УДК 004

Передача данных по сети с помощью RSocket

Еровлева Регина Викторовна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Еровлев Павел Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье будут рассмотрены вопросы, касающиеся передачи данных по сети. Для решения задачи будет использоваться RSocket и Spring Boot.
Ключевые слова: RSocket, Spring, Spring boot

Reactive streams over the network with RSocket

Erovleva Regina Viktorovna
Sholom-Aleichem Priamursky State University
Student

Erovlev Pavel Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article will address issues related to data transmission over the network. To solve the problem, RSocket and Spring Boot will be used.
Keywords: RSocket, Spring, Spring boot

RSocket — это протокол, который позволяет передавать данные по сети. Одним из преимуществ RSocket является то, что заголовок самого кадра отправляется в двоичном формате. Это снижает общую полезную нагрузку сети и снижает сетевую задержку.

RSocket не только охватывает спецификацию самого протокола, но также предоставляет несколько его реализаций. В настоящее время существуют реализации для Java, JavaScript, Go, Kotlin и других.

Цель данной работы – разобраться с передачей данных по сети. Создать клиента для отдачи сообщений и сервер для их приема.

В.И. Зарайский описывает разработку модуля автоматизации работы с конференциями в кафедральном приложении [1]. Д. Ю. Балакишиев рассказывает в своей статье о написании приложения с использованием технологий Java 8, Spring Boot, Angular 2 [2]. Р.И. Ибраимов, А.Р. Зайчик,

Н.С. Минзатов в статье описывают процесс разработки генеалогического дерева на языке Java с использованием фреймворка Spring и библиотеки gedcom4j [3]. Р.И. Ибраимов, А.Б. Джемалетдинов, А.А. Шевченко в своей работе описывают информацию для веб-разработчиков, целью которых является улучшение написанного ими кода путём его проверки и тестирования [4]. А.А. Голикова и В.А. Строганов в своей работе описывают систему, разработанную для пользователей новостных порталов в сети Интернет [5].

Первый шаг к настройке проекта - перейти к Spring Initializr и выбрать зависимость RSocke. Как и в случае с R2DBC, эта функция зависит от Spring boot 2.2.x , поэтому необходимо выбрать и его (рис.1).

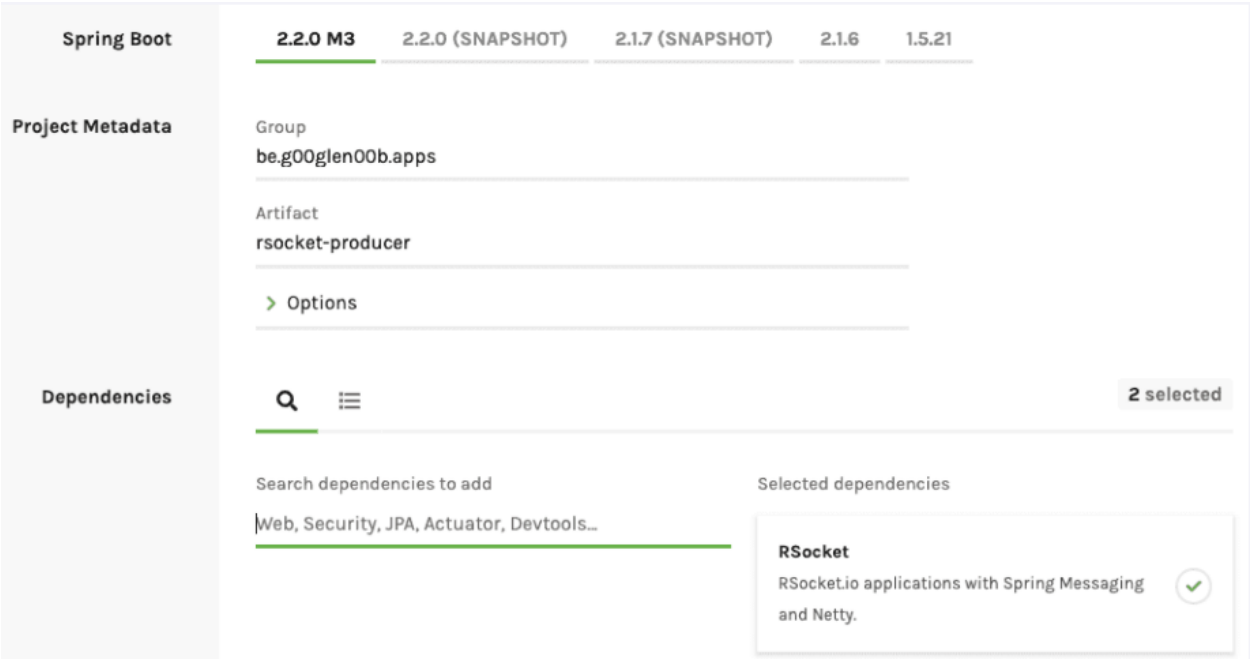

Рисунок 1 – Добавление зависимости

Работа с RSocket похожа на работу с другими протоколами обмена сообщениями, сначала нужно создать класс, представляющий данные, которые нужно передать (рис.2).

```
@Getter
@ToString
@RequiredArgsConstructor
public class PersonMessage {
    private final Long id;
    private final String firstname;
    private final String lastname;
}
```

Рисунок 2 – Создание класса

После этого можно создать контроллер и определить конечные точки, которые нужно предоставить, используя «@MessageMapping» аннотацию (рис.3).

```
@Controller
public class PersonController {
    @MessageMapping("findPeople")
    public Flux<PersonMessage> findAll() {
        return Flux.just(
            new PersonMessage(1L, "John", "Doe"),
            new PersonMessage(2L, "Jane", "Doe")
        );
    }
}
```

Рисунок 3 – Создание контроллера

Следующим шагом будет настройка RSocket клиента (рис.4).

```
@Bean
public Mono<RSocket> rSocket() {
    return RSocketFactory
        .connect()
        .dataMimeType(MediaType.APPLICATION_JSON_VALUE)
        .frameDecoder(PayloadDecoder.ZERO_COPY)
        .transport(TcpClientTransport.create(8000))
        .start()
        .doOnNext(socket -> log.info("🚀 Connected to RSocket"))
        .cache();
}
```

Рисунок 4 – Создание клиента

Как можно заметить в приведенном выше коде, идет обращение к RSocket, что будут отправляться данные JSON и что нужно подключиться к порту 8000. С помощью «PayloadDecoder.ZERO_COPY» сообщаем клиенту RSocket, что входящие данные не будут скопированы, что будет увеличена производительность, как указано в документации RSocket.

Кроме того, будем использовать «cache()» оператор, что означает, что если несколько bean-компонентов автоматически подключаются и подписываются на этот RSocket поток, исходный источник будет создан только один раз и кэширован для всех остальных подписчиков. Преимущество этого в том, что создается только одно RSocket соединение.

После этого нужно обернуть RSocket экземпляр в «SpringRSocketRequester», который предоставляет более удобный API для запроса данных из RSocket (рис.5).

```
@Bean
public Mono<RSocketRequester> rSocketRequester(Mono<RSocket> rSocket, RSocketStrategies strategies) {
    return rSocket
        .map(socket -> RSocketRequester.wrap(socket, MimeTypeUtils.APPLICATION_JSON, strategies))
        .cache();
}
```

Рисунок 5 – Создание метода

Параметры, предоставляемые этому методу, представляют собой RSocket поток, который был создан в предыдущем методе, и «RSocketStrategies» который представляет собой bean-компонент, созданный авто-конфигурацией RSocket.

Причина, по которой обертываем RSocket заключается в том, что он поддерживает реактивные потоки, но не содержит типов, представленных «Project Reactor», таких как, Mono и Flux, кроме того, пришлось бы выполнять сопоставление с «PersonMessage» объектами самостоятельно (рис.6).

```
private Flux<PersonMessage> findPeople(RSocketRequester requester) {
    return requester
        .route("findPeople")
        .data(DefaultPayload.create(""))
        .retrieveFlux(PersonMessage.class);
}
```

Рисунок 6 – Создание альтернативного метода

Теперь, когда были определены все строительные блоки для подключения к серверу RSocket, можно написать, «ApplicationRunner» который извлекает данные (рис.7).

```
@Bean
public ApplicationRunner consumer(Mono<RSocketRequester> requester) {
    return args -> requester
        .flatMapMany(this::findPeople)
        .map(PersonMessage::toString)
        .subscribe(log::info);
}
```

Рисунок 7 – Создание стартера

У RSocket есть подходящая альтернатива WebSockets или Server Sent Events, если необходимо отправлять данные по сети. Поддержка в Spring boot уже работает должным образом.

В данной работе был описан процесс передачи данных по сети, а так же создан сервер для отправки данных и клиент для их получения.

Библиографический список

1. Зарайский В.И. Разработка модуля автоматизации работы с конференциями в кафедральном приложении // Информатика, управляющие системы, математическое и компьютерное моделирование 2020. №3. С. 549-462.
2. Балакишиев Д. Ю. Использование java 8, spring boot и angular 2+ для разработки веб-приложения «справочник услуг» // Образовательные технологии и общество. 2019. №3. С. 145-154.
3. Ибраимов Р.И., Зайчик А.Р., Минзатов Н.С. Разработка генеалогического дерева средствами фреймвока spring boot // Молодость. Интеллект. Инициатива. 2017. С. 21-22.
4. Ибраимов Р.И., Джемалетдинов А.Б., Шевченко А.А. Spring boot: создание тестов для spring mvc контроллеров // Прикладная математика и фундаментальная информатика. 2020. №2. С. 23-28.
5. Голикова А.А., Строганов В.А. Система обработки и расширения данных новостной ленты // Теоретические основы программирования. Учебное пособие 2019. 137с.