

# The MathServices App Case Study: GCD Microservice Structure & Functionality

**Douglas C. Schmidt**

**[d.schmidt@vanderbilt.edu](mailto:d.schmidt@vanderbilt.edu)**

**[www.dre.vanderbilt.edu/~schmidt](http://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 GCDController/GCDService microservice implementation & how it applies Java parallel streams

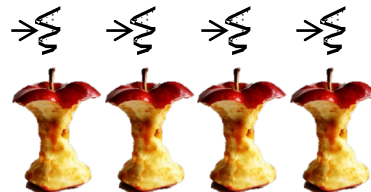
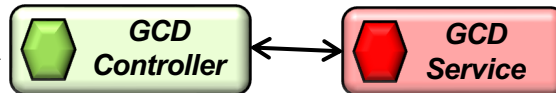
## MathServicesDriver

```
1 // This program tests the PrimeCheckClient and its ability to
2 // communicate with the PrimeCheckServerController.
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

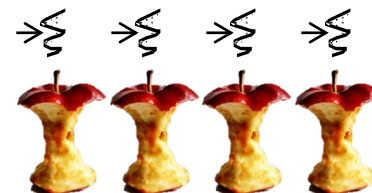
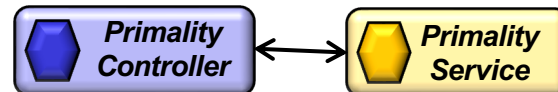
`@SpringBootTest`  
`@ContextConfiguration(classes = {`  
 `Components.class,`  
 `PrimeCheckClient.class,`  
 `PrimeCheckController.class`  
`})`  
`public class PrimeCheckTest {`  
 `//`  
 `// Debugging tag used by the logger.`  
 `private final String TAG = getClass().getSimpleName();`  
`}`

*HTTP GET  
requests/  
responses*

## GCDApplication



## PrimalityApplication



---

# Structure & Functionality of the GCDCController

# Structure & Functionality of the GCDController

---

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

```
@RestController
public class GCDController {
    @Autowired
    GCDService mService;

    @GetMapping("computeGCDList")
    public List<GCDResult>
    computeGCDs
        (@RequestParam List<Integer>
         integers) {
        mService
            .computeGCDs(integers);
    }
}
```

---

See [mathservices/microservices/gcd/GCDController.java](https://github.com/mathservices/microservices/gcd/GCDController.java)

# Structure & Functionality of the GCDController

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

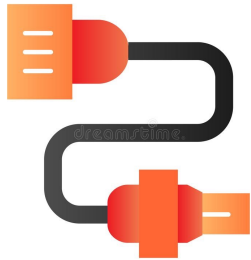
```
@RestController
public class GCDController {
    @Autowired
    GCDService mService;

    @GetMapping("computeGCDList")
    public List<GCDResult>
    computeGCDs
        (@RequestParam List<Integer>
         integers) {
        mService
            .computeGCDs(integers);
    }
}
```

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

# Structure & Functionality of the GCDController

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



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

```
@RestController
public class GCDController {
    @Autowired
    GCDService mService;

    @GetMapping("computeGCDList")
    public List<GCDResult>
    computeGCDs
        (@RequestParam List<Integer>
         integers) {
        mService
            .computeGCDs(integers);
    }
}
```

# Structure & Functionality of the GCDController

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

```
@RestController
public class GCDController {
    @Autowired
    GCDService mService;

    @GetMapping("computeGCDList")
    public List<GCDResult>
        computeGCDs
            (@RequestParam List<Integer>
             integers) {
        mService
            .computeGCDs(integers);
    }
}
```

*This method just forwards  
to the GCDService method  
& returns the results back*

# Structure & Functionality of the GCDController

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

```
@RestController
public class GCDController {
    @Autowired
    GCDService mService;

    @GetMapping("computeGCDList")
    public List<GCDResult>
    computeGCDs
        (@RequestParam List<Integer>
         integers) {
        mService
            .computeGCDs(integers);
    }
}
```

*This annotation maps  
HTTP GET requests onto  
endpoint handler methods*

# Structure & Functionality of the GCDController

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

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

```
@RestController
public class GCDController {
    @Autowired
    GCDService mService;

    @GetMapping("computeGCDList")
    public List<GCDResult>
    computeGCDs
        (@RequestParam List<Integer>
         integers) {
        mService
            .computeGCDs(integers);
    }
}
```

# Structure & Functionality of the GCDController

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

```
@RestController
public class GCDController {
    @Autowired
    GCDService mService;

    @GetMapping("computeGCDList")
    public List<GCDResult>
    computeGCDs
        (@RequestParam List<Integer>
         integers) {
        mService
            .computeGCDs(integers);
    }
}
```

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

---

# Structure & Functionality of the GCDSERVICE

# Structure & Functionality of the GCDSERVICE

---

- The GCDSERVICE class defines implementation methods that are called by the GCDController

```
@Service
public class GCDSERVICE
    public List<GCDResult> computeGCDs
        (List<Integer> integers) {
            return StreamSupport
                .stream(new ListSplitter(integers),
                    true)

                .map(GCDSERVICE::computeGCD)

                .toList();
        }
    ...
```

---

See [mathservices/microservices/gcd/GCDSERVICE.java](https://mathservices/microservices/gcd/GCDSERVICE.java)

# Structure & Functionality of the GCDSERVICE

- The GCDSERVICE class defines implementation methods that are called by the GCDController

**@Service**

```
public class GCDSERVICE
    public List<GCDResult> computeGCDs
        (List<Integer> integers) {
    return StreamSupport
        .stream(new ListSpliterator(integers),
            true)

        .map(GCDSERVICE::computeGCD)

        .toList();
    }
    ...
```

*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](http://www.baeldung.com/spring-component-repository-service)

# Structure & Functionality of the GCDSERVICE

- The GCDSERVICE class defines implementation methods that are called by the GCDController

```
@Service
public class GCDSERVICE
    public List<GCDResult> computeGCDs
        (List<Integer> integers) {
            return StreamSupport
                .stream(new ListSplitter(integers),
                    true)

                .map(GCDSERVICE::computeGCD)

                .toList();
        }
    ...
```

*Concurrently compute the GCD of the integers param & return GCDResult objects*

# Structure & Functionality of the GCDSERVICE

- The GCDSERVICE class defines implementation methods that are called by the GCDController

```
@Service
public class GCDSERVICE
    public List<GCDResult> computeGCDs
        (List<Integer> integers) {
            return StreamSupport
                .stream(new ListSplitter(integers),
                    true)
                .map(GCDSERVICE::computeGCD)
                .toList();
        }
    ...
```

*Use Java parallel streams  
to perform all the GCD  
computations in parallel*



---

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