(/)

Lee un InputStream usando el Java Server Socket

Última modificación: 24 de marzo de 2019

por baeldung (https://www.baeldung.com/author/baeldung/) (https://www.baeldung.com/author/baeldung/)

Java (https://www.baeldung.com/category/java/) +

1. Información general

Para enviar y recibir datos a través de una red, a menudo usamos sockets. Los sockets no son más que una combinación de una dirección IP y un número de puerto, que pueden identificar de forma única un programa que se ejecuta en cualquier máquina.

En este tutorial, mostraremos cómo podemos leer los datos que se nos envían a través de un socket.

2. Lectura de datos de un zócalo

Supongamos que tenemos una comprensión básica de la programación de socket (https://www.baeldung.com/a-guide-to-java-sockets) .

Ahora, profundizaremos en la lectura de datos en un puerto en el que nuestro servidor está escuchando.

En primer lugar, debemos declarar e inicializar las variables de *ServerSocket, Socket* y *DataInputStream* :

```
1 ServerSocket server = new ServerSocket(port);
2 Socket socket = server.accept();
3 DataInputStream in = new DataInputStream(new BufferedInputStream(socke
```

Tenga en cuenta que hemos elegido envolver *InputStream* del socket en un *DataInputStream.* Esto nos permite leer líneas de texto y tipos de datos primitivos de Java de manera portátil.

Esto es bueno ya que ahora, si conocemos el tipo de datos que recibiremos, **podemos usar métodos especializados como *readChar (), readInt (), readDouble ()* y *readLine ().***

Sin embargo, puede ser un desafío si el tipo y la longitud de los datos no se conocen de antemano.

En ese caso, obtendremos un flujo de bytes del socket, en su lugar, utilizando la función de *lectura () de* nivel inferior . **Pero, hay un pequeño problema en este enfoque: ¿Cómo sabemos la longitud y el tipo de datos que obtendremos?**

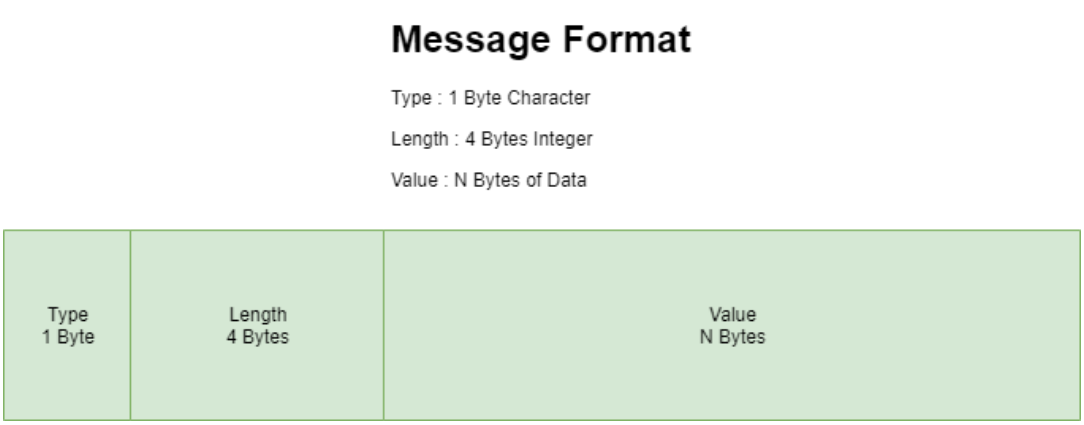
Vamos a explorar este escenario en la siguiente sección.

3. Leyendo datos binarios desde un socket

Al leer datos en bytes, debemos definir nuestro propio protocolo para la comunicación entre el servidor y el cliente. El protocolo más simple que podemos definir se llama TLV (Type Length Value). Significa que cada mensaje escrito en el socket tiene la forma del valor de longitud de tipo.

Así definimos cada mensaje enviado como:

- Un carácter de 1 *byte* que representa el tipo de datos, como *s* para *String*
- Un entero de 4 *bytes* que indica la longitud de los datos.
- Y luego los datos reales, cuya longitud se acaba de indicar.



(https://www.baeldung.com/wp-content/uploads/2019/03/figure1-1-1.png)
Una vez que el cliente y el servidor establezcan la conexión, cada mensaje seguirá este formato. Luego, podemos escribir nuestro código para analizar cada mensaje y leer *n* bytes de datos de un tipo específico.

Veamos cómo podemos implementar esto usando un ejemplo simple con un mensaje de *cadena* .

Primero, debemos llamar a la función *readChar ()* , para leer el tipo de datos y luego llamar a la función *readInt ()* para leer la longitud de la misma:

```
1 char dataType = in.readChar();
2 int length = in.readInt();
```

Después de eso, necesitamos leer los datos que estamos recibiendo. **Un punto importante a tener en cuenta aquí es que la función *read ()* podría no ser capaz de leer todos los datos en una llamada. Por lo tanto, debemos llamar a *read ()* en un bucle while:**

```
if(dataType == 's') {
    byte[] messageByte = new byte[length];
    boolean end = false;
    StringBuilder dataString = new StringBuilder(length);
    int totalBytesRead = 0;
    while(!end) {
        int currentBytesRead = in.read(messageByte);
        totalBytesRead = currentBytesRead + totalBytesRead;
        if(totalBytesRead <= length) {
            dataString
              .append(new String(messageByte, 0, currentBytesRead, St
        } else {
            dataString
              .append(new String(messageByte, 0, length - totalBytesR
              StandardCharsets.UTF_8));
        }
        if(dataString.length()>=length) {
            end = true;
        }
    }
}
```

4. Código de cliente para enviar datos

¿ Y **qué pasa con el código del lado del cliente?** En realidad, es bastante simple:

```
char type = 's'; // s for string
String data = "This is a string of length 29";
byte[] dataInBytes = data.getBytes(StandardCharsets.UTF_8);

out.writeChar(type);
out.writeInt(dataInBytes.length);
out.write(dataInBytes);
```

¡Eso es todo lo que nuestro cliente está haciendo!

5. Conclusión

En este artículo, discutimos cómo leer datos de un socket. Analizamos diferentes funciones que nos ayudan a leer datos de un tipo en particular. Además, vimos cómo leer datos binarios.

La implementación completa de este tutorial se puede encontrar en GitHub (https://github.com/eugenp/tutorials/tree/master/core-java-networking/src/main/java/com/baeldung) .

Deja una respuesta

Start the discussion...

Suscribir

CATEGORÍAS

PRIMAVERA (HTTPS://WWW.BAELDUNG.COM/CATEGORY/SPRING/)
DESCANSO (HTTPS://WWW.BAELDUNG.COM/CATEGORY/REST/)
JAVA (HTTPS://WWW.BAELDUNG.COM/CATEGORY/JAVA/)
SEGURIDAD (HTTPS://WWW.BAELDUNG.COM/CATEGORY/SECURITY-2/)
PERSISTENCIA (HTTPS://WWW.BAELDUNG.COM/CATEGORY/PERSISTENCE/)
JACKSON (HTTPS://WWW.BAELDUNG.COM/CATEGORY/JSON/JACKSON/)
CLIENTE HTTP (HTTPS://WWW.BAELDUNG.COM/CATEGORY/HTTP/)
KOTLIN (HTTPS://WWW.BAELDUNG.COM/CATEGORY/KOTLIN/)

SERIE

TUTORIAL "VOLVER A LO BÁSICO" DE JAVA (/JAVA-TUTORIAL)
JACKSON JSON TUTORIAL (/JACKSON)
TUTORIAL HTTPCLIENT 4 (/HTTPCLIENT-GUIDE)
DESCANSO CON TUTORIAL DE PRIMAVERA (/REST-WITH-SPRING-SERIES)
TUTORIAL DE PERSISTENCIA DE PRIMAVERA (/PERSISTENCE-WITH-SPRING-SERIES)
SEGURIDAD CON SPRING (/SECURITY-SPRING)

ACERCA DE

ACERCA DE BAELDUNG (/ABOUT)
LOS CURSOS (HTTPS://COURSES.BAELDUNG.COM)
TRABAJO DE CONSULTORÍA (/CONSULTING)
META BAELDUNG (HTTP://META.BAELDUNG.COM/)
EL ARCHIVO COMPLETO (/FULL_ARCHIVE)
ESCRIBIR PARA BAELDUNG (/CONTRIBUTION-GUIDELINES)
EDITORES (/EDITORS)
NUESTROS COMPAÑEROS (/PARTNERS)
PUBLICIDAD EN BAELDUNG (/ADVERTISE)

TÉRMINOS DE SERVICIO (/TERMS-OF-SERVICE)
POLÍTICA DE PRIVACIDAD (/PRIVACY-POLICY)
INFORMACIÓN DE LA COMPAÑÍA (/BAELDUNG-COMPANY-INFO)

CONTACTO (/CONTACT)