

Распределённые приложения, практика

Юрий Литвинов
yurii.litvinov@gmail.com

13.05.2019г

Задача 1, доделать gRPC

В командах по два человека разработать сетевой чат (наподобие Telegram) с помощью gRPC

- ▶ peer-to-peer, то есть соединение напрямую
 - ▶ Одно и то же приложение может быть и клиентом, и сервером
- ▶ Консольный пользовательский интерфейс
 - ▶ Отображение имени отправителя, даты и времени отправки и текста сообщения
- ▶ При запуске указываются:
 - ▶ Адрес peer-а и порт, если хотим подключиться
 - ▶ Должно быть можно не указывать, тогда работаем в режиме сервера
 - ▶ Своё имя пользователя
- ▶ Пишем адрес репозитория сюда: <https://forms.gle/NKEQKuJ2HAa8syhM7>
- ▶ За 10 минут до конца собираемся в общий чат и показываем, что получилось

RabbitMQ

- ▶ Сервер и клиенты системы надёжной передачи сообщений
 - ▶ Сообщение посылается на сервер и хранится там, пока его не заберут
 - ▶ Продвинутые возможности по маршрутизации сообщений
- ▶ Реализует протокол AMQP (Advanced Message Queuing Protocol), но может использовать и другие протоколы
- ▶ Сервер написан на Erlang, клиентские библиотеки доступны для практически чего угодно



Пример, отправитель

```
public class Sender {  
    private static final String QUEUE_NAME = "MyQueue";  
  
    public static void main(String[] args) throws IOException, TimeoutException {  
        var factory = new ConnectionFactory();  
        factory.setHost("localhost");  
        Connection connection = factory.newConnection();  
        Channel channel = connection.createChannel();  
  
        channel.queueDeclare(QUEUE_NAME, false, false, false, null);  
        String message = "Hello World!";  
        channel.basicPublish("", QUEUE_NAME, null, message.getBytes());  
        System.out.println(" [x] Sent " + message + "");  
  
        channel.close();  
        connection.close();  
    }  
}
```

Пример, получатель

```

public class Receiver {
    private static final String QUEUE_NAME = "MyQueue";

    public static void main(String[] args) throws IOException, TimeoutException {
        var factory = new ConnectionFactory();
        factory.setHost("localhost");
        Connection connection = factory.newConnection();
        Channel channel = connection.createChannel();

        channel.queueDeclare(QUEUE_NAME, false, false, false, null);
        System.out.println("[*] Waiting for messages. To exit press CTRL+C");

        Consumer consumer = new DefaultConsumer(channel) {
            @Override
            public void handleDelivery(String consumerTag, Envelope envelope,
                                     AMQP.BasicProperties properties, byte[] body)
                throws IOException {
                String message = new String(body, "UTF-8");
                System.out.println("[x] Received " + message + "");
            }
        };
        channel.basicConsume(QUEUE_NAME, true, consumer);
    }
}

```

Как всё собрать и запустить

- ▶ Поставить рантайм Erlang
- ▶ Поставить сервер RabbitMQ
 - ▶ <https://www.rabbitmq.com/download.html>
- ▶ Добавить зависимость от клиента RabbitMQ в проект
 - ▶ `compile group: 'com.rabbitmq', name: 'amqp-client', version: '5.7.0'`
- ▶ Пролистать Getting Started
 - ▶ <https://www.rabbitmq.com/tutorials/tutorial-one-java.html>

Задача 2, RabbitMQ

Переделать сетевой чат на RabbitMQ

- ▶ Сервер для обмена сообщениями, о котором договариваются клиенты
 - ▶ Центральный сервер, задаваемый при запуске (127.0.0.1 по умолчанию)
- ▶ Нет списка контактов, есть именованные каналы, на которые можно подписываться и постить туда
 - ▶ Должна быть кнопка подписки на канал и поле для ввода имени канала
 - ▶ Сообщения в разные каналы должны быть в разных вкладках
 - ▶ Подписка на несуществующий канал должна его создавать
- ▶ Нет истории, получать только те сообщения, что были опубликованы с момента подключения
 - ▶ Может помочь
<https://www.rabbitmq.com/tutorials/tutorial-three-java.html>