

The PrimeCheck App Case Study: Server Structure & Functionality

Douglas C. Schmidt

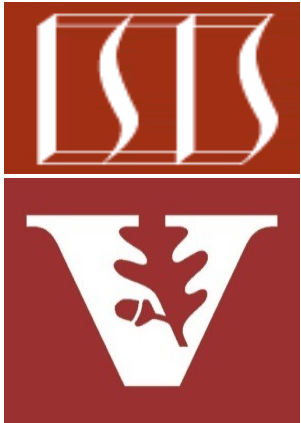
d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

**Institute for Software
Integrated Systems**

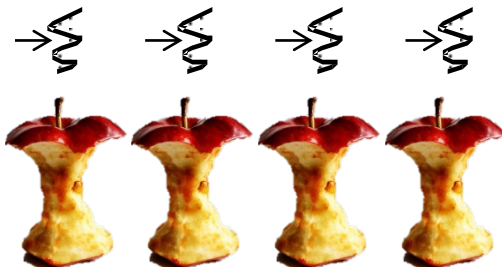
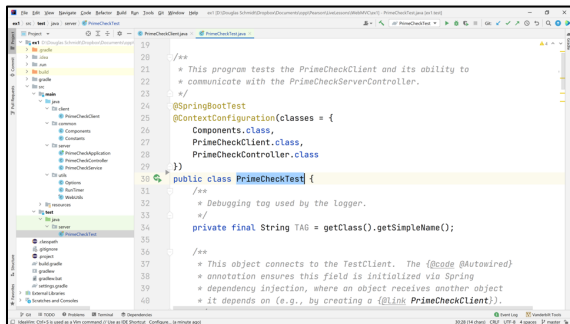
**Vanderbilt University
Nashville, Tennessee, USA**



Learning Objectives in this Part of the Lesson

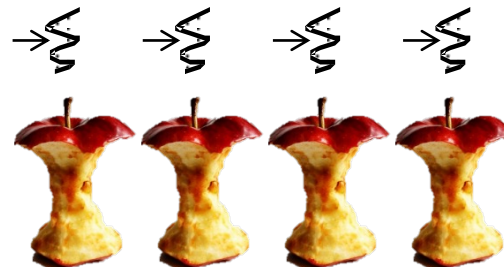
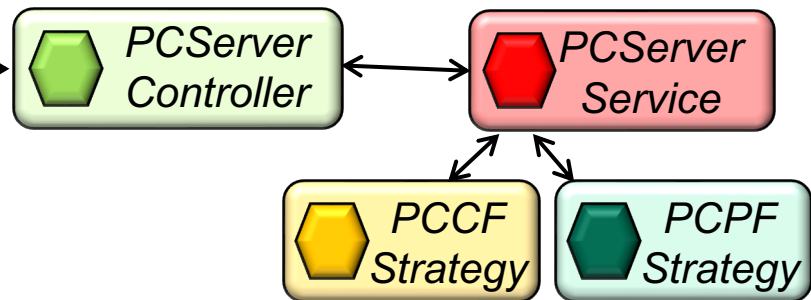
- Understand the structure & functionality of the PCServerController & the PC ServerService & how they check the primality of large integers asynchronously

PrimeCheckTest



**Asynchronous
HTTP GET
requests/
responses**

PCServerApp



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

Structure & Functionality of the PCServerController

Structure & Functionality of the PCServerController

- Client HTTP GET/POST requests are mapped to endpoint handlers via the PCServerController class

```
@RestController
public class PCServerController {
    @GetMapping(CHECK_IF_PRIME)
    public Integer checkIfPrime
        (Integer strategy,
         Integer primeCandidate) {...}

    @PostMapping(value = CHECK_IF_PRIME_FLUX,
                 consumes = APPLICATION_NDJSON_VALUE)
    public Flux<Integer>
        checkIfPrimeFlux
            (Integer Strategy,
             @RequestBody Flux<Integer>
              primeCandidates) {...}
}
```

See [WebFlux/ex2/src/main/java/server/PCServerController.java](#)

Structure & Functionality of the PCServerController

- Client HTTP GET/POST requests are mapped to endpoint handlers via the PCServerController class **@RestController**

This annotation ensures request handlers in the controller automatically serialize return objects into HttpResponse objects

```
public class PCServerController {  
    @GetMapping(CHECK_IF_PRIME)  
    public Integer checkIfPrime  
        (Integer strategy,  
         Integer primeCandidate) {...}  
  
    @PostMapping(value = CHECK_IF_PRIME_FLUX,  
                 consumes = APPLICATION_NDJSON_VALUE)  
    public Flux<Integer>  
        checkIfPrimeFlux  
        (Integer Strategy,  
         @RequestBody Flux<Integer>  
         primeCandidates) {...}  
}
```

Structure & Functionality of the PCServerController

- Client HTTP GET/POST requests are mapped to endpoint handlers via the PCServerController class

```
@RestController
public class PCServerController {
    @GetMapping(CHECK_IF_PRIME)
    public Integer checkIfPrime
        (Integer strategy,
         Integer primeCandidate) {...}

    @PostMapping(value = CHECK_IF_PRIME_FLUX,
                 consumes = APPLICATION_NDJSON_VALUE)
    public Flux<Integer>
        checkIfPrimeFlux
            (Integer Strategy,
             @RequestBody Flux<Integer>
              primeCandidates) {...}
}
```

These methods forward to PCServerService methods, which then forward to the designated strategy to determine the primality of params & return results

Structure & Functionality of the PCServerController

- Client HTTP GET/POST requests are mapped to endpoint handlers via the PCServerController class

*These annotations map
HTTP GET/POST requests
onto endpoint handlers*

```
@RestController
public class PCServerController {
    @GetMapping(CHECK_IF_PRIME)
    public Integer checkIfPrime
        (Integer strategy,
         Integer primeCandidate) {...}

    @PostMapping(value = CHECK_IF_PRIME_FLUX,
                 consumes = APPLICATION_NDJSON_VALUE)
    public Flux<Integer>
        checkIfPrimeFlux
            (Integer Strategy,
             @RequestBody Flux<Integer>
              primeCandidates) {...}
}
```

Structure & Functionality of the PCServerController

- Client HTTP GET/POST requests are mapped to endpoint handlers via the PCServerController class

```
@RestController
public class PCServerController {
    @GetMapping(CHECK_IF_PRIME)
    public Integer checkIfPrime
        (Integer strategy,
         Integer primeCandidate) {...}

    @PostMapping(value = CHECK_IF_PRIME_FLUX,
                 consumes = APPLICATION_NDJSON_VALUE)
    public Flux<Integer>
        checkIfPrimeFlux
            (Integer Strategy,
             @RequestBody Flux<Integer>
              primeCandidates) {...}
}
```

*These strings identify the
endpoint handlers from
HTTP GET/POST requests*

Structure & Functionality of the PCServerController

- Client HTTP GET/POST requests are mapped to endpoint handlers via the PCServerController class

```
@RestController
public class PCServerController {
    @GetMapping(CHECK_IF_PRIME)
    public Integer checkIfPrime
        (Integer strategy,
         Integer primeCandidate) {...}
```

*Enables passing
Flux as a param!*

```
{"value": 23}
{"value": 371}
{"value": 59}
{"value": 7893}
```

```
@PostMapping(value = CHECK_IF_PRIME_FLUX,
              consumes = APPLICATION_NDJSON_VALUE)
public Flux<Integer>
    checkIfPrimeFlux
        (Integer Strategy,
         @RequestBody Flux<Integer>
          primeCandidates) {...}
}
```

See nurkiewicz.com/2021/08/json-streaming-in-webflux.html

Structure & Functionality of the PCServerController

- Client HTTP GET/POST requests are mapped to endpoint handlers via the PCServerController class

```
@RestController
public class PCServerController {
    @GetMapping(CHECK_IF_PRIME)
    public Integer checkIfPrime
        (Integer strategy,
         Integer primeCandidate) {...}

    @PostMapping(value = CHECK_IF_PRIME_FLUX,
                 consumes = APPLICATION_NDJSON_VALUE)
    public Flux<Integer>
        checkIfPrimeFlux
            (Integer Strategy,
             @RequestBody Flux<Integer>
              primeCandidates) {...}
}
```

*This annotation maps
the HttpRequest object
into a Java object*

Structure & Functionality of the PCServerService

Structure & Functionality of the PCServerService

- The PCServerService class defines implementation methods that are called by the PCServerController

```
@Service
public class PCServerService
{
    ...
}
```

See <WebFlux/ex2/src/main/java/server/PCServerService.java>

Structure & Functionality of the PCServerService

- The PCServerService class defines implementation methods that are called by the PCServerController

```
@Service  
public class PCServerService  
    ...  
}
```

This annotation indicates the class implements "business logic" & enables auto-detection & wiring of dependent classes via classpath scanning

Structure & Functionality of the PCServerService

- The PCServerService class defines implementation methods that are called by the PCServerController

```
@Service
public class PCServerService
    PCAbstractStrategy[] mStrategy = {
        new PCConcurrentFluxStrategy(),
        new PCParallelFluxStrategy()
    };
    ...
}
```

*This array contains
concrete strategies whose
methods are implemented
to check for primality*

This solution could also be implemented via multiple microservices

Structure & Functionality of the PCServerService

- The PCServerService class defines implementation methods that are called by the PCServerController

```
@Service
public class PCServerService
{
    ...
    public Integer checkIfPrime
        (Integer strategy,
         Integer primeCandidate) {
        mStrategy[strategy]
            .checkIfPrime(primeCandidate);
    }
    ...
}
```

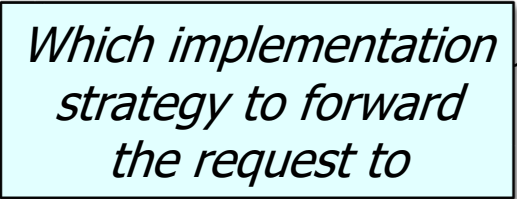
*Checks the primality
of a single Integer*

Structure & Functionality of the PCServerService

- The PCServerService class defines implementation methods that are called by the PCServerController

```
@Service
public class PCServerService
{
    ...
    public Integer checkIfPrime
        (Integer strategy,
         Integer primeCandidate) {
        mStrategy[strategy]
            .checkIfPrime (primeCandidate) ;
    }
    ...
}
```

*Which implementation
strategy to forward
the request to*



Structure & Functionality of the PCServerService

- The PCServerService class defines implementation methods that are called by the PCServerController

```
@Service
public class PCServerService
{
    ...
    public Integer checkIfPrime
        (Integer strategy,
         Integer primeCandidate) {
        mStrategy[strategy]
            .checkIfPrime(primeCandidate);
    }
    ...
}
```

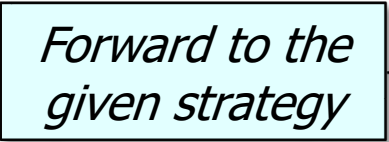
Checks the primeCandidate param for primality & returns 0 if it's prime or the smallest factor if it's not

Structure & Functionality of the PCServerService

- The PCServerService class defines implementation methods that are called by the PCServerController

```
@Service
public class PCServerService
{
    ...
    public Integer checkIfPrime
        (Integer strategy,
         Integer primeCandidate) {
        mStrategy[strategy]
            .checkIfPrime(primeCandidate) ;
    }
    ...
}
```

*Forward to the
given strategy*



Structure & Functionality of the PCServerService

- The PCServerService class defines implementation methods that are called by the PCServerController

```
@Service
public class PCServerService
{
    ...
    public Flux<Integer>
        checkIfPrimeFlux
        (Integer strategy,
         Flux<Integer> primeCandidates) {
        mStrategy[strategy]
            .checkIfPrimeFlux(primeCandidates);
    }
}
```

*Check all the elements
in a Flux for primality*



Structure & Functionality of the PCServerService

- The PCServerService class defines implementation methods that are called by the PCServerController

```
@Service
public class PCServerService
{
    ...
    public Flux<Integer>
    checkIfPrimeFlux
    (Integer strategy,
     Flux<Integer> primeCandidates) {
        mStrategy[strategy]
        .checkIfPrimeFlux(primeCandidates);
    }
}
```

*Which implementation
strategy to forward
the request to*



Structure & Functionality of the PCServerService

- The PCServerService class defines implementation methods that are called by the PCServerController

```
@Service
public class PCServerService
{
    ...
    public Flux<Integer>
    checkIfPrimeFlux
        (Integer strategy,
         Flux<Integer> primeCandidates) {
        mStrategy[strategy]
            .checkIfPrimeFlux(primeCandidates) ;
    }
}
```

Check all elements in the primeCandidates Flux param for primality & return a Flux whose results are 0 if an element is prime or the smallest factor if it's not

Structure & Functionality of the PCServerService

- The PCServerService class defines implementation methods that are called by the PCServerController

```
@Service
public class PCServerService
{
    ...
    public Flux<Integer>
    checkIfPrimeFlux
        (Integer strategy,
         Flux<Integer> primeCandidates) {
        mStrategy[strategy]
            .checkIfPrimeFlux(primeCandidates) ;
    }
}
```

*Forward to the
given strategy*

Structure & Functionality of the PCServerService

- PCAbstractStrategy defines a default method & an abstract method called by PCServerService

```
public interface PCAbstractStrategy {  
    default Integer checkIfPrime  
        (Integer primeCandidate) {  
        return isPrime(primeCandidate);  
    }  
  
    Flux<Integer> checkIfPrimeFlux  
        (Flux<Integer> primeCandidates);  
}
```

Structure & Functionality of the PCServerService

- PCAbstractStrategy defines a default method & an abstract method called by PCServerService

```
public interface PCAbstractStrategy {  
    default Integer checkIfPrime  
        (Integer primeCandidate) {  
        return isPrime(primeCandidate);  
    }  
  
    Flux<Integer> checkIfPrimeFlux  
        (Flux<Integer> primeCandidates);  
}
```

*This default method
can be used for all the
strategies since it's
very straightforward*

Structure & Functionality of the PCServerService

- PCAbstractStrategy defines a default method & an abstract method called by PCServerService

```
public interface PCAbstractStrategy {  
    default Integer checkIfPrime  
        (Integer primeCandidate) {  
        return isPrime(primeCandidate);  
    }  
}
```

*This 'abstract' method
must be defined by an
implementation class*

```
Flux<Integer> checkIfPrimeFlux  
    (Flux<Integer> primeCandidates);  
}
```

e.g., PCParallelFluxStrategy, PCConcurrentFluxStrategy, etc.

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