1)
End of the PrimeCheck App Case Study: Implementing Server Components (Part 2)