

# 실전 프로젝트1



12

# 외부 Library 연동(1) JSON

# 공지

- **출석확인**
  - 화요일 (대면 / 줌), 금요일 (줌), LMS 인 경우 HD LMS에서 자동출석체크
  - Hisnet 온라인 출석 각자 체크 : 수강과목 선택 > 강의정보 > 강의출석현황 메뉴에서 확인
- **개인프로젝트 제출**
  - 마감 : 10월 17일(일) 밤 12시 - HDLMS 과제로 제출
  - 제출물 : 제작보고서(PDF) + 데모(5분 동영상 또는 1Page 포스터 PDF)
  - 제작보고서 포함내용
    - 본인이 선택한 항목 목록과 항목별 희망 점수 기재
    - 최종 결과물이 올려진 github repository URI
    - 각 항목 목록별 구현 의도와 결과물(화면캡처 및 설명)
    - 개인프로젝트 제작과정 중 느낀 소감문 (50자 이상)

# 개인프로젝트 - SQLite 기반 TodoList App

- 제출물 : 제작보고서(PDF) + 데모(5분 동영상 또는 원페이지 포스터 PDF)
- 제출 마감 : 2021년 10월 17일(일) 밤 12시
- 수행 아이템 (100점 만점기준, 원하는 항목 선택할 것)

구분	기본필드(6개)	완료체크필드 추가	필드 2개 이상 추가
Item 추가	5	5	5
목록	5	5	5
카테고리	5	5	
수정	5	5	5
삭제	5	5	5

레코드 50개 이상 입력/사용 [10점]	Category Table 별도 생성/사용 [10점]
Multi-item 기능 (여러개 동시 삭제, 완료체크) [10점]	새로운 멋진 기능 제안/구현 [개당 최대 10점] [최대 3개 인정]

예1) JSON 내보내기/가져오기  
예2) 주기적인 항목을 한번에 추가하기  
매달 1일에 고향 다녀오기

# 외부 Library 연동 - JSON

- JSON 이란?
- JSON 의 특징 및 구조
- JSON 사용에 필요한 외부 라이브러리 소개
- GSON library 사용 예제
- JSON 을 이용한 데이터 가져오기/내보내기(객체 단위)
- JSON 을 이용한 데이터 가져오기/내보내기(ArrayList 단위)

# JSON이란?

- Javascript Object Notation
- 데이터 객체 전달을 위해 만들어진 텍스트 기반 표준 포맷으로 (키 or 속성, 값)의 쌍으로 구성됨
- 비동기 브라우저나 서버 통신을 위해 사용되며, XML 대체 포맷으로 사용됨
- <https://www.json.org/json-ko.html>

## XML

```
<menu>
  <name>콤비네이션 피자</name>
  <size>Large</size>
  <price>24000</price>
</menu>
```

## JSON

```
{
  "name": "콤비네이션 피자",
  "size": "Large",
  "price": 24000
}
```

# JSON 특징

- 내용이 함축적이고, 데이터 교환을 위한 최소한의 정보만을 포함
- XML에 비해 용량이 작으므로 전송 속도가 빠름
- 프로그래밍 언어에 독립적이고, 사용하기가 용이함
- 자바스크립트를 확장하여 만들어졌고, 자바스크립트 객체 표기법을 따름

Car.java

```
public class Car {  
    public String manufacturer;  
    public String model;  
    public double capacity;  
    public boolean accident;  
  
    public Car() {}  
}
```

User-defined format

```
Audi##A4##1.8##false  
Škoda##Octavia##2.0##true
```

```
"cars": [  
  {  
    "manufacturer": "Audi",  
    "model": "A4",  
    "capacity": 1.8,  
    "accident": false  
  },  
  {  
    "manufacturer": "Škoda",  
    "model": "Octavia",  
    "capacity": 2.0,  
    "accident": true  
  }  
]
```

JSON format

# JSON 구조

- 자료형
  - string, number, true, false, null, object, array
- 데이터 구조
  - Key : Value
- 데이터는 쉼표(,)로 나열됨
  - Object : {} 중괄호로 표시
    - { “name” : “홍길동”, “age” : 20 }
  - Array : [] 대괄호로 표시
    - { “name” : “홍길동”, “age” : 20 , “hobby” : [ “영화”, “독서” ] }



# JSON 예제

```
public class Car {  
    public String manufacturer;  
    public String model;  
    public double capacity;  
    public boolean accident;  
  
    public Car() {}  
  
    public Car(String manufacturer, String model,  
                Double capacity, boolean accident) {  
        this.manufacturer = manufacturer;  
        this.model = model;  
        this.capacity = capacity;  
        this.accident = accident;  
    }  
}
```

```
public class Person {  
    public String name;  
    public String surname;  
    public Car[] cars;  
    public int phone;  
    public transient int age;  
  
    public Person() {}  
  
    public Person(String name, String surname,  
                  int phone, int age, Car[] cars) {  
        this.name = name;  
        this.surname = surname;  
        this.cars = cars;  
        this.phone = phone;  
        this.age = age;  
    }  
}
```

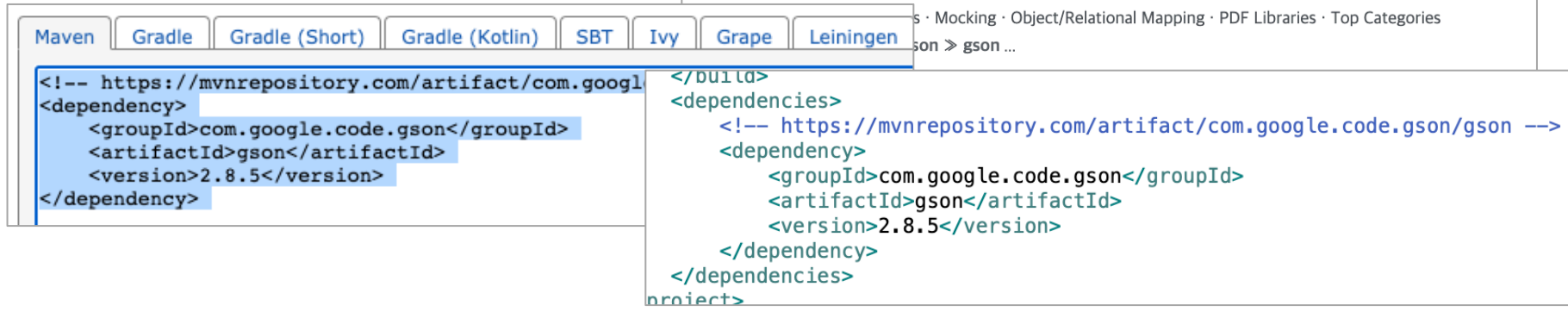
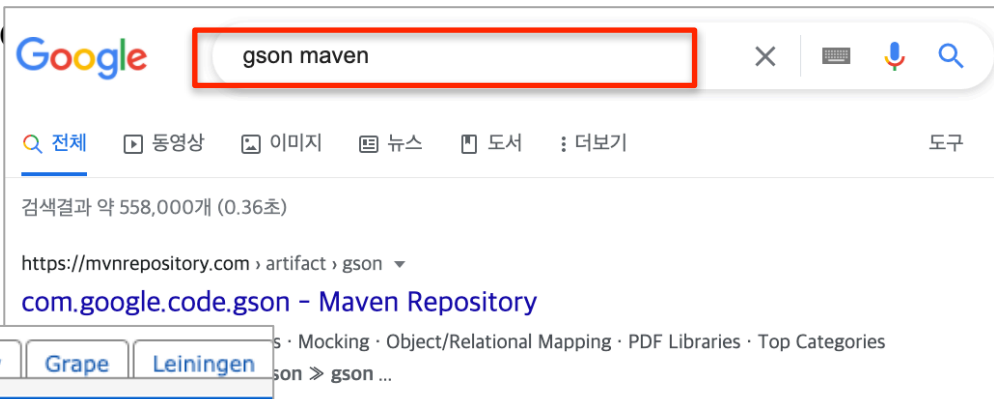
serializing 제외

JSON 저장 예시

```
{  
  "name": "John",  
  "surname": "Doe",  
  "cars": [  
    {"manufacturer": "Audi", "model": "A4", "capacity": 1.8, "accident": false},  
    {"manufacturer": "Škoda", "model": "Octavia", "capacity": 2.0, "accident": true}  
  ],  
  "phone": 2025550191  
}
```

# JSON Library

- 가장 많이 사용하는 Library : Json-simple library, GSON(Google Gson) library
- 자바 오브젝트를 JSON으로 직렬화, 역직렬화 해주는 오픈소스 라이브러리
- GSON library 다운로드 ( or maven dependency )
  - 원하는 라이브러리 검색
  - 버전 선택
  - Dependency 복사(Pom.xml)



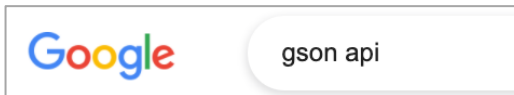
# GSON package

- GSON api 검색
- Module com.google.gson
  - Package com.google.gson(외 3개)
    - Interfaces/ Classes / Enums
    - Gson, GsonBuilder class
    - 직렬화(Serialization) : 객체 ➔ JSON

String **toJson**(Object src)

- 역직렬화(Deserialization) : JSON ➔ 객체

<T> T **fromJson**(String json, Class<T> classOfT) src)



# GSON 활용예제 project

1. 새 프로젝트 생성(Maven Project)
2. Gson library 추가
3. 데이터 클래스 생성
  - 이름 / 나이
4. Main함수 포함 클래스 생성
  - json 형식으로 파일 저장 기능
  - 파일에서 json형식으로 불러오기 기능

# GSON 사용예 : Serialize object to JSON

```
public class Student {
    private String name;
    private int age;

    public Student() {}

    public Student(String name, int age) {
        super();
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String toString() {
        return "Student [ name: " + name + ", age: " + age + " ]";
    }
}
```

```
public class GSONSample1 {

    public static void main(String[] args) {

        Gson gson = new Gson();

        //데이터 생성
        Student s1 = new Student("홍길동", 20);
        String jsonstr = gson.toJson(s1);
        System.out.println(jsonstr);

        //파일 저장
        try {
            FileWriter writer = new FileWriter("data.txt");
            writer.write(jsonstr);
            writer.close();
            System.out.println("파일에 저장되었습니다!");
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

Student.java GSONSample1.java data.txt ✕

```
1 {"name": "홍길동", "age": 20}
```

```
{"name": "홍길동", "age": 20}
파일에 저장되었습니다!
```

# GSON 사용예 : Deserialize JSON to Object

```
public class Student {  
    private String name;  
    private int age;  
  
    public Student() {}  
  
    public Student(String name, int age) {  
        super();  
        this.name = name;  
        this.age = age;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public int getAge() {  
        return age;  
    }  
  
    public void setAge(int age) {  
        this.age = age;  
    }  
  
    public String toString() {  
        return "Student [ name: " + name + ", age: " + age + " ]";  
    }  
}
```

```
// JSON String을 객체로 가져오기  
String jsonstr2 = null;  
try {  
    BufferedReader br = new BufferedReader(new FileReader("data.txt"));  
    jsonstr2 = br.readLine();  
    br.close();  
} catch (FileNotFoundException e) {  
    // TODO Auto-generated catch block  
    e.printStackTrace();  
} catch (IOException e) {  
    // TODO Auto-generated catch block  
    e.printStackTrace();  
}  
  
Student student = gson.fromJson(jsonstr2, Student.class);  
System.out.println(student);
```

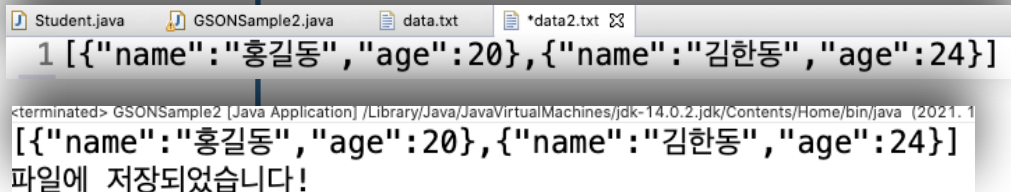
Student.java   GSONSample1.java   data.txt ✕

```
1 {"name": "홍길동", "age": 20}
```

파일에서 데이터를 가져왔습니다!  
Student [ name: 홍길동, age: 20 ]

# GSON 사용예 : ArrayList to json

```
public class GSONSample2 {  
  
    public static void main(String[] args) {  
  
        List<Student> list = new ArrayList<Student>();  
        Gson gson = new Gson();  
  
        //데이터 생성  
        list.add(new Student("홍길동", 20));  
        list.add(new Student("김한동", 24));  
        String jsonstr = gson.toJson(list);  
        System.out.println(jsonstr);  
  
        //파일 저장  
        try {  
            FileWriter writer = new FileWriter("data2.txt");  
            writer.write(jsonstr);  
            writer.close();  
            System.out.println("파일에 저장되었습니다!");  
        } catch (IOException e) {  
            // TODO Auto-generated catch block  
            e.printStackTrace();  
        }  
    }  
}
```



1 [{"name": "홍길동", "age": 20}, {"name": "김한동", "age": 24}]

<terminated> GSONSample2 [Java Application] /Library/Java/JavaVirtualMachines/jdk-14.0.2.jdk/Contents/Home/bin/java (2021. 11. 11. 14:00:00)  
[{"name": "홍길동", "age": 20}, {"name": "김한동", "age": 24}]  
파일에 저장되었습니다!

# GSON 사용예 : Json to ArrayList

```
// JSON String을 객체로 가져오기
String jsonstr2 = null;
try {
    BufferedReader br = new BufferedReader(new FileReader("data.txt"));
    jsonstr2 = br.readLine();
    br.close();

} catch (FileNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

```
System.out.println("파일에서 데이터를 가져왔습니다! ");
```

```
// case 1
```

```
Student[] array = gson.fromJson(jsonstr2, Student[].class);
List<Student> list1 = Arrays.asList(array);
System.out.println("list1 : " + list1);
```

```
// case 2
```

```
List<Student> list2 = gson.fromJson(jsonstr2, new TypeToken<List<Student>>(){}.getType());
System.out.println("list2 : " + list2);
```

파일에서 데이터를 가져왔습니다!

```
list1 : [Student [ name: 홍길동, age: 20 ], Student [ name: 김한동, age: 24 ]]
list2 : [Student [ name: 홍길동, age: 20 ], Student [ name: 김한동, age: 24 ]]
```