Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of the GCDController/GCDService microservice implementation & how it applies Java parallel streams

![Diagram of MathServicesDriver with GCDApplication, GCDController, GCDService, PrimalityApplication, PrimalityController, and PrimalityService nodes connected by HTTP GET requests/responses.](image-url)
Structure & Functionality of the GCDController
Structure & Functionality of the GCDController

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

```java
@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`
Structure & Functionality of the GCDController

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

```java
@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.

See [www.baeldung.com/spring-controller-vs-restcontroller](www.baeldung.com/spring-controller-vs-restcontroller)
Structure & Functionality of the GCDController

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

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

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

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

See [www.baeldung.com/spring-autowire](http://www.baeldung.com/spring-autowire)
Structure & Functionality of the GCDController

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

```java
@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

```java
@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.

See [www.baeldung.com/spring-new-requestmapping-shortcuts](http://www.baeldung.com/spring-new-requestmapping-shortcuts)
Structure & Functionality of the GCDController

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

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

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

This string is used to automatically identify the endpoint handler methods from incoming GET requests.
Structure & Functionality of the GCDController

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

```java
@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*

See [www.baeldung.com/spring-request-param](http://www.baeldung.com/spring-request-param)
Structure & Functionality of the GCDService
Structure & Functionality of the GCDService

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

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

See `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
Structure & Functionality of the GCDService

• The GCDService class defines implementation methods that are called by the GCDController

```java
@Service
public class GCDService {
    public List<GCDResult> computeGCDs(List<Integer> integers) {
        return StreamSupport.stream(new ListSpliterator(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

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

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

See [docs.oracle.com/javase/tutorial/collections/streams/parallelism.html](https://docs.oracle.com/javase/tutorial/collections/streams/parallelism.html)
End of the MathServices App
Case Study: GCD MicroService
Structure & Functionality