

The PrimeCheck App Case Study: Overview

Douglas C. Schmidt

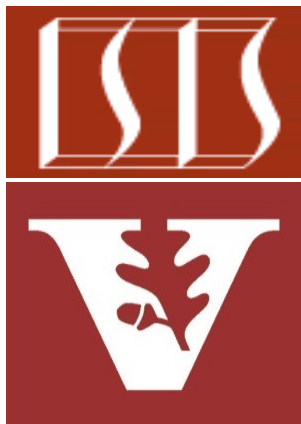
d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of ComPOSTer Science

**Institute for Software
Integrated Systems**

**Vanderbilt University
Nashville, Tennessee, USA**

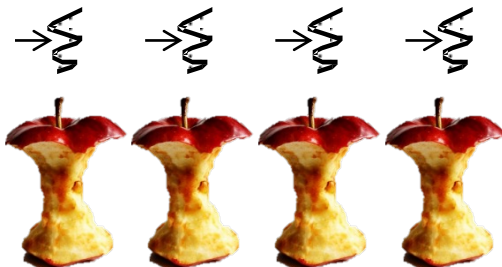


Learning Objectives in this Part of the Lesson

- Understand how various Project Reactor frameworks are applied in a case study using Spring WebFlux to check primality of large integers asynchronously

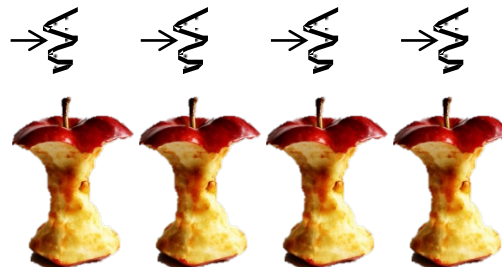
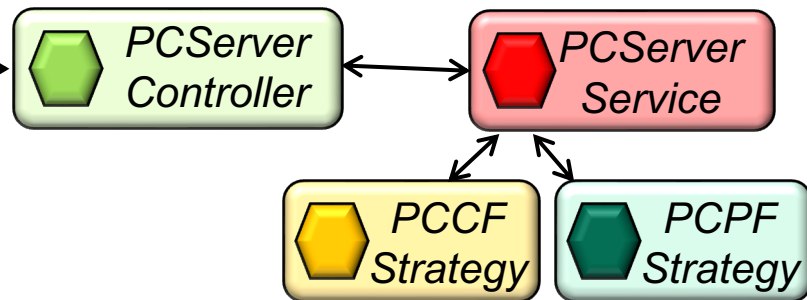
PrimeCheckTest

```
20 /**  
21  * This program tests the PrimeCheckClient and its ability to  
22  * communicate with the PrimeCheckServerController.  
23  */  
24 @SpringBootTest  
25 @ContextConfiguration(classes = {  
26     Components.class,  
27     PrimeCheckClient.class,  
28     PrimeCheckController.class  
29 })  
30 public class PrimeCheckTest {  
31     /**  
32      * Debugging tag used by the logger.  
33      */  
34     private final String TAG = getClass().getSimpleName();  
35 }  
36 /**  
37  * This object connects to the TestClient. The @code @Autowired  
38  * annotation ensures this field is initialized via Spring  
39  * dependency injection, where an object receives another object  
40  * it depends on (e.g., by creating a @Link PrimeCheckClient).
```



***Asynchronous
HTTP GET
requests/
responses***

PCServerApplication



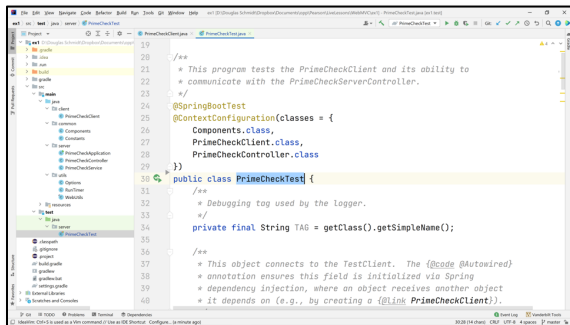
See github.com/douglasraigschmidt/LiveLessons/tree/master/WebFlux/ex2

Overview of the Prime Check App Case Study

Overview of the PrimeCheck App Case Study

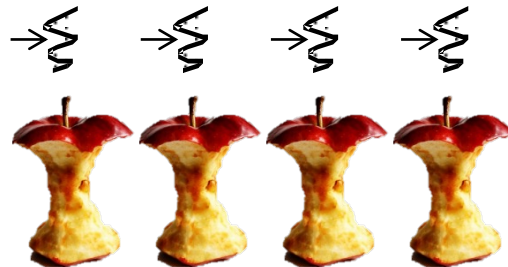
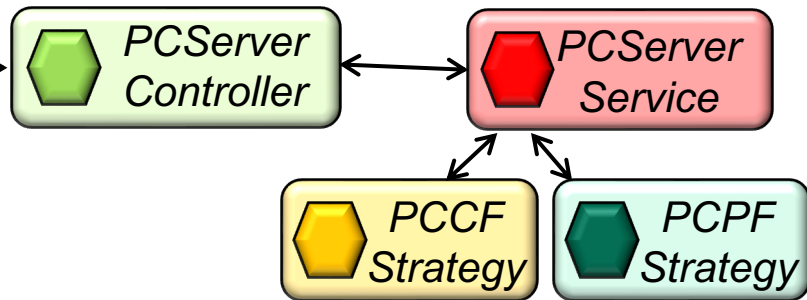
- This case study shows how Spring WebFlux can send & receive HTTP GET/POST requests to/from concurrent/parallel clients & servers asynchronously

PrimeCheckTest



*Asynchronous
HTTP GET
requests/
responses*

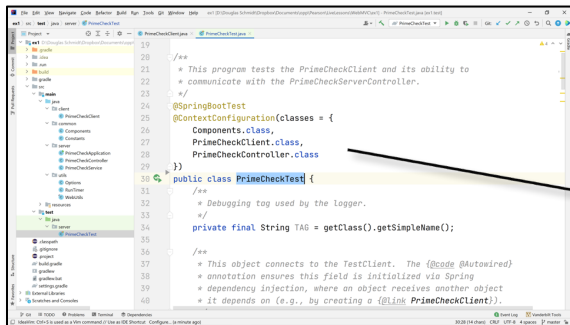
PCServerApplication



Overview of the PrimeCheck App Case Study

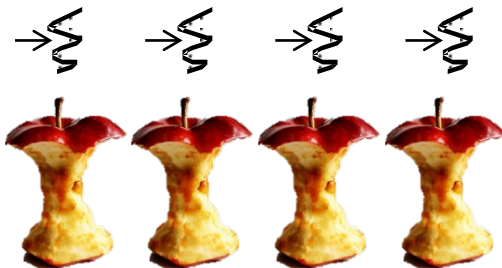
- This case study shows how Spring WebFlux can send & receive HTTP GET/POST requests to/from concurrent/parallel clients & servers asynchronously

PrimeCheckTest



```
19  /**
20   * This program tests the PrimeCheckClient and its ability to
21   * communicate with the PrimeCheckServerController.
22   */
23
24  @SpringBootTest
25  @ContextConfiguration(classes = {
26      Components.class,
27      PrimeCheckClient.class,
28      PrimeCheckController.class
29  })
30  public class PrimeCheckTest {
31
32      /**
33       * Debugging tag used by the logger.
34       */
35      private final String TAG = getClass().getSimpleName();
36
37
38      /**
39       * This object connects to the TestClient. The @Code @Autowired
40       * annotation ensures this field is initialized via Spring
41       * dependency injection, where an object receives another object
42       * it depends on (e.g., by creating a @Link PrimeCheckClient)).
43       */
44      @Autowired
45      PrimeCheckClient testClient;
```

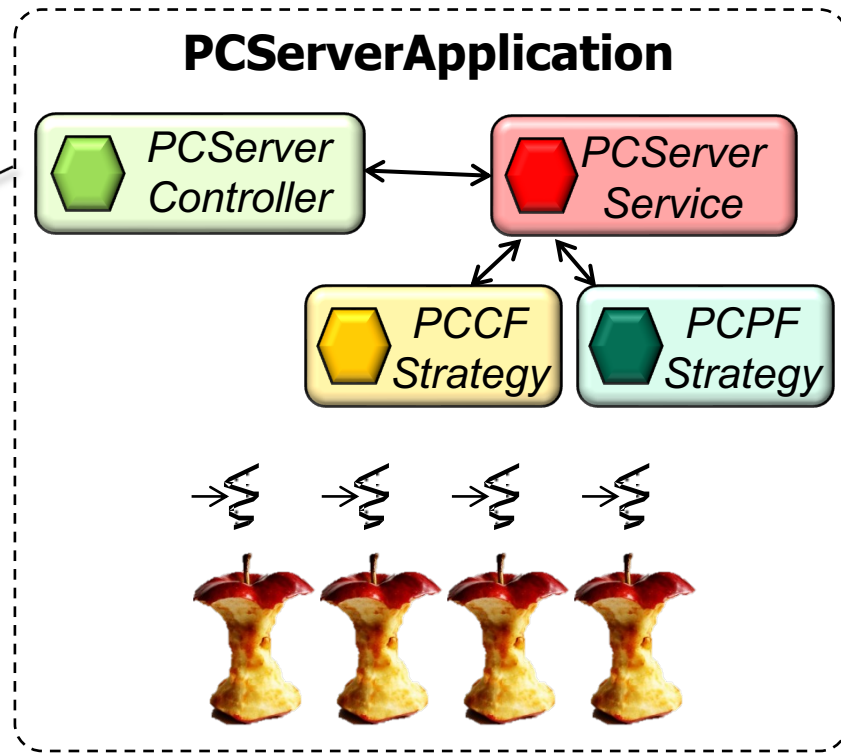
The client can send requests individually or in bulk, as well as in parallel using two Project Reactor concurrency/parallelism frameworks



See [WebFlux/ex2/src/test/java/primechecker/client](https://github.com/spring-projects/spring-framework/blob/master/spring-webflux/src/test/java/org/springframework/webflux/test/primechecker/client/PrimeCheckTest.java)

Overview of the PrimeCheck App Case Study

- This case study shows how Spring WebFlux can send & receive HTTP GET/POST requests to/from concurrent/parallel clients & servers asynchronously



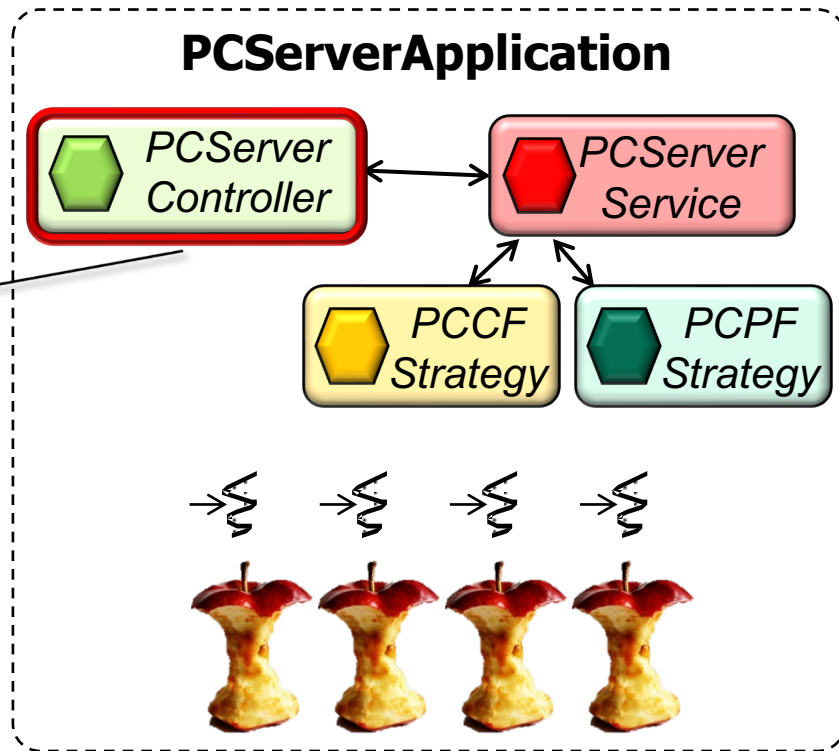
The server can receive requests individually or in bulk & process the requests in parallel using the two Project Reactor frameworks

See [WebFlux/ex2/src/main/java/server](https://github.com/spring-projects/spring-webflux/tree/master/spring-webflux-examples/src/main/java/server)

Overview of the PrimeCheck App Case Study

- This case study shows how Spring WebFlux can send & receive HTTP GET/POST requests to/from concurrent/parallel clients & servers asynchronously

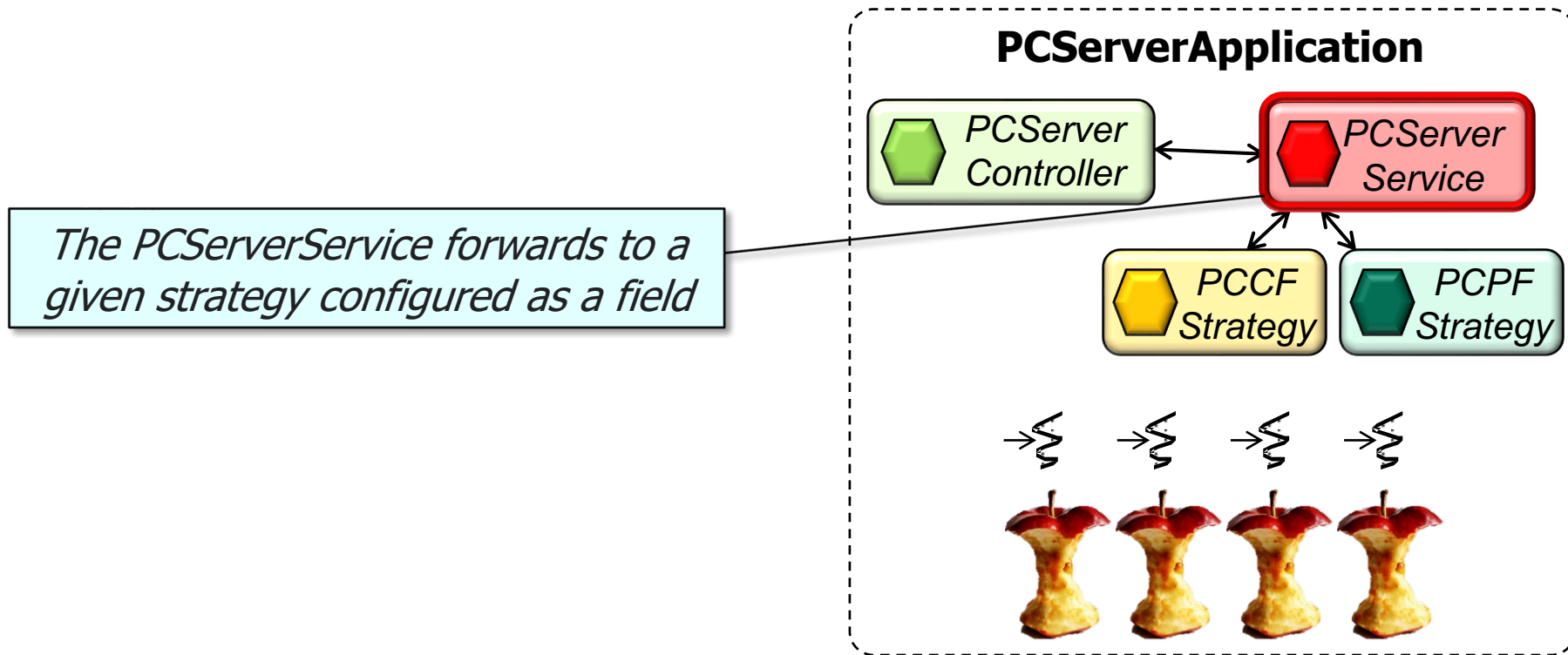
The PCServerController automatically converts HTTP GET/POST requests into Java types & forwards them to the PCServerService for processing



See [WebFlux/ex2/src/main/java/primechecker/server/PCServerController.java](https://github.com/spring-projects/spring-webflux/tree/master/spring-webflux-examples/src/main/java/primechecker/server)

Overview of the PrimeCheck App Case Study

- This case study shows how Spring WebFlux can send & receive HTTP GET/POST requests to/from concurrent/parallel clients & servers asynchronously

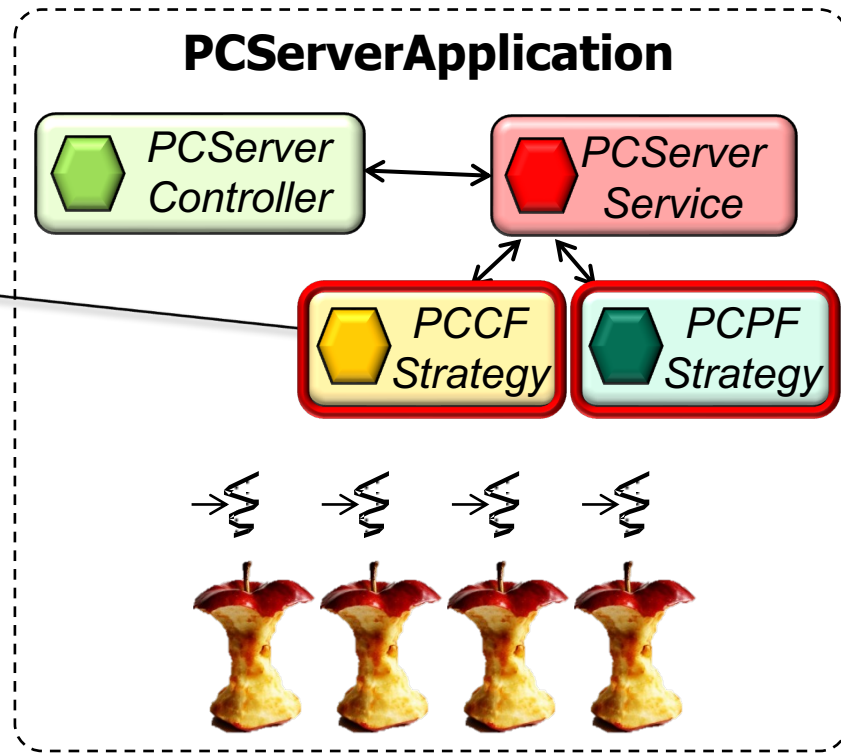


See [WebFlux/ex2/src/main/java/primechecker/server/PCServerService.java](https://github.com/spring-projects/spring-webflux/tree/master/spring-webflux-examples/src/main/java/primechecker/server/PCServerService.java)

Overview of the PrimeCheck App Case Study

- This case study shows how Spring WebFlux can send & receive HTTP GET/POST requests to/from concurrent/parallel clients & servers asynchronously

The given strategy checks the primality of Integers passed to it from the controller & service using one of the specified Project Reactor concurrency/parallelism frameworks

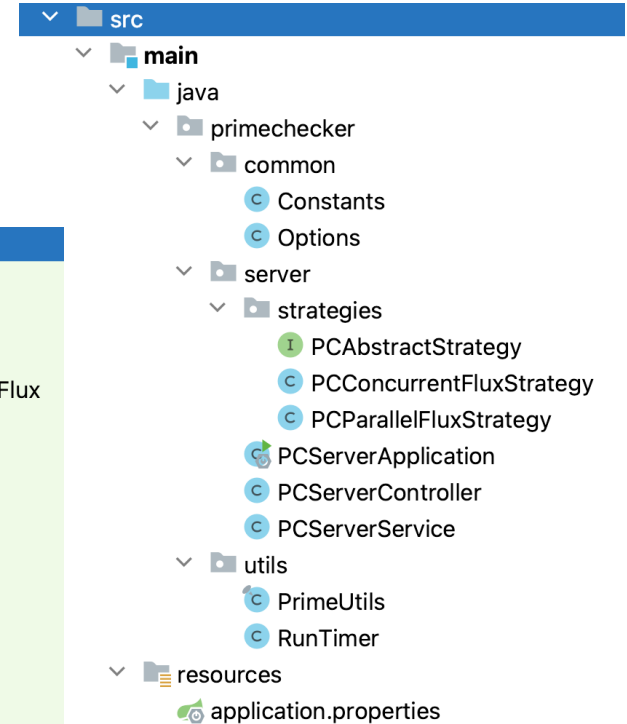
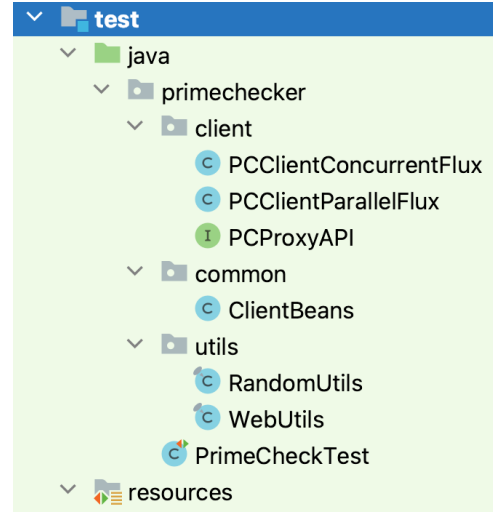


See [WebFlux/ex2/src/main/java/primechecker/server/strategies](https://github.com/spring-projects/spring-webflux/tree/master/samples/ex2/src/main/java/primechecker/server/strategies)

Structure of the PrimeCheck App Project

Structure of the PrimeCheck App Project

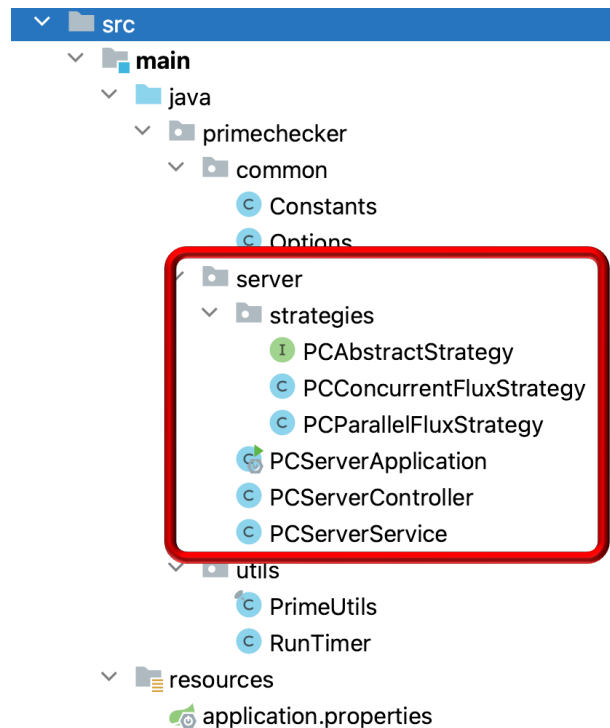
- The PrimeCheck App project source code is organized into several packages



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

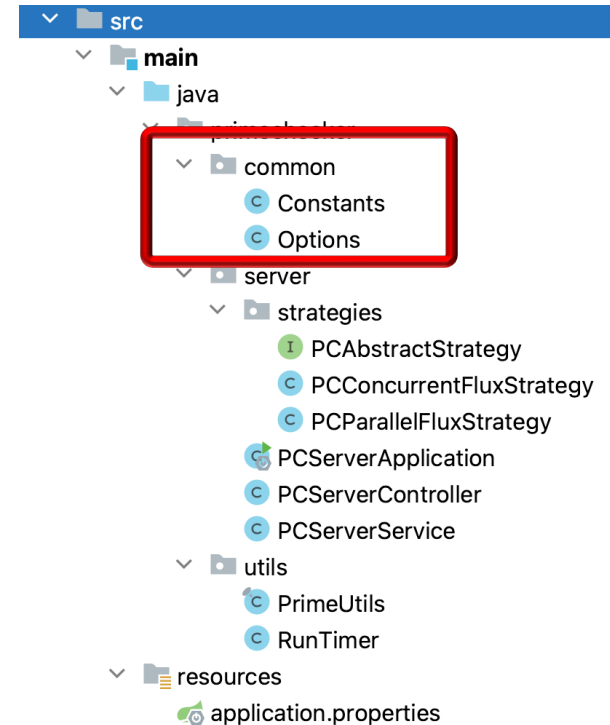
Structure of the PrimeCheck App Project

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



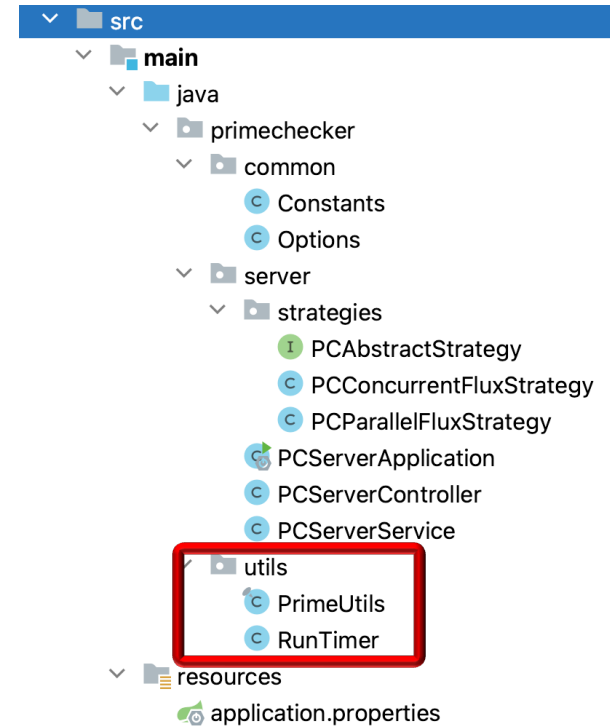
Structure of the PrimeCheck App Project

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



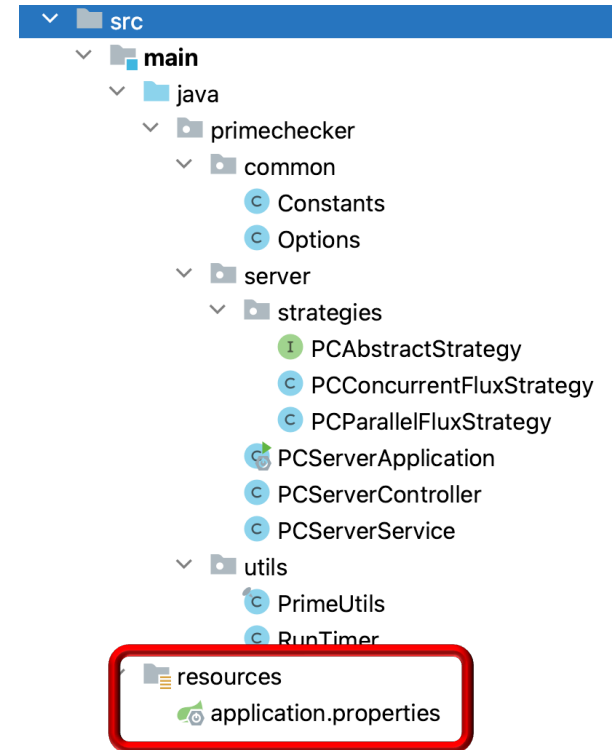
Structure of the PrimeCheck App Project

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



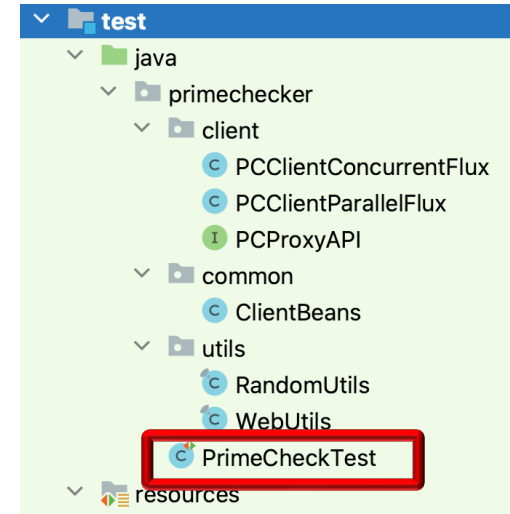
Structure of the PrimeCheck App Project

- 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



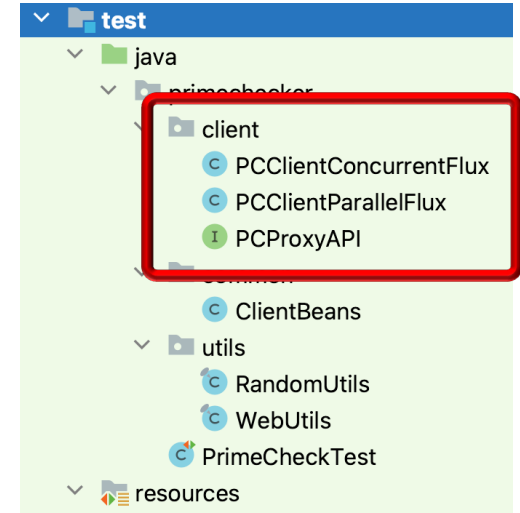
Structure of the PrimeCheck App Project

- 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 asynchronously to/from the microservice running on the server & displays the results



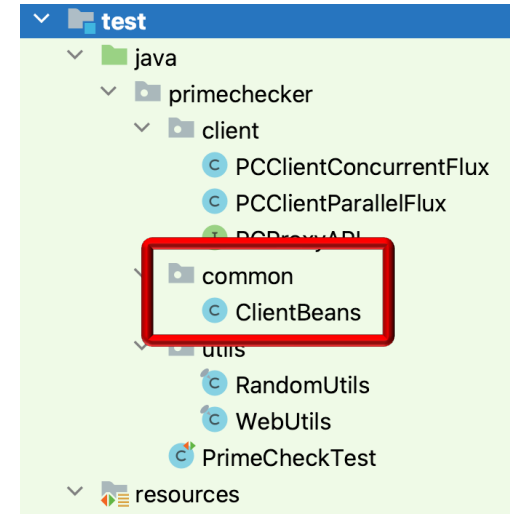
Structure of the PrimeCheck App Project

- The PrimeCheck App project source code is organized into several packages
 - test
 - PrimeCheckTest
 - client
 - Sends HTTP GET/POST requests to the server asynchronously using two Project Reactor concurrency/parallelism frameworks



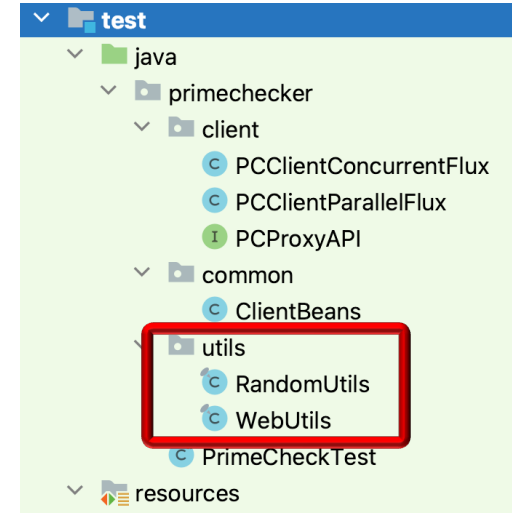
Structure of the PrimeCheck App Project

- The PrimeCheck App project source code is organized into several packages
 - test
 - PrimeCheckTest
 - client
 - PCClientConcurrentFlux
 - PCClientParallelFlux
 - PCProxyAPI
 - common
 - ClientBeans
 - utils
 - RandomUtils
 - WebUtils
 - common
 - Consolidates various project-specific reusable helper classes



Structure of the PrimeCheck App Project

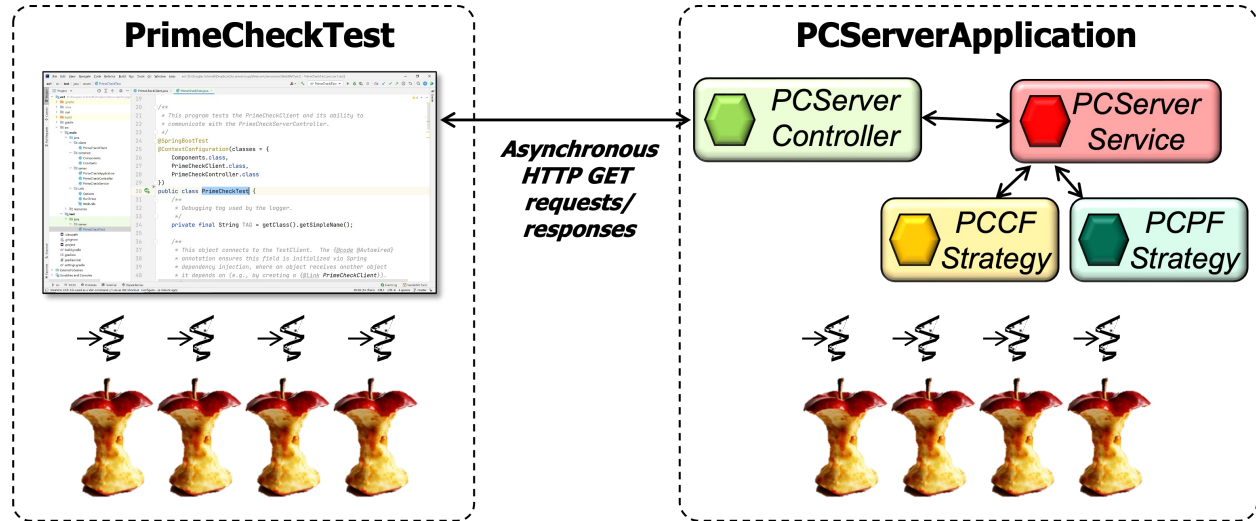
- The PrimeCheck App project source code is organized into several packages
 - test
 - PrimeCheckTest
 - client
 - common
 - utils
 - Consolidates various general-purpose reusable helper classes



Pros & Cons of the PrimeCheck App

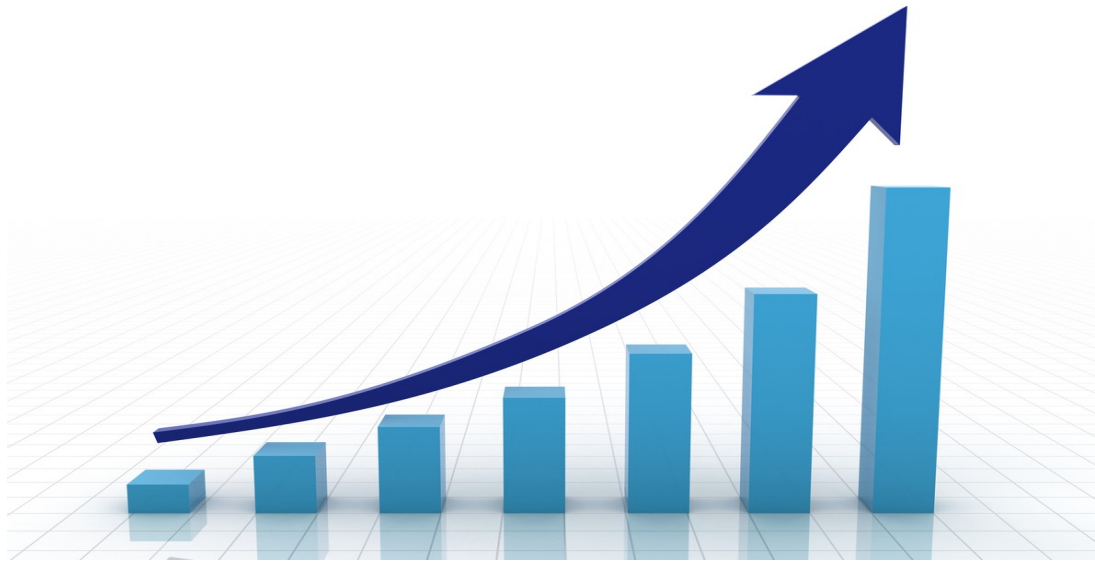
Pros & Cons of the PrimeCheck App

- Pros
 - All service implementations run in a single process, which simplifies configuration, deployment, testing, & security



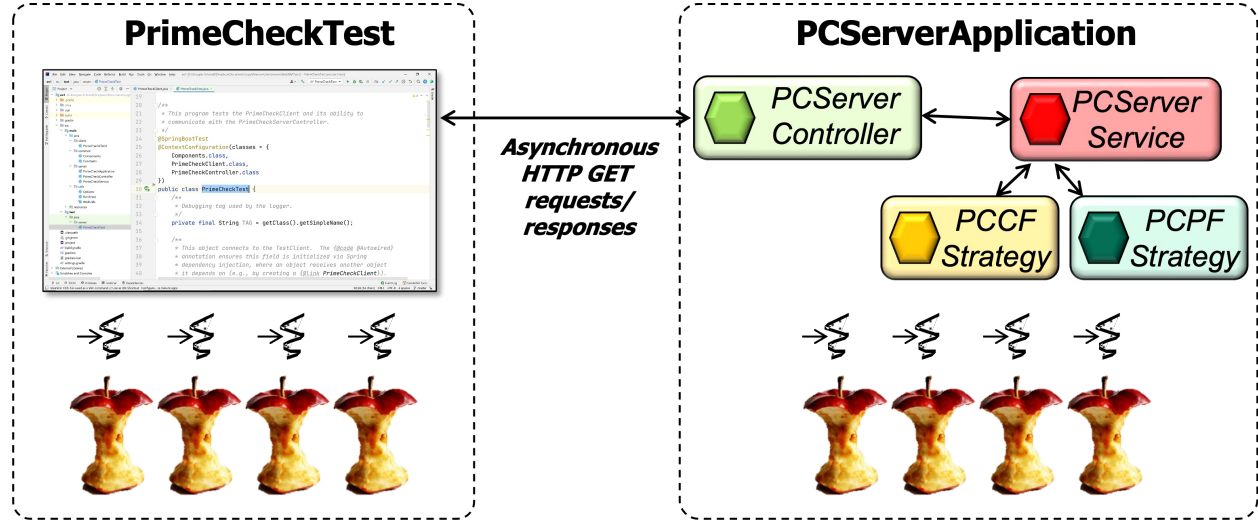
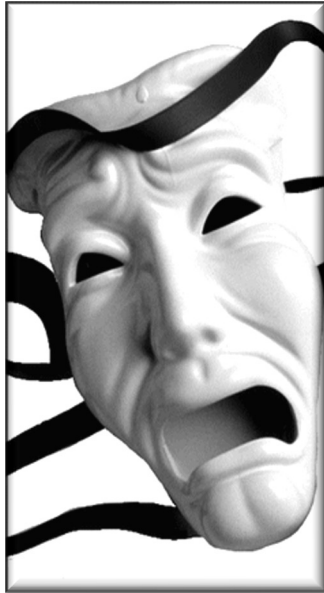
Pros & Cons of the PrimeCheck App

- Pros
 - All service implementations run in a single process, which simplifies configuration, deployment, testing, & security
 - Asynchrony may enable greater scalability



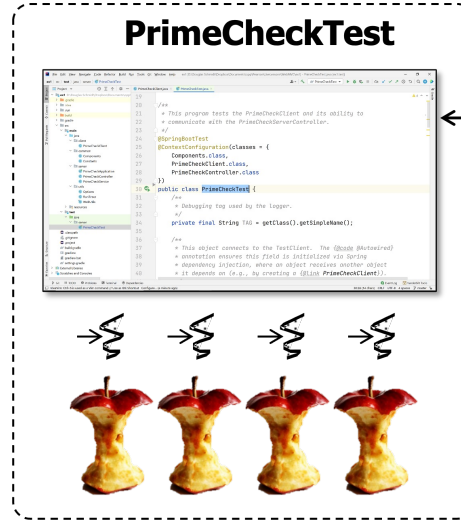
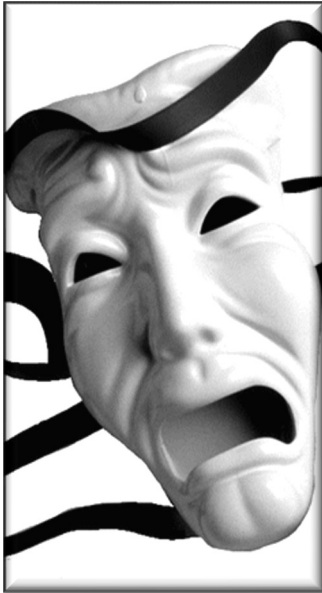
Pros & Cons of the PrimeCheck App

- Cons
 - All service implementations run in a single process, which can degrade system scalability & reliability

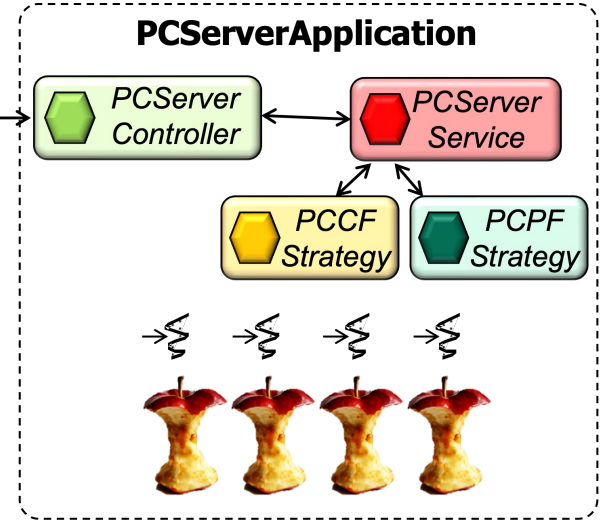


Pros & Cons of the PrimeCheck App

- Cons
 - All service implementations run in a single process, which can degrade system scalability & reliability
 - Asynchrony can be trickier to develop & debug



*Asynchronous
HTTP GET
requests/
responses*



End of the PrimeCheck App Case Study: Overview