(디지털 컨버전스) 스마트 콘텐츠와 웹 융합 응용 SW개발자 양성과정

훈련기간 : 2021.05.07 ~ 2021.12.08

KH정보교육원 | 전승리

비동기 패턴

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.
    Thread를 사용하지 않은 예를 보고싶다면 오래된 사이트를 확인해보면 알 수있다.
    Thread 사용시 탭간 이동시 부분적으로 바뀌지만 사용하지 않은경우는 화면전체가 새 창처럼 전환이 되어진다.

네트워크

```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());
    }
}
```

```
Protocol = http
authority = www.loanconsultant.or.kr
host = www.loanconsultant.or.kr
port = -1
path = /source/index.jsp
query = t=20191216
filename = /source/index.jsp?t=20191216
ref = null
```

현재는 http에서 보안측면에서 보완된 https를 주로 사용한다.
http의 경우 포트번호 80 https는 포트번호 443 이다. ex) http://www.naver.com:80

Collection FrameWork로 중복체크

```java
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;

class FrequencyChecker {
    HashSet<Integer> frequencySet;
    Map<Integer, Integer> frequancyMap;
    int[] backup;

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

        backup = arr; //입력받은 배열 백업배열에 저장

        for (Integer elem : arr) { //입력받은 배열 set과 map의 key값에 추가
            frequencySet.add(elem);
            frequancyMap.put(elem, 0);

        }
    }
```

```java
public Map<Integer, Integer> getFrequancyMap() { //중복된 수 확인하기위한 getter
    return frequancyMap;
}

public void allocRandomFrequency(int num) {
    for (int i = 0; i < num; i++) {
        int tmp = (int) (Math.random() * 10);
        int key = backup[tmp]; //랜덤생성번호에 위치한 배열정보를 key 값에 대입

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

        if (i % 5 == 4) { //줄 바꿈
            System.out.println();
        }

        if (frequencySet.contains(key)) { //키값이 포함되어있으면 true
            int cnt = frequancyMap.get(key);//해당 key 해당하는 value 값을 가져온다는 것이다
            frequancyMap.put(key, ++cnt); // 값 증가
        }
    }
```

```java
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.getFrequancyMap());
    }
}
```

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