코리아 서버 호스팅(oAuth API gateway 연동 프로젝트)

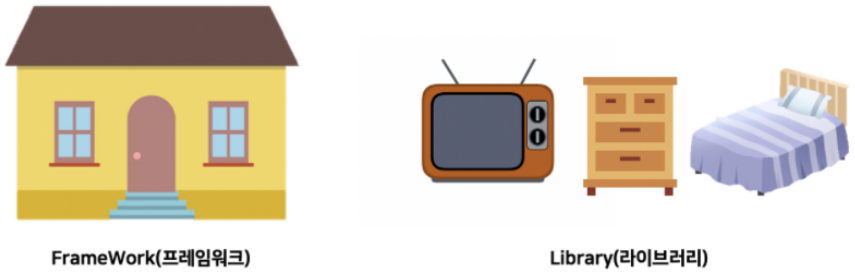
2021.07.05 프로젝트 관련 내용 사전 조사

FrameWork(프레임워크)와 Library(라이브러리) 차이점 -> 프레임 워크는 클래스와 인터페이스 집합이라고 볼 수 있음. 라이브러리는 프로그램 기능 수행을 위해 활용한 가능한 도구의 집합.

프레임워크와 라이브러리의 차이점

프레임워크와 라이브러리의 차이점은 흐름을 누가 지니고 있냐의 차이입니다. 프레임워크는 전체적인 흐름을 자체적으로 가지고 있어 프로그래머는 그 안에서 필요한 코드를 작성합니다. 반면에 라이브러리는 프로그래머가 전체적인 흐름을 가지고 있어 라이브러리를 자신이 원하는 기능을 구현하고 싶을 때 가져다 사용할 수 있다는 것이죠. 흐름에 대해서 잘 이해하시는 것이 가장 중요합니다.

프레임워크와 라이브러리의 차이



FrameWork(프레임워크) Library(라이브러리)

프레임워크를 집이라는 건물에 비유하였으며 라이브러리는 집 안에 포함되는 가구에 비유를하였습니다.

간단히 프레임워크는 가져다가 사용한다는 것보다는 프레임워크라는 특정 공간에 들어가서 사용한다는 느낌이 더 강하다고 말할 수 있으며 라이브러리는 라이브러리 자체를 가져가 사용하고 호출하는 용도로 사용된다고 생각하시면 쉽게 이해하실 수 있을 것입니다.

Rest란? -> Rest 아키텍처의 제약 조건을 준수하는 애플리케이션 프로그래밍 인터페이스임. "Representational State Transfer"의 약자로서, 자원의 이름(자원의 표현)으로 구분하여 해당 자원의 상태(정보)를 주고 받는 모든 것을 의미함.

정리하자면, HTTP URL(Uniform Resource Identifier)를 통해 자원(Resource)을 명시하고, HTTP Method(POST, GET, PUT, DELETE)를 통해 해당 자원에 대한 CRUD Operation을 적용하는 것을 의미한다.

여기서 장단점이 존재 ->

장점->

1.HTTP 프로토콜의 인프라를 그대로 사용하므로 REST API 사용을 위한 별도의 인프라를 구출할 필요가 없다.

2. HTTP 프로토콜의 표준을 최대한 활용하여 여러 추가적인 장점을 함께 가져갈 수 있게 해준다.

3. 표준에 따르는 모든 플랫폼에서 사용이 가능하다.

단점-> 메소드 4가지, 구형 브라우저 아직 제대로 지원 X, 브라우저를 통해 테스트 할일 많은 서

비스 X

결국 REST가 필요한 이유는 '애플리케이션 분리 및 통합', '다양한 클라이언트의 등장', '최근의 서버 프로그램은 다양한 브라우저와 안드로이드폰, 아이폰과 같은 모바일 디바이스에서도 통신을 할 수 있어야 한다.' ' 이러한 멀티 플랫폼에 대한 지원을 위해 서비스 자원에 대한 아키텍처를 세우고 이용하는 방법을 모색한 결과, REST에 관심을 가지게 된 것임.'

REST 구성요소->

1. 자원(Resource): URI 모든 자원에 고유한 ID가 존재하고, 이 자원이 Server에 존재함. 예를 들어 자원을 구별하는 ID가 '/group/:group_id'와 같은 HTTP URI 인거임.

2. 행위(Verb): HTTP Method / GET,POST,PUT,DELETE

3. 표현(Representation of Resource): jSON이랑 XML를 통해서 데이터를 주고 받는 것이 일반적임.

REST API란?

REST API란

- API(Application Programming Interface)란
  - 데이터와 기능의 집합을 제공하여 컴퓨터 프로그램간 상호작용을 촉진하며, 서로 정보를 교환 가능 하도록 하는 것
- REST API의 정의
  - REST 기반으로 서비스 API를 구현한 것
  - 최근 OpenAPI(누구나 사용할 수 있도록 공개된 API: 구글 맵, 공공 데이터 등), 마이크로 서비스(하나의 큰 애플리케이션을 여러 개의 작은 애플리케이션으로 쪼개어 변경과 조합이 가능하도록 만든 아키텍처) 등을 제공하는 업체 대부분은 REST API를 제공한다.

REST API의 특징

- 사내 시스템들도 REST 기반으로 시스템을 분산해 확장성과 재사용성을 높여 유지보수 및 운용을 편리하게 할 수 있다.
- REST는 HTTP 표준을 기반으로 구현하므로, HTTP를 지원하는 프로그램 언어로 클라이언트, 서버를 구현할 수 있다.
- 즉, REST API를 제작하면 델파이 클라이언트 뿐 아니라, 자바, C#, 웹 등을 이용해 클라이언트를 제작할 수 있다.

REST API 설계 기본 규칙

참고 리소스 원형

> - 도큐먼트 : 객체 인스턴스나 데이터베이스 레코드와 유사한 개념
> - 컬렉션 : 서버에서 관리하는 디렉터리라는 리소스
> - 스토어 : 클라이언트에서 관리하는 리소스 저장소

1. URI는 정보의 자원을 표현해야 한다.
   i. resource는 동사보다는 명사를, 대문자보다는 소문자를 사용한다.
   ii. resource의 도큐먼트 이름으로는 단수 명사를 사용해야 한다.
   iii. resource의 컬렉션 이름으로는 복수 명사를 사용해야 한다.
   iv. resource의 스토어 이름으로는 복수 명사를 사용해야 한다.
   - Ex) `GET /Member/1` -> `GET /members/1`
2. 자원에 대한 행위는 HTTP Method(GET, PUT, POST, DELETE 등)로 표현한다.
   i. URI에 HTTP Method가 들어가면 안된다.
   - Ex) `GET /members/delete/1` -> `DELETE /members/1`
   ii. URI에 행위에 대한 동사 표현이 들어가면 안된다.(즉, CRUD 기능을 나타내는 것은 URI에 사용하지 않는다.)
   - Ex) `GET /members/show/1` -> `GET /members/1`

ii. URI에 행위에 대한 동사 표현이 들어가면 안된다.(즉, CRUD 기능을 나타내는 것은 URI에 사용하지 않는다.)
   - Ex) `GET /members/show/1` -> `GET /members/1`
   - Ex) `GET /members/insert/2` -> `POST /members/2`

iii. 경로 부분 중 변하는 부분은 유일한 값으로 대체한다.(즉, :id는 하나의 특정 resource를 나타내는 고유값이다.)
   - Ex) student를 생성하는 route: POST /students
   - Ex) id=12인 student를 삭제하는 route: DELETE /students/12

REST API 설계 규칙

1. 슬래시 구분자(/)는 계층 관계를 나타내는데 사용한다.
   - Ex) `http://restapi.example.com/houses/apartments`
2. URI 마지막 문자로 슬래시(/)를 포함하지 않는다.
   - URI에 포함되는 모든 글자는 리소스의 유일한 식별자로 사용되어야 하며 URI가 다르다는 것은 리소스가 다르다는 것이고, 역으로 리소스가 다르면 URI도 달라져야 한다.
   - REST API는 분명한 URI를 만들어 통신을 해야 하기 때문에 혼동을 주지 않도록 URI 경로의 마지막에는 슬래시(/)를 사용하지 않는다.
   - Ex) `http://restapi.example.com/houses/apartments/ (X)`
3. 하이픈(-)은 URI 가독성을 높이는데 사용
   - 불가피하게 긴 URI경로를 사용하게 된다면 하이픈을 사용해 가독성을 높인다.
4. 밑줄(_)은 URI에 사용하지 않는다.
   - 밑줄은 보기 어렵거나 밑줄 때문에 문자가 가려지기도 하므로 가독성을 위해 밑줄은 사용하지 않는다.
5. URI 경로에는 소문자가 적합하다.

6. 파일확장자는 URI에 포함하지 않는다.
    - REST API에서는 메시지 바디 내용의 포맷을 나타내기 위한 파일 확장자를 URI 안에 포함시키지 않는다.
    - Accept header를 사용한다.
    - Ex) `http://restapi.example.com/members/soccer/345/photo.jpg (X)`
    - Ex) `GET / members/soccer/345/photo HTTP/1.1 Host: restapi.example.com Accept: image/jpg (O)`
7. 리소스 간에는 연관 관계가 있는 경우
    - /리소스명/리소스 ID/관계가 있는 다른 리소스명
    - Ex) `GET : /users/{userid}/devices (일반적으로 소유 'has'의 관계를 표현할 때)`

REST API 설계 예시

CRUD	HTTP verbs	Route
resource들의 목록을 표시	GET	*/resource*
resource 하나의 내용을 표시	GET	*/resource/:id*
resource를 생성	POST	*/resource*
resource를 수정	PUT	*/resource/:id*
resource를 삭제	DELETE	*/resource/:id*

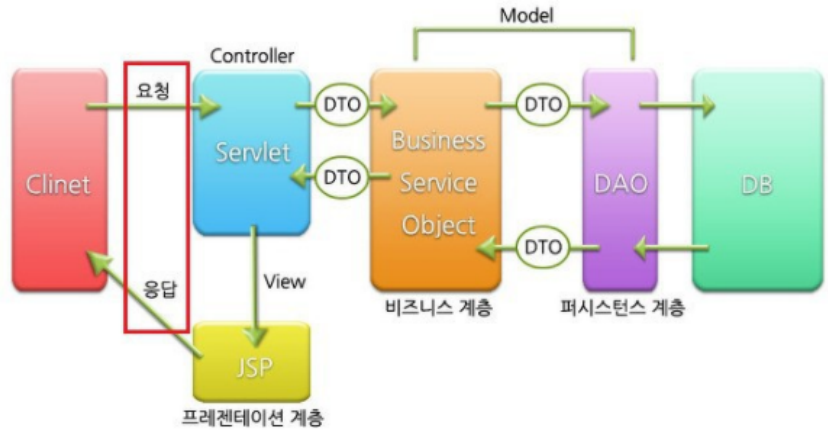
참고 응답상태코드

> - 1xx : 전송 프로토콜 수준의 정보 교환
> - 2xx : 클라어인트 요청이 성공적으로 수행됨
> - 3xx : 클라이언트는 요청을 완료하기 위해 추가적인 행동을 취해야 함

언어 선택 (총 3개 언어 선택)

-> Java => Java spring, spring boot 가 제일 최적의 프레임 워크로 보임.

Spring Framework의 특징 MVC (Model2)



MVC란 (Model View Controller) 구조로 사용자 인터페이스와 비지니스 로직을 분리하여 개발 하는 것 입니다. MVC에서는 Model1과 Model2로 나누어져 있으며 일반적인 MVC는 Model2를 지칭합니다.

Model

Model에서는 데이터처리를 담당하는 부분입니다. Model부분은 Serivce영역과 DAO영역으로 나누어지게 되고 여기서 중요한 것은 Service 부분은 불필요하게 HTTP통신을 하지 않아야하고 request나 response와 같은 객체를 매개변수로 받아선 안된다. 또한 Model 부분의 Service는 view에 종속적인 코드가 없어야 하고 View 부분이 변경되더라도 Service 부분은 그대로 재사용 할 수 있어야 한다.

Model에서는 View와 Controller 어떠한 정보도 가지고 있어서는 안된다.

View

View는 사용자 Interface를 담당하며 사용자에게 보여지는 부분입니다. View는 Controller를 통해 모델에 데이터에 대한 시각화를 담당하며 View는 자신이 요청을 보낼 Controller의 정보만 알고 있어야 하는 것이 핵심이다.

Model이 가지고 있는 정보를 저장해서는 안되며 Model, Controller에 구성 요소를 알아서는 안된다.

Controller

Controller에서는 View에 받은 요청을 가공하여 Model(Service 영역)에 이를 전달한다. 또한 Model로 부터 받은 결과를 View로 넘겨주는 역할을 합니다. Controller에서는 모든 요청 에러와 모델 에러를 처리하며 View와 Controller에 정보를 알고 있어야한다.

Model과 View의 정보에 대해 알고 있어야한다.

-> php => Laravel, Codeigniter, zend

Laravel

라라벨은 상대적으로 최근에 나온 PHP 프레임워크이지만, 논쟁의 여지 없이 현재 가장 인기있는 PHP 프레임워크이며, 거대한 생태계를 갖고 있습니다. 라라벨은 매 릴리즈마다 문서화되고 다수의 비디오, 튜토리얼, 블로그가 있기 때문에 쉽게 배울 수 있습니다.

라라벨은 많은 기능들로 빠른 어플리케이션 개발을 가능하게 합니다. Artisan 컴맨드 라인 인터페이스를 통해 개발 중 유용하게 쓰일 수 있는 명령어를 제공합니다. 또한, 라라벨은 강력한 템플릿 엔진을 통해 일상적인 인증이나 캐싱, 세션, RESTful 라우팅, 큐잉을 쉽게 처리할 수 있게 합니다.

라라벨은 여타 PHP 프레임워크에 비해 프레임워크 자체 오버해드가 매우 큰 편입니다. 매우 간단한 작업을 하는 요청의 경우, 실제 요청 처리에 걸리는 시간보다 프레임워크 및 내부 모듈의 초기화에 걸리는 시간이 더 많습니다. 개발자가 실제 사용하지 않는 모듈까지 불필요하게 모두 로드됩니다. 따라서 클라이언트로부터 빈번하게 대량의 요청이 들어오는 서비스의 경우, 프레임워크의 오버해드가 눈덩이처럼 커집니다.

CodeIgniter

코드이그나이터는 가장 오래된 프레임워크 중 하나이지만 심플하고 강력합니다. 설치가 쉬우며, 최소한의 환경 설정만 하면 됩니다. 거의 모든 공유 및 전용 호스팅 서버에서 완벽하게 작동합니다.

코드이그나이터는 전적으로 MVC 아키텍처에 기반하여 구성되어 있지는 않습니다. 컨트롤러 클래스는 필수이지만 모델과 뷰는 선택적입니다. 코드이그나이터의 또 다른 강점은 속도입니다. 다른 프레임워크와 비교하여 데이터베이스 작업이 더 빠르게 실행됩니다. 문서화가 잘 되어 있으며 PHP 초보자를 위한 훌륭한 프레임워크입니다.

-> python => Flask, Django

Python Web Framework django vs Flask

Python에서 django와 Flask는 가장 널리 사용되는 오픈소스 기반 웹 프레임워크입니다.
Django와 Flask는 각각 장단점이 존재할 텐데요 뭐가 더 좋다 나쁘다가 아니라 활용도에 따라 무엇을 써야 할지를 판단하는 게 중요합니다.
간단히 설명하자면 django는 Python의 full stack web framework인 반면 Flask는 가볍고 확장 가능한 web framework입니다. 즉 django는 기능이 훨씬 뛰어나지만 복잡하고, Flask는 매우 단순하고 가볍습니다.