

직렬화

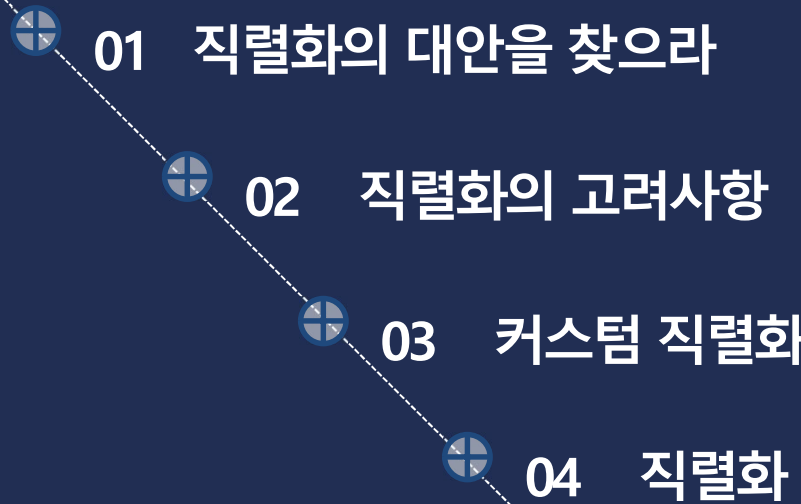
Serializable

JooYeon Han
한 주 연

직렬화

Serializable

CONTENTS

- 
- 01 직렬화의 대안을 찾으라
 - 02 직렬화의 고려사항
 - 03 커스텀 직렬화
 - 04 직렬화 프록시

01

Item 85

직렬화의 위험성

“직렬화는 위험하다.”

직렬화의 위험성을 회피하는 가장 좋은 방법은
아무것도 역 직렬화하지 않는 것이다.



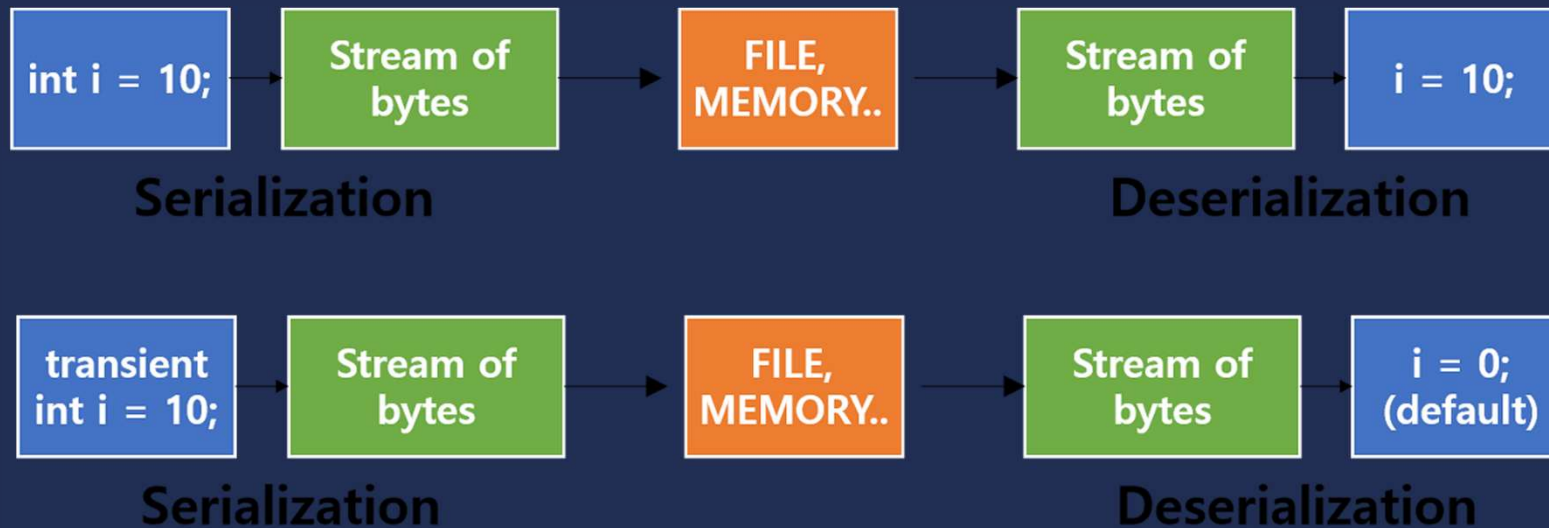
01

Item 85

직렬화란?

직렬화 : 객체를 직렬화하여 전송 가능한 형태로 만드는 것을 의미

역 직렬화 : 직렬화된 파일 등을 역으로 직렬화하여 다시 객체의 형태로 만드는 것



01

Item 85

역직렬화의 취약한 보안

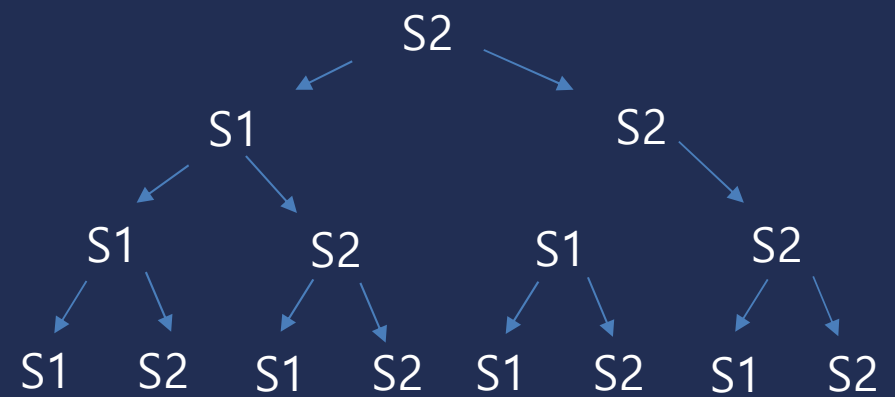
1. OuputInputStream의 readObject
2. Gaget (가젯)
3. 역 직렬화 폭탄



01

```

graph TD
    S1_0[S1] --> S1_1[S1]
    S1_0 --> S2_1[S2]
    S1_1 --> S1_2[S1]
    S1_1 --> S2_2[S2]
    S2_1 --> S1_3[S1]
    S2_1 --> S2_3[S2]
    S1_2 --> S1_4[S1]
    S1_2 --> S2_4[S2]
    S2_2 --> S1_5[S1]
    S2_2 --> S2_5[S2]
    S1_3 --> S1_6[S1]
    S1_3 --> S2_6[S2]
    S2_3 --> S1_7[S1]
    S2_3 --> S2_7[S2]
    S1_4 --> S1_8[S1]
    S1_4 --> S2_8[S2]
    S2_4 --> S1_9[S1]
    S2_4 --> S2_9[S2]
    S1_5 --> S1_10[S1]
    S1_5 --> S2_10[S2]
    S2_5 --> S1_11[S1]
    S2_5 --> S2_11[S2]
    S1_6 --> S1_12[S1]
    S1_6 --> S2_12[S2]
    S2_6 --> S1_13[S1]
    S2_6 --> S2_13[S2]
    S1_7 --> S1_14[S1]
    S1_7 --> S2_14[S2]
    S2_7 --> S1_15[S1]
    S2_7 --> S2_15[S2]
    S1_8 --> S1_16[S1]
    S1_8 --> S2_16[S2]
    S2_8 --> S1_17[S1]
    S2_8 --> S2_17[S2]
    S1_9 --> S1_18[S1]
    S1_9 --> S2_18[S2]
    S2_9 --> S1_19[S1]
    S2_9 --> S2_19[S2]
    S1_10 --> S1_20[S1]
    S1_10 --> S2_20[S2]
    S2_10 --> S1_21[S1]
    S2_10 --> S2_21[S2]
    S1_11 --> S1_22[S1]
    S1_11 --> S2_22[S2]
    S2_11 --> S1_23[S1]
    S2_11 --> S2_23[S2]
    S1_12 --> S1_24[S1]
    S1_12 --> S2_24[S2]
    S2_12 --> S1_25[S1]
    S2_12 --> S2_25[S2]
    S1_13 --> S1_26[S1]
    S1_13 --> S2_26[S2]
    S2_13 --> S1_27[S1]
    S2_13 --> S2_27[S2]
    S1_14 --> S1_28[S1]
    S1_14 --> S2_28[S2]
    S2_14 --> S1_29[S1]
    S2_14 --> S2_29[S2]
    S1_15 --> S1_30[S1]
    S1_15 --> S2_30[S2]
    S2_15 --> S1_31[S1]
    S2_15 --> S2_31[S2]
    S1_16 --> S1_32[S1]
    S1_16 --> S2_32[S2]
    S2_16 --> S1_33[S1]
    S2_16 --> S2_33[S2]
    S1_17 --> S1_34[S1]
    S1_17 --> S2_34[S2]
    S2_17 --> S1_35[S1]
    S2_17 --> S2_35[S2]
    S1_18 --> S1_36[S1]
    S1_18 --> S2_36[S2]
    S2_18 --> S1_37[S1]
    S2_18 --> S2_37[S2]
    S1_19 --> S1_38[S1]
    S1_19 --> S2_38[S2]
    S2_19 --> S1_39[S1]
    S2_19 --> S2_39[S2]
    S1_20 --> S1_40[S1]
    S1_20 --> S2_40[S2]
    S2_20 --> S1_41[S1]
    S2_20 --> S2_41[S2]
    S1_21 --> S1_42[S1]
    S1_21 --> S2_42[S2]
    S2_21 --> S1_43[S1]
    S2_21 --> S2_43[S2]
    S1_22 --> S1_44[S1]
    S1_22 --> S2_44[S2]
    S2_22 --> S1_45[S1]
    S2_22 --> S2_45[S2]
    S1_23 --> S1_46[S1]
    S1_23 --> S2_46[S2]
    S2_23 --> S1_47[S1]
    S2_23 --> S2_47[S2]
    S1_24 --> S1_48[S1]
    S1_24 --> S2_48[S2]
    S2_24 --> S1_49[S1]
    S2_24 --> S2_49[S2]
    S1_25 --> S1_50[S1]
    S1_25 --> S2_50[S2]
    S2_25 --> S1_51[S1]
    S2_25 --> S2_51[S2]
    S1_26 --> S1_52[S1]
    S1_26 --> S2_52[S2]
    S2_26 --> S1_53[S1]
    S2_26 --> S2_53[S2]
    S1_27 --> S1_54[S1]
    S1_27 --> S2_54[S2]
    S2_27 --> S1_55[S1]
    S2_27 --> S2_55[S2]
    S1_28 --> S1_56[S1]
    S1_28 --> S2_56[S2]
    S2_28 --> S1_57[S1]
    S2_28 --> S2_57[S2]
    S1_29 --> S1_58[S1]
    S1_29 --> S2_58[S2]
    S2_29 --> S1_59[S1]
    S2_29 --> S2_59[S2]
    S1_30 --> S1_60[S1]
    S1_30 --> S2_60[S2]
    S2_30 --> S1_61[S1]
    S2_30 --> S2_61[S2]
    S1_31 --> S1_62[S1]
    S1_31 --> S2_62[S2]
    S2_31 --> S1_63[S1]
    S2_31 --> S2_63[S2]
    S1_32 --> S1_64[S1]
    S1_32 --> S2_64[S2]
    S2_32 --> S1_65[S1]
    S2_32 --> S2_65[S2]
    S1_33 --> S1_66[S1]
    S1_33 --> S2_66[S2]
    S2_33 --> S1_67[S1]
    S2_33 --> S2_67[S2]
    S1_34 --> S1_68[S1]
    S1_34 --> S2_68[S2]
    S2_34 --> S1_69[S1]
    S2_34 --> S2_69[S2]
    S1_35 --> S1_70[S1]
    S1_35 --> S2_70[S2]
    S2_35 --> S1_71[S1]
    S2_35 --> S2_71[S2]
    S1_36 --> S1_72[S1]
    S1_36 --> S2_72[S2]
    S2_36 --> S1_73[S1]
    S2_36 --> S2_73[S2]
    S1_37 --> S1_74[S1]
    S1_37 --> S2_74[S2]
    S2_37 --> S1_75[S1]
    S2_37 --> S2_75[S2]
    S1_38 --> S1_76[S1]
    S1_38 --> S2_76[S2]
    S2_38 --> S1_77[S1]
    S2_38 --> S2_77[S2]
    S1_39 --> S1_78[S1]
    S1_39 --> S2_78[S2]
    S2_39 --> S1_79[S1]
    S2_39 --> S2_79[S2]
    S1_40 --> S1_80[S1]
    S1_40 --> S2_80[S2]
    S2_40 --> S1_81[S1]
    S2_40 --> S2_81[S2]
    S1_41 --> S1_82[S1]
    S1_41 --> S2_82[S2]
    S2_41 --> S1_83[S1]
    S2_41 --> S2_83[S2]
    S1_42 --> S1_84[S1]
    S1_42 --> S2_84[S2]
    S2_42 --> S1_85[S1]
    S2_42 --> S2_85[S2]
    S1_43 --> S1_86[S1]
    S1_43 --> S2_86[S2]
    S2_43 --> S1_87[S1]
    S2_43 --> S2_87[S2]
    S1_44 --> S1_88[S1]
    S1_44 --> S2_88[S2]
    S2_44 --> S1_89[S1]
    S2_44 --> S2_89[S2]
    S1_45 --> S1_90[S1]
    S1_45 --> S2_90[S2]
    S2_45 --> S1_91[S1]
    S2_45 --> S2_91[S2]
    S1_46 --> S1_92[S1]
    S1_46 --> S2_92[S2]
    S2_46 --> S1_93[S1]
    S2_46 --> S2_93[S2]
    S1_47 --> S1_94[S1]
    S1_47 --> S2_94[S2]
    S2_47 --> S1_95[S1]
    S2_47 --> S2_95[S2]
    S1_48 --> S1_96[S1]
    S1_48 --> S2_96[S2]
    S2_48 --> S1_97[S1]
    S2_48 --> S2_97[S2]
    S1_49 --> S1_98[S1]
    S1_49 --> S2_98[S2]
    S2_49 --> S1_99[S1]
    S2_49 --> S2_99[S2]
    S1_50 --> S1_100[S1]
    S1_50 --> S2_100[S2]
    S2_50 --> S1_101[S1]
    S2_50 --> S2_101[S2]
    S1_51 --> S1_102[S1]
    S1_51 --> S2_102[S2]
    S2_51 --> S1_103[S1]
    S2_51 --> S2_103[S2]
    S1_52 --> S1_104[S1]
    S1_52 --> S2_104[S2]
    S2_52 --> S1_105[S1]
    S2_52 --> S2_105[S2]
    S1_53 --> S1_106[S1]
    S1_53 --> S
```



01

Item 85

직렬화의 대안

1. 크로스 - 플랫폼 데이터 표현

Json / Protocol Buffer

2. 객체 역직렬화 필터링



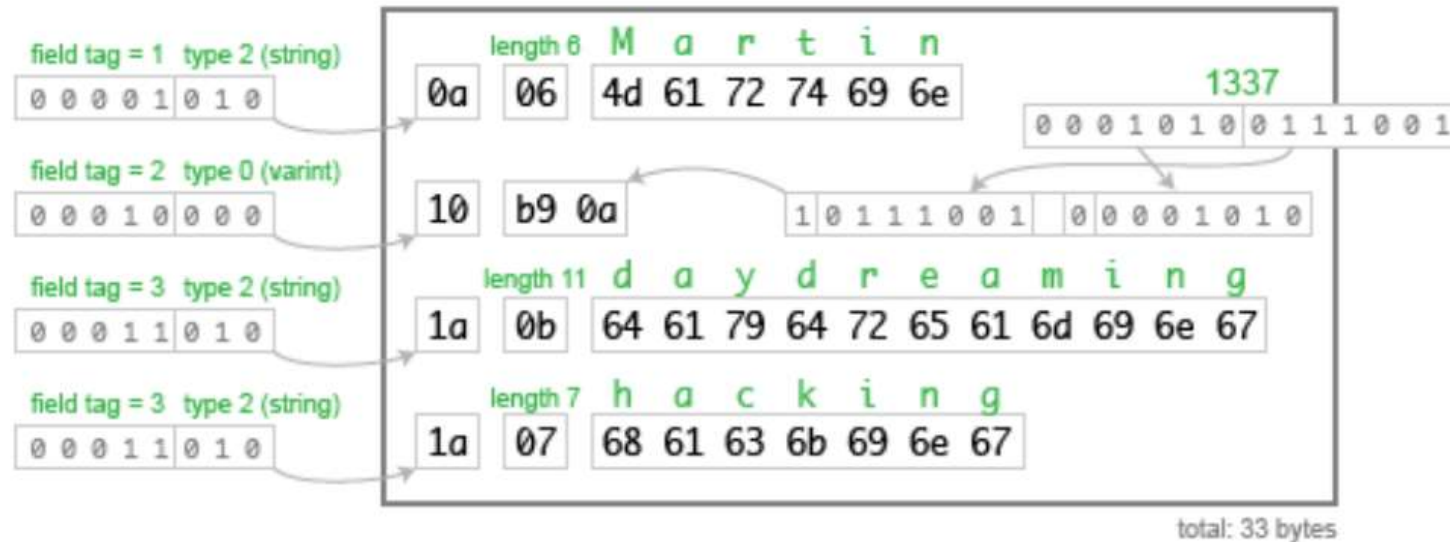
01

Item 85

Protocol Buffer

```
{  
  "userName": "Martin",  
  "favouriteNumber": 1337,  
  "interests": ["daydreaming", "hacking"]  
}
```

Protocol Buffers



- 릴리즈 후에 수정이 어렵다
- 역직렬화는 숨은 생성자이다
 - 신 버전을 릴리즈할 때 테스트 요소가 많아진다
- 상속용 클래스와 인터페이스는 직렬화를 구현하면 안된다
 - 내부 클래스는 직렬화를 구현하면 안된다



02

Item 86

SerialVersionUID

- SUID : 직렬화 된 클래스가 갖는 고유 식별 번호

```
class Article implements Serializable {  
    // 간단한 예를 들기 위해 간단한 값으로 선언합니다.  
    private static final long serialVersionUID = 1L;  
  
    private String title;  
    private String pressName;  
    private String reporterName;  
  
    // ... 이하 생략  
}
```



02

Item 86

SerialVersionUID

개발자가 SUID를 관리한 상황에서 클래스의 필드를 수정 한 뒤 역직렬화

- 멤버 변수를 추가할 때
객체의 해당 필드가 기본값으로 초기화 된다.
- 멤버 변수가 삭제될 때
값 자체가 사라진다.
- 멤버 변수의 이름이 바뀔 때
값이 할당 되지 않는다
- 멤버 변수의 타입이 바뀔 때
기존 멤버 변수의 타입이 변경되면 역직렬화 과정에서 `ClassCastException`이 발생할 수 있다.
- 접근 지정자의 변경
직렬화에 영향을 주지 않는다.



03

Item 87

커스텀 직렬화

```
public final class StringList implements Serializable {
    private transient int size = 0;
    private transient Entry head = null;
    private static class Entry {
        String data;
        Entry next;
        Entry previous;
    }

    // 문자열을 리스트에 추가한다.
    public final void add(String s) { ... }

    private void writeObject(ObjectOutputStream stream)
        throws IOException {
        stream.defaultWriteObject();
        stream.writeInt(size);

        // 모든 원소를 순서대로 기록한다.
        for (Entry e = head; e != null; e = e.next) {
            s.writeObject(e.data);
        }
    }

    private void readObject(ObjectInputStream stream)
        throws IOException, ClassNotFoundException {
        stream.defaultReadObject();
        int numElements = stream.readInt();

        for (int i = 0; i < numElements; i++) {
            add((String) stream.readObject());
        }
    }
    // ... 생략
}
```

- **Transient**
Serialize하는 과정에 제외하고 싶은
경우 선언하는 키워드
- **defaultWriteObject**
non-static and non-transient 필드
를 현재 스트림에 작성한다.
- **defaultReadObject**
non-static and non-transient 필드
를 현재 스트림에서 읽는다.

```
class Period implements Serializable {  
    private final Date start; // final로 불변 선언 가능  
    private final Date end;  
  
    public Period(Date start, Date end) {  
        this.start = start;  
        this.end = end;  
    }  
  
    private static class SerializationProxy implements Serializable {  
        private static final long serialVersionUID = 2123123123;  
        private final Date start;  
        private final Date end;  
  
        public SerializationProxy(Period p) {  
            this.start = p.start;  
            this.end = p.end;  
        }  
  
        private Object readResolve() { // Deserialize 할 때 호출된다.  
            return new Period(start, end); // Object 생성  
        }  
    }  
  
    private Object writeReplace() { // Serialize할 때 호출된다.  
        return new SerializationProxy(this);  
    }  
    // deserialize할 때, 직접 Period로 역직렬화를 할 수 없게 한다.  
    private void readObject(ObjectInputStream stream) throws InvalidObjectException {  
        throw new InvalidObjectException("프록시가 필요해요.");  
    }  
}
```

감사합니다

THANK YOU

JooYeon Han
한 주 연