Smart Green Platform 개발

REST 방식 개발가이드

문서번호 입력

Ver. 1.0

관리부서 : 통합팀

개 정 이 력

버전	작성일	변경내용[1]	작성자	승인자
1.0.0	2013-03-05	기존 개발가이드 v1.8.14 에서 이관 및 주제에 따라 별도 문서로 분리	신창훈	

[1] 변경 내용: 변경이 발생되는 위치와 변경 내용을 자세히 기록(장/절과 변경 내용을 기술한다.)

목 차

1. 개요

SGP Enterprise 에서 REST 방식을 이용하여 타 시스템과 연계 개발할 때의 가이드를 제시한다.

연계할 대상 시스템과의 역할 관계에 따라, 서버로서 서비스를 제공하는 모듈을 구현하는 경우와 클라이언트로서 서비스를 호출하는 모듈을 구현하는 것으로 구분할 수 있다.

2. REST 서비스의 구현

Rest 방식의 연계 서비스를 구현하기 위해서는 사전에 URI 와 HTTP 메소드, 송수신 데이터의 포맷, 응답 메시지 등이 정의되어야 한다.

2.1 URI 정의

Rest 방식의 연계에서 사용하는 URI 패턴은 *.do, *.json 과 같은 확장자는 붙이지 않고 /interface, /rclient 와 같은 접두어로 구분한다.

해당 접두어는 클라이언트가 서버를 호출할 때만 사용되고, 서버의 Controller 내부 RequestMapping 정보에서는 제외해야 한다.

/[접두어]/[구분코드]/[resource-path-1]/[resource-path-2]/...

접두어	용도	비고
interface	외부 연계	외부 연계 대상 시스템별로 구분 코드를 정의함
rclient	내부 RClient 연계	내부 RClient 별로 구분 코드를 정의함

구분코드는 외부 연계 개발가이드와 RClient 개발가이드에 정의된 내용을 참고한다.

2.2 URI 와 HTTP 메소드 매핑

Rest 방식의 서비스 구현에서는 서버 자원의 접근 경로인 URI 에 서버 자원의 식별자를 포함하는 것이 일반적이다. 또한 동일한 URI 를 갖더라도 요청의 HTTP 메소드(GET, POST,

PUT, DELETE 등)에 따라 동작을 다르게 정의할 수 있다. Rest 방식에서는 각 메소드별로 CRUD 를 다음과 같이 매핑하는 것이 일반적이다.

- GET – Retrieve
- POST – Create
- PUT – Update
- DELETE - Delete

단, HTTP 프로토콜 스펙에 따르면 POST 를 제외한 메소드들은 멱등(Idempotent) 메소드로서 네트워크 상태가 고르지 못한 경우 내부적으로 재요청(retry)될 수 있음에 주의해야 한다. 멱등 메소드를 통해 수행하는 기능은 재수행되어도 문제가 발생하지 않는 것으로 한정하거나, 그런 제한이 어려운 경우 POST 메소드를 사용한다.

2.3 송수신 데이터 정의

모델링 표준에 따라 컴포넌트간에는 모델 객체를 주고 받도록 정의되어 있으므로, Rest 방식의 연계에서도 JSON 형태의 메시지를 사용하도록 한다.

메시지 유형	Content-Type	메소드 파라미터
JSON	application/json	모델 객체

송수신 대상이 되는 모델 객체는 직렬화(Serialization)와 역직렬화(Deserialization)가 가능해야 한다. 직렬화가 가능한 모델을 만들려면, 다음 예시와 같이 Serializable 인터페이스를 구현하도록 선언하고 serialVersionUID 상수값을 정의하도록 한다.

[모델 예시]

```
public class TemplateModel implements Serializable {

    private static final long serialVersionUID = 1L;

    private String interfaceId;
    private String responseCode;
    private String id;
    private String name;

    public String getInterfaceId() {
        return interfaceId;
    }
    public void setInterfaceId(String interfaceId) {
        this.interfaceId = interfaceId;
    }
    public String getResponseCode() {
        return responseCode;
    }
```

```
    public void setResponseCode(String responseCode) {
        this.responseCode = responseCode;
    }
    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

송수신 데이터 모델 객체에서는 공통 파라미터로서 interfaceId 와 responseCode 항목을 사용하도록 하며, 해당 값의 지정이나 로깅은 각 도메인 서비스에서 처리한다.

- interfaceId

  - 요청에 대한 고유식별자로 클라이언트가 전송한 전문을 서버가 수신하였는지 식별하거나, 서버가 보낸 응답을 클라이언트가 제대로 수신하였는지 확인하기 위해 사용한다.

- responseCode

  - 성공여부, 오류 및 상태 정보 전달을 위한 코드값이며, Response body 에 담아 전달한다. Response body 를 갖지 않는 PUT, DELETE 에서는 사용할 수 없다. 0-99 까지는 공통 영역이며, 100 이상은 도메인에서 각 업무 특성에 따라 정의한다. 코드별 상태 정보는 다음과 같다

ResponseCode	이름	설명
0	Success	요청이 성공적으로 처리됨
1	Unknown Exception	SGP 내부 오류 발생
2	Core Exception	SGP Core Exception 이 발생
3	Data Exception	SGP Data Exception 이 발생
4	Device Exception	SGP Device Exception 이 발생
5	Push Exception	SGP Push Exception 이 발생
6	Service Exception	SGP Service Exception 이 발생
7-19		미정
20	Validation Exception	수신된 데이터가 유효하지 않음
21	Insufficient Parameter	파라미터가 부족함
22	Undefined Parameter	정의되지 않은 파라미터가 존재함
23	Null Parameter	파라미터값이 Null 임

ResponseCode	이름	설명
24-99		미정
100 -		업무별로 정의하는 코드 영역

2.4 Controller 구현

외부 연계, RClient 와 같이 대상별로 정의되어 있는 가이드에 따라 Controller 를 정의하도록 한다.

2.5 예외 처리

예외는 각 컨트롤러 안에서 처리하도록 한다. 예외가 발생하면 @ExceptionHandler 어노테이션으로 선언한 메소드 내에서 예외를 받아서 로그를 남기고 연계 대상 시스템에 전달할 리턴 메시지를 만들어서 리턴한다.

아래는 Controller 내에 선언한 예외 처리 메소드에 대한 예시이다.

```
@ExceptionHandler(Exception.class)
@ResponseBody
public ResponseModel handleException(Exception ex) {
    logger.error("RestClient error", ex);
    ResponseModel response = new ResponseModel();
    response.setInterfaceId(interfaceId);
    response.setResponseCode("errorCode");
    return response;
}
```

3. REST 클라이언트의 구현

Rest 클라이언트의 역할은 PBI, PBC 에서 담당한다. Rest 방식의 요청/응답 처리를 위해서는 SGPRestTemplate 을 이용한다. 아래 예시는 defaultRestTemplate 이라는 이름으로 등록되어 있는 SGPRestTemplate Bean 을 주입받는 방법을 보여준다.

```
@Autowired
@Qualifier("defaultRestTemplate")
private SGPRestTemplate sgpRestTemplate;
```