[h1][REST API 개발] 2. JAX_RS 개요

노트북:	0074_jsp 개요		
만든 날짜:	2023-01-21 오전 1:34	수정한 날짜:	2023-01-25 오후 12:35
작성자:	nomad.lab.kr@gmail.com		
URL:	about:blank		

1. JAX-RS란?

REST 서비스를 제공하기 위해서는 여러 HTTP Method (GET, POST, PUT등)을 지원하면서
다양한 URI 요청을 처리할 수 있는 서버 프로그램 구조가 필요.
단순히 서브릿만 이용해도 어느 정도 REST 형태의 서비스를 개발할 수 있지만 여러 URI 요청을 구조적으로 손쉽게 처리하려면 규격이 필요.
이에 따라 자바는 JAX-RS라고 하는 표준 규격을 만듬.

JAX-RS Java API for RESTful Web Services는 REST 원칙을 사용하는 개발 메커니즘을 제공하는 자바 표준 API.
즉 JAX-RS는 서비스 측 REST 애플리케이션 개발을 단순화하는 인터페이스 및 Java 애너테이션의 집합체.

스프링 프레임워크의 경우 RestController 라고 하는 자체 규격을 제공하며 필요에 따라 JAX-RS를 사용할 수 있음.
시스템 자체가 스프링 기반이라고 한다면 JAX-RS를 사용하는 것보다는 스프링의 RestController가 좀 더 편리.

* 구현체가 필요한 JAX-RS
JDBC와 유사하게 JAX-RS 역시 표준 규격만 인터페이스 형태로 JDK에 포함되어 있고 실제 사용을 위해서는 구현체가 필요.
대표적으로 아파치 Jersey, JBoss RESTEasy 등이 있음.

2. JAX-RS 사용 과정

1) API 서비스 등록

먼저 제공하려고 하는 API 서비스에 대한 정보를 컨테이너에 전달.
@Application 애너테이션을 사용해 어떤 시작을 URL을 사용할지 지정해야 하며 해당 요청에 대한 구현 클래스의 패지키를 등록.

서블릿이나 JSP의 경우에도 콘텍스트 루트라고 해서 시작 경로가 정해져 있음.
다음 코드는 모든 REST API 요청이 /api로 시작하도록 지정.

```
@ApplicationPath("/api") // REST API 서비스 진입점을 의미.
public class RestConfig extends Application {
    public Map<String, Object> getProperties() {
        Map<String, Object> properties = new
HashMap<String, Object>();

properties.put("jersey.config.server.provider.packages",
"restAPI");
        return properties;
    }
}
```

주의할 점은 RestConfig 클래스에서 속성으로 저장하는 패키지에 대해서만 REST API 클래스로 사용할 수 있음.

2) API 클래스 구현
API 클래스는 서블릿과 달리 **별도의 클래스 상속 없이 일반 자바 클래스로 구현**.
클래스 앞부분에 @Path 애너테이션으로 하위 URI 시작점을 지정하고(선택 사항) 각각의 요청에 대한 처리를 담당하는 메서드를 구현.

REST API 클래스 구현에 필요한 구성요소

* 메서드 : GET, POST, PUT, DELETE 등 어떤 HTTP 메서드 요청을 처리할 것인지 지정.

* 요청 경로 : 어떤 URI 요청에 동작할 것인지 지정

* 응답 컨텐트 타입 : 어떤 컨텐트 타입 (예 : text, html, json, xml 등)으로 응답 메시지를 구성할것인지 지정

* 파라미터 : 클라이언트 요청 파라미터 및 경로 파라미터 값을 메서드에 전달

REST API를 구성한다는 것은 어떤 HTTP 메서드로 요청되는 특정 URI에 동작할 메서드를 구현하고
요청 파라미터를 메서드에서 사용해 프로그램을 구현한 다음, 처리 결과를 어떤 형태로 전달할지 정의한다로 이해.

예를 들어 /api/addrbook/list 요청을 처리하는 API 클래스는 다음과 같이 구현

```
@Path("/addrbook")
public class AddrBookApiService {
    Logger logger =
Logger.getLogger("RestApiService");
    AddrBookDAO dao = new AddrBookDAO();

    @GET
    @Path("list")
    @Produces(MediaType.APPLICATION_JSON)
    public List<AddrBook> getList() {
        List<AddrBook> datas = dao.getAll();
        logger.info("API call: /list");
        return datas;
    }
}
```

* @Path("/addrbook") : 클래스부에 선언된 애너테이션의 경우 현재 클래스에서 처리할 서브 경로를 지정.
즉 /api/addrbook 요청에 응답.

* @GET : GET 요청을 처리.

* @Path("list") : /api/addrbook/list 요청에 응답.

* @Produces(MediaType.APPLICATION_JSON) : 메서드 처리 후 생성되는 데이터 타입으로,
여기서는 JSON 형식을 리턴하게 된다는 의미.
내부적으로 List 타입이 JSON으로 변경.

3. REST API 설계 원칙

REST API를 구현하는 과정은 비교적 단순하지만 프로그램 문법과 같이 정해진 규칙이 있는 것이 아니기 때문에 잘 설계하기 위해서는 경험과 노력이 필요.

일반적인 REST API 설계 원칙

* 동사 대신 명사 사용

예를 들어 전체 상품 목록을 제공하는 API의 경우 getProductLIst, getAll, getProduct와 같은 동사 형태보다는 products와 같은 명서 형태가 적합.

/getProductList -> /products
/getAllStudent -> /students

* 상태 변경시 GET 메서드와 쿼리 파라미터 사용 금지

수정, 삭제와 같이 데이터의 상태를 변경하는 경우 파라미터 형태가 아닌 HTTP 메서드와 경로 파라미터로 처리하는 것이 좋음.

GET /api/products?action=delete&id=1021 -> DELETE /api/products/1021

* 복수 명사 사용

집합형 데이터를 다루는 API의 경우 단수형보다는 복수형을 사용하는 것이 좋음.

GET /product -> GET /products
GET /studentAll -> GET /students

* 관계 형태 표현에 하위 리소스 사용

예를 들어 컴퓨터공학과 학생들 중 홍길동 학생을 찾는 경우 다음과 같이 작성.

GET /depts/CE/students/홍길동

*** 에러 정보 제공**

에러 발생 시 단순히 서버 에러 코드 500만 표시하는 것이 아니라 여러 HTTP 상태 코드를 활용해야 하며,
별도의 에러 정보를 응답 메시지 구조에 포함할 수 있어야 함.

```
{
   "errors":
      {
         "userMessage": "요청 데이터를 찾을 수 없습니다.!!",
         "internalMessage": "데이터베이스 검색 오류",
         "code": 211,
         "more info": "http://xxxx.xxx.com/api/v1/errors/12345",
      }
   ]
}
```