GRAB YOUR TICKETS APPLICATION AND PAYMENT HANDLING MICROSERVICE DOCUMENTATION

Grab Your Tickets Application

Introduction

The Grab Your Tickets application is an online platform that allows users to browse and book tickets for movies and events. This documentation provides detailed information on setting up and using the Grab Your Tickets application.

Technologies Used

- Java Spring Boot
- Spring Web MVC
- Thymeleaf
- MySQL
- RestTemplate

Configuration

YAML Configuration (application.yml)

```yaml
Server configuration
server:
  port: 8080

Database configuration
spring:
  datasource:
    url: jdbc:mysql://localhost:3306/grab_your_tickets_db
```

```
  username: root
  password: password
  driver-class-name: com.mysql.cj.jdbc.Driver

Payment Handling Microservice Configuration
payment:
 service:
  url: http://payment-service:8081/api/payments/process
```

Controller

```java
@RestController
@RequestMapping("/api/tickets")
public class TicketController {

  @Autowired
  private RestTemplate restTemplate;

  @PostMapping("/book")
  public ResponseEntity<String> bookTickets(@RequestBody SeatSelectionRequest seatSelectionRequest) {
      // Call Payment Handling microservice to process payment
      HttpHeaders headers = new HttpHeaders();
      headers.setContentType(MediaType.APPLICATION_JSON);

      HttpEntity<SeatSelectionRequest> requestEntity = new HttpEntity<>(seatSelectionRequest, headers);
      ResponseEntity<String> responseEntity = restTemplate.exchange(
          "http://payment-service:8081/api/payments/process",
          HttpMethod.POST,
```

```
            requestEntity,
            String.class);

        return ResponseEntity.ok(responseEntity.getBody());
    }
}
```

Model

```java
public class SeatSelectionRequest {
    private double ticketPrice;
    private int numSeats;

    // Getters and setters
}
```

Payment Handling Microservice

Introduction

The Payment Handling microservice is responsible for processing payment transactions initiated by the Grab Your Tickets platform. This documentation provides detailed information on setting up and using the Payment Handling microservice.

Technologies Used

- Java Spring Boot
- Spring Web MVC
- MySQL

Configuration

YAML Configuration (application.yml)

```yaml
Server configuration
server:
  port: 8081

Database configuration
spring:
  datasource:
    url: jdbc:mysql://localhost:3306/payment_db
    username: root
    password: password
    driver-class-name: com.mysql.cj.jdbc.Driver
```

Controller

```java
@RestController
@RequestMapping("/api/payments")
public class PaymentController {

    @Autowired
    private PaymentService paymentService;

    @PostMapping("/process")
```

```
    public ResponseEntity<PaymentResponse> processPayment(@RequestBody SeatSelectionRequest seatSelectionRequest) {
        double paymentAmount = seatSelectionRequest.getTicketPrice() * seatSelectionRequest.getNumSeats();
        boolean paymentSuccess = paymentService.processPayment(paymentAmount, seatSelectionRequest);

        PaymentResponse paymentResponse = new PaymentResponse();
        if (paymentSuccess) {
            paymentResponse.setSuccess(true);
            paymentResponse.setMessage("Payment processed successfully.");
        } else {
            paymentResponse.setSuccess(false);
            paymentResponse.setMessage("Payment failed. Please try again.");
        }
        return ResponseEntity.ok(paymentResponse);
    }
}
```

Service

```java
@Service
public class PaymentService {

    public boolean processPayment(double amount, SeatSelectionRequest seatSelectionRequest) {
        // Calculate expected payment amount
        double expectedAmount = seatSelectionRequest.getTicketPrice() * seatSelectionRequest.getNumSeats();

        // Validate if the calculated amount matches the expected amount
```

```
        if (amount != expectedAmount) {
            return false; // Payment amount mismatch
        }

        // Dummy payment processing logic
        return true; // Payment successful
    }
}
```

Model

```java
public class PaymentResponse {

    private boolean success;
    private String message;

    // Getters and setters
}
```

The details such as numberOfSeats and ticketPrice can indeed be selected from HTML/JavaScript and sent to the `/bookings` endpoint as a request body. This process typically involves capturing user input through HTML forms or JavaScript user interfaces, then using JavaScript to make an AJAX request to the backend API endpoint with the selected data as the request body.

HTML Form (booking.html)

```html
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Book Tickets</title>
</head>
<body>
    <h1>Book Tickets</h1>
    <form id="bookingForm">
        <label for="ticketPrice">Ticket Price:</label>
        <input type="number" id="ticketPrice" name="ticketPrice" required><br><br>

        <label for="numberOfSeats">Number of Seats:</label>
        <input type="number" id="numberOfSeats" name="numberOfSeats" required><br><br>

        <button type="submit">Book Tickets</button>
    </form>

    <div id="message"></div>

    <script>
        document.getElementById('bookingForm').addEventListener('submit', function(event) {
            event.preventDefault(); // Prevent default form submission

            // Get values from form
            const ticketPrice = document.getElementById('ticketPrice').value;
            const numberOfSeats = document.getElementById('numberOfSeats').value;

            // Create request body
            const requestBody = {
                ticketPrice: ticketPrice,
```

```
            numberOfSeats: numberOfSeats
        };

        // Make AJAX request
        fetch('/api/tickets/book', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify(requestBody)
        })
        .then(response => response.json())
        .then(data => {
            // Display response message
            document.getElementById('message').innerText = data.message;
        })
        .catch(error => {
            console.error('Error:', error);
            document.getElementById('message').innerText = 'An error occurred. Please try again.';
        });
    });
  </script>
</body>
</html>
```

In this HTML form, users can input the ticket price and the number of seats they wish to book. When the form is submitted, JavaScript captures the form data, constructs a JSON request body, and sends an AJAX POST request to the `/api/tickets/book` endpoint. The backend API can then process this request and handle the booking accordingly.