HTTP 프로토콜

-인터넷 통신을 위해 사용되는 프로토콜

-클라이언트는 서버에 요청(Request)하고 서버는 그 요청에 대한 반응(Response)을 돌려준다

-요청할 때 URL사용

쿠키

🡪클라이언트의 컴퓨터에 파일로 저장되는 State정보

-클라이언트 컴퓨터에 파일 형태로 존재하기 때문에 해킹 위험성 존재

세션(쿠키해결 용도)

🡪서버의 메모리에 저장되는 정보 –서버에서만 읽고 쓸 수 있다.

캐시

🡪자주 사용하는 데이터나 값을 미리 복사해 놓는 임시 장소

\*로컬캐시 : 로컬 장비내에서만 사용되는 캐시

글로벌캐시: 여러 서버에서 캐시 서버에 접근하여 사용하는 캐시로 분산된 서버에서 데이터를 저장 조회

REST

: HTTP URL을 통해 자원을 명시하고 HTTP Method(POST,GET,PUT,DELETE)를 통해 해당 자원에 대한 CRUD Operation을 적용하는 것을 의미

CRUD Operation

-Create: 생성(POST)

-Read: 조회(GET)

-Update: 수정(PUT)

-Delete: 삭제(DELETE)

REST의 특징

-서버-클라이언트 구조: 자원이 있는 쪽(Server),자원을 요청하는 쪽(Client) 서버는 API를 제공하고 비즈니스 로직 처리 및 저장을 책임, 클라이언트는 사용자 인증,세션,로그인 정보를 관리

-Stateless(무상태): 통신이 끝나면 현재 상태가 저장이 되지 않음

-Cacheable(캐시 처리 가능): 웹 표준 HTTP프로토콜을 그대로 사용하므로 웹에서 사용하는 인프라를 그대로 활용가능, 캐시 사용을 통해 응답시간 빨리지고 REST 서버 트랜잭션이 발생하지 않기 때문에 전체 응답시간,성능,서버의 자원 이용률을 향상 시킬수 있다.

-Layered System(계층화): 클라이언트는 REST API 서버만 호출. REST 서버는 다중 계층 구성 가능

프록시,게이트웨이 같은 네트워크 기반의 중간 매체 사용 가능

-Addressability(URL접근): REST는 모든 오브젝트에 유일, 직관적인 URL을 통해 접근

-Uniform Interface(인터페이스 일관성): URL로 지정한 리소스 조작을 통일되고 한정적인 인터페이스로 수행

REST의 장점

-HTTP표준 프로토콜에 따르는 모든 플랫폼에서 사용 가능

-REST API 메시지가 의도하는 바를 명확하게 나타내므로 의도하는 바를 쉽게 파악 가능

REST의 단점

-사용 가능 메소드가 4가지

-표준이 없음

REST가 필요한 시점

-다양한 클라이언트들 등장

-모바일 디바이스에서도 통신 가능한 서버 프로그램 필요시

-애플리케이션 분리 및 통합

REST API

:REST기반으로 서비스 API를 구현하는 것, OpenAPI,마이크로 서비스(하나의 큰 애플리케이션을 여러개의 작은 애플리케이션으로 쪼개어 변경과 조합이 가능하도록 만든 아키텍처)등을 제공하는 업체 대부분 REST API제공

REST API 특징

-HTTP표준을 기반으로 구현하므로, HTTP를 지원하는 프로그램 언어로 클라이언트,서버를 구현

-확장성과 재사용성 높여 유지보수 및 운용을 편리하게 함

Restful

:REST원리를 따르는 시스템

Restful 목적

:이해하기 쉽고 사용하기 쉬운 RestAPI를 만드는 것이 목적

JSON

:데이터를 저장하거나 전송할 때 많이 경량의 DATA 교환 형식,텍스트 형식

Key/Value 형식

Pathvariable 방식

:경로의 특정 위치값이 고정되지 않고 달라질 때 사용하는 것으로 , URL경로에 변수를 넣어주는 기능,URL path로 들어오는 값을 해석 그대로 변수로 사용하겠다는 의미

사용방식

@RequestMapping(“[경로]/{변수이름}”)

메소드의 파라미터에 pathvariable를 이용해서 값을 받아온다.

사용이유

동일하고 여러 개의 컨트롤러를 사용할 때 좋음, 응용해서 사용하면 여러가지 폭넓게 계발 가능

예시)

예를 들어 특정 유저의 정보를 조회하고 싶다면 GET /user/123 이런 식으로 url 패스가 들어갑니다.

이때 123은 124, 125 등 리소스 번호를 뜻하고 이는 변할 수 있는 값입니다.

Spring 에서는/user/{userId} 와 같은 식으로 매핑하고

@PathVariable Long userId 와 같은 식으로 메서드 파라미터로 그대로 받을 수 있습니다.

바로 보일러 플레이트기 때문입니다.

그리고 제공하는 api (컨트롤러의 메서드)가 많아질수록 이런 보일러 플레이트 코드량은 증가하게 되고, 결국 유지보수의 어려움을 야기할 수 있는 원인이 됩니다.

프로그래머 또한 사람이기에 해당 정보를 얻기 위한 코드를 작성하다가 실수할 수도 있고요.

따라서 @PathVariable 이라는 어노테이션을 사용함으로써 보일러 플레이트가 하는 역할을 스프링이 알아서 하게끔 하는것이죠.

이에 따라 같은 작업을 하지만 전체적인 코드량의 감소, 이에 따른 실수 확률의 감소, 유지보수 편의성 증대, 신뢰성 역시 보장되구요.

자바

객체지향프로그래밍 – OOP : 객체들끼리 메시지를 주고받으며 데이터를 처리

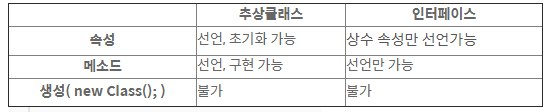
상속 : 다른 클래스의 메소드들을 물려받는 것, extends, 단일 상속만 가능

다형성 : 하나의 클래스 메소드가 다양한 방법으로 동작한다

오버로딩:같은 이름의 메소드를 여러 개 가지면서 매개변수의 유형과 개수가 다른 것

오버라이딩:상위 클래스가 가지고 있는 메소드를 하위 클래스가 재정의해서 사용

추상화: 여러 객체에 공통적으로 사용되는 내용을 뽑아낸 것 ,abstract



인터페이스: implements 상수와 추상메소드로 구성, 일종의 양식을 제공 그 양식에 따라 클래스들은 해당하는 항목의 조건에 따라 자유롭게 코드 정의

캡슐화: 중요 데이터 보존 보호를 위해 외부 접근을 막을 수 있음 ,private 🡪 set/get