

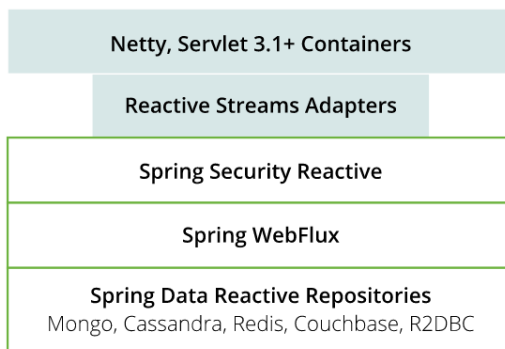
Spring Boot Reactive - Mr. RAGHU

<https://www.facebook.com/groups/thejavatemple>



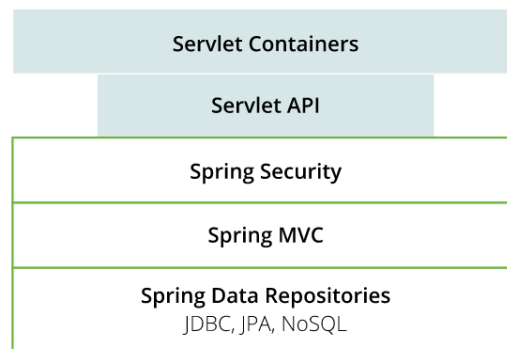
Reactive Stack

Spring WebFlux is a non-blocking web framework built from the ground up to take advantage of multi-core, next-generation processors and handle massive numbers of concurrent connections.

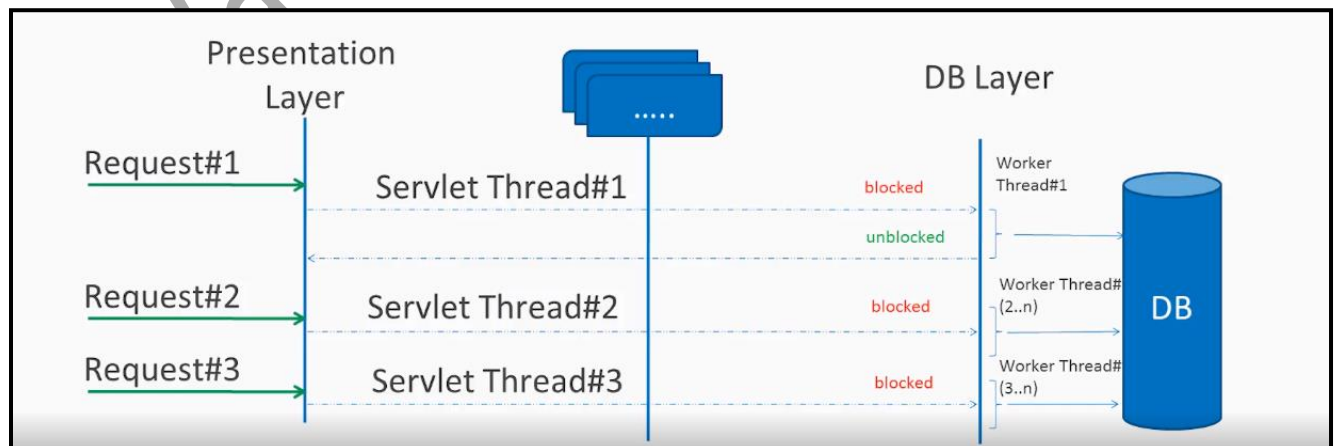


Servlet Stack

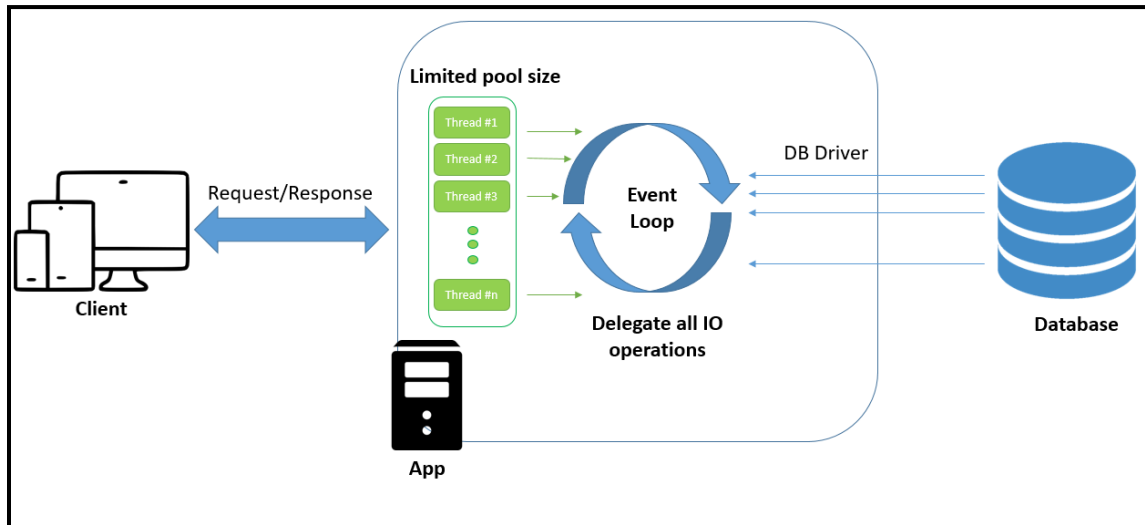
Spring MVC is built on the Servlet API and uses a synchronous blocking I/O architecture with a one-request-per-thread model.



Blocking:



Non Blocking:



Spring WebFlux

Spring WebFlux is the new reactive web framework that comes with Spring 5. It is not a replacement for Spring MVC but a fully non-blocking, Reactive Streams back pressure supporting framework that is run on servers like Netty, Jetty, Undertow, etc. Spring WebFlux majorly uses two publishers:

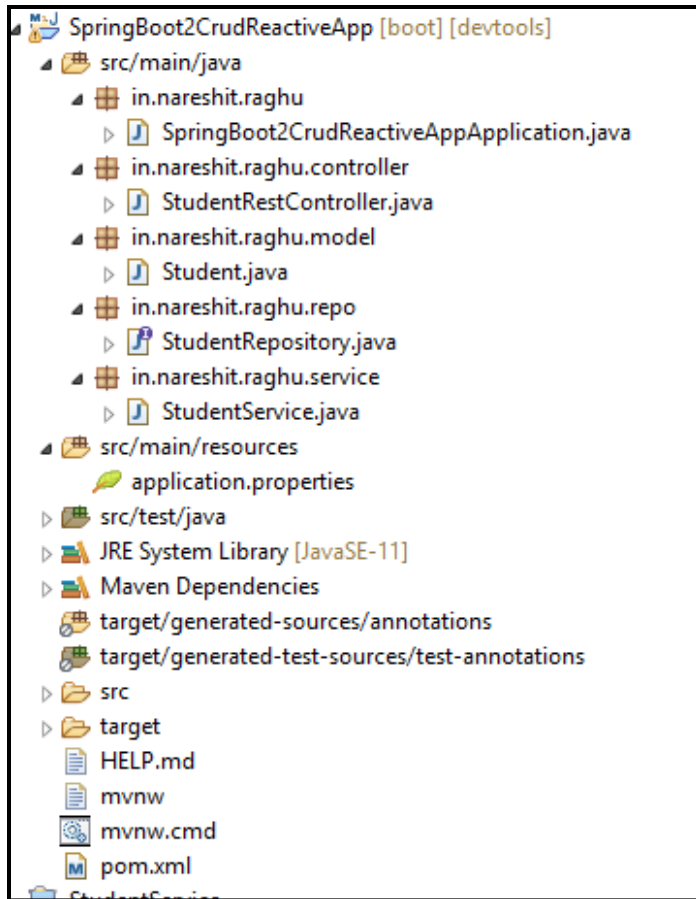
The Mono

Mono: Returns 0 or 1 element. The Mono API allows producing only one value. `Mono mono = Mono.just("Spring Framework");` `Mono mono = Mono.empty();` This is limited to no more than one element.

The Flux

Flux: Returns 0...N elements. The Flux can be endless, it can produce multiple values. Here, we have a static stream of the three elements. `Flux flux = Flux.just("One", "Two", "Three");`

Provider Application



pom.xml:-

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.3.6.RELEASE</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>in.nareshit.raghu</groupId>
  <artifactId>SpringBoot2CrudReactiveApp</artifactId>
  <version>1.0</version>
  <name>SpringBoot2CrudReactiveApp</name>
  <description>Demo project for Spring Boot</description>

  <properties>
```

```
<java.version>11</java.version>
</properties>

<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-mongodb-
reactive</artifactId>
    </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-
webflux</artifactId>
    </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
  </dependency>
  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
    <exclusions>
      <exclusion>
        <groupId>org.junit.vintage</groupId>
        <artifactId>junit-vintage-
engine</artifactId>
      </exclusion>
    </exclusions>
  </dependency>
  <dependency>
    <groupId>io.projectreactor</groupId>
    <artifactId>reactor-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>

<build>
  <plugins>
```

```
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-
plugin</artifactId>
        </plugin>
    </plugins>
</build>

</project>
```

application.properties :-

```
spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
spring.data.mongodb.database=nitdb
```

Model class:-

```
package in.nareshit.raghu.model;

import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;

@Data
@AllArgsConstructor
@RequiredArgsConstructor
@NoArgsConstructor
@Document
public class Student {

    @Id
    private String id;
    @NonNull
    private String name;
    @NonNull
    private Double fee;

}
```

Repository Interface:-

```
package in.nareshit.raghu.repo;

import
org.springframework.data.mongodb.repository.ReactiveMongoRepository;

import in.nareshit.raghu.model.Student;

public interface StudentRepository extends
ReactiveMongoRepository<Student, String> {

}
```

Service class:-

```
package in.nareshit.raghu.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import in.nareshit.raghu.model.Student;
import in.nareshit.raghu.repo.StudentRepository;
import reactor.core.publisher.Flux;
import reactor.core.publisher.Mono;

@Service
public class StudentService {
    @Autowired
    private StudentRepository repo;

    public Mono<Student> save(Student student) {
        return repo.save(student);
    }

    public Mono<Student> getOne(String id) {
        return repo.findById(id).switchIfEmpty(Mono.empty());
    }

    public Flux<Student> findAll() {
        return repo.findAll().switchIfEmpty(Flux.empty());
    }
}
```

```
    }  
  
    public Mono<Void> delete(String id) {  
        return repo.deleteById(id);  
    }  
}
```

Controller class:-

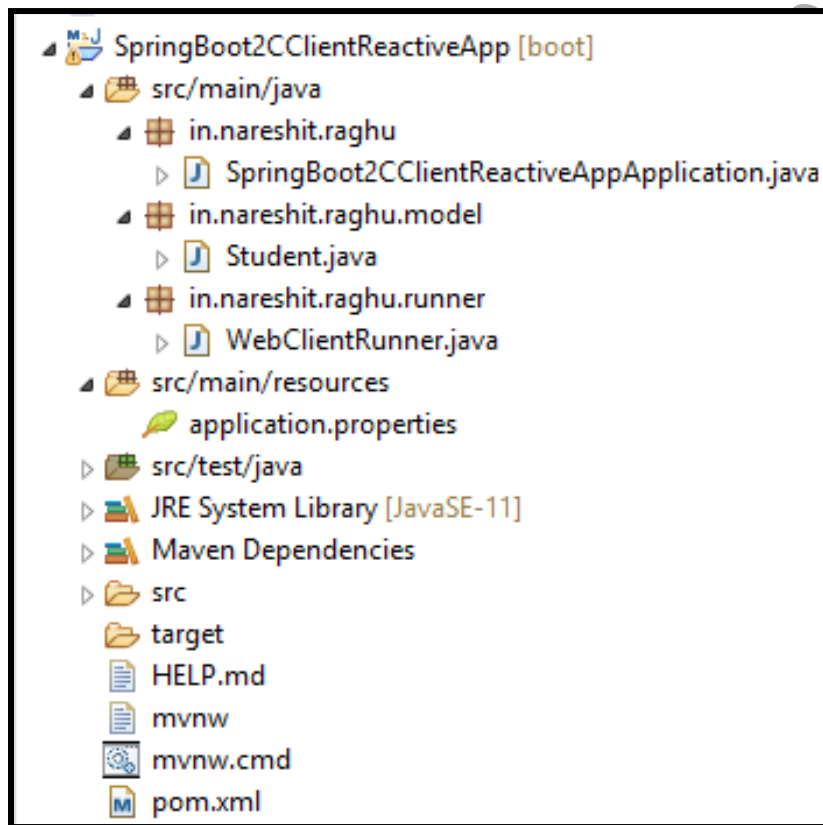
```
package in.nareshit.raghu.controller;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.DeleteMapping;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
import in.nareshit.raghu.model.Student;  
import in.nareshit.raghu.service.StudentService;  
import reactor.core.publisher.Flux;  
import reactor.core.publisher.Mono;  
  
@RestController  
@RequestMapping("/student")  
public class StudentRestController {  
    @Autowired  
    private StudentService service;  
  
    @PostMapping  
    public Mono<Student> save(@RequestBody Student student) {  
        return service.save(student);  
    }  
  
    @GetMapping("/{id}")  
    public Mono<Student> getOne(@PathVariable String id) {  
        return service.getOne(id);  
    }  
  
    @GetMapping  
    public Flux<Student> findAll() {  
        return service.findAll();  
    }  
}
```

```
}

@DeleteMapping("/{id}")
public Mono<Void> delete(@PathVariable String id) {
    return service.delete(id);
}

}
```

Client Application



pom.xml:-

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```



```
xsi:schemaLocation="http://maven.apache.org/POM/4.0
.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-
parent</artifactId>
    <version>2.3.6.RELEASE</version>
    <relativePath/> <!-- lookup parent from
repository -->
  </parent>
  <groupId>in.nareshit.raghu</groupId>
  <artifactId>SpringBoot2CCClientReactiveApp</artifact
Id>
  <version>1.0</version>
  <name>SpringBoot2CCClientReactiveApp</name>
  <description>Demo project for Spring
Boot</description>

  <properties>
    <java.version>11</java.version>
  </properties>

  <dependencies>
    <dependency>

      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-
webflux</artifactId>
    </dependency>
    <dependency>

      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-
test</artifactId>
      <scope>test</scope>
      <exclusions>
        <exclusion>
```

```
<groupId>org.junit.vintage</groupId>
    <artifactId>junit-vintage-
engine</artifactId>
    </exclusion>
</exclusions>
</dependency>
<dependency>
    <groupId>io.projectreactor</groupId>
    <artifactId>reactor-test</artifactId>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
</dependency>
</dependencies>

<build>
    <plugins>
        <plugin>

            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-
plugin</artifactId>
        </plugin>
    </plugins>
</build>
</project>
```

Model class:-

```
package in.nareshit.raghu.model;

import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;
```

```
@Data
@NoArgsConstructor
@RequiredArgsConstructor
public class Student {

    private String id;
    @NotNull
    private String name;
    @NotNull
    private Double fee;

}
```

Runner class:-

```
package in.nareshit.raghu.runner;

import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;
import org.springframework.web.reactive.function.client.WebClient;

import reactor.core.publisher.Mono;

@Component
public class WebClientRunner implements
CommandLineRunner {

    @Override
    public void run(String... args) throws Exception {

        WebClient client =
WebClient.create("http://localhost:8080");

        //Get All
        /*
```

```
Flux<Student> flux = client
    .get()
    .uri("/student")
    .retrieve()
    .bodyToFlux(Student.class);

flux.doOnNext(System.out::println).blockLast();
*/
/*
Mono<Student> mono = client
    .get()

    .uri("/student/5fc3debaa3e8810895545f74")
        .retrieve()
        .bodyToMono(Student.class);

mono.subscribe(System.out::println);
*/

/*
Mono<Student> mono =client.post()
    .uri("/student")
    .body(Mono.just(new Student("JJ",2500.0)),
Student.class)
    .retrieve()
    .bodyToMono(Student.class);

mono.subscribe(System.out::println);
*/
Mono<Void> mono =client.delete()

    .uri("/student/5fc3e0aca3e8810895545f75")
        .retrieve()
        .bodyToMono(Void.class);

mono.subscribe(System.out::println);
}

}
```

application.properties:-

`server.port=3000`

<https://www.facebook.com/groups/thejavatemple>

javabyraghu@gmail.com

THANK YOU!!