The PrimeCheck App Case Study: Overview

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

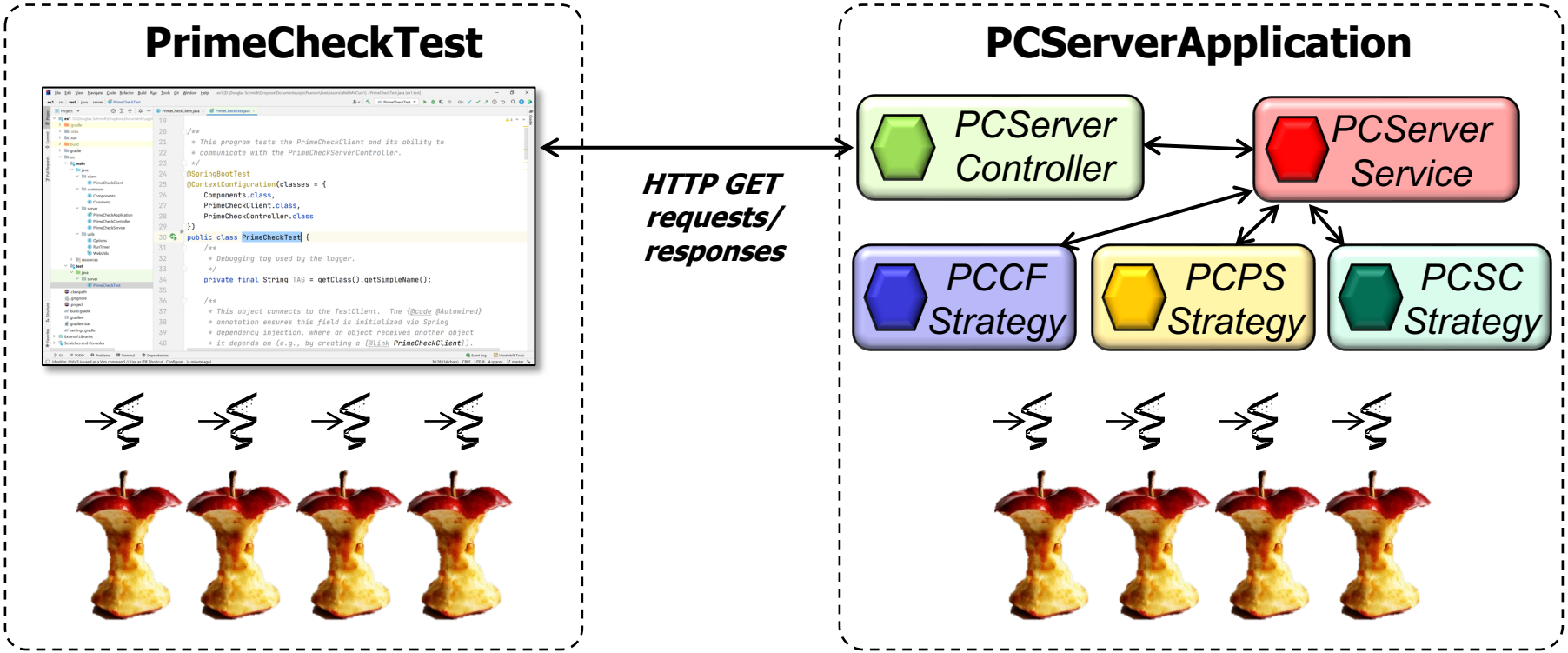
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand how functional programming & various Java frameworks are applied in a case study using Spring WebMVC to check primality of large integers

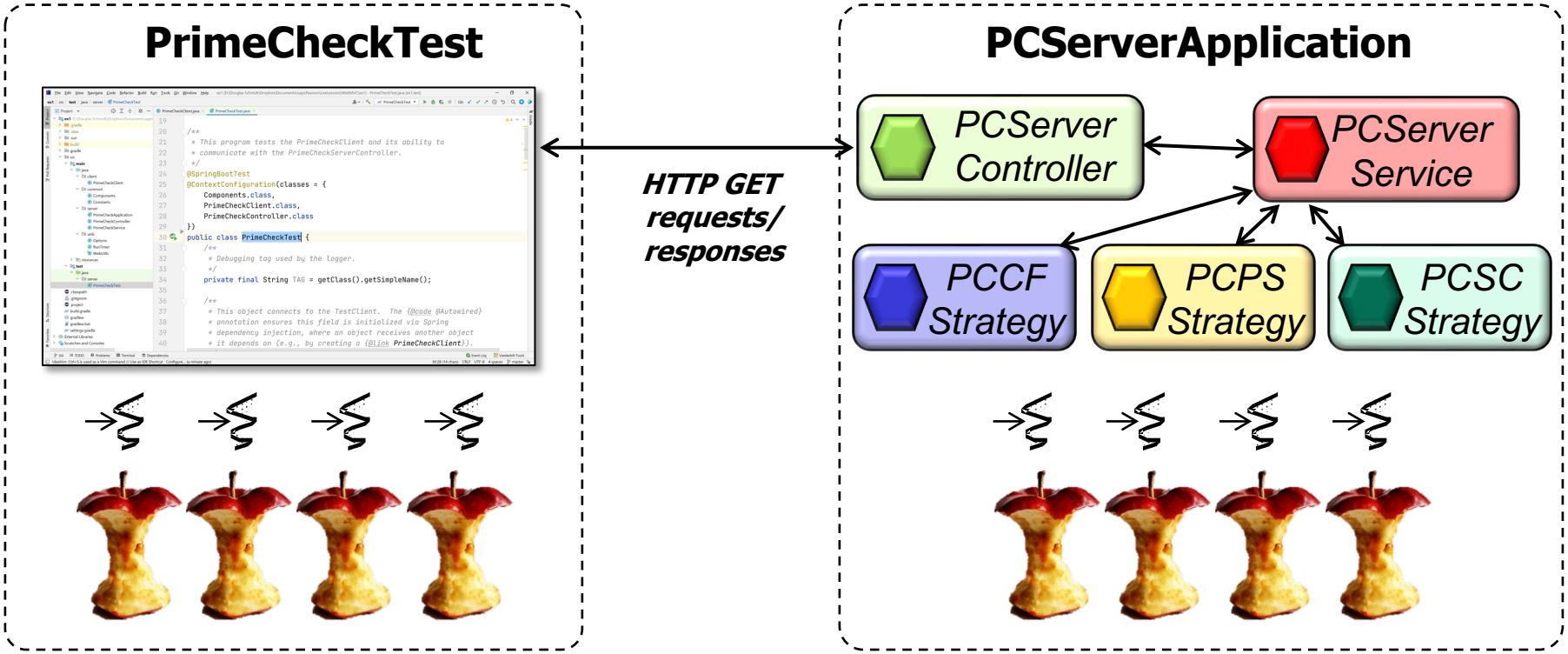


See github.com/douglascraigschmidt/LiveLessons/tree/master/WebMVC/ex1

Overview of the Prime Check App Case Study

Overview of the PrimeCheck App Case Study

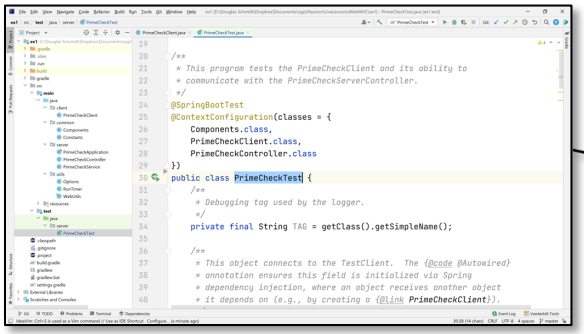
- This case study shows how Spring WebMVC can be used to send & receive HTTP GET requests to/from sequential and/or parallel clients & servers



Overview of the PrimeCheck App Case Study

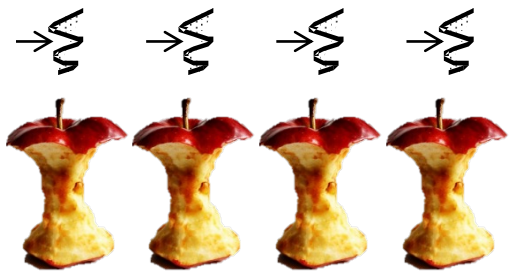
- This case study shows how Spring WebMVC can be used to send & receive HTTP GET requests to/from sequential and/or parallel clients & servers

PrimeCheckTest



```
/**
 * 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}).
```

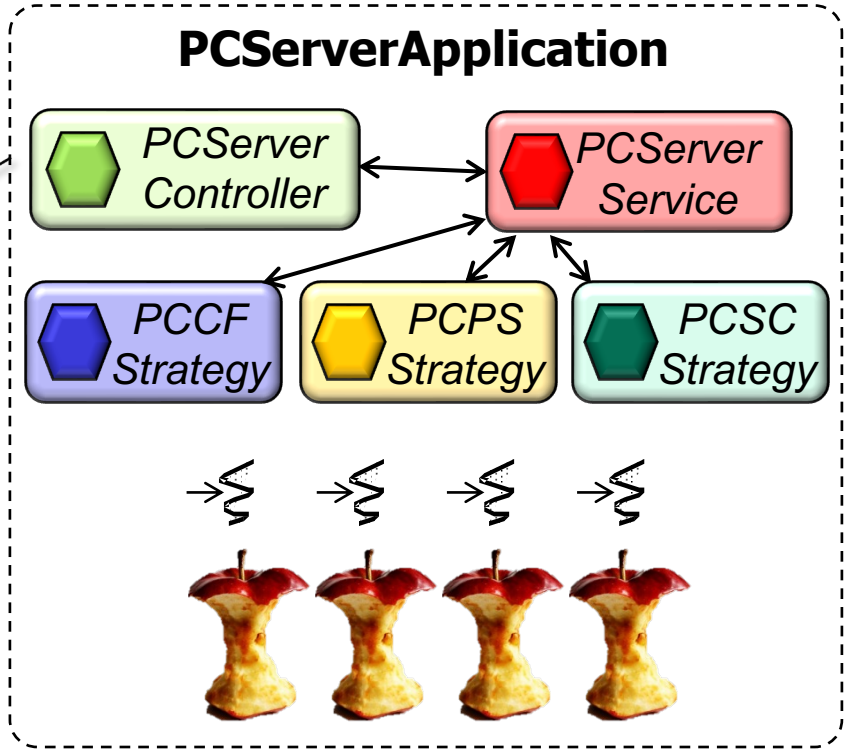


The client can send requests individually or in bulk, as well as sequentially or in parallel using Java Streams

See WebMVC/ex1/src/test/java/primechecker/client

Overview of the PrimeCheck App Case Study

- This case study shows how Spring WebMVC can be used to send & receive HTTP GET requests to/from sequential and/or parallel clients & servers



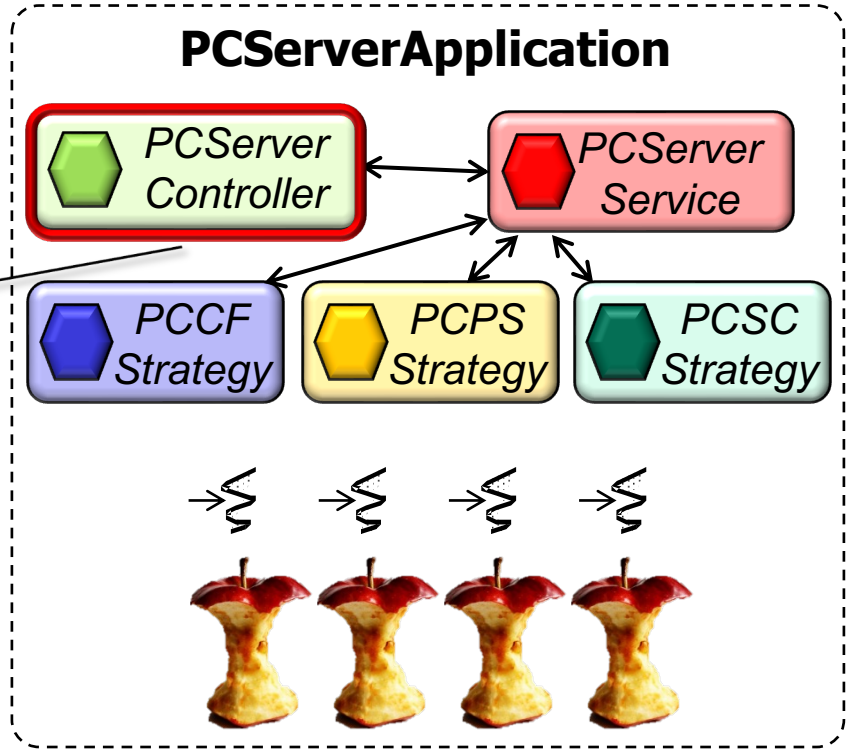
The server can receive requests individually or in bulk, as well as process the requests sequentially or in parallel using various Java frameworks

See WebMVC/ex1/src/main/java/server

Overview of the PrimeCheck App Case Study

- This case study shows how Spring WebMVC can be used to send & receive HTTP GET requests to/from sequential and/or parallel clients & servers

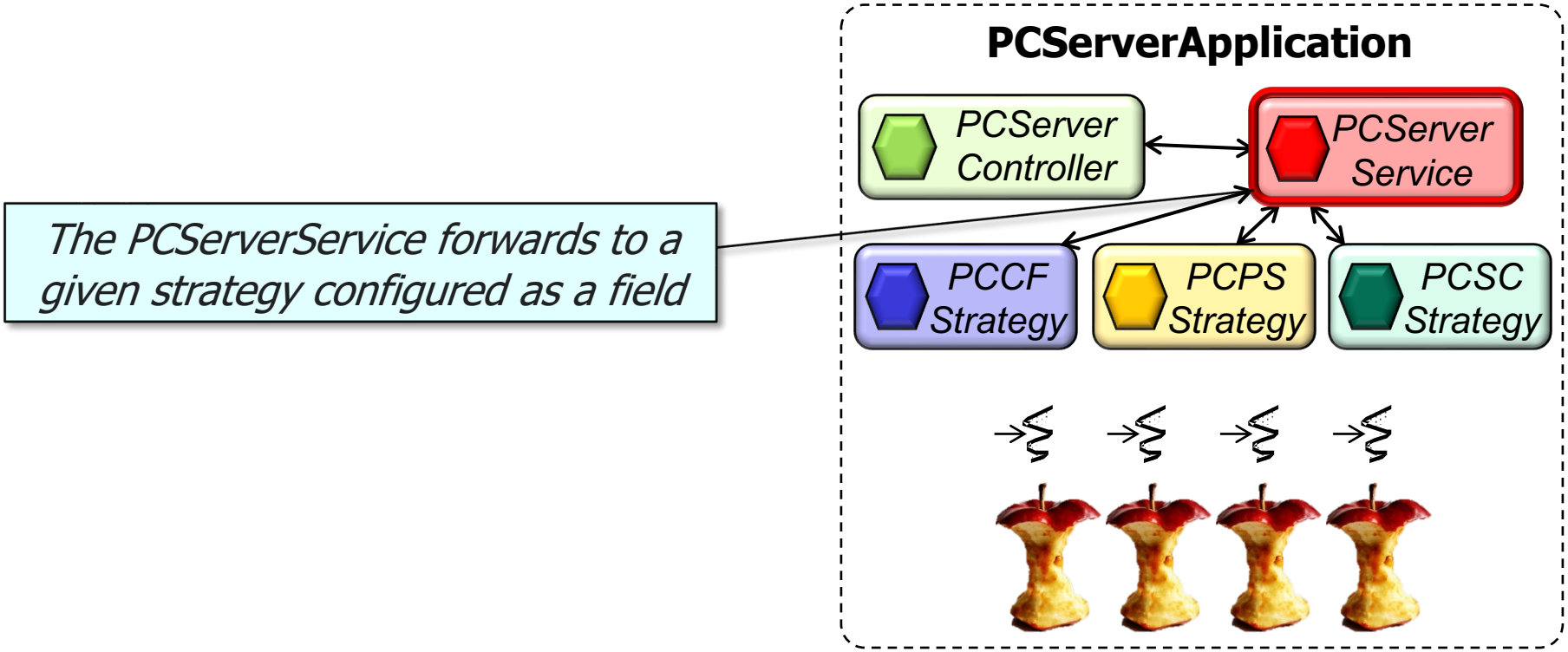
The PCServerController automatically converts HTTP GET requests into Java types & forwards them to the PCServerService for processing



See WebMVC/ex1/src/main/java/primechecker/server/PCServerController.java

Overview of the PrimeCheck App Case Study

- This case study shows how Spring WebMVC can be used to send & receive HTTP GET requests to/from sequential and/or parallel clients & servers

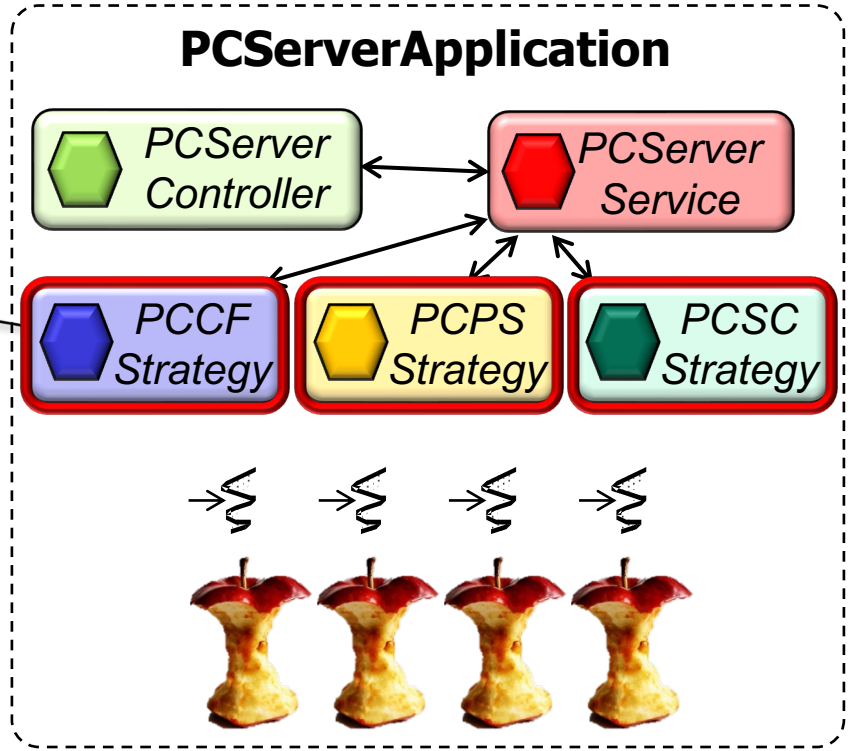


See WebMVC/ex1/src/main/java/primechecker/server/PCServerService.java

Overview of the PrimeCheck App Case Study

- This case study shows how Spring WebMVC can be used to send & receive HTTP GET requests to/from sequential and/or parallel clients & servers

The given strategy then checks the primality of Integers passed to it from the controller & service using a Java concurrency/parallelism framework

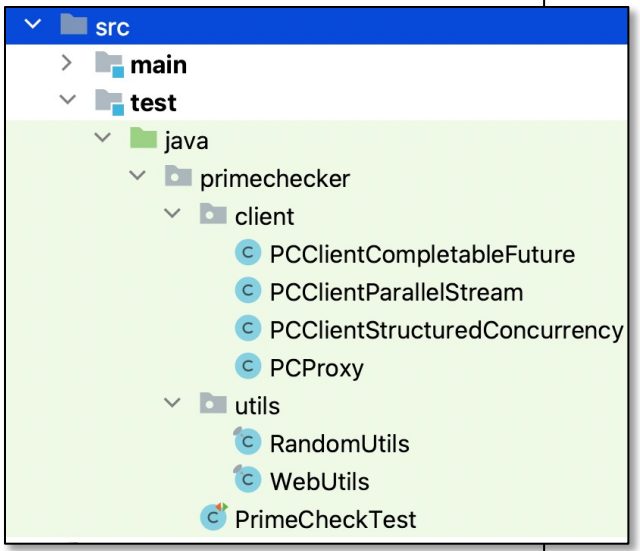


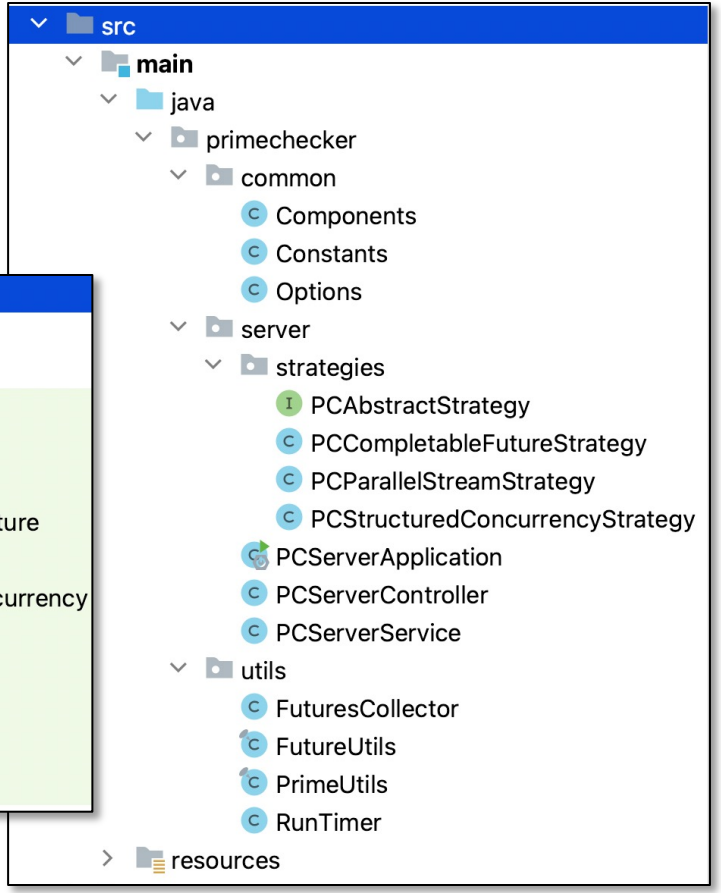
See WebMVC/ex1/src/main/java/primechecker/server/strategies

Structure of the PrimeCheck App Project

Structure of the PrimeCheck App Project

- The PrimeCheck App project source code is organized into several packages

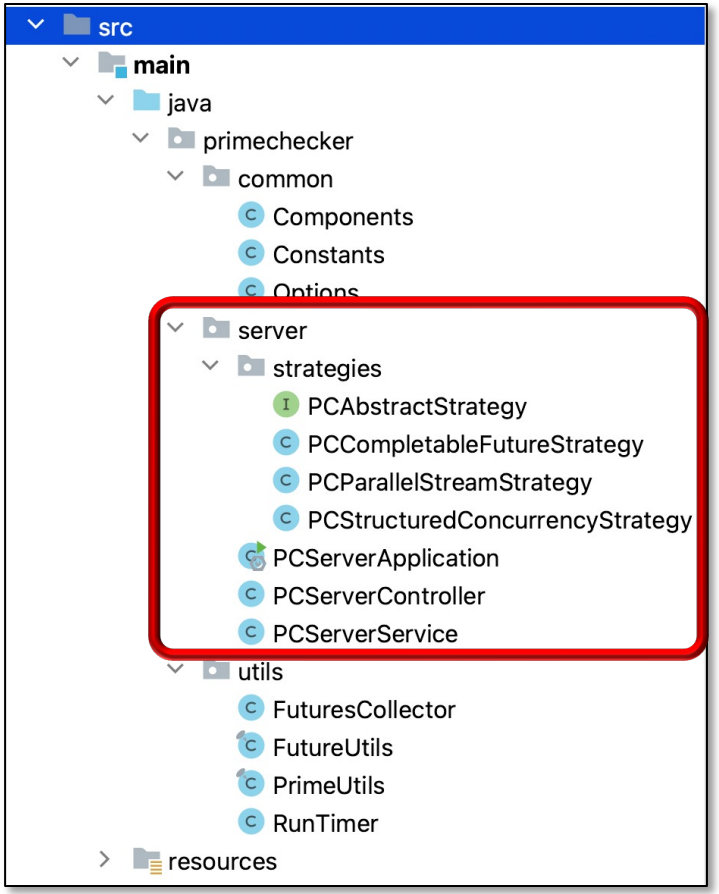




See github.com/douglascraigschmidt/LiveLessons/tree/master/WebMVC/ex1

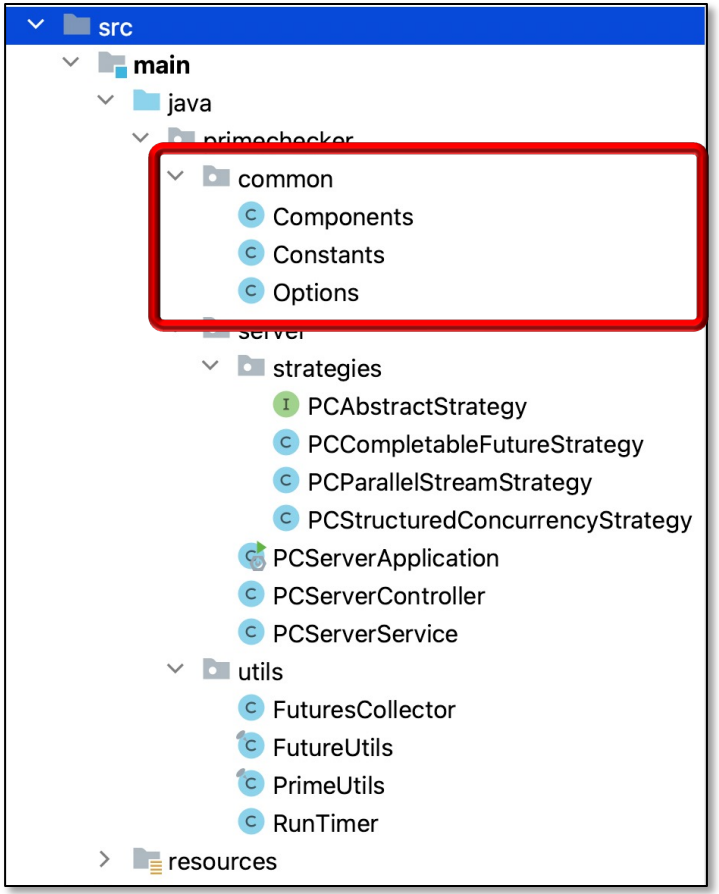
Structure of the PrimeCheck App Project

- The PrimeCheck App project source code is organized into several packages
  - main
    - server
      - Contains the "app" entry point, the controller, & the service implementation strategies



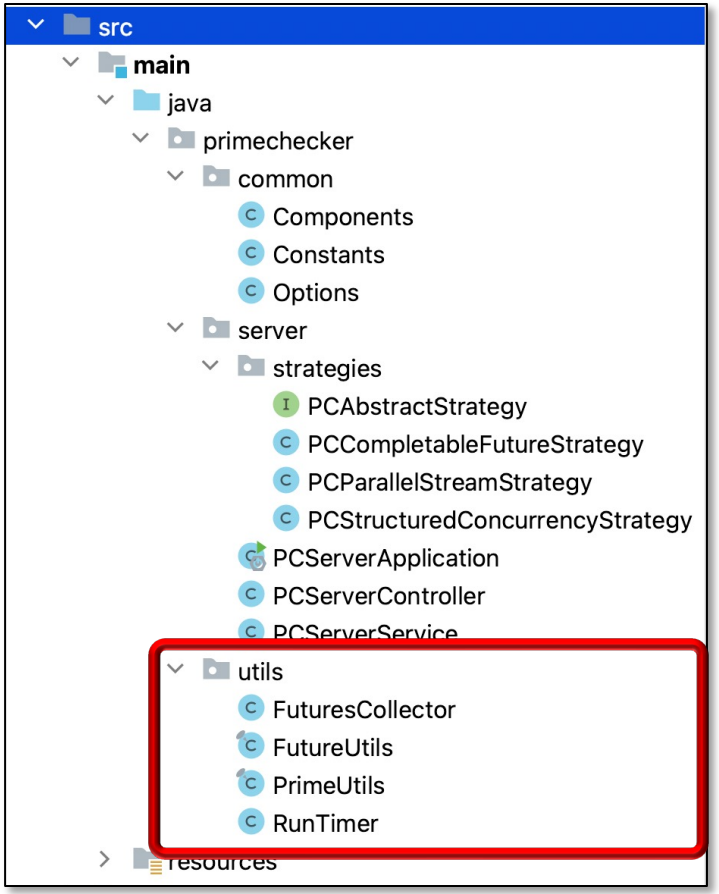
Structure of the PrimeCheck App Project

- The PrimeCheck App project source code is organized into several packages
  - main
    - server
    - common
      - Consolidates various project-specific helper classes
        - Also shows how to implement a connection pool



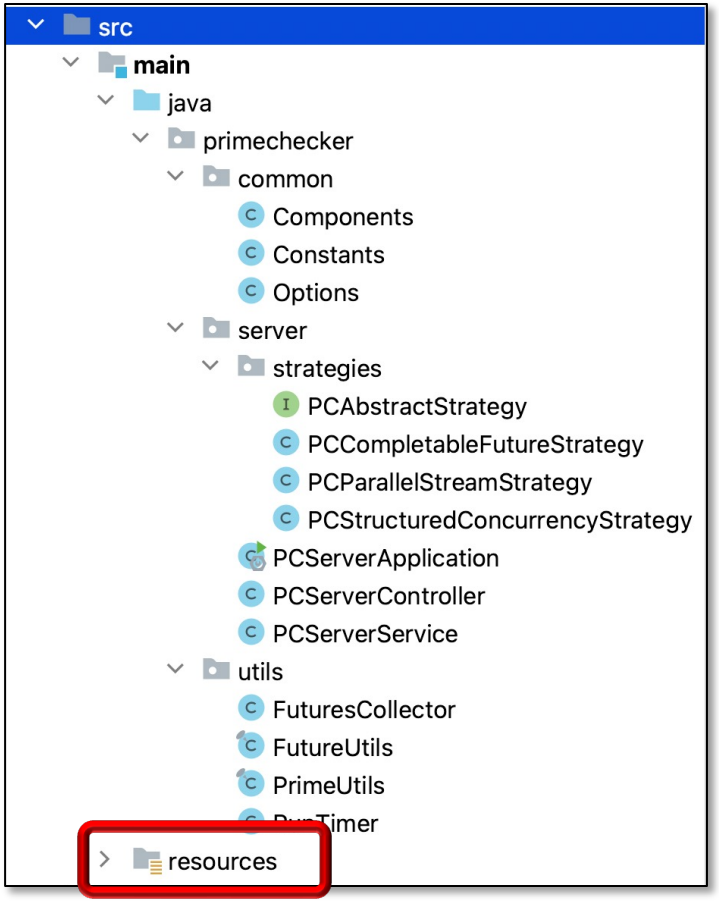
Structure of the PrimeCheck App Project

- The PrimeCheck App project source code is organized into several packages
  - main
    - server
    - common
    - utils
      - Consolidates various reusable helper classes



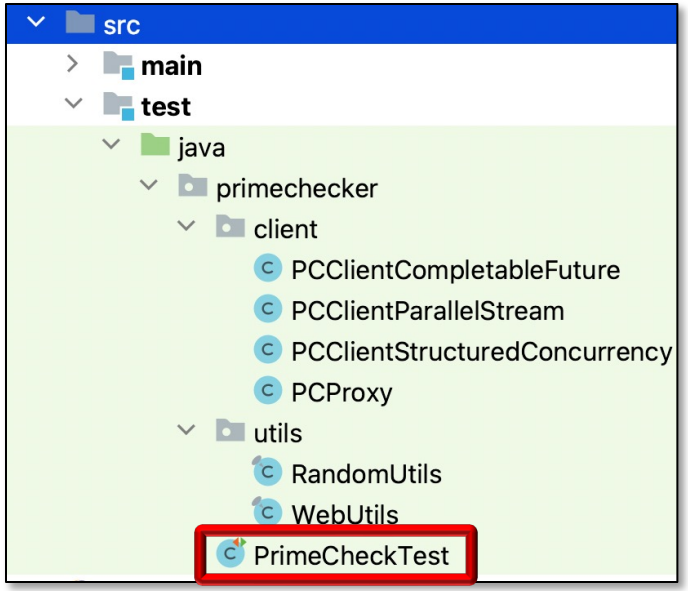
Structure of the PrimeCheck App Project

- The PrimeCheck App project source code is organized into several packages
  - main
    - server
    - common
    - utils
    - resources
      - Defines various application properties
        - e.g., name & port number



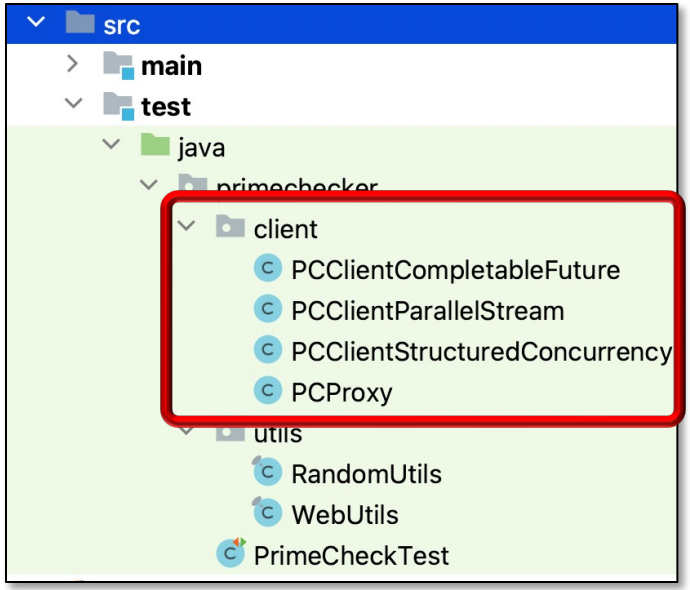
Structure of the PrimeCheck App Project

- The PrimeCheck App project source code is organized into several packages
  - test
    - PrimeCheckTest
      - This test driver measures the time taken by the client to send/receive requests/responses to/from the microservice running on the server & displays the results



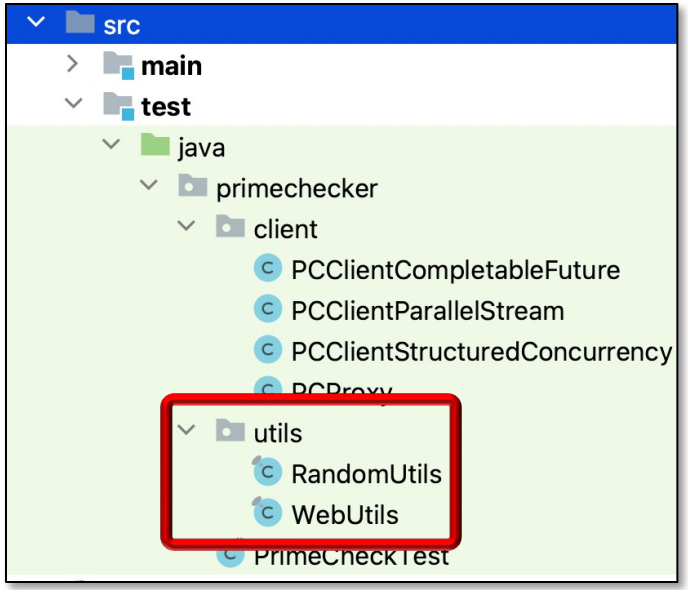
Structure of the PrimeCheck App Project

- The PrimeCheck App project source code is organized into several packages
  - test
    - PrimeCheckTest
  - client
    - Sends HTTP GET requests to the server using various Java frameworks



Structure of the PrimeCheck App Project

- The PrimeCheck App project source code is organized into several packages
  - test
    - PrimeCheckTest
    - client
    - utils
      - Consolidates various reusable helper classes



End of the PrimeCheck App Case Study: Overview