(디지털컨버전스) 스마트 콘텐츠와 웹 융합 응용 SW개발자 양성과정

-21일차 학습 및 질문 노트-

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – Kyeonghwan Lee(이경환)
airtrade7@naver.com

```
public class AsynchronousPatternDoc {
}

/* 여기서 가져가야할 주요 개념

1. Thread를 활용하는 이유는 성능을 빠르게 만들기 위함이다.
2. 비동기 패턴(Asynchronous Pattern)이란 전부 Thread를 기반으로 한다.
3. 자바 스크립트 또한 Multi Thread 모델을 지원한다(자체적으로)
   (이건 최신 자바스크립트 ECMA 6 부터 서포트인것 같음) - Promise를 활용하여 증명
4. Thread를 사용할 때는 Critical Section에 대한 방어가 무엇보다도 중요하다(데이터 무결성)
5. 또한 스레드는 비동기 처리를 하기 때문에 데이터의 완전한 전송을 보장하지 못할 수도 있다.
   (말이 좀 어려운데 이 부분은 자바스크립트의 Promise를 통해 살펴볼 예정)

ex) 전화 통화: 동기 처리
    왜 ? 친구한테 전화를 걸었음. 친구가 통화 허용을 안하면 통화가 안됨

ex) 카카오톡 메시지: 비동기 처리
    왜 ? 상대방이 확인하던 안하던 난 보낸다.
    나는 니가 뭘 하던 내 할 일을 하겠다.
```

동기방식(Synchronous)

동기 방식은 서버에서 요청을 보냈을 때 응답이 돌아와야 다음 동작을 수행할 수 있다.
즉 A작업이 모두 진행 될 때까지 B작업은 대기해야한다.

비동기방식(Asynchronous)

비동기 방식은 반대로 요청을 보냈을 때 응답 상태와 상관없이 다음 동작을 수행 할 수 있다.
즉 A작업이 시작하면 동시에 B작업이 실행된다.
A작업은 결과값이 나오는 대로 출력된다.

```java
public class SortingTest {
    public static void main(String[] args) {
        String[] sample = {"I", "walk", "the", "line", "Apple", "hit", "me", "Ground", "attack", "you"};

        List<String> list = Arrays.asList(sample);

        // 정렬 법칙(대문자 우선, 그 다음 소문자)
        Collections.sort(list);

        System.out.println(list);

        Integer[] numbers = {1, 2, 3, 100, 77, 2342, 2342354, 345, 12323, 12, 4};

        List<Integer> numList = Arrays.asList(numbers);

        Collections.sort(numList);

        System.out.println(numList);

        Set fruits = new HashSet();

        fruits.add("strawberry");
        fruits.add("watermelon");
        fruits.add("grape");
        fruits.add("orange");
        fruits.add("apple");
        fruits.add("banana");

        List fruitsList = new ArrayList(fruits);
        fruitsList.add("ofcourse");

        Collections.sort(fruitsList);
        System.out.println(fruitsList);
        Collections.reverse(fruitsList);
        System.out.println(fruitsList);
```

Collections.sort(배열)
->오름차순으로 정렬된다.

Collections.reverse(배열)
->내림차순으로 정렬된다.

***Collections.sort()는 리스트로 변환하는 과정이 반드시 필요하다.**

```
[Apple, Ground, I, attack, hit, line, me, the, walk, you]
[1, 2, 3, 4, 12, 77, 100, 345, 2342, 12323, 2342354]
[apple, banana, grape, ofcourse, orange, strawberry, watermelon]
[watermelon, strawberry, orange, ofcourse, grape, banana, apple]

Process finished with exit code 0
```

```java
import java.net.MalformedURLException;
import java.net.URL;

public class NetworkUrlTest {
    // Malform 이라는것이 악성 코드에 해당해서
    // 이상한 URL로 링크를 태워서 공격을 할 수 있기 때문에 그것에 대한 방어 조치라 보면 됨
    // www.daum.net, http://www.daum.net
    // URL을 반드시 후자로 줘야 합니다.
    // 이유는 www.daum.net 으로 하면 위와 같이 악성코드 공격이 가능함
    public static void main(String[] args) throws MalformedURLException {
        URL myURL = new URL( spec: "http://www.loanconsultant.or.kr/source/index.jsp?t=20191216");

        // Protocol: HTTP(웹 애플리케이션 전용 프로토콜입니다)
        System.out.println("Protocol = " + myURL.getProtocol());
        System.out.println("authority = " + myURL.getAuthority());
        System.out.println("host = " + myURL.getHost());
        System.out.println("port = " + myURL.getPort());
        System.out.println("path = " + myURL.getPath());
        System.out.println("query = " + myURL.getQuery());
        System.out.println("filename = " + myURL.getFile());
        System.out.println("ref = " + myURL.getRef());
    }
}
```

Malform

악성코드에 대한 방어 조치

http vs https

**HTTP는 암호화가 추가되지 않았기 때문에 보안에 취약한 반면, HTTPS는 안전하게 데이터를 주고받을 수 있다. 하지만 HTTPS를 이용하면 암호화/복호화의 과정이 필요하기 때문에 HTTP보다 속도가 느리다.
또한 HTTPS는 인증서를 발급하고 유지하기 위한 추가 비용이 발생하다.**

21일차(06.07)

```java
class FrequencyChecker {
    Set<Integer> frequencySet;
    Map<Integer, Integer> frequencyMap;
    int[] backUp;

    public FrequencyChecker (int[] arr) {
        frequencySet = new HashSet<Integer>();
        frequencyMap = new HashMap<Integer, Integer>();

        backUp = arr;

        for (Integer elem : arr) {
            frequencySet.add(elem);
            frequencyMap.put(elem, 0);
        }

        System.out.println("frequencySet: " + frequencySet);
        System.out.println("frequencyMap: " + frequencyMap);
    }

    public void allocRandomFrequency (int num) {
        for (int i = 0; i < num; i++) {
            int tmp = (int) (Math.random() * 10);
            int key = backUp[tmp];

            System.out.printf("%6d", key);

            if (i % 20 == 19) {
                System.out.println();
            }

            if (frequencySet.contains(key)) {
                int cnt = frequencyMap.get(key);
                frequencyMap.put(key, ++cnt);
            }
            /*
            else {
                frequencyMap.put(key, 1);
            }
```

```java
    public Map<Integer, Integer> getFrequencyMap() {
        return frequencyMap;
    }
}

public class Prob60 {
    public static void main(String[] args) {
        int[] testSet = {
                2400,  5000, 1000, 200, 6000,
                77000, 434,  768,  20,  50
        };

        FrequencyChecker fc = new FrequencyChecker(testSet);

        fc.allocRandomFrequency( num: 20);

        System.out.println(fc.getFrequencyMap());
        //System.out.println(fc);
    }
}
```

```
frequencySet: [2400, 6000, 768, 434, 50, 20, 5000, 1000, 200, 77000]
frequencyMap: {2400=0, 6000=0, 768=0, 434=0, 50=0, 20=0, 5000=0, 1000=0, 200=0, 77000=0}
  5000  5000  1000  1000   200  5000  6000    20  5000  6000   434  1000  6000  5000   200 77000 77000  5000  1000  6000
{2400=0, 6000=4, 768=0, 434=1, 50=0, 20=1, 5000=6, 1000=4, 200=2, 77000=2}

Process finished with exit code 0
```