컴퓨터 기본 용어 쉽게 이해하깅

- 코어의 수 = 내가 부릴 노예의 수
- 쓰레드 = 내가 부릴 노예의 손 갯수
- 오버블럭 = 노예가 발까지 써서 옥수수 수확
- 캐쉬 메모리 = 노예가 짊어지고 있는 가방
- 램 = 노예가 준비한 리어카. —> 많으면 좋다. 옥수수 수확 많이 했는데 리어카가 작아서 한번에 창고로 못가져가는 상황 생김
- 하드 디스크 = 수확한 옥수수의 저장 창고
- ex) i7 10900 뜻은 2010년도생의 7마리 노예가 손이 900개라서 옥수수 수확 빠르다. 10900은 10년 900개의 손갯수임

컴파일

- 컴파일이란 프로그래머가 작성한 소스코드를 바이너리 파일로 변환하는 과정이다
- 컴파일은 사람이 이해하는 언어를 컴퓨터가 이해할 수 있는 언어로 바꾸어 주는 과정

컴파일러는 컴파일을 자동으로 수행해주는 소프트웨어
빌드는 컴파일, 패키징 등 최종 소프트웨어 실행파일을 만들기 위한 일련의 과정을 포함한다.

빌드 동작 순서

1.개발자가 원시코드(우리가 작성한 코드) 작성 실시
2.원시코드를 목적코드(컴퓨터가 이해할 수 있도록 번역)로 변경 실시(컴파일러)
3.목적코드를 실행 파일로 변경하는 링크 과정 진행

로직 : 어떠한 목표 결과물을 만들기 위한 과정.
Ex) 1부터 10까지 합계를 구하는 로직을 만든다고 하면
가장 무식한 로직으로 1+2+3+4+5+6+7+8+9+10을 해서 합계를 구하는 과정이 있고
두번째로는 반복문을 통해서 1부터 10까지 반복시킨후 값을 누적시키는 과정이 있고
세번째로는 첫수와 끝수를 더하고 전체 길이의 반으로 곱하는 공식화 방법이 있다(1+10) *(10/2)

===> 이렇게 목적을 위한 과정을 로직이라 한다.

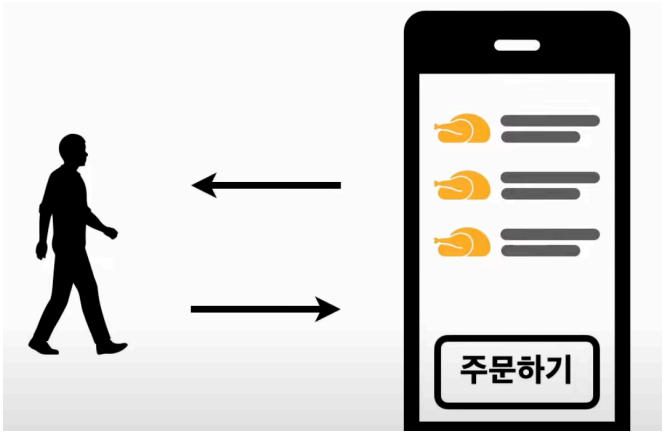
정적 컨텐츠 : 서버에서 하는것 없이 파일을 웹브라우저에 그대로 내보내는것
mvc(model View Controller)와 템플릿 엔진 : html에 해논것만 보내주는게 아니라 html을 서버가 프로그래밍을 해서 동적으로 바꿔서 보내줌

인터페이스

- 상호간에(사물과 사물or 사물과 사람 or 사람과 사람) 소통을 위해 만들어진 접점이라 생각하면 된다.
- 인터페이스에 소통이라 하면 쉽게 말해 읽거나 쓰는 걸 말한다.

위 사진 예시 처럼 사람과 자동차와의 소통의 접접인 키가 인터페이스라 할수 있다.
또한 우리가 컴퓨터 메모장에 일기를 적으려면 키보드가 필요한데 여기서도 키보드(쓰는 write)가 인터페이스이다. 또한 컴퓨터가 사람에게 화면을 보여주는 모니터(읽기 read)도 인터페이스가 된다.

좀 더 나아가 어플에서 보이는 메뉴 목록, 주문 버튼등 모든 화면을 인터페이스라 할 수 있다.
배달앱이라는 시스템은 화면을 통해서 사용자의 소통의 역할을 해주고 있기 때문이다.



이처럼
UI(User

사용자가 사용하는 인터페이스를 Interface)라고 한다.

앞서 말
웨어적
프트웨어

한 자동차 키와 모니터 키보드는 하드인 ui라고 보고 앱과 같은 것들은 소적인ui라고 한다.

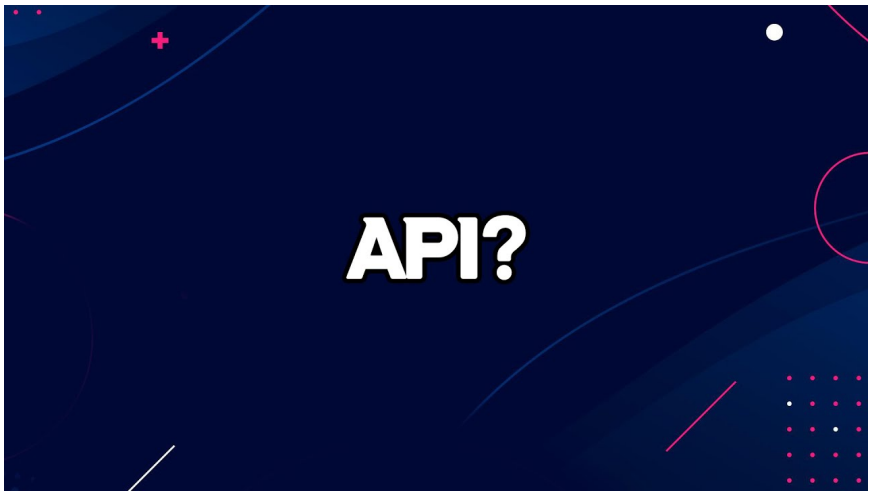
정리하면
인터페이스는 소통을 하기 위한 접점이고
ui는 사용자가 소통을 하기 위한 접점이라고 보면 된다.

Api(Application Programming Interface) :

-서버끼리 통신할수 있게 하는 메커니즘이다. 쉽게 말해 기상청의 소프트웨어 시스템에는 일일 기상 데이터가 들어있고 핸드폰의 날씨 앱은 api를 통해 이 시스템과 대화 하여 핸드폰에 매일 최신 날씨 정보를 표시한다.
- 애플리케이션에서 데이터를 읽거나 쓰기 위해 사용하는 인터페이스이다.

정적컨텐츠 : welcome페이지 처럼 서버에서 뭔가를 하는것 없이 파일을 그냥 웹브라우저에 보여주는 것.
파일을 그냥 그대로 사용자에게 전해주는것.
MVC와 템플릿 엔진(가장 많이 하는 것) : jsp, php가 템플릿 엔진인데 html을 서버에서 뭔가 가공하고 동작한 뒤 웹 브라우저에 보여줌
그것을 위해 model, view, controller를 이용함.
API : 안드로이드, 아이폰 클라이언트랑 개발시 json 데이터 포맷으로 클라이언트에게 데이터를 전달함.
보통 서버끼리 데이터 주고 받는것을(이때는 html이런게 필요가 없으니..) API방식이라함.

api설명 영상



- **컨트롤러가 정적(STATIC) 파일보다 우선순위가 높기 때문에 정적 파일은 무시된다**

컨트롤러가 정적파일보다 높으면 정적파일은 무시되고
컨트롤러가 정적파일보다 우선순위가 낮으면 정적파일이 실행된다.
(Welcome 파일이 static index.html이 아니라 다른곳에 만든 home.html이 되는 이유는 컨트롤러가 정적파일보다 우선순위가 높기 때문이다.—> 웹브라우저에서 요청이 들어오면 톰캣은 스프링 컨테이너에서 관련 컨트롤러가 있는지 찾고 없으면 static의 index.html을 웰켐페이지로 연다)—>hello-spring 연습한 프젝

스프링특징

DI(Dipendency Injection)
의존관계 주입 & 의존성 주입이라고도 한다.
내가 원하는 모든 곳에서 클래스나 메서드를 가져와 사용할 수있다.
Ex) 생성자에 @autowired가 있으면 스프링이 연관된 객체를 스프링 컨테이너에서 찾아서 넣어준다. 이렇게 객체 의존관계를 외부에서 넣어주는 것을 DI의존송주입이라 한다.

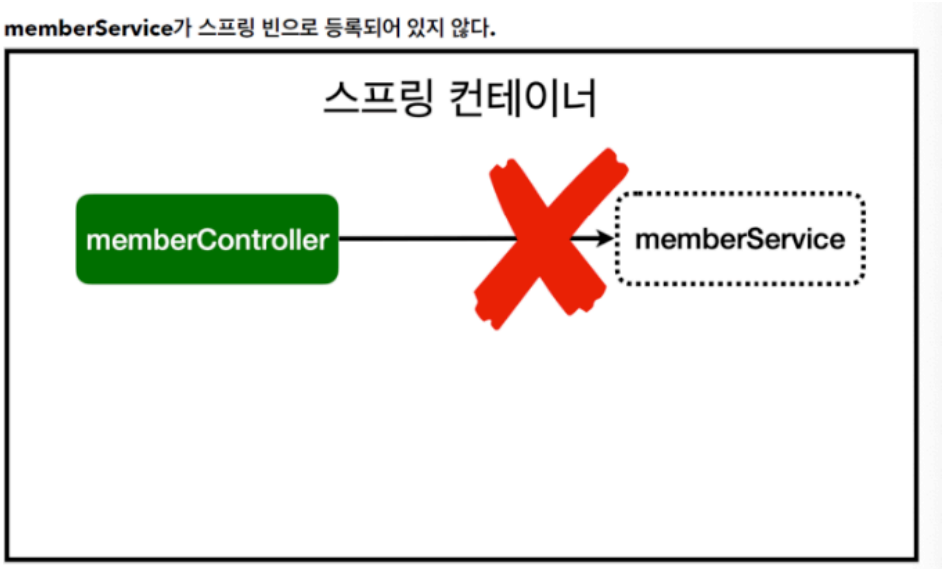
스프링 컨테이너
- 자바 객체의 생명 주기를 관리하고 생성된 자바 객체들에게 추가적인 기능을 제공하는 역할을 한다.
- 스프링에서 자바객체들을 관리하는 공간을 말한다.
- 스프링 컨테이너에서는 이 빈의 생성부터 소멸까지를 개발자 대신 관리해준다.

—> 여기서 자바 객체를 스프링에서는 빈이라고 한다.

*빈 등록방법(자동)
가장쉬운 방법은 클래스 선언 윗 부분에 @Component 어노테이션을 사용하는 것이다.
@Controller, @Service, @Repository는 모두 @Component를 포함하고 있으며 해당 어노테이션으로 등록된 클래스들은 스프링컨테이너에 의해 자동으로 생성되어 스프링빈으로 등록된다. (해당 어노테이션들이 스프링빈으로 자동 등록되는 이유도 컴포넌트 스캔 때문이다)

빈 수동 등록방법
: 수동으로 스프링 빈을 등록하려면 자바 설정 클래스를 만들어 사용해야한다.
설정클래스를 만들고 @Configuration 어노테이션을 클래스 윗부분에 추가하면된다. 그리고 특정 타입을 리턴하는 메소드를 만들고 @Bean 어노테이션을 붙여주면 자동으로 해당 타입의 빈 객체가 생성된다.

1.



<참고로 helloController는 스프링이 제공하는 컨트롤러여서 스프링 빈이 자동 등록된다.
@Controller가 있으면 자동 등록됨.>

@RequestMapping이란?
특정 url로부터 요청을 받으면 어떤 Controller에서 처리할 지 알아야 한다.
이 때, 특정 url을 요청을 수행할 Controller과 매핑하여 지정하는 어노테이션이 @RequestMapping이다.
요청 URL을 어떤 method가 처리할지 mapping해주는 Annotation이다.

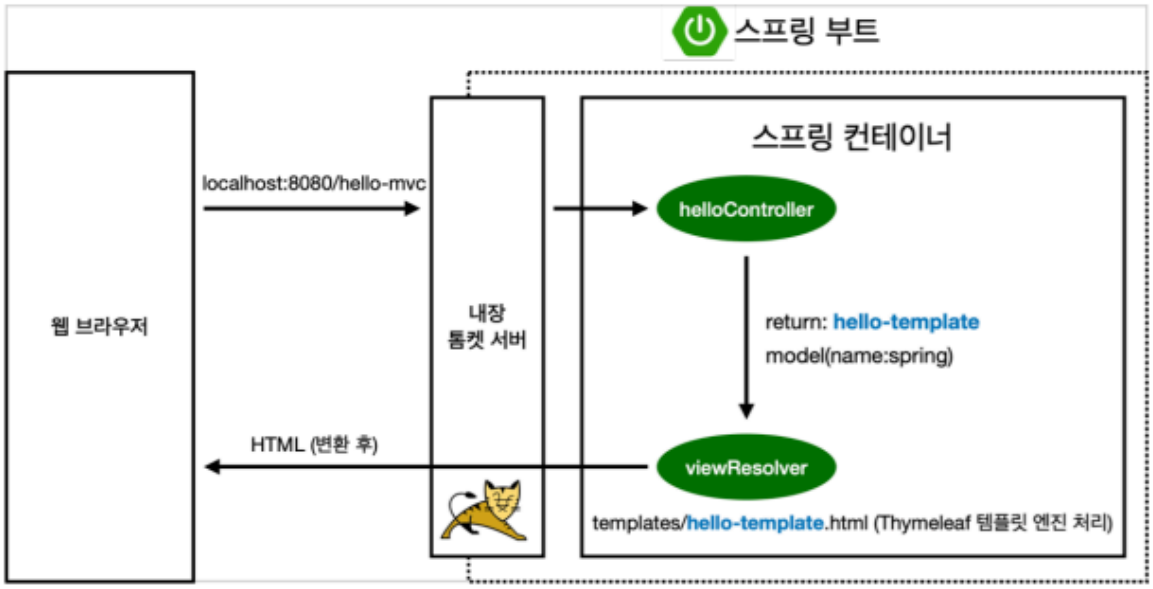
2. @ResponsBody이란
http(통신프로토콜)에서 header부분과 body부분이 있는데 그 body부분을 직접 넣어주겠단의미
Ex) @GetMapping("hello-string")

```
        @ResponseBody
        public String helloString(@RequestParam("name") String
name) {
            return "hello " + name;
        }
}
```

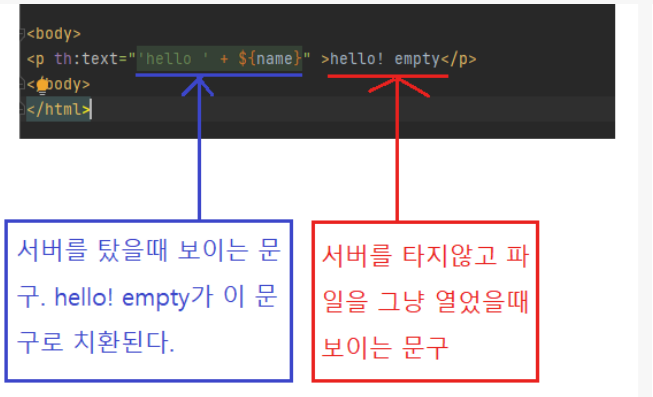
브라우저 url에 우리가 http://localhost:8080/hello-static.html (이 페이지를 보여달라고 사용자인 내가 요청한것)
내장 톰캣이 항시 우리가 명령을 내리기 기다리고 있다가 우리가 요청하는 명령을 듣고 스프링 컨테이너에게 hello-static.html페이지 좀 달래!! 라고 요청한다.
그러면 스프링 컨테이너는 일단 hello-static관련 맵핑이 있는 컨트롤러가 있는지 우선 검색한다.(컨트롤러가 우선순위가 더 높음)
컨트롤러가 없으므로 resources: static/hello-static.html이 있는지 찾고 있네!!
하고 반환해 준다. 그럼 톰캣(웹 서버)이 그걸 받아서 사용자에게 보여줌.

사용자 : 손님 (짜장면 주세용!!)
톰캣 : 서빙하는 종업원 (주방장님 짜장면 한그릇이요!)
스프링 컨테이너 : 주방장 (요리중...요리완료.. 여기)
톰캣 : (넵...손님 짜장면 나왔습니다.)
사용자 : 짜장면에 고춧가루 뿌려주세용!
톰캣 : 오..간단한건 제가 할 수 있음..
사용자 : 탕슉 한그릇 추가요..
톰캣 : 그건 내가 못 만들어유..주방장님!!! 탕슉 한그릇 추가요.

MVC, 템플릿 엔진 이미지



viewResolver : 뷰를 찾아 템플릿 엔진과 연결 시켜줌




```
@GetMapping("hello")//hello요청이 들어오면 return인 hello.html을 띄어준다
public String hello(Model model){
    model.addAttribute("data", "hello!!");
    return "hello";//model이란 것을 hello.html로 넘겨라라는 뜻
}
```

controller에 아래 부분을 추가해 주자.

```
@GetMapping("hello-mvc")
public String helloMvc(@RequestParam("name") String name, Model model){
    model.addAttribute( attributeName: "name", name);
    return "hello-template";
}
```

위의 의미는 url에 localhost:8080/hello-mvc 요청이 들어오면
스프링 컨테이너가 일단 컨트롤러중에 해당하는것이 있는지 찾아...오 있네...
그럼 이 컨트롤러가 실행됨.
@RequestParam("name") String name : 이 함수가 실행할때 RequestParam을 통해 name을 받는다.
return "hello-template"이므로
templates 폴더 밑에 hello-template.html파일을 찾아 들어간다.
빈손으로 가는 것이 아니라 model.addAttribute("name", name);을 통해 name을 받아서 이 name을 들고 hello-template.html에 간다.

컨트롤러는 mvc 에서 c인 controller이고
뷰는 mvc에서 v이다. 이것은 말그대로 웹페이지에 뜨는것 위 코드에서는 html파일이 뷰가된다

프레임 워크 : 그 안에서 쓰여야 되는 것(누가 만들어 놓은 프레임워크안에 있는 틀 안에서만 쓰는것)
라이브러리 : 도서관 처럼 아무거나 막 쓸수 있는 것 (누가 만들어놓은걸 우리가 가져다 쓰는 거임)

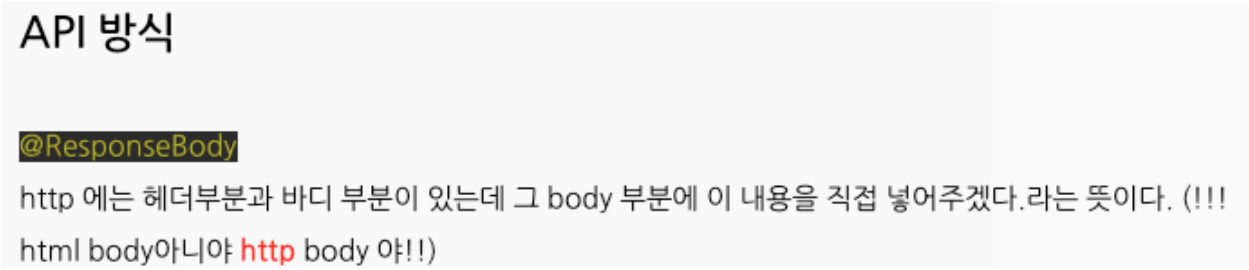
동기식이 전체적으로 시작하는 것
비동기식이 부분적으로 시작하는 것
(로봇생각하면 죽은 로봇동기화 시켜서 일으키고 비동기식은 카메라만 재생시키는 부분적인 것)

인터페이스는 '약속'이다
투입**A**를 넣으면 리턴**B**가 '반드시' 나온다는 약속이다
'함수', '**API**' 등이 인터페이스 개념에 부합되는 존재일 수도 있는데, 프로그램을 개발할때 '**A**를주면 **B**가온다'라는 기능부품/약속 으로 사용되기 때문이다
(티비리모컨/고깃집 벨/에어컨리모컨 등과 같은 약속기능)
정리하면 인터페이스는 프로그램을 개발할때 쓰이는 약속(어떤결과가보장된 약속기능) 이다.

인터페이스의 장점

인터페이스를 사용하면 다중 상속이 가능할 뿐만 아니라 다음과 같은 장점을 가질 수 있습니다.

1. **대규모 프로젝트 개발 시 일관되고 정형화된 개발을 위한 표준화가 가능합니다.**
2. **클래스의 작성과 인터페이스의 구현을 동시에 진행할 수 있으므로, 개발 시간을 단축할 수 있습니다.**
3. **클래스와 클래스 간의 관계를 인터페이스로 연결하면, 클래스마다 독립적인 프로그래밍이 가능합니다.**



Html form 전송방식(get/post)

get방식
url에 파라미터를 포함시켜 요청하는 방식
※데이터를 헤더에 포함시켜 전송한다.
Ex)

```
<form action="" method ="get">
                텍스트를 입력하세요 :
        <input type ="text" name ="search">
```

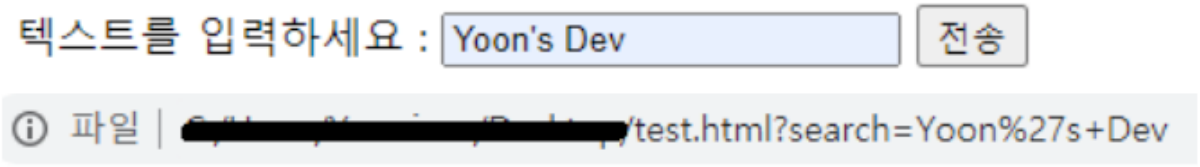
```
        <input type = “submit” value = “전송”>
</form>
```

장점

1.전송 송도가 post 방식보다 빠르다.

2.데이터베이스에 대한 질의어 데이터와 같은 요청 자체를 위한 정보를 전송할 때 사용된다.

3.데이터가 url뒤에 붙기 보안이 취약하다.



위와 같이 URL을 보면 test.html? 이후에 데이터를 포함시킨 것을 확인할 수 있다.

post방식

url에 붙여 보내지 않고 body에 데이터를 넣어서 보내는 방식

클라이언트와 서버간 인코딩하여 서버로 전송한다.

데이터를 body에 포함시켜 전송한다.

- header에 content-type 필드안에 데이터 타입을 명시하여야 한다.

데이터 길이 제한 없음

장점

1.복잡한 형태의 데이터를 전송할때 유용하다

2.데이터베이스에 대한 갱신 작업과 같은 서버측에서의 정보 갱식 작업을 원할때 사용

3.입력한 데이터가 url에 보이지 않아서 보안에서 get방식보다 우수하다.

Ex)

```
<form action=“” method =“post”>
        텍스트를 입력하세요 :
    <input type =“text” name =“search”>
        <input type = “submit” value = “전송”>
</form>
```

다음과 같이 URL에 아무것도 뜨지 않는 것을 확인할 수 있다.

JPA

- 자바에서 제공하는 API이다.
- 특징
  - SQL을 직접 자바 어플리케이션 내에서 사용 경우가 적다
  - SQL 구조를 자바 어플리케이션 내에 적용하지 않아도 된다
  - 테이블 생성, 변경, 삭제 관리가 쉽다.
  - 빠른 개발 가능

생성자에 `@AUTOWIRED` 가 있으면 스프링이 연관된 객체를 스프링 컨테이너에서 찾아서 넣어준다. 이렇게 객체 의존관계를 외부에서 넣어주는 것을 DI (DEPENDENCY INJECTION), 의존성 주입이라 한다.
이전 테스트에서는 개발자가 직접 주입했고, 여기서는 @AUTOWIRED에 의해 스프링이 주입해준다.

스프링 부트와 JPA만 사용해도 개발 생산성이 정말 많이 증가하고, 개발해야할 코드도 확연히 줄어듭니다. 여기에 스프링 데이터 JPA를 사용하면, 기존의 한계를 넘어 마치 마법처럼, 리포지토리에 구현 클래스 없이 인터페이스 만으로 개발을 완료할 수 있습니다. 그리고 반복 개발해온 기본 CRUD 기능도 스프링 데이터 JPA가 모두 제공합니다.
스프링 부트와 JPA라는 기반 위에, 스프링 데이터 JPA라는 환상적인 프레임워크를 더하면 개발이 정말 즐거워집니다. 지금까지 조금이라도 단순하고 반복이라 생각했던 개발 코드들이 확연하게 줄어듭니다.

따라서 개발자는 핵심 비즈니스 로직을 개발하는데, 집중할 수 있습니다.
실무에서 관계형 데이터베이스를 사용한다면 스프링 데이터 JPA는 이제 선택이 아니라 필수 입니다.

스프링의 DI(DEPENDENCY INJECTION)을 사용하면 기존 코드를 전혀 손대지 않고 설정만으로 구현 클래스를 변경할수 있다.

JDBC

- connection관리를 해줘야한다.
- dao 관리를 해주어야하고 , preparestatement로 sql을 전달하여 result 객체로 값을 받아온다.
  —> connection 객체가 db와 app의 연결을 관리하고 preparedstatement가 sql을 전달하며 resultset객체를 통해 결과값을 전달한다.

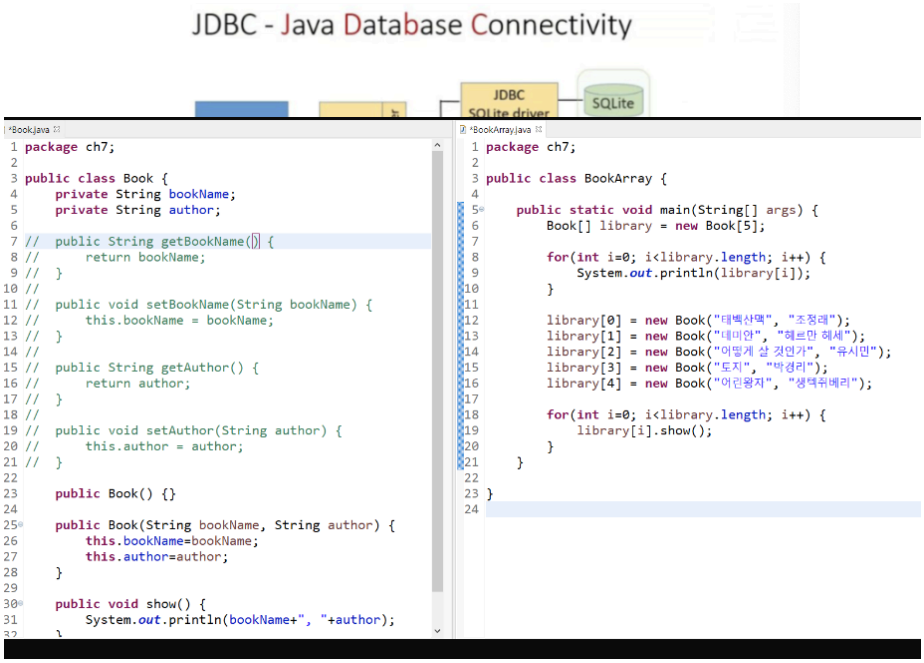
Ex)

```
private Connection conn;
// DB 접근 객체

private PreparedStatement pstmt;
private ResultSet rs;
// 정보를 담을 객체
```

```
String dbURL = "jdbc:mysql://localhost:3306/DB?useSSL=false";

// localhost:3306 -> 컴퓨터에 설치된 mysql, port 3306의 DB(내가 저장한이)라는 데이터베이스에 접근



String dbID = "root";
String dbPassword = "thsdudwls";
Class.forName("com.mysql.jdbc.Driver");
// mysql에 접속할 수 있도록 매개체 역할을 해주는 하나의 라이브러리, jdbc 드라이 로드
```

```java
public int write(String bbsTitle, String userID, String bbsContent) {
        String SQL = "INSERT INTO bbs VALUES (? ,? ,? ,? ,? ,?)";
        try {
                PreparedStatement pstmt = conn.prepareStatement(SQL);
                pstmt.setInt(1, getNext()); // bbsID
                pstmt.setString(2, bbsTitle);
                pstmt.setString(3, userID);
                pstmt.setString(4, getDate());
                pstmt.setString(5, bbsContent);
                pstmt.setInt(6, 1); //
                return pstmt.executeUpdate();

        } catch (Exception e) {
                e.printStackTrace();
        }
        return -1; // 데이터베이스 오류
```

위와 같이 dao관리를 해 줘야한다.



json과 xml 두 형식의 궁극적인 목적은 데이터를 쉽게 분류, 찾기 위한 목적이다.

json(JavaScriptObjectNotation)

- 쉽게 데이터를 교환하고 저장하기 위하여 만들어진 텍스트 기반의 데이터 교환 표준이다.
- 자바스크립트 기반으로 만들어짐
- json은 텍스트 기반이므로 어떠한 프로그래밍 언어에서도 json데이터를 읽고 사용할 수 있다.
- 특징
  - 사람과 기계가 모두 읽기 편하도록 고안되었다.
  - 데이터를 빨리 읽고 쓸수 있다.
  - json은 프로그래밍 언어와 운영체제에 독립적이다.
  - 배열을 사용할 수 있다.
    Ex) 챗봇

=>

위 그림에서 버튼을 정의한 부분에 배열이 사용되 었고, 여기서 배열이란 여러 개의 데이터가 순서를 가지고 나열된 집합을 의미하며, 대괄호([])로 둘러싸여 있다. 배열은 xml에서는 사용할 수 없고 json에서만 가능하다.

- key, value로 이루어져 있어 간편하고 속성태그를 붙이지 않아도된다.
  Ex)

```
{
    "name": "남주혁",
    "instagramID": "skawngur",
    "Birth": 1994.2.22
    "height": 187
}
```

Get = 가져오다, set = 저장 하다

Ex) userId라는 변수를 선언하였을 때 userId의 변수에 어떠한 값을 저장할때 set을 사용,
저장된 userId의 값을 불러오는 것을 get을 사용한다.

Get,set함수 쓰는 이유(접근지정자를 private으로 지정하고 쓰는이유!)

- 외부로부터 변수값에 직접적으로 접근하는 것을 막기 위해 사용한다.
- 자바의 캡슐화 때문이다. -> 자바에서는 함수를 통해 값을 전달 받거나 전달하는 방식을 권장하고 있다(get, set 함수는 Accessor할수라한다. 다시말해 접근자이다.)

캡슐화 :캡슐화를 통해 외부에서 내부의 정보에 접근하거나 변경할 수 없게 직접적인 접근을 막고 객체가 제공하는 필드와 메소드를 통해서만 접근 가능하다. 이로인해 객체 내 정보 손상과 오용을 방지하고 변경되어도 다른 객체에 영향을 주지 않아 독립성이 좋다.

캡슐화는 접근제어자를 통해 이루어진다. (Public : 접근 제한 없음, protected : 동일한 패키지 내or 파생클래스에서만 접근가능, default : 아무런 접근 제한자를 명시하지 않으면 default값이 되며 동일한 패키지 내에서만 접근 가능,
private : 자기 자신의 클래스 내에서만 접근 가능

== 내가 이해한 방식은

위 그림에서 bookName과 author은 private 접근 제어자로 동일한 클래스 내에서만 접근이 가능하다.
그런데 bookName과 author을 get,set 즉 접근자로 지정해줌으로 써 bookArray 클래스에서 온 정보를 book클래스의 get함수가 가져오고 가져온 정보를 set함수로 저장을 시킨다. Get과 set 은 접근자 함수이기도 하고 pulic인 접근 제한이 없는 접근제어자를 가지고 있기 때문이다.