RabbitMQ with Spring Boot Application

#Setting up RabbitMQ Server

- **Software Requirement for RabbitMQ Server**
  - Erlang 27.x.x
  - RabbitMQ Server 4.x.x
- **Enable the management plugin**
  - The management UI is not enabled by default. If you are running RabbitMQ locally, open your terminal and enable the plugin with this command:

    rabbitmq-plugins enable rabbitmq_management
- **Access the management UI**
  - After enabling the plugin, navigate to the web interface by opening a web browser and going to http://<your-rabbitmq-server-host>:15672/.
  - For a local installation, this is typically http://localhost:15672/.
- **Log in**
  - By default, you can log in with the username guest and password guest. This user is restricted to localhost only.
- **Navigate to the Queues tab**
  - Once you are logged in, click on the "Queues" tab. This will show you a list of all the queues on the server.
- **Inspect a specific queue**
  - Clicking on an individual queue name will take you to a detailed page for that queue, where you can see:
  - **Real-time metrics:** Charts showing message rates and queue size.
  - **Contents of the queue:** Scroll down to the "Get messages" panel to pull and inspect messages. This is a destructive operation that removes messages unless you tell RabbitMQ to requeue them.
  - **Queue details:** Information on consumers, bindings, and other configurations.

#Spring Boot App using RabbitMQ

- Make Spring Boot Application with following dependencies:
  - Spring Web
  - Spring RabbitMQ
- Add RabbitMQConfiguration class under config package of app base package
  - RabbitMQConfiguration.java

```java
@Configuration
public class RabbitMQConfiguration {

// Define the name of the queue
public static final String QUEUE_NAME = "my.simple.queue";
// Define the name of the exchange
public static final String EXCHANGE_NAME = "my.simple.exchange";
// Define the routing key
public static final String ROUTING_KEY = "my.simple.routingkey";
```

```
@Bean
public Queue myQueue() {
// The queue will survive a broker restart (durable = true)
return new Queue(QUEUE_NAME, true);
}

@Bean
public DirectExchange myExchange() {
return new DirectExchange(EXCHANGE_NAME);
}

@Bean
public Binding binding(Queue myQueue, DirectExchange myExchange) {
return BindingBuilder.bind(myQueue).to(myExchange).with(ROUTING_KEY);
}

@Bean
public RabbitAdmin rabbitAdmin(ConnectionFactory connectionFactory) {
return new RabbitAdmin(connectionFactory);
}

@Bean
public RabbitTemplate rabbitTemplate(ConnectionFactory connectionFactory) {
return new RabbitTemplate(connectionFactory);
}
}
```

- Add MessageProducer class under producer package of app base package
  - MessageProducer.java

```
@Component
public class MessageProducer {


private static final Logger log = LoggerFactory.getLogger(MessageProducer.class);


@Autowired
private RabbitTemplate rabbitTemplate;
@Scheduled(fixedDelay = 5000)
public void sendMessage(String msg) {
String message = msg + System.currentTimeMillis();
log.info("Sending message: '{}'", message);
// Sends the message to the specified exchange with the routing key
rabbitTemplate.convertAndSend(EXCHANGE_NAME, ROUTING_KEY, message);
}
}
```

- Add MessageListener class under consumer package of app base package
  - MessageListener.java

```
@Component
public class MessageListener {


private static final Logger log = LoggerFactory.getLogger(MessageListener.class);
```

```
@RabbitListener(queues = QUEUE_NAME)
public void receiveMessage(String message) { // You can add your message
processing logic here // e.g., save to a database, call another service, etc.
log.info("Received message: '{}'", message);
}
}
```

- Add MessageController class under controller package of app base package
  - MessageController.java

```
@RestController
@RequestMapping("/api/message")
public class MessageController {

@Autowired
private MessageProducer messageProducer;
@PostMapping("/send")
public ResponseEntity<String> sendMessage(@RequestBody String message) {
messageProducer.sendMessage(message);
return ResponseEntity.ok("Message sent to RabbitMQ successfully!");
}
}
```