**Reactive programming - Consume Rest service using Spring 5 WebClient**

------------------------------------------------------------------------------------------------------------

App Name- **bookMyShow**

Dependencies- Devtool, Web, lombok, mysql

**Controller code-**

@SpringBootApplication

@RestController

@RequestMapping("BookMyShow/Service")

**public** **class** BookMyShowApplication {

@Autowired

**private** BookMyShowRepository repository;

@PostMapping("/bookingShow")

**public** String bookShow(@RequestBody BookRequest bookRequest) {

BookRequest response = repository.save(bookRequest);

**return** "Hi " + response.getUserName() + " your Request for " + response.getShowName() + " on date "

+ response.getBookingTime() + "Booking successfully..";

}

@GetMapping("/getAllBooking")

**public** List<BookRequest> getAllBooking() {

**return** repository.findAll();

}

@GetMapping("/getBooking/{bookingId}")

**public** BookRequest getBooking(@PathVariable **int** bookingId) {

**return** repository.findById(bookingId).get();

}

@DeleteMapping("/cancelBooking/{bookingId}")

**public** String cancelBooking(@PathVariable **int** bookingId) {

repository.deleteById(bookingId);

**return** "Booking cancelled with bookingId : " + bookingId;

}

@PutMapping("/updateBooking/{bookingId}")

**public** BookRequest updateBooking(@RequestBody BookRequest updateBookRequest, @PathVariable **int** bookingId) {

BookRequest dbResponse = repository.findById(bookingId).get();

dbResponse.setBookingTime(updateBookRequest.getBookingTime());

dbResponse.setPrice(updateBookRequest.getPrice());

dbResponse.setShowName(updateBookRequest.getShowName());

dbResponse.setUserCount(updateBookRequest.getUserCount());

repository.saveAndFlush(dbResponse);

**return** dbResponse;

}

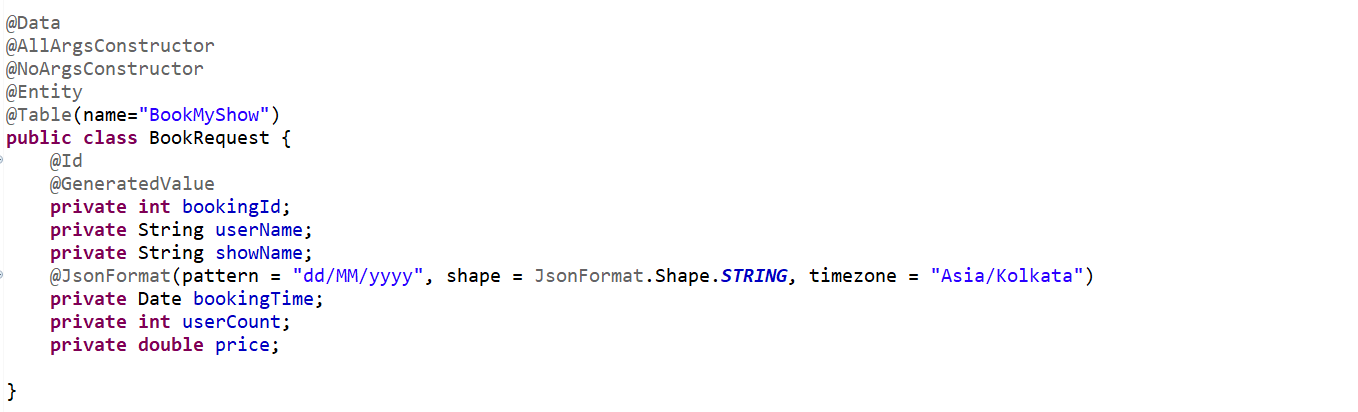
**public** **static** **void** main(String[] args) {

SpringApplication.*run*(BookMyShowApplication.**class**, args);

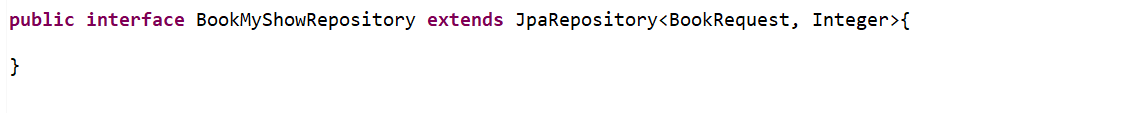
}

}

**Model class**



**Repository Class**



**Application.properties**

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url = jdbc:mysql://localhost:3306/selfdeveloped

spring.datasource.username = root

spring.datasource.password = KrTum5h87@7

spring.jpa.show-sql = true

spring.jpa.hibernate.ddl-auto = update

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect

server.port=9090

**Postman-**

**POST** <http://localhost:9090/BookMyShow/Service/bookingShow>

{

    "userName": "kanishk",

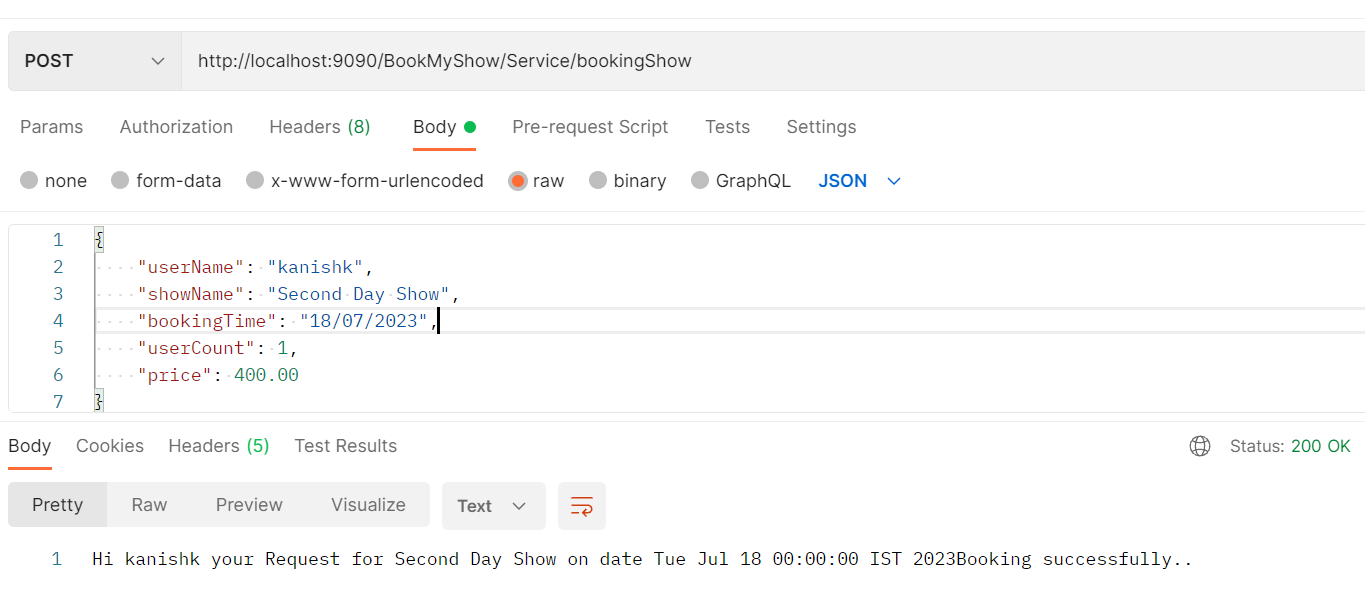
    "showName": "Second Day Show",

    "bookingTime": "18/07/2023",

    "userCount": 1,

    "price": 400.00

}



**GET** <http://localhost:9090/BookMyShow/Service/getAllBooking>

Response-

[

    {

        "bookingId": 218,

        "userName": "anand",

        "showName": "New Show",

        "bookingTime": "17/07/2023",

        "userCount": 2,

        "price": 350.0

    },

    {

        "bookingId": 219,

        "userName": "Bharat",

        "showName": "First Day Show",

        "bookingTime": "17/07/2023",

        "userCount": 4,

        "price": 700.0

    },

    {

        "bookingId": 220,

        "userName": "kanishk",

        "showName": "Second Day Show",

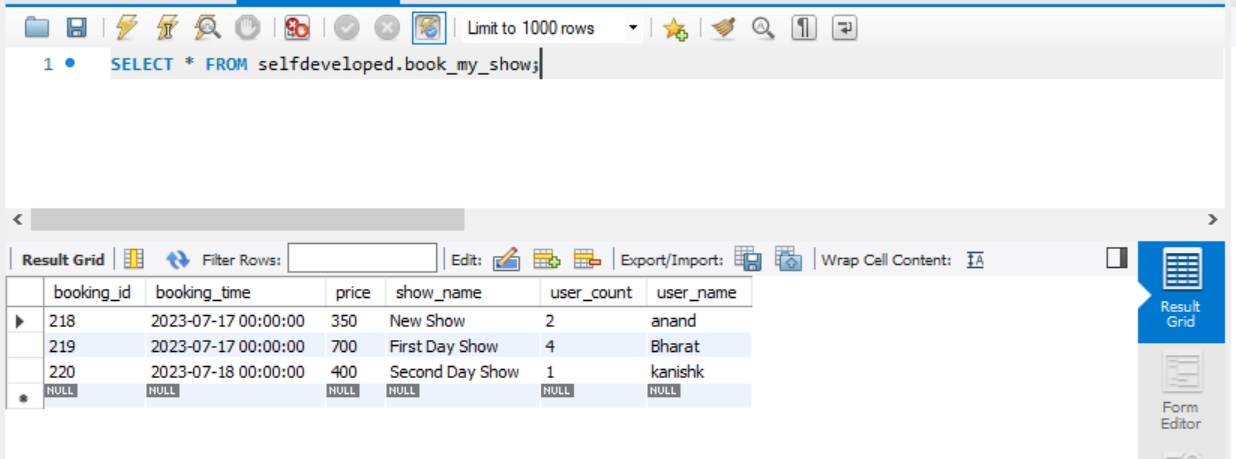
        "bookingTime": "18/07/2023",

        "userCount": 1,

        "price": 400.0

    }

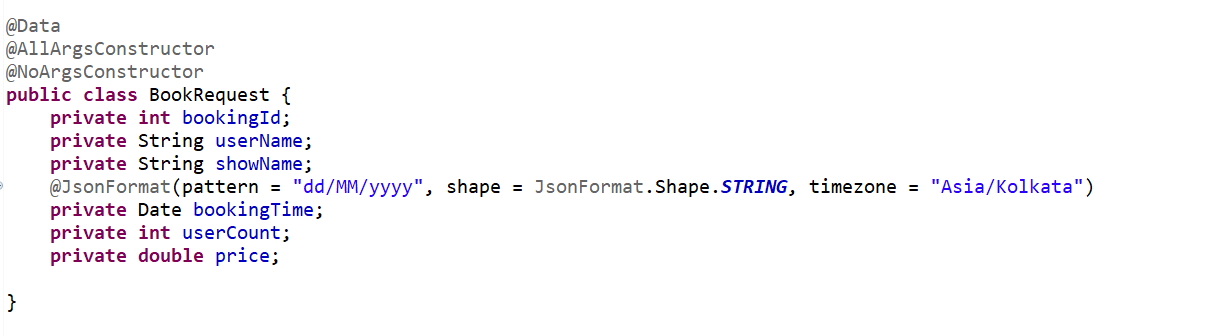
]



========================**XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX**=======================

App Name- **bookmyshow-webclient**

Dependencies- Webflux, devtool, lombok



------------------------------------------------------------------------------------------------------------------------------------------

@**SpringBootApplication**

@**RestController**

@**RequestMapping**("/bookMyShow-client")

**public** **class** BookmyshowWebclientApplication {

WebClient webClient;

@PostConstruct

**public** **void** init() {

webClient = WebClient.*builder*().baseUrl("http://localhost:9090/BookMyShow/Service")

.defaultHeader(HttpHeaders.***CONTENT\_TYPE***, MediaType.***APPLICATION\_JSON\_VALUE***).build();

}

@PostMapping("/bookNow")

**public** Mono<String> BookNow(@RequestBody BookRequest request) {

**return** webClient.post().uri("/bookingShow").~~syncBody~~(request).retrieve().bodyToMono(String.**class**);

}

@GetMapping("/trackBookings")

**public** Flux<BookRequest> trackAllBooking() {

**return** webClient.get().uri("/getAllBooking").retrieve().bodyToFlux(BookRequest.**class**);

}

@GetMapping("/trackBooking/{bookingId}")

**public** Mono<BookRequest> getBookingById(@PathVariable **int** bookingId) {

**return** webClient.get().uri("/getBooking/" + bookingId).retrieve().bodyToMono(BookRequest.**class**);

}

@DeleteMapping("/removeBooking/{bookingId}")

**public** Mono<String> cancelBooking(@PathVariable **int** bookingId) {

**return** webClient.delete().uri("/cancelBooking/" + bookingId).retrieve().bodyToMono(String.**class**);

}

@PutMapping("/changeBooking/{bookingId}")

**public** Mono<BookRequest> updateBooking(@PathVariable **int** bookingId, @RequestBody BookRequest request) {

**return** webClient.put().uri("/updateBooking/" + bookingId).~~syncBody~~(request).retrieve()

.bodyToMono(BookRequest.**class**);

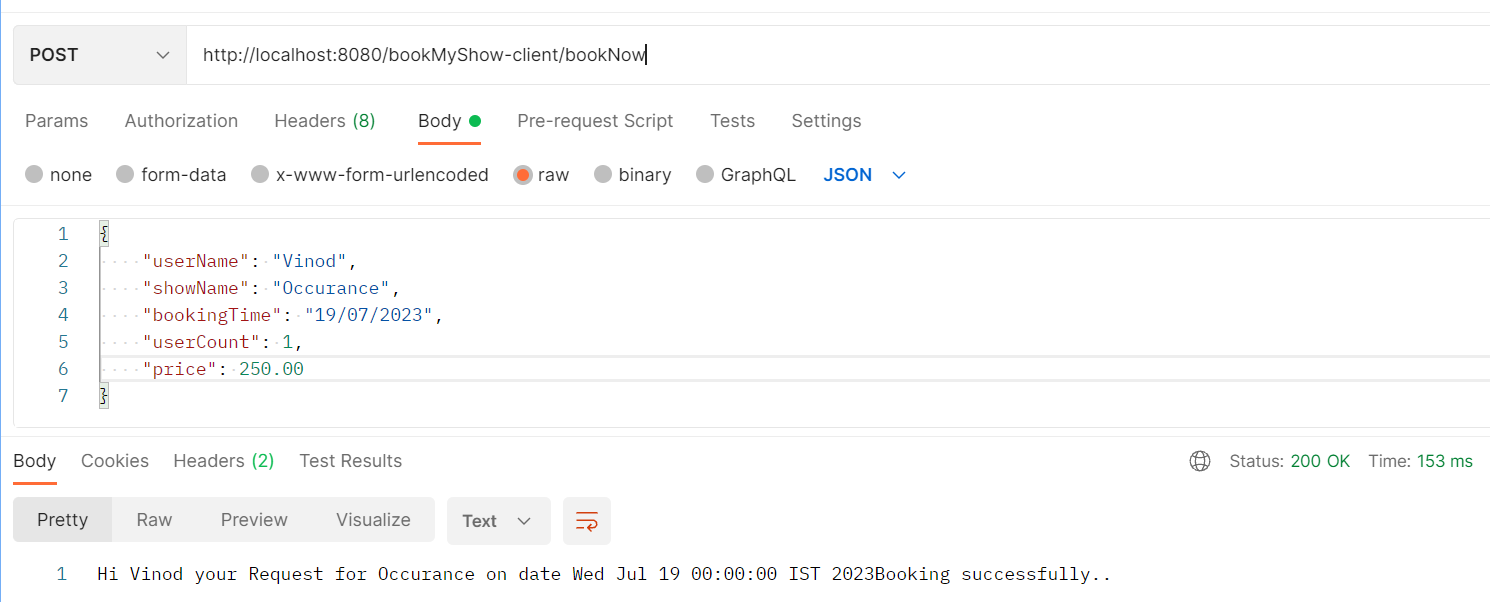
}

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(BookmyshowWebclientApplication.**class**, args);

}}

**POST** <http://localhost:8080/bookMyShow-client/bookNow>



**GET** <http://localhost:8080/bookMyShow-client/trackBookings>

[

    {

        "bookingId": 218,

        "userName": "anand",

        "showName": "New Show",

        "bookingTime": "17/07/2023",

        "userCount": 2,

        "price": 350.0

    },

    {

        "bookingId": 219,

        "userName": "Bharat",

        "showName": "First Day Show",

        "bookingTime": "17/07/2023",

        "userCount": 4,

        "price": 700.0

    },

    {

        "bookingId": 220,

        "userName": "kanishk",

        "showName": "Second Day Show",

        "bookingTime": "18/07/2023",

        "userCount": 1,

        "price": 400.0

    },

    {

        "bookingId": 221,

        "userName": "Vinod",

        "showName": "Occurance",

        "bookingTime": "19/07/2023",

        "userCount": 1,

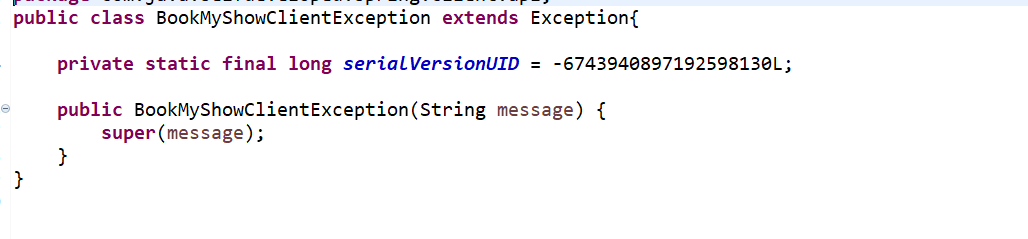
        "price": 250.0

    }

]

**Web Client Logging and Exception Handling**

**---------------------------------------------------------------**



**------------------------------------------------------------------------------------------------------------------------------------------**

@SpringBootApplication

@RestController

@RequestMapping("/bookMyShow-client")

**public** **class** BookmyshowWebclientApplication {

Logger log = LoggerFactory.*getLogger*(BookmyshowWebclientApplication.**class**);

WebClient webClient;

@PostConstruct

**public** **void** init() {

webClient = WebClient.*builder*().baseUrl("http://localhost:9090/BookMyShow/Service")

.defaultHeader(HttpHeaders.***CONTENT\_TYPE***, MediaType.***APPLICATION\_JSON\_VALUE***).filter(logRequest())

.filter(logResponse()).build();

}

@PostMapping("/bookNow")

**public** Mono<String> BookNow(@RequestBody BookRequest request) {

**return** webClient.post().uri("/bookingShow").~~syncBody~~(request).retrieve().bodyToMono(String.**class**);

}

@GetMapping("/trackBookings")

**public** Flux<BookRequest> trackAllBooking() {

**return** webClient.get().uri("/getAllBooking").retrieve().bodyToFlux(BookRequest.**class**);

}

@GetMapping("/trackBooking/{bookingId}")

**public** Mono<BookRequest> getBookingById(@PathVariable **int** bookingId) {

**return** webClient.get().uri("/getBooking/" + bookingId).retrieve()

.onStatus(HttpStatus::is4xxClientError,

clientResponse-> Mono.*error*(**new** BookMyShowClientException("404 unsported exception")))

.onStatus(HttpStatus::is5xxServerError,

clientResponse-> Mono.*error*(**new** BookMyShowClientException(" 505 Server exception")))

.bodyToMono(BookRequest.**class**);

}

@DeleteMapping("/removeBooking/{bookingId}")

**public** Mono<String> cancelBooking(@PathVariable **int** bookingId) {

**return** webClient.delete().uri("/cancelBooking/" + bookingId).retrieve().bodyToMono(String.**class**);

}

@PutMapping("/changeBooking/{bookingId}")

**public** Mono<BookRequest> updateBooking(@PathVariable **int** bookingId, @RequestBody BookRequest request) {

**return** webClient.put().uri("/updateBooking/" + bookingId).~~syncBody~~(request).retrieve()

.bodyToMono(BookRequest.**class**);

}

**private** ExchangeFilterFunction logRequest() {

**return** ExchangeFilterFunction.*ofRequestProcessor*(clientRequest-> {

log.info("Request {} {}", clientRequest.method(), clientRequest.url());

**return** Mono.*just*(clientRequest);

});

}

**private** ExchangeFilterFunction logResponse() {

**return** ExchangeFilterFunction.*ofResponseProcessor*(clientResponse-> {

log.info("Response {} {}", clientResponse.statusCode());

**return** Mono.*just*(clientResponse);

});

}

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(BookmyshowWebclientApplication.**class**, args);

}

}

**------------------------------------------------------------------------------------------------------------------------------------------**

