The PrimeCheck App Case Study: Server Structure & Functionality

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

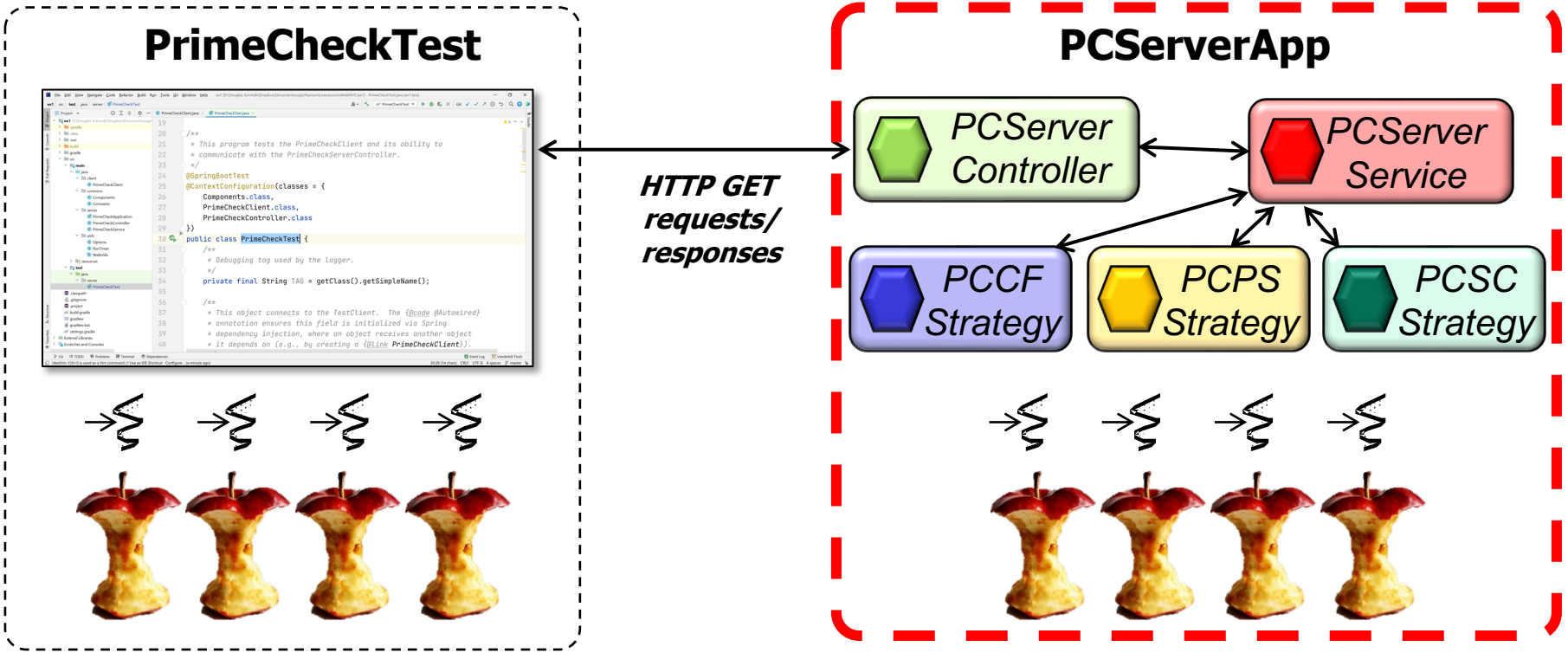
Institute for Software Integrated Systems

Vanderbilt University

Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of the PCServerController & the PCServerService & how they check the primality of large integers in the Web



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

Structure & Functionality of the PCServerController

Structure & Functionality of the PCServerController

- Client HTTP GET requests are mapped to endpoint handler methods via the PCServerController class

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

  @GetMapping(CHECK_IF_PRIME_LIST)
  public List<Integer>
  checkIfPrimeList
    (Integer Strategy,
     @RequestParam List<Integer>
       primeCandidates,
     Boolean parallel) {...}
}
```

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

Structure & Functionality of the PCServerController

- Client HTTP GET requests are mapped to endpoint handler methods via the PCServerController class

This annotation ensures request handling methods in the controller class automatically serialize return objects into HttpResponse objects

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

  @GetMapping(CHECK_IF_PRIME_LIST)
  public List<Integer>
  checkIfPrimeList
    (Integer Strategy,
     @RequestParam List<Integer>
       primeCandidates,
     Boolean parallel) {...}
}
```

See www.baeldung.com/spring-controller-vs-restcontroller

Structure & Functionality of the PCServerController

- Client HTTP GET requests are mapped to endpoint handler methods via the PCServerController class

These methods just forward to the PCServerService methods, which then in turn forward to the designated strategy to determine the primality of the parameters & return the results

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

  @GetMapping(CHECK_IF_PRIME_LIST)
  public List<Integer>
  checkIfPrimeList
    (Integer Strategy,
     @RequestParam List<Integer>
       primeCandidates,
     Boolean parallel) {...}
}
```

Structure & Functionality of the PCServerController

- Client HTTP GET requests are mapped to endpoint handler methods via the PCServerController class

This annotation maps HTTP GET requests onto endpoint handler methods

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

   @GetMapping(CHECK_IF_PRIME_LIST)
   public List<Integer>
   checkIfPrimeList
      (Integer Strategy,
       @RequestParam List<Integer>
         primeCandidates,
       Boolean parallel) {...}
}
```

See www.baeldung.com/spring-new-requestmapping-shortcuts

Structure & Functionality of the PCServerController

- Client HTTP GET requests are mapped to endpoint handler methods via the PCServerController class

These strings are used to automatically identify the endpoint handler methods from incoming GET requests

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

  @GetMapping(CHECK_IF_PRIME_LIST)
  public List<Integer>
  checkIfPrimeList
    (Integer Strategy,
     @RequestParam List<Integer>
       primeCandidates,
     Boolean parallel) {...}
}
```

See www.baeldung.com/spring-new-requestmapping-shortcuts

Structure & Functionality of the PCServerController

- Client HTTP GET requests are mapped to endpoint handler methods via the PCServerController class

This annotation maps to query parameters, form data, & parts in multipart requests

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

  @GetMapping(CHECK_IF_PRIME_LIST)
  public List<Integer>
  checkIfPrimeList
    (Integer Strategy,
     @RequestParam List<Integer>
       primeCandidates,
     Boolean parallel) {...}
}
```

See www.baeldung.com/spring-request-param

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 WebMVC/ex1/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

See www.baeldung.com/spring-component-repository-service

Structure & Functionality of the PCServerService

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

```
@Service
public class PCServerService
  PCAbstractStrategy[] mStrategy = {
    new PCStructuredConcurrencyStrategy(),
    new PCParallelStreamStrategy(),
    new PCCompletableFutureStrategy()
  };

  ...
}
```

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 List<Integer>
  checkIfPrimeList
    (Integer strategy,
     List<Integer> primeCandidates,
     Boolean parallel) {
     mStrategy[strategy]
       .checkIfPrimeList(primeCandidates,
                         parallel);
  }
}
```

Check all the elements in a List for primality

Structure & Functionality of the PCServerService

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

```
@Service
public class PCServerService
  ...
  public List<Integer>
  checkIfPrimeList
    (Integer strategy,
     List<Integer> primeCandidates,
     Boolean parallel) {
     mStrategy[strategy]
      .checkIfPrimeList(primeCandidates,
                        parallel);
  }
}
```

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 List<Integer>
  checkIfPrimeList
    (Integer strategy,
     List<Integer> primeCandidates,
     Boolean parallel) {
     mStrategy[strategy]
      .checkIfPrimeList(primeCandidates,
                        parallel);
  }
}
```

Check all elements in the primeCandidates List param for primality & return a List 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

If this param is 'true' a parallel strategy is used, else a sequential strategy is used

```
@Service
public class PCServerService
   ...
   public List<Integer>
   checkIfPrimeList
      (Integer strategy,
       List<Integer> primeCandidates,
       Boolean parallel) {
       mStrategy[strategy]
         .checkIfPrimeList(primeCandidates,
                           parallel);
   }
}
```

Structure & Functionality of the PCServerService

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

```
@Service
public class PCServerService
  ...
  public List<Integer>
  checkIfPrimeList
    (Integer strategy,
     List<Integer> primeCandidates,
     Boolean parallel) {
     mStrategy[strategy]
       .checkIfPrimeList(primeCandidates,
                         parallel);
  }
}
```

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);
  }

  List<Integer> checkIfPrimeList
          (List<Integer> primeCandidates,
           Boolean parallel);
}
```

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);
  }

  List<Integer> checkIfPrimeList
                (List<Integer> primeCandidates,
                 Boolean parallel);
}
```

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

See docs.oracle.com/javase/tutorial/java/IandI/defaultmethods.html

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);
  }

  List<Integer> checkIfPrimeList
          (List<Integer> primeCandidates,
           Boolean parallel);
}
```

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

e.g., PCParallelStreamStrategy, PCStructuredConcurrencyStrategy, etc.

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