The MathServices App Case Study: Primality Microservice Structure & Functionality

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

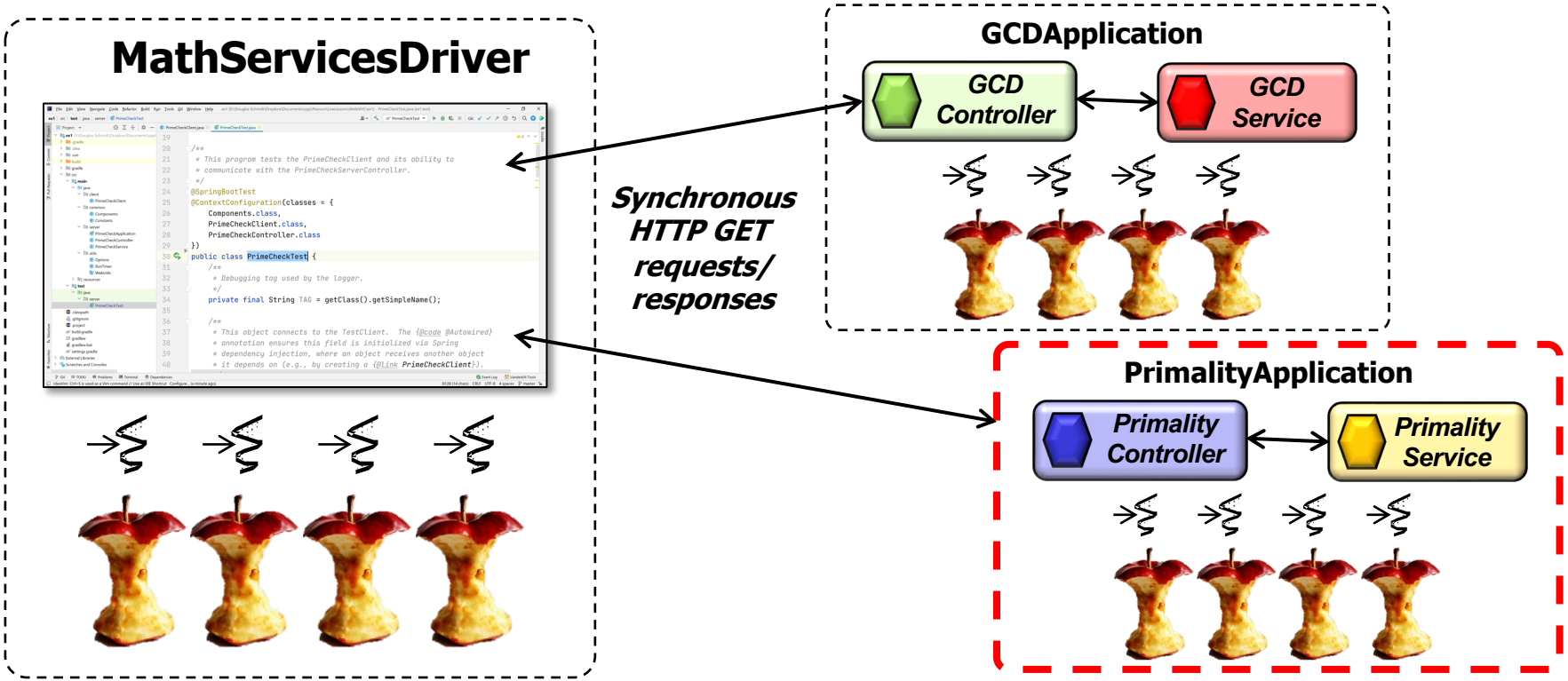
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of the PrimalityController/Primality Service microservice & how it applies Java structured concurrency



This microservice uses the Executors VirtualThreadPerTaskExecutor model

Structure & Functionality of the PrimalityController

Structure & Functionality of the PrimalityController

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

```
@RestController
public class PrimalityController {
  @Autowired
  PrimalityService mService;

  @GetMapping(CHECK_PRIMALITY_LIST)
  public List<PrimeResult>
  checkPrimalities
    (@RequestParam List<Integer>
       primeCandidatres) {
    mService
      .checkPrimalities
        (primeCandidatres);
  }
}
```

See WebMVC/ex3/server/src/main/java/edu/vandy/mathservices/microservices/primality

Structure & Functionality of the PrimalityController

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

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

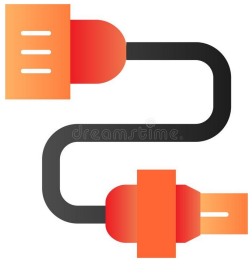
```
@RestController
public class PrimalityController {
   @Autowired
   PrimalityService mService;

   @GetMapping(CHECK_PRIMALITY_LIST)
   public List<PrimeResult>
   checkPrimalities
      (@RequestParam List<Integer>
         primeCandidatres) {
      mService
         .checkPrimalities
            (primeCandidatres);
   }
}
```

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

Structure & Functionality of the PrimalityController

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



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

```
@RestController
public class PrimalityController {
   @Autowired
   PrimalityService mService;

   @GetMapping(CHECK_PRIMALITY_LIST)
   public List<PrimeResult>
   checkPrimalities
      (@RequestParam List<Integer>
         primeCandidatres) {
      mService
         .checkPrimalities
            (primeCandidatres);
   }
}
```

See www.baeldung.com/spring-autowire

Structure & Functionality of the PrimalityController

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

```
@RestController
public class PrimalityController {
   @Autowired
   PrimalityService mService;

   @GetMapping(CHECK_PRIMALITY_LIST)
   public List<PrimeResult>
   checkPrimalities
      (@RequestParam List<Integer>
         primeCandidatres) {
     mService
        .checkPrimalities
           (primeCandidatres);
   }
}
```

This method just forwards to the PrimalityService method & returns the results back

Structure & Functionality of the PrimalityController

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

```
@RestController
public class PrimalityController {
   @Autowired
   PrimalityService mService;

   @GetMapping(CHECK_PRIMALITY_LIST)
   public List<PrimeResult>
   checkPrimalities
      (@RequestParam List<Integer>
         primeCandidatres) {
      mService
         .checkPrimalities
            (primeCandidatres);
   }
}
```

This annotation maps HTTP GET requests onto endpoint handler methods

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

Structure & Functionality of the PrimalityController

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

This string is used to automatically identify the endpoint handler methods from incoming GET requests

```
@RestController
public class PrimalityController {
  @Autowired
  PrimalityService mService;

  @GetMapping(CHECK_PRIMALITY_LIST)
  public List<PrimeResult>
  checkPrimalities
    (@RequestParam List<Integer>
       primeCandidatres) {
    mService
      .checkPrimalities
        (primeCandidatres);
  }
}
```

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

Structure & Functionality of the PrimalityController

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

```
@RestController
public class PrimalityController {
   @Autowired
   PrimalityService mService;

   @GetMapping(CHECK_PRIMALITY_LIST)
   public List<PrimeResult>
   checkPrimalities
      (@RequestParam List<Integer>
         primeCandidatres) {
      mService
         .checkPrimalities
            (primeCandidatres);
   }
}
```

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

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

Structure & Functionality of the PrimalityService

Structure & Functionality of the PrimalityService

- The PrimalityService class defines implementation methods that are called by the PrimalityController

```
@Service
public class PrimalityService
  public List<PrimeResult> checkPrimalities
    (List<Integer> primeCandidates) {
    List<Future<PrimeResult>> results;

    try (var executor = Executors
      .newVirtualThreadPerTaskExecutor()){
      results = getFutures
        (primeCandidates, executor);
    }

    return convertFutures(results);
  } ...
```

See ex3/server/src/main/java/edu/vandy/mathservices/microservices/primality/PrimalityService.java

Structure & Functionality of the PrimalityService

- The PrimalityService class defines implementation methods that are called by the PrimalityController

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

```
@Service
public class PrimalityService
  public List<PrimeResult> checkPrimalities
    (List<Integer> primeCandidates) {
    List<Future<PrimeResult>> results;

    try (var executor = Executors
      .newVirtualThreadPerTaskExecutor()){
      results = getFutures
        (primeCandidates, executor);
    }

    return convertFutures(results);
  } ...
```

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

Structure & Functionality of the PrimalityService

- The PrimalityService class defines implementation methods that are called by the PrimalityController

```
@Service
public class PrimalityService
  public List<PrimeResult> checkPrimalities
    (List<Integer> primeCandidates) {
    List<Future<PrimeResult>> results;

    try (var executor = Executors
      .newVirtualThreadPerTaskExecutor()){
      results = getFutures
        (primeCandidates, executor);
    }

    return convertFutures(results);
  } ...
```

Concurrently compute the primality of the primeCandidates & return PrimeResult objects

Structure & Functionality of the PrimalityService

- The PrimalityService class defines implementation methods that are called by the PrimalityController

```
@Service
public class PrimalityService
  public List<PrimeResult> checkPrimalities
    (List<Integer> primeCandidates) {
    List<Future<PrimeResult>> results;

    try (var executor = Executors
      .newVirtualThreadPerTaskExecutor()){
      results = getFutures
        (primeCandidates, executor);
    }

    return convertFutures(results);
  } ...
```

Use Java structured concurrency to perform computations in parallel

See java/util/concurrent/Executors.html#newVirtualThreadPerTaskExecutor

Structure & Functionality of the PrimalityService

- The PrimalityService class defines implementation methods that are called by the PrimalityController

```
@Service
public class PrimalityService
  public List<PrimeResult> checkPrimalities
    (List<Integer> primeCandidates) {
    List<Future<PrimeResult>> results;

    try (var executor = Executors
      .newVirtualThreadPerTaskExecutor()){
      results = getFutures
        (primeCandidates, executor);
    }

    return convertFutures(results);
  } ...
```

Return the results back to the client

End of the MathServices App Case Study: GCD MicroService Structure & Functionality

The MathServices App Case Study: Implementing the Primality Microservice

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

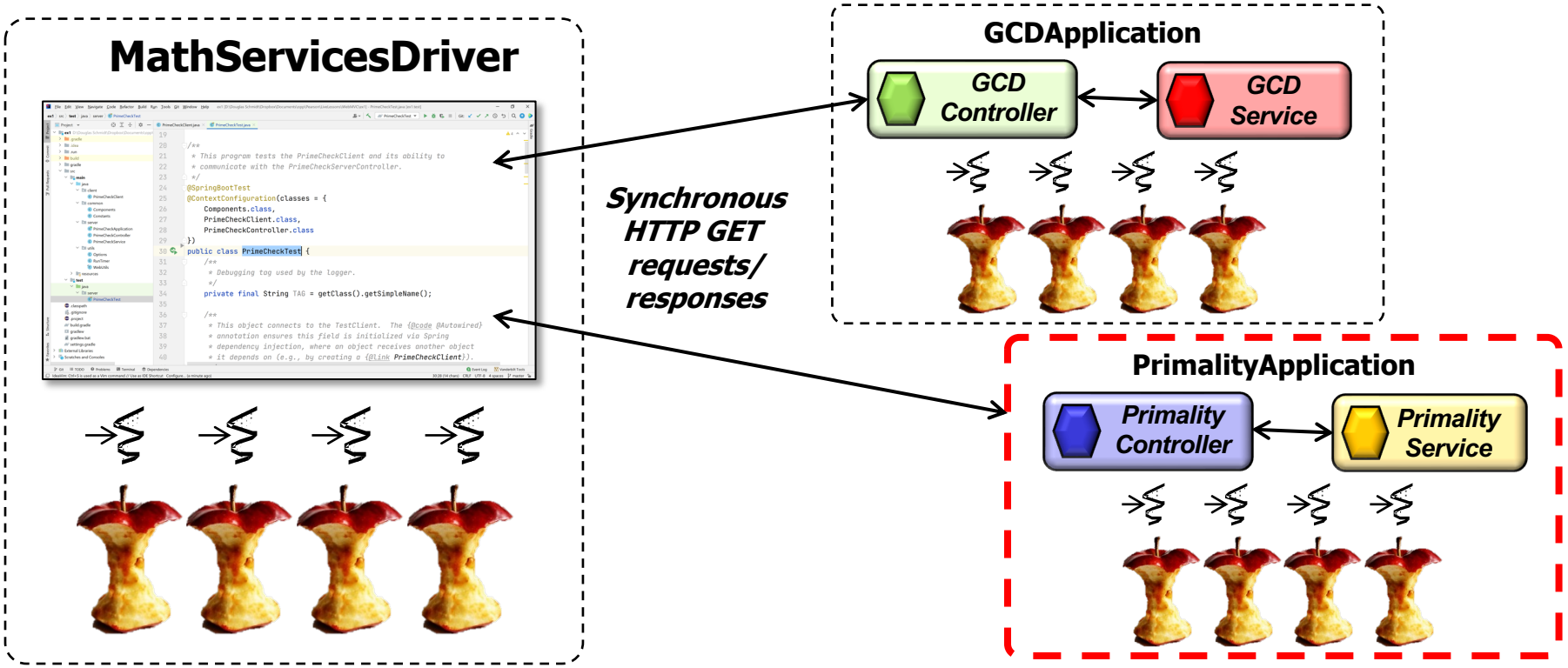
Institute for Software Integrated Systems

Vanderbilt University

Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

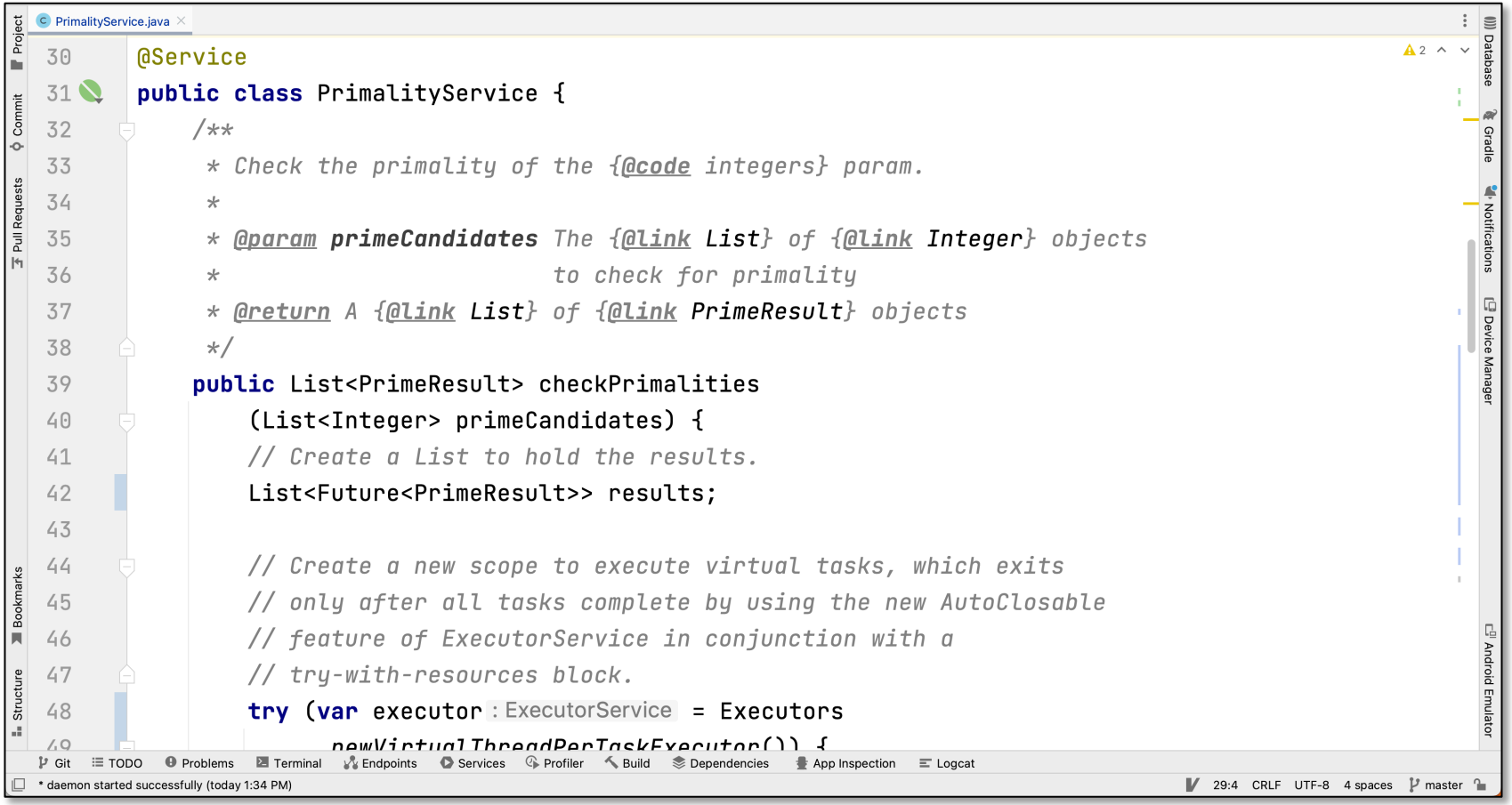
- Understand the concurrent implementation of the PrimalityController & PrimalityService classes that run in the PrimalityApplication microservice



The focus is on the Java Executors VirtualThreadPerTaskExecutor model

Implementing the Primality Application Microservice

Implementing the PrimalityApplication Microservice



See github.com/douglascraigschmidt/LiveLessons/tree/master/WebMVC/ex3

End of the MathServices App Case Study: Implementing the Primality Microservice