The PrimeCheck App Case Study: Client Structure & Functionality

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

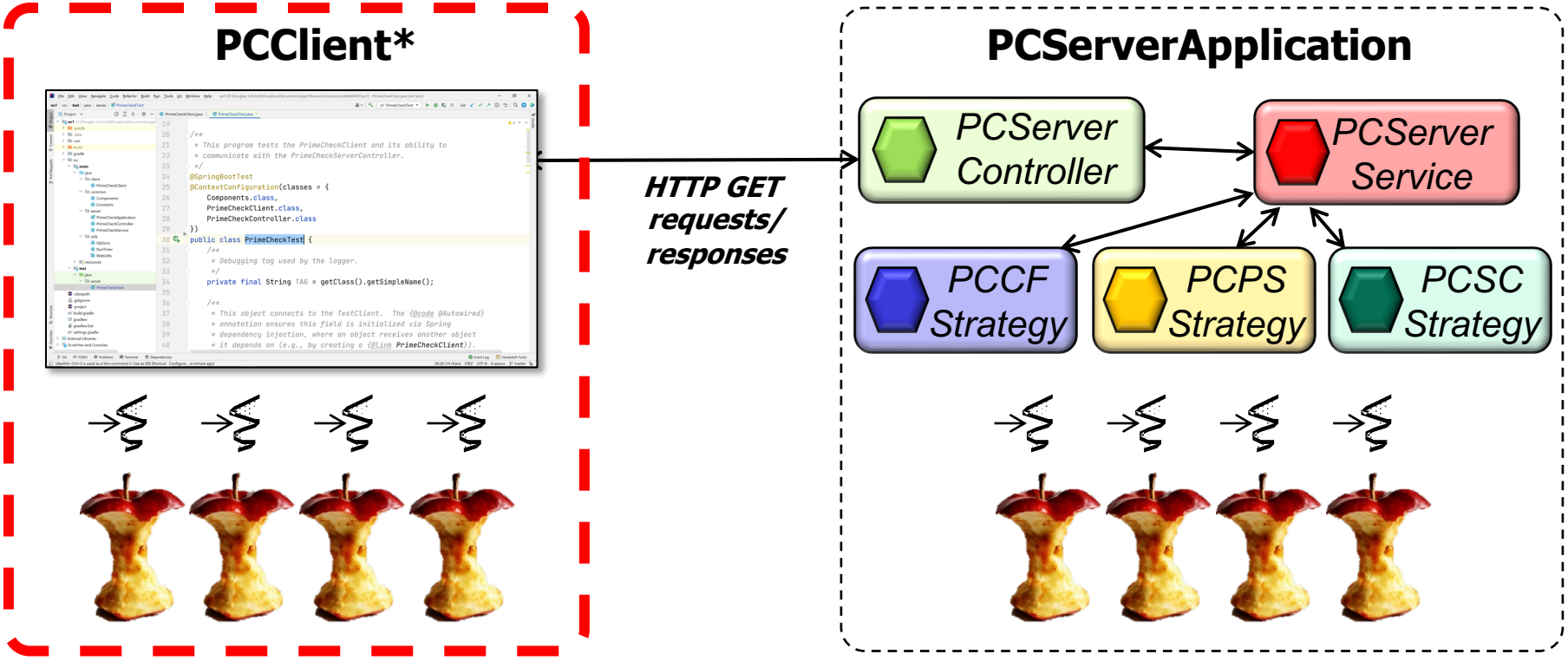
Institute for Software Integrated Systems

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



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

The Structure & Functionality of PCClient* Classes

The Structure & Functionality of PCClient* Classes

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

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

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

  public List<Integer>
  testListCall
    (List<Integer> primeCandidates,
     boolean parallel) {...}
}
```

See WebMVC/ex1/src/main/java/client/PCClientParallelStream.java

The Structure & Functionality of PCClient* Classes

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

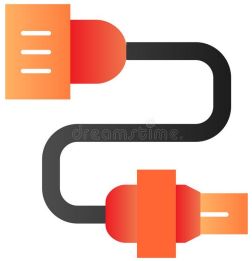
This annotation enables the auto-detection & wiring of dependent implementation classes via classpath scanning

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

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

   public List<Integer>
   testListCall
      (List<Integer> primeCandidates,
       boolean parallel) {...}
}
```

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

The Structure & Functionality of PCClient* Classes

- The PCClient* classes performs synchronous 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 PCClientParallelStream
  @Autowired PCProxy mPCProxy;

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

  public List<Integer>
  testListCall
    (List<Integer> primeCandidates,
     boolean parallel) {...}
}
```

See www.baeldung.com/spring-autowire

The Structure & Functionality of PCClient* Classes

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

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

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

  public List<Integer>
  testListCall
    (List<Integer> primeCandidates,
     boolean parallel) {...}
}
```

Sends individual HTTP GET requests to the server

The Structure & Functionality of PCClient* Classes

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

```
List<Integer> testIndividualCalls(List<Integer> primeCandidates,
                                  boolean parallel) {
  return StreamSupport
    .stream(primeCandidates.spliterator(),
            parallel)

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

    .toList();
```

Conditionally enable a parallel or sequential stream using the 'parallel' param

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

The Structure & Functionality of PCClient* Classes

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

```
List<Integer> testIndividualCalls(List<Integer> primeCandidates,
                                  boolean parallel) {
  return StreamSupport
    .stream(primeCandidates.spliterator(),
            parallel)

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

    .toList();
```

Perform a remote method invocation via a proxy

The Structure & Functionality of PCClient* Classes

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

```
List<Integer> testIndividualCalls(List<Integer> primeCandidates,
                                  boolean parallel) {
  return StreamSupport
    .stream(primeCandidates.spliterator(),
            parallel)

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

    .toList();
```

Convert the Stream of results into a List

The Structure & Functionality of PCClient* Classes

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

Sends a List of Integer objects in one HTTP GET request to the server

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

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

  public List<Integer>
  testListCall
    (List<Integer> primeCandidates,
     boolean parallel) {...}
}
```

The Structure & Functionality of PCClient* Classes

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

```
List<Integer> testListCalls(List<Integer> primeCandidates,
                            boolean parallel) {

  return mPCProxy
    .checkIfPrimeList(PARALLEL_STREAM,
                      primeCandidates,
                      parallel);
}
```

Perform a remote method invocation via a proxy that passes a list of Integer objects

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

The Structure & Functionality of PCProxy Class

The Structure & Functionality of PCProxy

- PCProxy abstracts low-level details of remote method invocations using HTTP

```
@Component
public class PCProxy {
  @Autowired RestTemplate mRestTemplate;

  public Integer checkIfPrime(int strategy, int primeCandidate){
    ...
  }
```

See WebMVC/ex1/src/main/java/client/PCProxy.java

The Structure & Functionality of PCProxy

- PCProxy abstracts low-level details of remote method invocations using HTTP

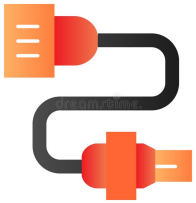
```
@Component
public class PCProxy {
   @Autowired RestTemplate mRestTemplate;

   public Integer checkIfPrime(int strategy, int primeCandidate){
      ...
   }
```

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

The Structure & Functionality of PCProxy

- PCProxy abstracts low-level details of remote method invocations using HTTP

```
@Component
public class PCProxy {
   @Autowired RestTemplate mRestTemplate;

   public Integer checkIfPrime(int strategy, int primeCandidate){
      ...
   }
```



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

The Structure & Functionality of PCProxy

- PCProxy abstracts low-level details of remote method invocations using HTTP

```
@Component
public class PCProxy {
  @Autowired RestTemplate mRestTemplate;

  public Integer checkIfPrime(int strategy, int primeCandidate){
    ...
  }

  public List<Integer> checkIfPrimeList(int strategy,
                                        List<Integer> primeCandidates,
                                        Boolean parallel) { ... }
}
```

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

The Structure & Functionality of PCProxy

- PCProxy abstracts low-level details of remote method invocations using HTTP

```
@Component
public class PCProxy {
  ...
  public Integer checkIfPrime(int strategy, int primeCandidate){
    var uri = UriComponentsBuilder
      .fromPath(CHECK_IF_PRIME)
      .queryParam("strategy", strategy)
      .queryParam("primeCandidate", primeCandidate)
      .build().toUriString();


    return WebUtils
       .makeGetRequest(mRestTemplate, uri, Integer.class);
  }
```

The params to pass to the server

The Structure & Functionality of PCProxy

- PCProxy abstracts low-level details of remote method invocations using HTTP

```
@Component
public class PCProxy {
   ...
   public Integer checkIfPrime(int strategy, int primeCandidate){
     var uri = UriComponentsBuilder
       .fromPath(CHECK_IF_PRIME)
       .queryParam("strategy", strategy)
       .queryParam("primeCandidate", primeCandidate)
       .build().toUriString();


     return WebUtils
        .makeGetRequest(mRestTemplate, uri, Integer.class);
   }
```

Create a URI passed in an HTTP GET request to determine if an Integer is prime

e.g., /checkIfPrime?strategy=1&primeCandidate=2147483515

The Structure & Functionality of PCProxy

- PCProxy abstracts low-level details of remote method invocations using HTTP

```
@Component
public class PCProxy {
   ...
   public Integer checkIfPrime(int strategy, int primeCandidate){
     var uri = UriComponentsBuilder
        .fromPath(CHECK_IF_PRIME)
        .queryParam("strategy", strategy)
        .queryParam("primeCandidate", primeCandidate)
        .build().toUriString();

     return WebUtils
         .makeGetRequest(mRestTemplate, uri, Integer.class);
   }
```

Make an HTTP GET call to the server at the designed URL for each primeCandidate

e.g., http://localhost:8081/checkIfPrime?strategy=1&primeCandidate=2147483515

The Structure & Functionality of PCProxy

- PCProxy abstracts low-level details of remote method invocations using HTTP

```
@Component
public class PCProxy {
   ...
   public Integer checkIfPrime(int strategy, int primeCandidate){
     var uri = UriComponentsBuilder
        .fromPath(CHECK_IF_PRIME)
        .queryParam("strategy", strategy)
        .queryParam("primeCandidate", primeCandidate)
        .build().toUriString();

     return WebUtils
         .makeGetRequest(mRestTemplate, uri, Integer.class);
   }
```

Make an HTTP GET call to the server at the designed URL for each primeCandidate

e.g., http://localhost:8081/checkIfPrime?strategy=1&primeCandidate=2147483515

The Structure & Functionality of PCProxy

- PCProxy abstracts low-level details of remote method invocations using HTTP

```
@Component
public class PCProxy {
   ...
   public List<Integer> checkIfPrimeList(int strategy,
                                         List<Integer> primeCandidates,
                                         Boolean parallel) {
     var uri = UriComponentsBuilder
       .fromPath(CHECK_IF_PRIME_LIST)
       .queryParam("strategy", strategy)
       .queryParam("primeCandidates", WebUtils
                   .list2String(primeCandidates))
       .queryParam("parallel", parallel)
       .build().toUriString();
     ...
```

The params to pass to the server

The Structure & Functionality of PCProxy

- PCProxy abstracts low-level details of remote method invocations using HTTP

Create a URI passed in an HTTP GET request to determine if a List of Integers are prime

```
@Component
public class PCProxy {
  ...
  public List<Integer> checkIfPrimeList(int strategy,
                                        List<Integer> primeCandidates,
                                        Boolean parallel) {
    var uri = UriComponentsBuilder
      .fromPath(CHECK_IF_PRIME_LIST)
      .queryParam("strategy", strategy)
      .queryParam("primeCandidates", WebUtils
                  .list2String(primeCandidates))
      .queryParam("parallel", parallel)
      .build().toUriString();
    ...
```

The Structure & Functionality of PCProxy

- PCProxy abstracts low-level details of remote method invocations using HTTP

```
@Component
public class PCProxy {
   ...
   public List<Integer> checkIfPrimeList(int strategy,
                                         List<Integer> primeCandidates,
                                         Boolean parallel) {
     var uri = UriComponentsBuilder
       .fromPath(CHECK_IF_PRIME_LIST)
       .queryParam("strategy", strategy)
       .queryParam("primeCandidates", WebUtils
                   .list2String(primeCandidates))
       .queryParam("parallel", parallel)
       .build().toUriString();
     ...
```

Convert the List of Integers into a String of comma-separated integers encodings

e.g., "218315,42673259,212438568,147483,5489341,81931857,..."

The Structure & Functionality of PCProxy

- PCProxy abstracts low-level details of remote method invocations using HTTP

```
@Component
public class PCProxy {
   ...
   public List<Integer> checkIfPrimeList(int strategy,
                                         List<Integer> primeCandidates,
                                         Boolean parallel) {
     ...
     return WebUtils.makeGetRequestList(mRestTemplate,
                                        uri,
                                        Integer[].class);
   }
```

Make one HTTP GET call to the server that contains all encoded primeCandidates

e.g., http://localhost:8081/checkIfPrimeList?...primeCandidates=218315,147483,...&parallel=true

The Structure & Functionality of PCProxy

- PCProxy abstracts low-level details of remote method invocations using HTTP

```
@Component
public class PCProxy {
   ...
   public List<Integer> checkIfPrimeList(int strategy,
                                         List<Integer> primeCandidates,
                                         Boolean parallel) {
     ...
     return WebUtils.makeGetRequestList(mRestTemplate,
                                        uri,
                                        Integer[].class);
   }
```

A more common way to pass List params is in the body of an HTTP POST request

See www.baeldung.com/cs/http-get-vs-post

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