This document contains continuation of Functional web in webflux for MovieReview Application, unit testing for functional web, bean validation in functional web, custom global exception handler in functional web. Handling resources not found in functional web , Movie Service which talks with MovieInfo and MovieReview Service and Exception service to service communication.

Post method with functional style.

```
    @Bean
    public RouterFunction<ServerResponse> reviewsRoute(ReviewHandler reviewHandler){

        return route()
                .GET( pattern: "/v1/helloworld",(request -> ServerResponse.ok().bodyValue("helloworld")))
                .POST( pattern: "/v1/reviews",request -> reviewHandler.addReview(request))
                .build();

    }
}
```

Two changes here are passing the handler object as parameter in the bean method.

Handler class with component annotation

```
ReviewRouter.java  ReviewHandler.java  ReactiveCrudRepository.java
@Component
public class ReviewHandler {

    private ReviewReactiveRepository reviewReactiveRepository;

    public ReviewHandler(ReviewReactiveRepository reviewReactiveRepository) {
        this.reviewReactiveRepository = reviewReactiveRepository;
    }

    public Mono<ServerResponse> addReview(ServerRequest request) {

        request.bodyToMono(Review.class)
                .flatMap(review -> {
                    return  reviewReactiveRepository.save(review);
                });


    }
}
```

We can change it to method reference

```
21          request.bodyToMono(Review.class)
22                  .flatMap(reviewReactiveRepository::save);
23
```

Here it is writing Mono of object instead of mono of server response.

```
17      }
18
19  @   public Mono<ServerResponse> addReview(ServerRequest request) {
20
21          return request.bodyToMono(Review.class)
22                  .flatMap(reviewReactiveRepository::save);
23
24      }
25  }
```

Required type: Mono <ServerResponse>
Provided: Mono <Object>
no instance(s) of type variable(s) exist so that Review conforms to ServerResponse
inference variable R has incompatible bounds: equality constraints: ServerResponse lower bounds: Review

```
0  @    public Mono<ServerResponse> addReview(ServerRequest request) {
1
2           return request.bodyToMono(Review.class)
3                   .flatMap(reviewReactiveRepository::save)
4                   .flatMap(savedReview -> {
5                       ServerResponse.status(HttpStatus.CREATED)
6                               .bodyValue(savedReview);
7                   });
8
9       }
0   }
```

After replacing with method reference

```
@   public Mono<ServerResponse> addReview(ServerRequest request) {

        return request.bodyToMono(Review.class)
                .flatMap(reviewReactiveRepository::save)
                .flatMap(ServerResponse.status(HttpStatus.CREATED)::bodyValue);

    }
}
```

Integration test:

```java
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
@ActiveProfiles("test")
@AutoConfigureWebTestClient
public class ReviewsIntgTest {

    @Autowired
    WebTestClient webTestClient;

    @Autowired
    ReviewReactiveRepository reviewReactiveRepository;

    static  String REVIEWS_URL = "/v1/reviews";

    @BeforeEach
    void setUp() {

        var reviewsList = List.of(
                new Review( reviewId null,  movieInfoId 1L,  comment:"Awesome Movie",  rating:9.0),
                new Review( reviewId null,  movieInfoId 1L,  comment:"Awesome Movie1",  rating:9.0),
                new Review( reviewId null,  movieInfoId 2L,  comment:"Excellent Movie",  rating:8.0));
        reviewReactiveRepository.saveAll(reviewsList)
                .blockLast();
    }
```

```java
    @AfterEach
    void tearDown() {
        reviewReactiveRepository.deleteAll().block();
    }
}
```

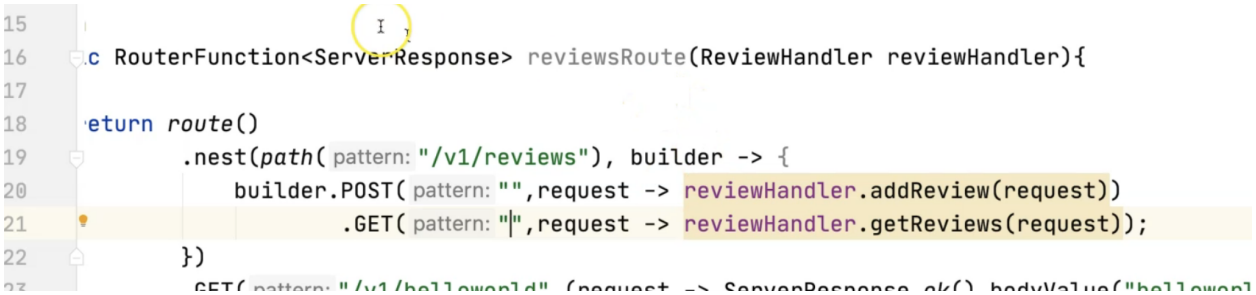
```java

    @Test
    void addReview() {

        //given
        var review = new Review( reviewId null,  movieInfoId 1L,  comment:"Awesome Movie",  rating:9.0);

        //when

        webTestClient
                .post() WebTestClient.RequestBodyUriSpec
                .uri(REVIEWS_URL) WebTestClient.RequestBodySpec
                .bodyValue(review) WebTestClient.RequestHeadersSpec<capture of ?>
                .exchange() WebTestClient.ResponseSpec
                .expectStatus() StatusAssertions
                .isCreated() WebTestClient.ResponseSpec
                .expectBody(Review.class) WebTestClient.BodySpec<Review, capture of ?>
                .consumeWith(movieInfoEntityExchangeResult -> {

                    var savedReview = movieInfoEntityExchangeResult.getResponseBody();
                    assert savedReview!=null;
                    assert savedReview.getReviewId()!=null;
                });

```

Get method with functional style.

Line number 20

Integration testing for the above method is assignment

Nest function used for grouping the same endpoint with different methods as shown below.



Run the integration test and check all are posing if not passing something wrong with our changes. That is usefulness and fastest way of doing testing.

Put method with functional style:

Line number 22

```java
@    public Mono<ServerResponse> updateReview(ServerRequest request) {

        var reviewId = request.pathVariable( name: "id");

        var existingReview = reviewReactiveRepository.findById(reviewId);

        return existingReview
                .flatMap(review -> request.bodyToMono(Review.class)
                        .map(reqReview -> {
                            review.setComment(reqReview.getComment());
                            review.setRating(reqReview.getRating());
                            return review;
                        })
                        .flatMap(reviewReactiveRepository::save)
                        .flatMap(savedReview -> ServerResponse.ok().bodyValue(savedReview))
                );
    }
}

```

```
/System/Volumes/Data/Dilip/udemy/reactive-spring-webflux/course-codebase/reactive-spring-webflux » curl -i \
ps-MacBook-Pro
\ -d '{"reviewId":1, "movieInfoId":1, "comment": "Excellent Movie", "rating":8.0}' \
\ -H "Content-Type: application/json" \
\ -X POST http://localhost:8081/v1/reviews
HTTP/1.1 201 Created
Content-Type: application/json
Content-Length: 73

{"reviewId":"1","movieInfoId":1,"comment":"Excellent Movie","rating":8.0}%
```

```
/System/Volumes/Data/Dilip/udemy/reactive-spring-webflux/course-codebase/reactive-spring-webflux » curl -i \
\ -d '{"reviewId":1, "movieInfoId":1, "comment": "Excellent Movie Update", "rating":8.5}' \
\ -H "Content-Type: application/json" \
\ -X PUT http://localhost:8081/v1/reviews/1
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 80

{"reviewId":"1","movieInfoId":1,"comment":"Excellent Movie Update","rating":8.5}%
```

Update reflected everywhere.

```
/System/Volumes/Data/Dilip/udemy/reactive-spring-webflux/course-codebase/reactive-spring-webflux » curl -i http://localhost:8081/v1/reviews
HTTP/1.1 200 OK
transfer-encoding: chunked
Content-Type: application/json

[{"reviewId":"612f53cd80720e6ae6bb2809","movieInfoId":1,"comment":"Awesome Movie","rating":9.0},{"reviewId":"612f566db85b70267793973f","movieInfoId":1,"comment":"Awesome Movie","rating":9.0},{"reviewId":"1","movieInfoId":1,"comment":"Excellent Movie Update","rating":8.5}]%
```

Delete method with functional style:

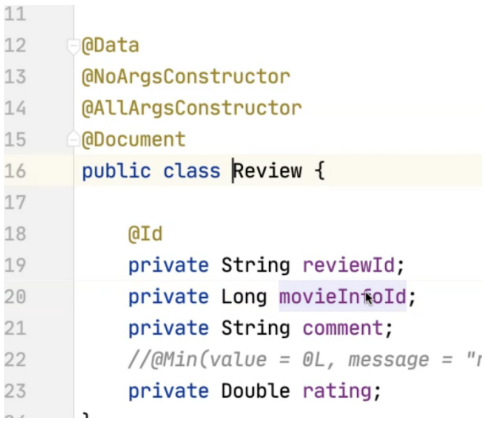
```
18          return route()
19                  .nest(path( pattern: "/v1/reviews"), builder -> {
20                      builder.POST( pattern: "",request -> reviewHandler.addReview(request))
21                              .GET( pattern: "",request -> reviewHandler.getReviews(request))
22                      .PUT( pattern: "/{id}", request -> reviewHandler.updateReview(request))
23                      .DELETE( pattern: "/{id}", request -> reviewHandler.deleteReview(request));
24                  })
```

Line number 23

```
95
96 @    public Mono<ServerResponse> deleteReview(ServerRequest request) {
97
98          var reviewId = request.pathVariable( name: "id");
99
100         var existingReview = reviewReactiveRepository.findById(reviewId);
101
102         return existingReview
103                 .flatMap(review -> reviewReactiveRepository.deleteById(reviewId)
104                 .then(ServerResponse.noContent().build()));
105
106     }
107 }
108
```

```
{"reviewId":"1","movieInfoId":1,"comment":"Excellent Movie","rating":8.0}%
--------------------------------------------------------------------------------------------------------------------------------
/System/Volumes/Data/Dilip/udemy/reactive-spring-webflux/course-codebase/reactive-spring-webflux » curl -i -X DELETE http://
localhost:8081/v1/reviews/1
HTTP/1.1 204 No Content
```

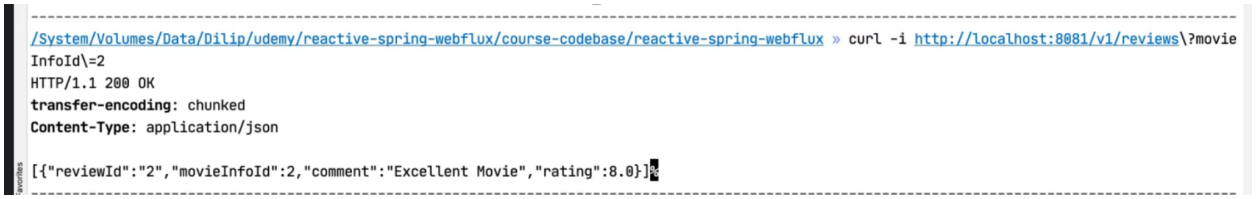
Based on movieInfoId we should retrieve all the reviews.



```
11
12  @Data
13  @NoArgsConstructor
14  @AllArgsConstructor
15  @Document
16  public class Review {
17
18      @Id
19      private String reviewId;
20      private Long movieInfoId;
21      private String comment;
22      //@Min(value = 0L, message = "I
23      private Double rating;
```

```
6
7   public interface ReviewReactiveRepository extends ReactiveMongoRepository<Review, String> {
8
9       Flux<Review> findReviewsByMovieInfoId(Long movieInfoId);
10  }
11
```

```
public Mono<ServerResponse> getReviews(ServerRequest request) {

    var movieInfoId = request.queryParam( name: "movieInfoId");

    if(movieInfoId.isPresent()){
        var reviewsFlux = reviewReactiveRepository.findReviewsByMovieInfoId(Long.valueOf(movieInfoId.get()));
        return buildReviewsResponse(reviewsFlux);
    }else {
        var reviewsFlux = reviewReactiveRepository.findAll();
        return buildReviewsResponse(reviewsFlux);
    }


}

private Mono<ServerResponse> buildReviewsResponse(Flux<Review> reviewsFlux) {
    return ServerResponse.ok().body(reviewsFlux, Review.class);
}
```



```
/System/Volumes/Data/Dilip/udemy/reactive-spring-webflux/course-codebase/reactive-spring-webflux » curl -i http://localhost:8081/v1/reviews\?movie
InfoId\=2
HTTP/1.1 200 OK
transfer-encoding: chunked
Content-Type: application/json

[{"reviewId":"2","movieInfoId":2,"comment":"Excellent Movie","rating":8.0}]%
```

Assignment: Integration Test Case for the Reviews Service in the ReviewsIntgTest class

30 minutes to complete | 103 student solutions

Write an integration test case for the below endpoints in the 1) Get All Reviews -> /v1/reviews 2) Update a Review using the PUT endpoint -> /v1/reviews/{id} 3) Delete a Review using the DELTE endpoint - > /v1/reviews/{id} 4) Get all reviews using the MovieInfoId -> /v1/reviews?movieInfoId=1

Unit test cases for functional web.

Because here no controller we need to remove it from @WebFluxTest @ContextConfiguration is important. Because it is a unit test case it is not automatically called the classes. We have to specify what classes are needed as part of unit testing.

```java
import org.springframework.test.web.reactive.server.WebTestClient;

@WebFluxTest
@ContextConfiguration(classes = {ReviewRouter.class, ReviewHandler.class})
@AutoConfigureWebTestClient
public class ReviewsUnitTest {

    @MockBean
    private ReviewReactiveRepository reviewReactiveRepository;

    @Autowired
    private WebTestClient webTestClient;

}
```

Unit testing for save (post call)

```java
@Test
void addReview() {
    //given
    var review = new Review( reviewId: null,  movieInfoId: 1L,  comment: "Awesome Movie",  rating: 9.0);

    when(reviewReactiveRepository.save(isA(Review.class)))
            .thenReturn(Mono.just(new Review( reviewId: "abc",  movieInfoId: 1L,  comment: "Awesome Movie",  rating: 9.0)));

    //when

    webTestClient
            .post() WebTestClient.RequestBodyUriSpec
            .uri(REVIEWS_URL) WebTestClient.RequestBodySpec
            .bodyValue(review) WebTestClient.RequestHeadersSpec<capture of ?>
            .exchange() WebTestClient.ResponseSpec
            .expectStatus() StatusAssertions
            .isCreated() WebTestClient.ResponseSpec
            .expectBody(Review.class) WebTestClient.BodySpec<Review, capture of ?>
            .consumeWith(movieInfoEntityExchangeResult -> {
                var savedReview = movieInfoEntityExchangeResult.getResponseBody();
                assert savedReview!=null;
                assert savedReview.getReviewId()!=null;
            });
}
```

Assignment: Unit Test Cases for the Reviews Service in the ReviewsUnitTest class

60 minutes to complete | 88 student solutions

Write an Unit test case for the below endpoints in the 1) Get All Reviews -> /v1/reviews 2) Update a Review using the PUT endpoint -> /v1/reviews/{id} 3) Delete a Review using the DELTE endpoint - > /v1/reviews/{id} 4) Get all reviews using the MovieInfoId -> /v1/reviews?movieInf

Bean Validation in Functional Web.

```java

@Data
@NoArgsConstructor
@AllArgsConstructor
@Document
public class Review {

    @Id
    private String reviewId;
    @NotNull(message = "rating.movieInfoId : must not be null")
    private Long movieInfoId;
    private String comment;
    @Min(value = 0L, message = "rating.negative : please pass a non-negative value")
    private Double rating;
}

```

Annotations are out of scope for functional web unlike @valid in @RequestBody

In the functional web we have to handle the below fashion.

doOnNext() is a side effect function where we can raise exceptions or do validations as shown above.

```java
package com.reactivespring.exception;

public class ReviewDataException extends RuntimeException {
    private String message;
    public ReviewDataException(String s) {
        super(s);
        this.message=s;
    }
}

```

Introduce validator in the handler class

```java
Component
ublic class ReviewHandler {


    @Autowired
    private Validator validator;

```

```java
public Mono<ServerResponse> addReview(ServerRequest request) {

    return request.bodyToMono(Review.class)
            .doOnNext(this::validate)
            .flatMap(reviewReactiveRepository::save)
            .flatMap(ServerResponse.status(HttpStatus.CREATED)::bodyValue);

}
```

```java
private void validate(Review review) {

    var constraintViolations = validator.validate(review);
    log.info("constraintViolations : {}" , constraintViolations);
    if(constraintViolations.size() > 0) {
        var errorMessage = constraintViolations
                .stream() Stream<ConstraintViolation<Review>>
                .map(ConstraintViolation::getMessage) Stream<String>
                .sorted()
                .collect(Collectors.joining( delimiter: ","));
        throw  new ReviewDataException(errorMessage);

    }
```

Unit testing for bean validation

```java
@Test
void addReview_validation() {
    //given
    var review = new Review( reviewId: null,  movieInfoId: null,  comment: "Awesome Movie",  rating: -9.0);

    when(reviewReactiveRepository.save(isA(Review.class)))
            .thenReturn(Mono.just(new Review( reviewId: "abc",  movieInfoId: 1L,  comment: "Awesome Movie",  rating: 9.0)));

    //when

    webTestClient
            .post() WebTestClient.RequestBodyUriSpec
            .uri(REVIEWS_URL) WebTestClient.RequestBodySpec
            .bodyValue(review) WebTestClient.RequestHeadersSpec<capture of ?>
            .exchange() WebTestClient.ResponseSpec
            .expectStatus() StatusAssertions
            .isBadRequest();
}
```

```
Run:  MoviesReviewServiceApplication  ReviewsUnitTest.addReview_validation
Tests failed: 1 of 1 test – 464 ms
Test Results 464 ms
ReviewsUnitTest 464 ms
addReview_validation() 464 ms
java.lang.AssertionError: Status expected:<400 BAD_REQUEST> but was:<500 INTERNAL_SERVER_ERROR>
Expected :400 BAD_REQUEST
Actual   :500 INTERNAL_SERVER_ERROR
Click to see difference>
```

We are expecting a bad request but we are receiving an internal server error because it is throwing a runtime exception.

custom global exception handler in functional web.

```java
@Component
@Slf4j
public class GlobalErrorHandler implements ErrorWebExceptionHandler {
    @Override
    public Mono<Void> handle(ServerWebExchange exchange, Throwable ex) {
        log.error("Exception message is {} ", ex.getMessage(), ex);
        DataBufferFactory dataBufferFactory = exchange.getResponse().bufferFactory();
        var errorMessage = dataBufferFactory.wrap(ex.getMessage().getBytes());

        if(ex instanceof ReviewDataException){
            exchange.getResponse().setStatusCode(HttpStatus.BAD_REQUEST);
            return  exchange.getResponse().writeWith(Mono.just(errorMessage));
        }
        exchange.getResponse().setStatusCode(HttpStatus.INTERNAL_SERVER_ERROR);
        return  exchange.getResponse().writeWith(Mono.just(errorMessage));
    }
}
```

Add this class in the unit test class.

```java
@WebFluxTest
@ContextConfiguration(classes = {ReviewRouter.class, ReviewHandler.class, GlobalErrorHandler.class})
@AutoConfigureWebTestClient
public class ReviewsUnitTest {
```

Refactor the unit test case

```java
    @Test
    void addReview_validation() {
        //given
        var review = new Review( reviewId: null,  movieInfoId: null,  comment: "Awesome Movie",  rating: -9.0);

        when(reviewReactiveRepository.save(isA(Review.class)))
                .thenReturn(Mono.just(new Review( reviewId: "abc",  movieInfoId: 1L,  comment: "Awesome Movie",  rating: 9.0)));

        //when

        webTestClient
                .post() WebTestClient.RequestBodyUriSpec
                .uri(REVIEWS_URL) WebTestClient.RequestBodySpec
                .bodyValue(review) WebTestClient.RequestHeadersSpec<capture of ?>
                .exchange() WebTestClient.ResponseSpec
                .expectStatus() StatusAssertions
                .isBadRequest() WebTestClient.ResponseSpec
        .expectBody(String.class) WebTestClient.BodySpec<String, capture of ?>
        .isEqualTo("rating.movieInfoId : must not be null,rating.negative : please pass a non-negative value ");
    }

}
```

Handling resources not found in functional web.

Resource NotFound(404) in Update Review using GlobalErrorHandler

```java
package com.reactivespring.exception;

public class ReviewNotFoundException extends RuntimeException{

    private String message;
    private Throwable ex;

    public ReviewNotFoundException( String message, Throwable ex) {
        super(message, ex);
        this.message = message;
        this.ex = ex;
    }

    public ReviewNotFoundException(String message) {
        super(message);
        this.message = message;
    }
}
```

Just add logic if review id not found

```java
    public Mono<ServerResponse> updateReview(ServerRequest request) {

        var reviewId = request.pathVariable( name: "id");

        var existingReview = reviewReactiveRepository.findById(reviewId)
                .switchIfEmpty(Mono.error(new ReviewNotFoundException("Review not found for the given Review id "+ reviewId)));

        return existingReview
                .flatMap(review -> request.bodyToMono(Review.class)
                        .map(reqReview -> {
                            review.setComment(reqReview.getComment());
                            review.setRating(reqReview.getRating());
                            return review;
                        })
                        .flatMap(reviewReactiveRepository::save)
                        .flatMap(savedReview -> ServerResponse.ok().bodyValue(savedReview))
                );
    }
```

Add this if statement in the global exception handler

```java
            return  exchange.getResponse().writeWith(Mono.just(errorMessage));
        }
        if(ex instanceof ReviewNotFoundException){
            exchange.getResponse().setStatusCode(HttpStatus.NOT_FOUND);
            return  exchange.getResponse().writeWith(Mono.just(errorMessage));
        }
        exchange.getResponse().setStatusCode(HttpStatus.INTERNAL_SERVER_ERROR);
        return  exchange.getResponse().writeWith(Mono.just(errorMessage));
    }
```

```
----------------------------------------------------------------------------------------------------------------------
/System/Volumes/Data/Dilip/udemy/reactive-spring-webflux/course-codebase/reactive-spring-webflux » curl -i \
\ -d '{"reviewId":1, "movieInfoId":1, "comment": "Excellent Movie Update", "rating":8.5}' \
\ -H "Content-Type: application/json" \
\ -X PUT http://localhost:8081/v1/reviews/100
HTTP/1.1 404 Not Found
content-length: 44
```

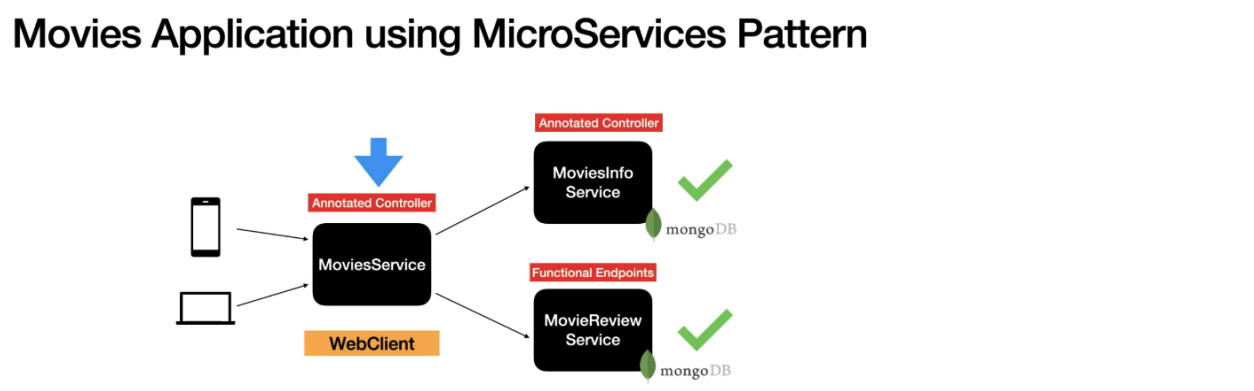
Another approach:

```
76
77 @    public Mono<ServerResponse> updateReview(ServerRequest request) {
78
79          var reviewId = request.pathVariable( name: "id");
80
81          var existingReview = reviewReactiveRepository.findById(reviewId);
82                  //.switchIfEmpty(Mono.error(new ReviewNotFoundException("Review not found for the given Review id "+ reviewId)));
83
84          return existingReview
85                  .flatMap(review -> request.bodyToMono(Review.class)
86                          .map(reqReview -> {
87                              review.setComment(reqReview.getComment());
88                              review.setRating(reqReview.getRating());
89                              return review;
90                          })
91                          .flatMap(reviewReactiveRepository::save)
92                          .flatMap(savedReview -> ServerResponse.ok().bodyValue(savedReview)))
93                  .switchIfEmpty(ServerResponse.notFound().build());
94      }
```

Here we gave not found at the last line. But disadvantage is no custom message.

```
----------------------------------------------------------------------------------------------------------------------------------
/System/Volumes/Data/Dilip/udemy/reactive-spring-webflux/course-codebase/reactive-spring-webflux » curl -i \
-d '{"reviewId":1, "movieInfoId":1, "comment": "Excellent Movie Update", "rating":8.5}' \
-H "Content-Type: application/json" \
-X PUT http://localhost:8081/v1/reviews/100
HTTP/1.1 404 Not Found
content-length: 0
```

Movie Service which talks with MovieInfo and MovieReview Service.



We use webclient here for connecting other services.

The application does not have dependency on DB.

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-validation'
    implementation 'org.springframework.boot:spring-boot-starter-webflux'

    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'

    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testImplementation 'io.projectreactor:reactor-test'
}
```

We are trying to get this data after connecting external service.

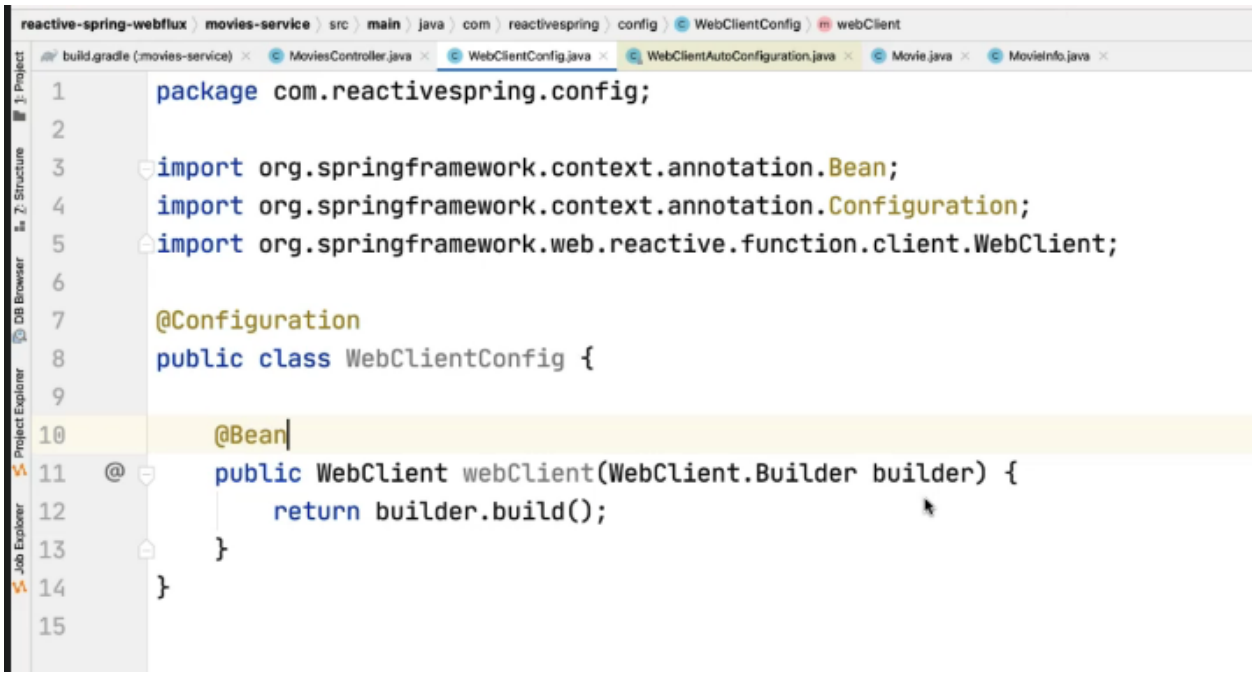
```
import ...

@Data
@NoArgsConstructor
@AllArgsConstructor
public class Movie {

    private MovieInfo movieInfo;
    private List<Review> reviewList;
}
```

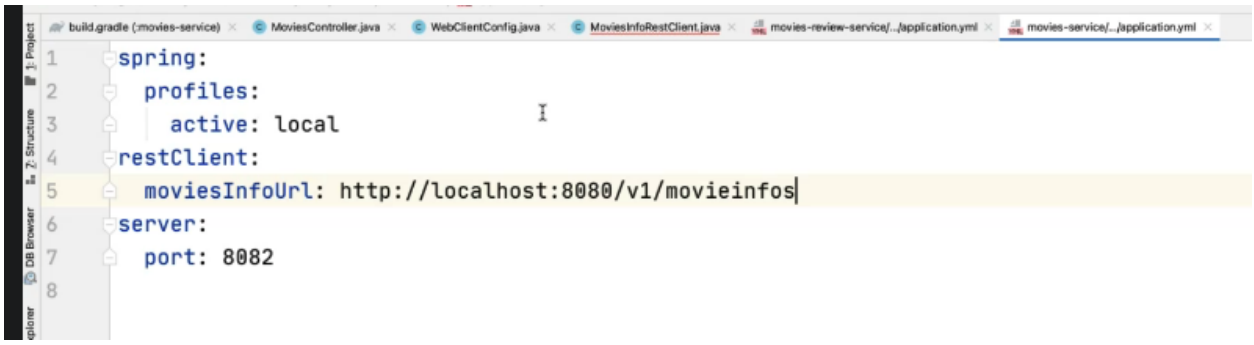
```
package com.reactivespring.domain;


import ...

@Data
@NoArgsConstructor
@AllArgsConstructor
@Validated
public class MovieInfo {
    private String movieInfoId;
    @NotBlank(message = "movieInfo.name must be present")
    private String name;
    @NotNull
    @Positive(message = "movieInfo.year must be a Positive Value")
    private Integer year;

    @NotNull
    private List<@NotBlank(message = "movieInfo.cast must be present") String> cast;
    private LocalDate release_date;
}
```

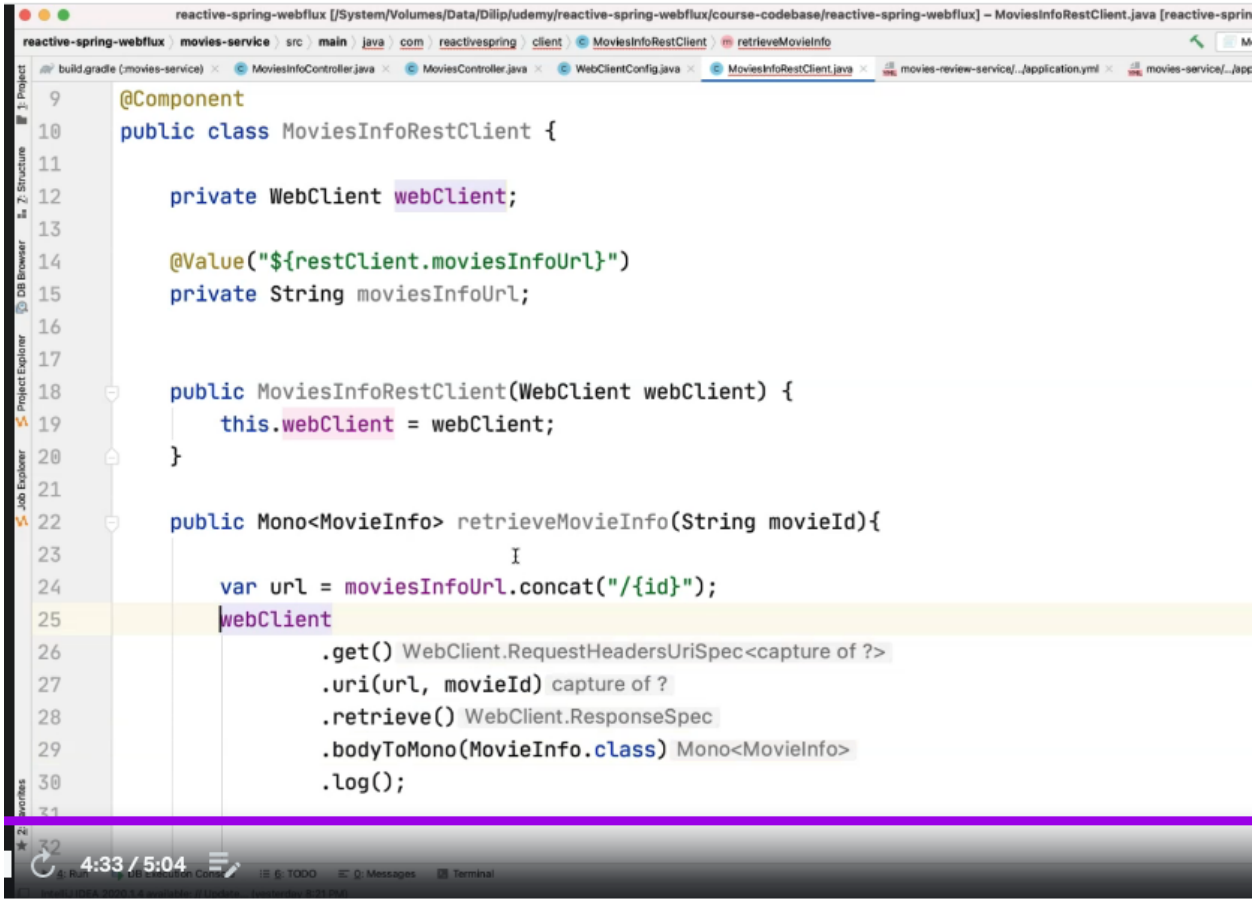
WebClient

- It is a reactive non-blocking Rest Client
- It uses a functional style API
- It allows the application to interact with other services in a non-blocking fashion
- Its auto configured in to the application by Spring Boot

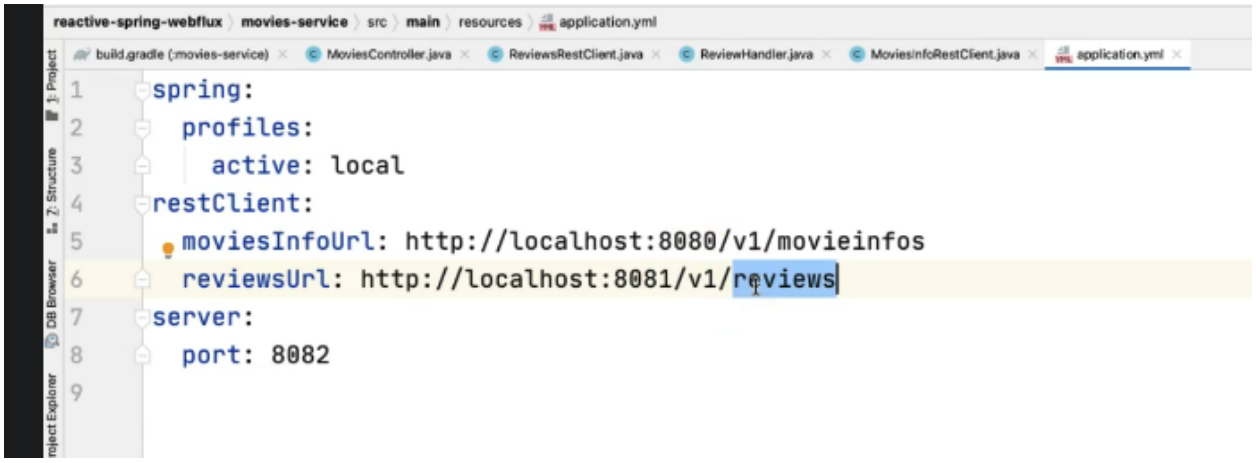
https://docs.spring.io/spring-framework/docs/current/reference/html/web-reactive.html#webflux-client

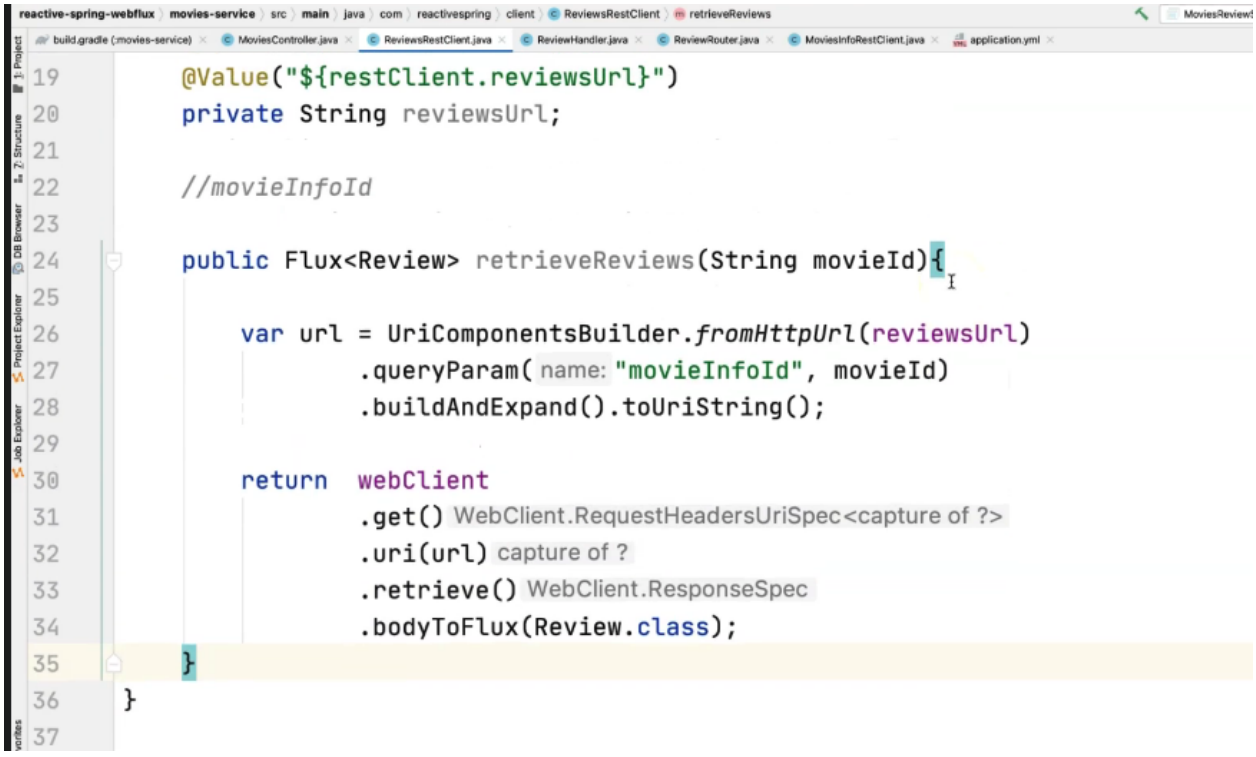


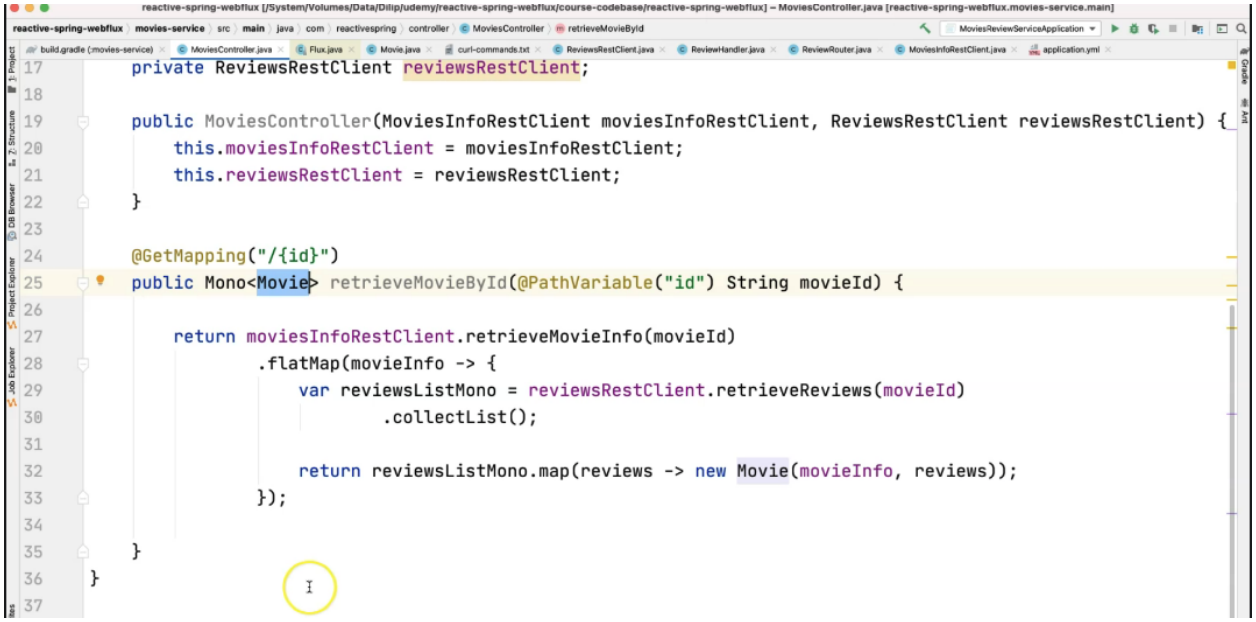




Add return in the line number 25.


reactive-spring-webflux movies-service src main resources application.yml
build.gradle (:movies-service)
MoviesController.java
ReviewsRestClient.java
ReviewHandler.java
MoviesInfoRestClient.java
application.yml
spring:
profiles:
active: local
restClient:
moviesInfoUrl: http://localhost:8080/v1/movieinfos
reviewsUrl: http://localhost:8081/v1/reviews
server:
port: 8082

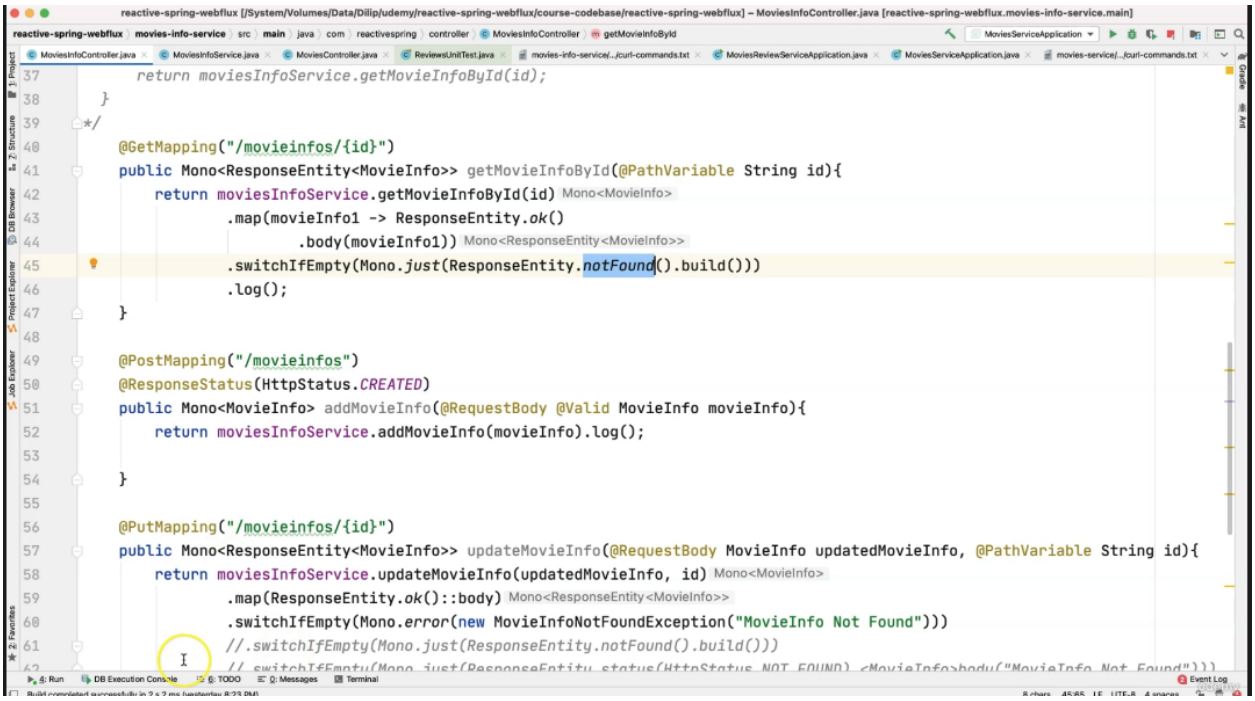

reactive-spring-webflux movies-service src main java com reactivespring client ReviewsRestClient retrieveReviews
@Value("${restClient.reviewsUrl}")
private String reviewsUrl;
//movieInfoId
public Flux<Review> retrieveReviews(String movieId){
var url = UriComponentsBuilder.fromHttpUrl(reviewsUrl)
.queryParam(name: "movieInfoId", movieId)
.buildAndExpand().toUriString();
return webClient
.get() WebClient.RequestHeadersUriSpec<capture of ?>
.uri(url) capture of ?
.retrieve() WebClient.ResponseSpec
.bodyToFlux(Review.class);
}
}



Exceptions in Service to Service Communication

Http Failures

- Http Failure falls in to two categories:
  - 4xx - Client Error
    - 400(Bad Request), 404 (Not Found) and etc.,
  - 5xx - Server Error
    - 500(Internal Server Error), 503 (Service Unavailable) and etc.,



```
return moviesInfoService.getMovieInfoById(id);
}
*/
@GetMapping("/movieinfos/{id}")
public Mono<ResponseEntity<MovieInfo>> getMovieInfoById(@PathVariable String id){
    return moviesInfoService.getMovieInfoById(id) Mono<MovieInfo>
            .map(movieInfo1 -> ResponseEntity.ok()
                    .body(movieInfo1)) Mono<ResponseEntity<MovieInfo>>
            .switchIfEmpty(Mono.just(ResponseEntity.notFound().build()))
            .log();
}

@PostMapping("/movieinfos")
@ResponseStatus(HttpStatus.CREATED)
public Mono<MovieInfo> addMovieInfo(@RequestBody @Valid MovieInfo movieInfo){
    return moviesInfoService.addMovieInfo(movieInfo).log();

}

@PutMapping("/movieinfos/{id}")
public Mono<ResponseEntity<MovieInfo>> updateMovieInfo(@RequestBody MovieInfo updatedMovieInfo, @PathVariable String id){
    return moviesInfoService.updateMovieInfo(updatedMovieInfo, id) Mono<MovieInfo>
            .map(ResponseEntity.ok()::body) Mono<ResponseEntity<MovieInfo>>
            .switchIfEmpty(Mono.error(new MovieInfoNotFoundException("MovieInfo Not Found")))
            //.switchIfEmpty(Mono.just(ResponseEntity.notFound().build()))
            // switchIfEmpty(Mono.just(ResponseEntity.status(HttpStatus.NOT_FOUND).<MovieInfo>body("MovieInfo Not Found")))
```

```
{"timestamp":"2021-09-21T01:25:15.671+00:00","path":"/v1/movies/3","status":500,"error":"Internal Server Error","requestId":"7b6ab51e-1"}
------------------------------------------------------------------------------------------------------------------------------------------
/System/Volumes/Data/Dilip/udemy/reactive-spring-webflux/course-codebase/reactive-spring-webflux » curl -i http://localhost:8082/v1/movies/3
HTTP/1.1 500 Internal Server Error
Content-Type: application/json
Content-Length: 137

{"timestamp":"2021-09-21T09:51:16.757+00:00","path":"/v1/movies/3","status":500,"error":"Internal Server Error","requestId":"aaa5f982-1"}
------------------------------------------------------------------------------------------------------------------------------------------
```

But we get 500 error instead of 404 (as there is no data with 3)

```
public Mono<MovieInfo> retrieveMovieInfo(String movieId){

    var url = moviesInfoUrl.concat("/{id}");
    return webClient
            .get() WebClient.RequestHeadersUriSpec<capture of ?>
            .uri(url, movieId) capture of ?
            .retrieve() WebClient.ResponseSpec
            .onStatus(HttpStatus::is4xxClientError, clientResponse -> {
                if(clientResponse.statusCode().equals(HttpStatus.NOT_FOUND)){
                    return Mono.error(new MoviesInfoClientException(
                            "There is no MovieInfo Available for the passed in Id : " + movieId,
                            clientResponse.statusCode().value()));
                }

                return clientResponse.bodyToMono(String.class)
                        .flatMap(responseMessage -> Mono.error(new MoviesInfoClientException(
                                responseMessage, clientResponse.statusCode().value()
                        )));
            })
            .bodyToMono(MovieInfo.class) Mono<MovieInfo>
            .log();

}
```

```java
package com.reactivespring.globalerrorhandler;

import com.reactivespring.exception.MoviesInfoClientException;
import lombok.extern.slf4j.Slf4j;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;

@ControllerAdvice
@Slf4j
public class GlobalErrorHandler {

    @ExceptionHandler(MoviesInfoClientException.class)
    public ResponseEntity<String> handleClientException(MoviesInfoClientException exception) {
        log.error("Exception Caught in handleClientException : {}", exception.getMessage());
        return ResponseEntity.status(exception.getStatusCode()).body(exception.getMessage());
    }
}
```

Add line number 34 in the below code.

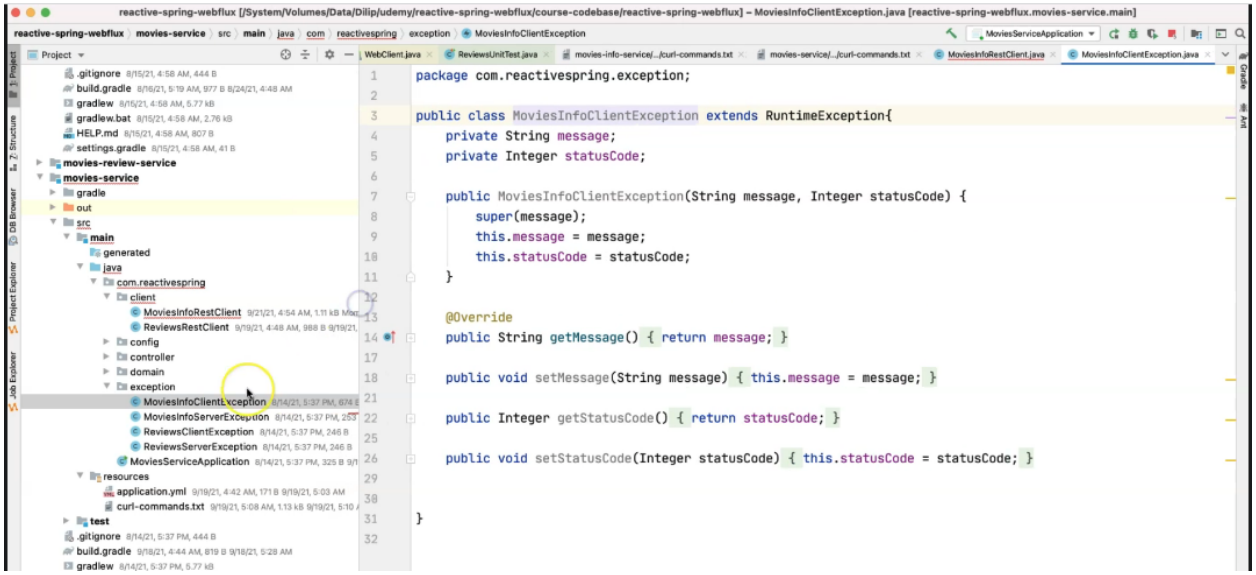
```java
public Mono<MovieInfo> retrieveMovieInfo(String movieId){

    var url = moviesInfoUrl.concat("/{id}");
    return webClient
            .get() WebClient.RequestHeadersUriSpec<capture of ?>
            .uri(url, movieId) capture of ?
            .retrieve() WebClient.ResponseSpec
            .onStatus(HttpStatus::is4xxClientError, clientResponse -> {
                log.info("Status code is : {}", clientResponse.statusCode().value());
                if(clientResponse.statusCode().equals(HttpStatus.NOT_FOUND)){
```

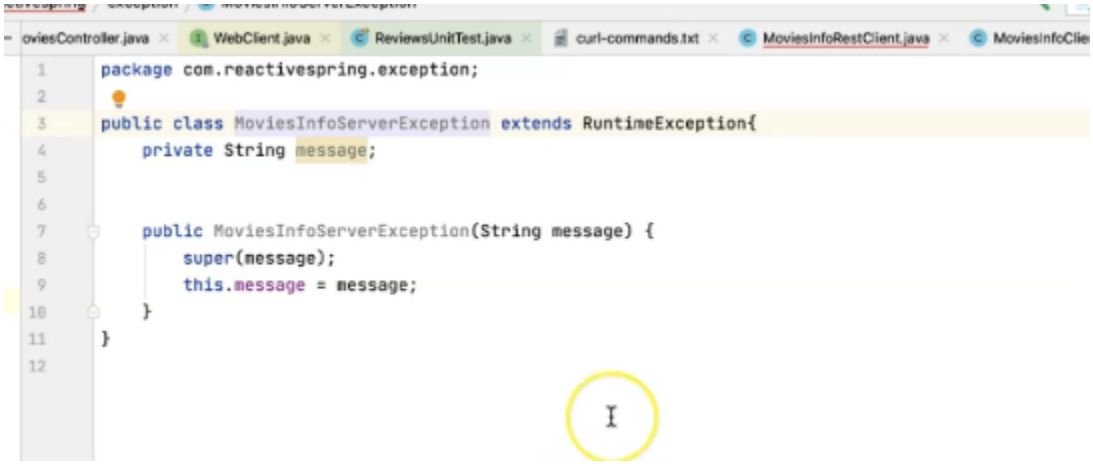
```java
            .onStatus(HttpStatus::is4xxClientError, clientResponse -> {
                log.info("Status code is : {}", clientResponse.statusCode().value());
                if(clientResponse.statusCode().equals(HttpStatus.NOT_FOUND)){
                    return Mono.error(new MoviesInfoClientException(
                            "There is no MovieInfo Available for the passed in Id : " + movieId,
                            clientResponse.statusCode().value()));
                }
```

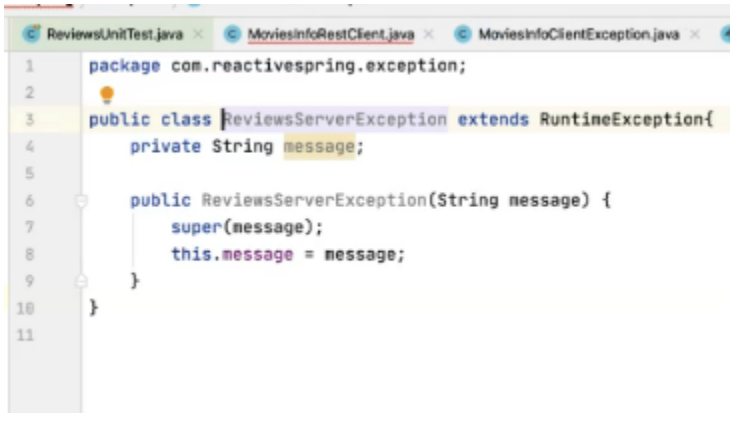
```
Terminal: Local
HTTP/1.1 500 Internal Server Error
Content-Type: application/json
Content-Length: 137

{"timestamp":"2021-09-21T09:51:16.757+00:00","path":"/v1/movies/3","status":500,"error":"Internal Server Error","requestId":"aaa5f982-1"}
---------------------------------------------------------------------------------------------------------------------------------------------
/System/Volumes/Data/Dilip/udemy/reactive-spring-webflux/course-codebase/reactive-spring-webflux » curl -i http://localhost:8082/v1/movies/3
HTTP/1.1 404 Not Found
Content-Type: text/plain;charset=UTF-8
Content-Length: 56

There is no MovieInfo Available for the passed in Id : 3
---------------------------------------------------------------------------------------------------------------------------------------------
```

Now we got a 404 error and with an appropriate message.
onStatus if the first argument is true then function Interface will be called.

Different exception classes







Stop movieInfo service

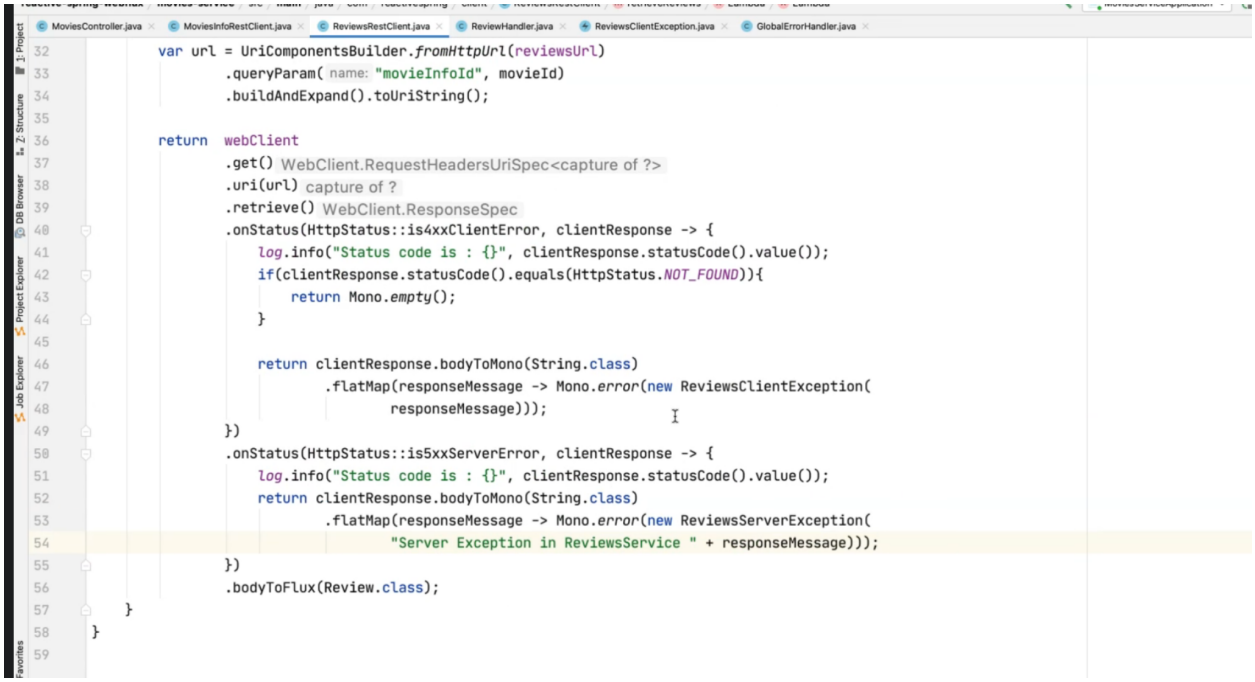
```
There is no MovieInfo Available for the passed in Id : 3
--------------------------------------------------------------------------------------------------------------------------------------------------------------
/System/Volumes/Data/Dilip/udemy/reactive-spring-webflux/course-codebase/reactive-spring-webflux » curl -i http://localhost:8082/v1/movies/2
HTTP/1.1 500 Internal Server Error
Content-Type: application/json
Content-Length: 137

{"timestamp":"2021-09-21T10:10:53.851+00:00","path":"/v1/movies/2","status":500,"error":"Internal Server Error","requestId":"e600ebc8-1"}
--------------------------------------------------------------------------------------------------------------------------------------------------------------
```

```
org.springframework.web.reactive.function.client.WebClientRequestException: Connection refused: localhost/127.0.0.1:8080; n
    at org.springframework.web.reactive.function.client.ExchangeFunctions$DefaultExchangeFunction.lambda$wrapException$9(Ex
    Suppressed: reactor.core.publisher.FluxOnAssembly$OnAssemblyException:
Error has been observed at the following site(s):
    |_ checkpoint ⇢ Request to GET http://localhost:8080/v1/movieinfos/2 [DefaultWebClient]
Stack trace:
```

```
                return clientResponse.bodyToMono(String.class)
                        .flatMap(responseMessage -> Mono.error(new MoviesInfoClientException(
                                responseMessage, clientResponse.statusCode().value()
                        )));
            })
            .onStatus(HttpStatus::is5xxServerError, clientResponse -> {
                log.info("Status code is : {}", clientResponse.statusCode().value());

                return clientResponse.bodyToMono(String.class)
                        .flatMap(responseMessage -> Mono.error(new MoviesInfoServerException(
                                "Server Exception in MoviesInfoService " + responseMessage)));
            })
            .bodyToMono(MovieInfo.class) Mono<MovieInfo>
            .log();
```

```
@ControllerAdvice
@Slf4j
public class GlobalErrorHandler {

    @ExceptionHandler(MoviesInfoClientException.class)
    public ResponseEntity<String> handleClientException(MoviesInfoClientException exception) {
        log.error("Exception Caught in handleClientException : {}", exception.getMessage());
        return ResponseEntity.status(exception.getStatusCode()).body(exception.getMessage());
    }

    @ExceptionHandler(RuntimeException.class)
    public ResponseEntity<String> handleRunTimeException(RuntimeException exception) {
        log.error("Exception Caught in handleClientException : {}", exception.getMessage());
        return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(exception.getMessage());
    }
}
```

It is difficult to test the above scenario. Because our server response is not 500. So in the above case it will try to capture controller advise.



Review test client code with 4XX and 5XX error