RabbitMQ의 기본 비동기 메시지 전송

각 단계별 메시지 전송 예제 살펴보기

각 단계별 Step Tutorial 강의로 단순 메시지 전송부터 Work Queue, Pub/Sub, Routing, Dead Letter Queue 등에 해당하는 샘플 코드들을 작성하면서 각 개념에 대해서 이해하도록 한다.

각 단계별 예제는 Git (https://github.com/villainscode/HelloMessageQueue)의 Branch 별로 총 12개의 예제 코드들을 참고하여 따라할 수 있도록 구성하였다.

단순 메시지 전송 (Producer to Consumer)

메시지 발행자(Producer)가 큐에 메시지를 전달하면 소비자(Consumer)는 큐에 들어온 메시지를 소진한다.

튜토리얼 Step 1. 단순 메시지 전송 (Producer to Consumer)

1. 서버 실행 : rabbitmq 가 인스톨된 위치에서 sbin 하위의 ./rabbitmq-server를 실행
    a. brew 인스톨 위치 찾기 : brew --prefix rabbitmq
2. 유저 및 VHost 추가 (guestuser / guestuser)

    VHost와 권한의 역할

    ```
    VHost는 RabbitMQ에서 사용자를 격리하는 기본 단위로 다음과 같은 역할을 한다
        1.  리소스 격리: VHost는 큐, 익스체인지, 바인딩을 포함한 RabbitMQ 리소스
                각 VHost는 독립된 큐 및 익스체인지 세트를 가지므로, 서로 다른 VH
        2.  멀티 테넌시(Multi-Tenancy): 여러 애플리케이션이나 사용자 그룹이 하나
              각 애플리케이션이 서로 간섭하지 않고 독립적으로 운영될 수 있도록 함.
        3.  접근 제어: VHost 단위로 사용자 접근 권한을 부여할 수 있어, 특정 사용자
    ```

    VHost 권한의 세 가지 종류

    ```
    1.  Configure (.*): 익스체인지 및 큐의 설정 권한.
    2.  Write (.*): 큐에 메시지를 쓰는 권한. 큐 전송
    3.  Read (.*): 큐에서 메시지를 읽는 권한. 큐 소비
    ```

3. IntelliJ 프로젝트 생성

a. rabbitmq 관련 의존성추가 (기존에는 Spring AMQP)

4. 정상적으로 실행 되는지 테스트
   a. rest api 샘플 작성
5. application.yml 작성

```
spring:
  rabbitmq:
    host: localhost
    port: 5672  // 기본 통신 포트, 15672는 주로 관리 및 모니터링(admin) 용
    username: guestuser
    password: guestuser
  application:
    name: HelloWorldMessageQueue
server:
  port: 8080
```

6. RabbitMQConfig.java 작성
   a. https://github.com/villainscode/HelloMessageQueue/blob/tutorial-step1/src/main/java/net/harunote/hellomessagequeue/step1/RabbitMQConfig.java
   b. Bean 생성
      i. Queue queue() : Queue 인스턴스를 생성하고, 애플리케이션이 사용할 큐를 정의, 메시지를 전달하고 처리하는 기본 큐 세팅
         - QUEUE_NAME은 메시지가 쌓이고 처리될 큐의 이름을 정의
         - 다음 인자로 넘길 false 파라미터는 큐가 휘발성(volatile)인지 영속성(persistent)인지 여부를 지정하는 옵션. (false로 설정하면 서버가 종료되거나 재시작될 때 큐의 메시지가 사라짐)

      ii. RabbitTemplate rabbitTemplate(ConnectionFactory connectionFactory) : RabbitMQ와 통신하기 위한 템플릿 인스턴스 생성, 메시지 송수신용
         - JdbcTemplate과 비슷하게, RabbitMQ와 상호작용하기 위한 간단한 API를 제공합니다. 주로 메시지 전송을 담당
         - ConnectionFactory는 RabbitMQ와의 연결을 관리하는 객체로, rabbitTemplate에 주입하여 메시지를 전송할 때 사용할 연결을 제공

- 메시지를 전송하는 Sender가 rabbitTemplate.convertAndSend() 메서드를 사용해 큐에 메시지를 넣는 데 사용

iii. SimpleMessageListenerContainer container(ConnectionFactory connectionFactory, MessageListenerAdapter listenerAdapter) :

- RabbitMQ 메시지를 비동기적으로 수신하기 위해 SimpleMessageListenerContainer 를 생성, 이 컨테이너가 특정 큐를 지속적으로 모니터링하고 매시지를 수신하면 지정된 리스너(MessageListenerAdapter)를 통해 처리
- ConnectionFactory는 RabbitMQ와 연결을 유지하며, 수신하는 메시지를 이 연결을 통해 가져옴
- setQueueNames(QUEUE_NAME) 메서드는 특정 큐 이름을 설정. 이 컨테이너는 코드에서 설명한 큐 네임인 helloQueue에서 수신되는 메시지를 모니터링
- setMessageListener(listenerAdapter)는 listenerAdapter를 설정하여, 메시지가 수신될 때 호출할 리스너를 지정

iv. MessageListenerAdapter listenerAdapter(Receiver receiver) : 수신한 메시지를 특정 클래스의 특정 메서드로 전달하는 어댑터, 인자로 전달된 메서드를 자동으로 호출

- receiver 객체는 메시지를 처리하는 역할을 하는 빈이며, receiveMessage 메서드를 호출
- MessageListenerAdapter는 RabbitMQ에서 수신된 메시지를 특정 메서드에 전달할 수 있도록 해줌
- 이 경우, receiveMessage 메서드가 자동으로 호출되며, 메시지 내용을 인자로 받음
- Receiver 클래스의 receiveMessage 메서드가 메시지를 수신하여 처리할 수 있도록 설정 (RabbitMQ에서 수신된 메시지가 receiver.receiveMessage(String message) 메서드로 전달)

7. Message Sender 구현

   https://github.com/villainscode/HelloMessageQueue/blob/tutorial-step1/src/main/java/net/harunote/hellomessagequeue/step1/Sender.java

8. Message Receiver 구현

   https://github.com/villainscode/HelloMessageQueue/blob/tutorial-step1/src/main/java/net/harunote/hellomessagequeue/step1/Receiver.java

9. REST API 호출 Endpoint 작성

https://github.com/villainscode/HelloMessageQueue/blob/tutorial-step1/src/main/java/net/harunote/hellomessagequeue/step1/MessageController.java

10. 메시지 전송 테스트 (인텔리제이 내장 http client)

```
###
GET http://localhost:8080/hello?q=world!!

<> 2024-11-20T201307.200.txt

###
POST http://localhost:8080/api/send
Content-Type: application/json

"MyQueue Hello World Test"
```

코드들은 https://github.com/villainscode/HelloMessageQueue/tree/tutorial-step1 확인

코드 작성 전에 git 에 푸시하기

github 에 들어가서 repository 생성 (여기서는 HelloMessageQueue로 생성)

인텔리제이 로컬 터미널로 이동해서 아래와 같이 초기화 및 remote 설정 후 rebase 한 뒤 push

```
git init
git remote -v  // 원격 url 확인
git remote remove origin // 있을 경우 모든 원격 url 삭제

git remote add origin https://github.com/[MyAccount]/HelloMessageQueu
git remote -v
git add .
git commit -m "init"
git pull origin main --rebase // rebase 옵션은 로컬의 변경사항을 원격의 최신
git push origin main
```

각 단계별 메시지 전송 예제 살펴보기

각 단계별 Step Tutorial 강의로 단순 메시지 전송부터 Work Queue, Pub/Sub, Routing, Dead Letter Queue 등에 해당하는 샘플 코드들을 작성하면서 각 개념에 대해서 이해하도록 한다.

각 단계별 예제는 Git (https://github.com/villainscode/HelloMessageQueue)의 Branch 별로 총 12개의 예제 코드들을 참고하여 따라할 수 있도록 구성하였다.

단순 메시지 전송 (Producer to Consumer)

메시지 발행자(Producer)가 큐에 메시지를 전달하면 소비자(Consumer)는 큐에 들어온 메시지를 소진한다.

튜토리얼 Step 1. 단순 메시지 전송 (Producer to Consumer)

1. 서버 실행 : rabbitmq 가 인스톨된 위치에서 sbin 하위의 ./rabbitmq-server를 실행
    a. brew 인스톨 위치 찾기 : brew --prefix rabbitmq
2. 유저 및 VHost 추가 (guestuser / guestuser)

    VHost와 권한의 역할

    ```
    VHost는 RabbitMQ에서 사용자를 격리하는 기본 단위로 다음과 같은 역할을 한다
        1.  리소스 격리: VHost는 큐, 익스체인지, 바인딩을 포함한 RabbitMQ 리소스
                각 VHost는 독립된 큐 및 익스체인지 세트를 가지므로, 서로 다른 VH
        2.  멀티 테넌시(Multi-Tenancy): 여러 애플리케이션이나 사용자 그룹이 하나
              각 애플리케이션이 서로 간섭하지 않고 독립적으로 운영될 수 있도록 함.
        3.  접근 제어: VHost 단위로 사용자 접근 권한을 부여할 수 있어, 특정 사용자
    ```

    VHost 권한의 세 가지 종류

    ```
    1.  Configure (.*): 익스체인지 및 큐의 설정 권한.
    2.  Write (.*): 큐에 메시지를 쓰는 권한. 큐 전송
    3.  Read (.*): 큐에서 메시지를 읽는 권한. 큐 소비
    ```

3. IntelliJ 프로젝트 생성
    a. rabbitmq 관련 의존성추가 (기존에는 Spring AMQP)
4. 정상적으로 실행 되는지 테스트

a. rest api 샘플 작성

5. application.yml 작성

```
spring:
  rabbitmq:
    host: localhost
    port: 5672  // 기본 통신 포트, 15672는 주로 관리 및 모니터링(admin) 용
    username: guestuser
    password: guestuser
  application:
    name: HelloWorldMessageQueue
server:
  port: 8080
```

6. RabbitMQConfig.java 작성
   a. https://github.com/villainscode/HelloMessageQueue/blob/tutorial-step1/src/main/java/net/harunote/hellomessagequeue/step1/RabbitMQConfig.java
   b. Bean 생성
      i. Queue queue() : Queue 인스턴스를 생성하고, 애플리케이션이 사용할 큐를 정의, 메시지를 전달하고 처리하는 기본 큐 세팅
         - QUEUE_NAME은 메시지가 쌓이고 처리될 큐의 이름을 정의
         - 다음 인자로 넘길 false 파라미터는 큐가 휘발성(volatile)인지 영속성(persistent)인지 여부를 지정하는 옵션. (false로 설정하면 서버가 종료되거나 재시작될 때 큐의 메시지가 사라짐)

      ii. RabbitTemplate rabbitTemplate(ConnectionFactory connectionFactory) : RabbitMQ와 통신하기 위한 템플릿 인스턴스 생성, 메시지 송수신용
         - JdbcTemplate과 비슷하게, RabbitMQ와 상호작용하기 위한 간단한 API를 제공합니다. 주로 메시지 전송을 담당
         - ConnectionFactory는 RabbitMQ와의 연결을 관리하는 객체로, rabbitTemplate에 주입하여 메시지를 전송할 때 사용할 연결을 제공
         - 메시지를 전송하는 Sender가 rabbitTemplate.convertAndSend() 메서드를 사용해 큐에 메시지를 넣는 데 사용

iii. SimpleMessageListenerContainer container(ConnectionFactory connectionFactory, MessageListenerAdapter listenerAdapter) :

- RabbitMQ 메시지를 비동기적으로 수신하기 위해 SimpleMessageListenerContainer 를 생성, 이 컨테이너가 특정 큐를 지속적으로 모니터링하고 매시지를 수신하면 지정된 리스너(MessageListenerAdapter)를 통해 처리
- ConnectionFactory는 RabbitMQ와 연결을 유지하며, 수신하는 메시지를 이 연결을 통해 가져옴
- setQueueNames(QUEUE_NAME) 메서드는 특정 큐 이름을 설정. 이 컨테이너는 코드에서 설명한 큐 네임인 helloQueue에서 수신되는 메시지를 모니터링
- setMessageListener(listenerAdapter)는 listenerAdapter를 설정하여, 메시지가 수신될 때 호출할 리스너를 지정

iv. MessageListenerAdapter listenerAdapter(Receiver receiver) : 수신한 메시지를 특정 클래스의 특정 메서드로 전달하는 어댑터, 인자로 전달된 메서드를 자동으로 호출

- receiver 객체는 메시지를 처리하는 역할을 하는 빈이며, receiveMessage 메서드를 호출
- MessageListenerAdapter는 RabbitMQ에서 수신된 메시지를 특정 메서드에 전달할 수 있도록 해줌
- 이 경우, receiveMessage 메서드가 자동으로 호출되며, 메시지 내용을 인자로 받음
- Receiver 클래스의 receiveMessage 메서드가 메시지를 수신하여 처리할 수 있도록 설정 (RabbitMQ에서 수신된 메시지가 receiver.receiveMessage(String message) 메서드로 전달)

7. Message Sender 구현

   https://github.com/villainscode/HelloMessageQueue/blob/tutorial-step1/src/main/java/net/harunote/hellomessagequeue/step1/Sender.java

8. Message Receiver 구현

   https://github.com/villainscode/HelloMessageQueue/blob/tutorial-step1/src/main/java/net/harunote/hellomessagequeue/step1/Receiver.java

9. REST API 호출 Endpoint 작성

   https://github.com/villainscode/HelloMessageQueue/blob/tutorial-step1/src/main/java/net/harunote/hellomessagequeue/step1/MessageController.java

10. 메시지 전송 테스트 (인텔리제이 내장 http client)

```
###
GET http://localhost:8080/hello?q=world!!

<> 2024-11-20T201307.200.txt

###
POST http://localhost:8080/api/send
Content-Type: application/json

"MyQueue Hello World Test"
```

코드들은 https://github.com/villainscode/HelloMessageQueue/tree/tutorial-step1 확인

코드 작성 전에 git 에 푸시하기

github 에 들어가서 repository 생성 (여기서는 HelloMessageQueue로 생성)

인텔리제이 로컬 터미널로 이동해서 아래와 같이 초기화 및 remote 설정 후 rebase 한 뒤 push

```
git init
git remote -v  // 원격 url 확인
git remote remove origin // 있을 경우 모든 원격 url 삭제

git remote add origin https://github.com/[MyAccount]/HelloMessageQueu
git remote -v
git add .
git commit -m "init"
git pull origin main --rebase // rebase 옵션은 로컬의 변경사항을 원격의 최신
git push origin main
```