Netty-Starter-With-Spring-Boot (사전 학습)

1. bit와 byte

왜 bit와 byte를 알아야 할까요?

📖 네트워크 통신 이해:
- 네트워크 통신은 데이터를 비트와 바이트 단위로 전송합니다.
- 비트와 바이트에 대한 이해가 없다면 네트워크 통신의 기본 원리를 이해하기 어려울 수 있습니다.

데이터 표현 이해:
- Netty는 데이터를 버퍼(buffer)에 저장하고 처리합니다.
- 이때 비트와 바이트 단위로 데이터를 다루게 됩니다.
- 비트와 바이트에 대한 이해가 없다면 Netty에서 데이터를 효과적으로 처리하기 어려울 수 있습니다.

성능 최적화:
- 네트워크 통신에서 데이터 크기는 성능에 큰 영향을 미칩니다.
- 비트와 바이트에 대한 이해를 바탕으로 데이터 크기를 최소화하여 성능을 향상시킬 수 있습니다.

디버깅 및 문제 해결:

- 네트워크 통신 문제를 해결할 때 비트와 바이트 단위로 데이터를 분석해야 할 경우가 있습니다.
- 이때 비트와 바이트에 대한 이해가 필수적입니다.

비트(bit)란?

📖 비트는 `Binary Digit`의 줄임말로 컴퓨터 과학에서 가장 작은 단위이며 0과 1의 값으로 구성(이진수)되어 있습니다.
이 bit는 컴퓨터에서 가장 기본이 되는 개념이며, 데이터를 표현하고 처리하는 데 사용됩니다.

비트의 특징

1. 이진 체계
    a. 이친 체계는 0과 1 두 개의 값을 사용하여 모든 데이터를 표현할 수 있는 방법(체계)입니다.
    b. 컴퓨터는 전기 신호(=전구)나 자기장(=자석) 등의 물리적인 수단을 사용하여 `bit`를 나타낼 수 있습니다.
    c. 만약 전구 스위치가 꺼져서 전류가 흐르지 않는 상태를 0으로 표현할 수 있을 것이고, 전구 스위치가 켜져서 전류가 흐르는 상태를 1로 표현할 수 있을 것입니다.
2. 데이터 표현
    a. `bit`를 사용하여 다양한 종류의 데이터를 표현할 수 있습니다.
    b. 예를 들면, 숫자, 문자, 이미지, 음성 등의 데이터를 이진 형태로 나타낼 수 있습니다.
    c. 컴퓨터에서 문자를 표현하기 위해서는 문자 인코딩 방식(ASCII, UTF-8)을 사용해서 `bit`로 매핑한 뒤 문자를 표현합니다.
    d. 예를 들면, 이진수 101을 인코딩 방식으로 변환하면 숫자 5가 됩니다.
3. 비트 묶음과 바이트
    a. `bit`는 컴퓨터에서 가장 작은 단위이며, 컴퓨터의 데이터를 나타내는 데 중요한 역할을 합니다.
    b. 여러 개의 `bit`를 조합해서 더 큰 데이터 단위를 표현할 수 있습니다.
    c. 8개의 `bit`를 묶어서 하나의 단위로 표현할 수 있는데, 이를 바이트(`byte`)라고 합니다.
    d. `byte`가 가장 일반적인 데이터 단위이지만, 다른 단위도 존재합니다.
4. 바이트(Byte)
    a. 컴퓨터는 일반적으로 8개 단위의 비트(`8 bit`)를 하나의 그룹으로 사용하는데 이를 바이트(`byte`)라 합니다. 즉 (`8 bit` = `1 byte`)

b. `byte`는 많은 컴퓨터 시스템에서 기본적인 데이터 단위로 사용된다. `byte`가 가장 일반적인 데이터 단위입니다.

c. `byte`는 256(2^8)가지의 서로 다른 값을 나타낼 수 있으며, 문자, 숫자, 그래픽 등을 표현하는 데 사용됩니다.

d. 예를 들면, 파일 크기, 메모리 용량 등을 `byte` 단위로 표현할 수 있습니다.

5. 컴퓨터 구성

a. 컴퓨터의 CPU(중앙 처리 장치), 메모리(Memory), 저장 장치(Storage) 등은 모두 `bit`를 사용하여 데이터를 처리합니다.

b. `bit`는 컴퓨터의 처리 속도, 저장 용량, 통신 속도 등을 결정하는 중요한 요소입니다.

c. 컴퓨터의 비트 수는 해당 컴퓨터 시스템이 처리할 수 있는 데이터의 크기와 범위를 결정합니다.

d. 비트 수가 많을수록 컴퓨터의 처리 속도, 저장 용량, 통신 속도 등이 향상됩니다.

6. 컴퓨터 아키텍처(Architecture)

a. 컴퓨터 시스템의 아키텍처는 bit의 크기에 따라 결정됩니다.

b. 예를 들어, 32비트 아키텍처는 각 워드(`word`)가 32비트로 구성되어 있고, 64비트 아키텍처는 각 워드(`word`)가 64비트로 구성되어 있다.

i. 워드(`word`) : 여기서 워드는 컴퓨터 구조에서 하나의 연산을 통해 저장 장치로부터 프로세서의 레지스터에 옮겨놓을 수 있는 데이터 단위를 의미합니다.

ii. 즉, 컴퓨터에서 저장할 수 있는 데이터(연산)의 단위입니다.

c. 윈도우 `32 bit` : `4,294,967,296`의 데이터 크기 처리 가능합니다.

d. 윈도우 `64 bit` : `18,446,744,073,709,600,000`의 데이터 크기 처리 가능합니다.

---

bit 연산 실습 예제

```
public class BitOperationExample {
    public static void main(String[] args) {
                // 비트 연산 실습
        int a = 0b1010;// 10진수로 10
        int b = 0b0110;// 10진수로 6// AND 연산
        int andResult = a & b;// 결과: 0b0010 (2)
        System.out.println("AND 연산 결과: " + andResult);
```

```
                // OR 연산
        int orResult = a | b;// 결과: 0b1110 (14)
        System.out.println("OR 연산 결과: " + orResult);

                // XOR 연산
        int xorResult = a ^ b;// 결과: 0b1100 (12)
        System.out.println("XOR 연산 결과: " + xorResult);

                // NOT 연산
        int notResult = ~a;// 결과: 0b0101 (-11)
        System.out.println("NOT 연산 결과: " + notResult);
    }
}
```

바이트 단위 데이터 처리 실습 예제

```
public class ByteOperationExample {
    public static void main(String[] args) {
                // 바이트 단위 데이터 처리 실습
        byte b1 = 0b01000001;// 65 (ASCII 'A')
        byte b2 = 0b00110100;// 52 (ASCII '4')// 바이트 값 확
인
        System.out.println("b1: " + b1);
        System.out.println("b2: " + b2);

                // 바이트 배열 생성
        byte[] bytes = {b1, b2};

                // 바이트 배열 출력
        for (byte b : bytes) {
            System.out.print((char) b);// 출력: A4
        }
        System.out.println();
    }
}
```