

The PrimeCheck App Case Study: Client Structure & Functionality

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

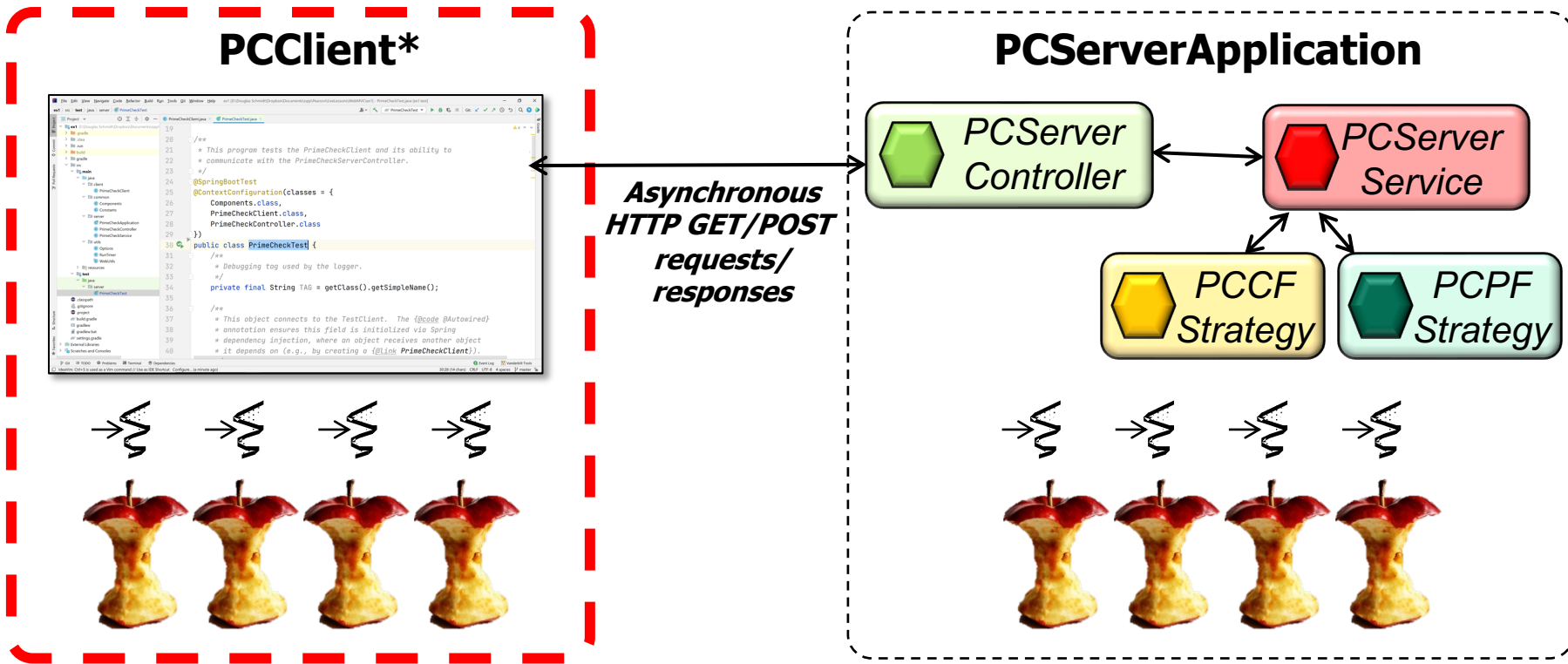
**Institute for Software
Integrated Systems**

**Vanderbilt University
Nashville, Tennessee, USA**



Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of PCClient* classes that send/receive HTTP GET requests/responses to/from the PCServerApplication microservice



The Structure & Functionality of PCClient* Classes

The Structure & Functionality of PCClient* Classes

- The PCClient* classes performs asynchronous remote method invocations on the PCServerController to determine the primality of large integers

```
@Component
public class PCClientParallelFlux
    @Autowired PCProxyAPI mPCProxy;

    public Flux<Integer> testIndividualCalls
        (Flux<Integer> primeCandidates) {...}

    public Flux<Integer> testFluxCall
        (Flux<Integer> primeCandidates) {...}
}
```

See [WebFlux/ex2/src/main/java/client/PCClientParallelFlux.java](#)

The Structure & Functionality of PCClient* Classes

- The PCClient* classes performs asynchronous remote method invocations on the PCServerController to determine the primality of large integers

@Component

```
public class PCClientParallelFlux
    @Autowired PCProxyAPI mPCProxy;

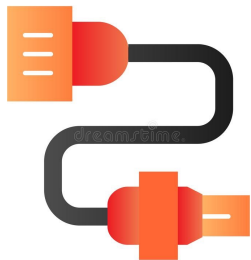
    public Flux<Integer> testIndividualCalls
        (Flux<Integer> primeCandidates) {...}

    public Flux<Integer> testFluxCall
        (Flux<Integer> primeCandidates) {...}
}
```

This annotation enables the auto-detection & wiring of dependent implementation classes via classpath scanning

The Structure & Functionality of PCClient* Classes

- The PCClient* classes performs asynchronous remote method invocations on the PCServerController to determine the primality of large integers



This field is auto-wired by Spring's dependency injection framework

```
@Component
public class PCClientParallelFlux
    @Autowired PCProxyAPI mPCProxy;

    public Flux<Integer> testIndividualCalls
        (Flux<Integer> primeCandidates) {...}

    public Flux<Integer> testFluxCall
        (Flux<Integer> primeCandidates) {...}
}
```

The Structure & Functionality of PCClient* Classes

- The PCClient* classes performs asynchronous remote method invocations on the PCServerController to determine the primality of large integers

```
@Component
public class PCClientParallelFlux
    @Autowired PCProxyAPI mPCProxy;

    public Flux<Integer> testIndividualCalls
        (Flux<Integer> primeCandidates) {...}

    public Flux<Integer> testFluxCall
        (Flux<Integer> primeCandidates) {...}
}
```

*Sends individual
HTTP GET requests
to the server*

The Structure & Functionality of PCClient* Classes

- The PCClient* classes performs asynchronous remote method invocations on the PCServerController to determine the primality of large integers

```
Flux<Integer> testIndividualCalls (Flux<Integer> primeCandidates)
{
    return primeCandidates
        .parallel ()

        .runOn (Schedulers.boundedElastic ())

        .map (primeCandidate -> mPCProxy
            .checkIfPrime (PARALLEL_FLUX,
                primeCandidate))

        .sequential ();
}
```

*Test a Flux of prime candidate
Integers in parallel one at a time*

The Structure & Functionality of PCClient* Classes

- The PCClient* classes performs asynchronous remote method invocations on the PCServerController to determine the primality of large integers

```
Flux<Integer> testIndividualCalls(Flux<Integer> primeCandidates)
{
    return primeCandidates
        .parallel()

        .runOn(Schedulers.boundedElastic())

        .map(primeCandidate -> mPCProxy
            .checkIfPrime(PARALLEL_FLUX,
                primeCandidate))

        .sequential();
}
```

*Use a ParallelFlux to make
all the calls in parallel*

Parallelism is performed by the client rather than by the server

The Structure & Functionality of PCClient* Classes

- The PCClient* classes performs asynchronous remote method invocations on the PCServerController to determine the primality of large integers

```
Flux<Integer> testIndividualCalls(Flux<Integer> primeCandidates)
{
    return primeCandidates
        .parallel()
        .runOn(Schedulers.boundedElastic())
        .map(primeCandidate -> mPCProxy
            .checkIfPrime(PARALLEL_FLUX,
                primeCandidate))
        .sequential();
}
```

Use the BoundedElastic Scheduler since these calls go over the network

The Structure & Functionality of PCClient* Classes

- The PCClient* classes performs asynchronous remote method invocations on the PCServerController to determine the primality of large integers

```
Flux<Integer> testIndividualCalls(Flux<Integer> primeCandidates)
{
    return primeCandidates
        .parallel()

        .runOn(Schedulers.boundedElastic())

        .map(primeCandidate -> mPCProxy
            .checkIfPrime(PARALLEL_FLUX,
                primeCandidate))

        .sequential();
}
```

Perform a remote method invocation via a proxy

The Structure & Functionality of PCClient* Classes

- The PCClient* classes performs asynchronous remote method invocations on the PCServerController to determine the primality of large integers

```
@Component
public class PCClientParallelFlux
    @Autowired PCProxy mPCProxy;

    public Flux<Integer> testIndividualCalls
        (Flux<Integer> primeCandidates,
         boolean parallel) {...}

    public Flux<Integer> testFluxCall
        (Flux<Integer> primeCandidates) {...}
}
```

Sends a Flux of Integer objects in a single HTTP POST request to server

The Structure & Functionality of PCClient* Classes

- The PCClient* classes performs asynchronous remote method invocations on the PCServerController to determine the primality of large integers

```
Flux<Integer> testFluxCalls (Flux<Integer> primeCandidates) {  
  
    return mPCProxy  
        .checkIfPrimeFlux (PARALLEL_FLUX,  
                           primeCandidates) ;  
}
```

Test a Flux of Integer prime candidates in one fell swoop

The Structure & Functionality of PCClient* Classes

- The PCClient* classes performs asynchronous remote method invocations on the PCServerController to determine the primality of large integers

```
Flux<Integer> testFluxCalls(Flux<Integer> primeCandidates) {  
  
    return mPCProxy  
        .checkIfPrimeFlux(PARALLEL_FLUX,  
                          primeCandidates);  
}
```

Perform a remote method invocation via a proxy

Any requested parallelism is performed by the server rather than by the client

The Structure & Functionality of the PCProxyAPI Interface

The Structure & Functionality of the PCProxyAPI Interface

- PCProxyAPI uses the declarative HTTP interface features in Spring 6+ to abstract low-level details of remote method invocations using HTTP

```
public interface PCProxyAPI {  
    @GetExchange(CHECK_IF_PRIME)  
    Integer checkIfPrime(@RequestParam Integer strategy,  
                        @RequestParam Integer primeCandidate);  
  
    @PostExchange(value = CHECK_IF_PRIME_FLUX,  
                 contentType = APPLICATION_NDJSON_VALUE)  
    Flux<Integer> checkIfPrimeFlux  
        (@RequestParam Integer strategy,  
         @RequestBody Flux<Integer> primeCandidates);  
}
```

See [WebFlux/ex2/src/main/java/client/PCProxy.java](#)

The Structure & Functionality of the PCProxyAPI Interface

- PCProxyAPI uses the declarative HTTP interface features in Spring 6+ to abstract low-level details of remote method invocations using HTTP

```
public interface PCProxyAPI {  
    @GetExchange(CHECK_IF_PRIME)  
    Integer checkIfPrime(@RequestParam Integer strategy,  
                          @RequestParam Integer primeCandidate);
```

These proxy methods shield clients from low-level HTTP programming details

```
@PostExchange(value = CHECK_IF_PRIME_FLUX,  
              contentType = APPLICATION_NDJSON_VALUE)  
Flux<Integer> checkIfPrimeFlux  
    (@RequestParam Integer strategy,  
     @RequestBody Flux<Integer> primeCandidates);
```

The Structure & Functionality of the PCProxyAPI Interface

- PCProxyAPI uses the declarative HTTP interface features in Spring 6+ to abstract low-level details of remote method invocations using HTTP

```
public interface PCProxyAPI {  
    @GetExchange (CHECK_IF_PRIME)  
    Integer checkIfPrime (@RequestParam Integer strategy,  
                          @RequestParam Integer primeCandidate);
```

*These annotations generate
HTTP GET & POST requests*

```
@PostExchange (value = CHECK_IF_PRIME_FLUX,  
                contentType = APPLICATION_NDJSON_VALUE)  
Flux<Integer> checkIfPrimeFlux  
    (@RequestParam Integer strategy,  
     @RequestBody Flux<Integer> primeCandidates);
```

The Structure & Functionality of the PCProxyAPI Interface

- PCProxyAPI uses the declarative HTTP interface features in Spring 6+ to abstract low-level details of remote method invocations using HTTP

```
public interface PCProxyAPI {  
    @GetExchange(CHECK_IF_PRIME)  
    Integer checkIfPrime(@RequestParam Integer strategy,  
                        @RequestParam Integer primeCandidate);
```

These path variables route the GET & POST requests to the appropriate endpoint handlers

```
@PostExchange(value = CHECK_IF_PRIME_FLUX,  
              contentType = APPLICATION_NDJSON_VALUE)  
Flux<Integer> checkIfPrimeFlux  
    (@RequestParam Integer strategy,  
     @RequestBody Flux<Integer> primeCandidates);
```

The Structure & Functionality of the PCProxyAPI Interface

- PCProxyAPI uses the declarative HTTP interface features in Spring 6+ to abstract low-level details of remote method invocations using HTTP

```
public interface PCProxyAPI {  
    @GetExchange(CHECK_IF_PRIME)  
    Integer checkIfPrime(@RequestParam Integer strategy,  
                        @RequestParam Integer primeCandidate);  
}
```

This value enables passing Flux as a param

```
@PostExchange(value = CHECK_IF_PRIME_FLUX,  
              contentType = APPLICATION_NDJSON_VALUE)  
Flux<Integer> checkIfPrimeFlux  
    (@RequestParam Integer strategy,  
     @RequestBody Flux<Integer> primeCandidates);
```

See [org/springframework/web/service/annotation/PostExchange.html](https://org.springframework/web/service/annotation/PostExchange.html)

The Structure & Functionality of the PCProxyAPI Interface

- PCProxyAPI uses the declarative HTTP interface features in Spring 6+ to abstract low-level details of remote method invocations using HTTP

```
public interface PCProxyAPI {  
    @GetExchange(CHECK_IF_PRIME)  
    Integer checkIfPrime(@RequestParam Integer strategy,  
                        @RequestParam Integer primeCandidate);
```

*These annotations enable passing
params via HTTP GET & POST requests*

```
@PostExchange(value = CHECK_IF_PRIME_FLUX,  
              contentType = APPLICATION_NDJSON_VALUE)  
Flux<Integer> checkIfPrimeFlux  
    (@RequestParam Integer strategy,  
     @RequestBody Flux<Integer> primeCandidates);
```

The Structure & Functionality of the PCProxyAPI Interface

- PCProxyAPI uses the declarative HTTP interface features in Spring 6+ to abstract low-level details of remote method invocations using HTTP

```
public interface PCProxyAPI {  
    @GetExchange(CHECK_IF_PRIME)  
    Integer checkIfPrime(@RequestParam Integer strategy,  
                        @RequestParam Integer primeCandidate);  
}
```

This proxy method returns a Flux that emits Integer results

```
@PostExchange(value = CHECK_IF_PRIME_FLUX,  
              contentType = APPLICATION_NDJSON_VALUE)  
Flux<Integer> checkIfPrimeFlux  
    (@RequestParam Integer strategy,  
     @RequestBody Flux<Integer> primeCandidates);
```

End of the PrimeCheck App Case Study: Structure & Functionality of the Client