Part 3 Ajax 댓글 처리

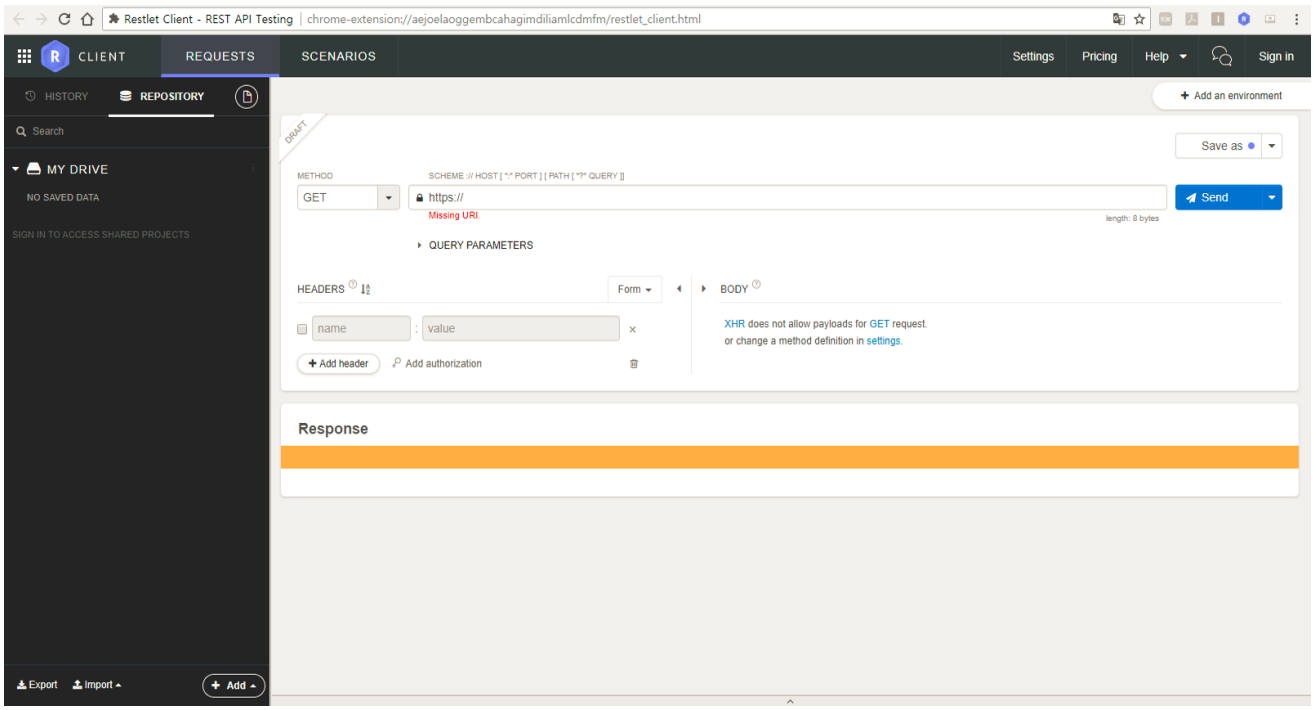
Chapter 02 댓글 처리와 REST

- REST방식은 URI + Http의 메서드(get, put, post) 를 이용하여 필요한 작업을 처리한다.
- REST 방식을 사용하는 원칙은 크게 두 가지
  - URI가 원하는 리소스
  - URI에는 식별할 수 있는 데이터를 같이 전달
- 전송 방식

Http method	설명
GET(/boards/123)	자료의 조회용
DELETE (/boards/123)	자료의 삭제
POST (/boards 혹은 /board/new) + 데이터	신규 자료의 등록
PUT (/boards/123) + 데이터	신규 자료의 수정 혹은 등록
PATCH	간혹 PUT 방식 대용으로 사용

2.1 Advanced REST Client를 이용한 테스트

- REST 클라이언트 프로그램을 통한 테스트



2.2 REST와 Ajax

- Ajax (Asynchronous JavaScript and XML, 비동기화된 자바스크립트와 XML)는 주로 브라우저에서 대화형으로 서버와 데이터를 주고받는 형태의 메시지 전송 방식을 의미한다.
- 대표적인 예로 포털사이트의 자동완성 기능들이 Ajax를 활용한 가장 대표적인 서비스이다.
- Ajax의 가장 큰 특징
    - 브라우저의 화면 전환이 없기 때문에 사용자 경험 측면에서 좋다.
    - 서버에서 화면에 필요한 모든 데이터를 만드는 대신 서버는 필요한 데이터만 전달하기 때문에, 개발의 무게 중심이 브라우저 쪽으로 많이 배분된다.
- REST 방식
    - 데이터를 호출하고 사용하는 방식
- Ajax
    - 실제로 그를 이용하는 수단

2.3 댓글 처리를 위한 준비

2.3.1 전달 방식과 처리 방식의 결정

- 댓글 처리의 방식 결정

URI	전송방식	설명
/replies/all/123	GET	게시물 123번의 모든 댓글 리스트
/replies/ + 데이터	POST	새로운 댓글 추가
/replies/456 + 데이터	PUT/PATCH	456 댓글의 수정
/replies/456	DELETE	456 댓글의 삭제

2.3.2 데이터 전송 방식의 결정

- REST 방식으로 데이터를 서비스 받는 경우는 주로 웹과 모바일이다.
- 데이터의 전송 형태로는 XML이나 JSON을 사용하는 경우가 많다.

2.3.3 댓글을 위한 테이블 설정

- tbl_reply

```
-- tbl_reply 테이블의 생성 스크립트
create table tbl_reply (
    rno int not null auto_increment
    , bno int not null default 0
    , replytext varchar(1000) not null
    , replyer varchar(50) not null
    , regdate timestamp not null default now()
    , updatedate timestamp not null default now()
    , primary key(rno)
);
-- tbl_reply에 외래키(foreign key) 추가
alter table tbl_reply add constraint fk_board foreign key (bno) references tbl_board (bno);
```

2.3.4 댓글을 위한 도메인 객체 설계

- ReplyVO

```
public class ReplyVO {
    private Integer rno;
    private Integer bno;
    private String replytext;
    private String replyer;

    private Date regdate;
    private Date updatedate;
// ... getter / setter, toString()
}
```

2.3.5 ReplyDAO

2.3.5.1 ReplyDAO 인터페이스의 작성

- ReplyDAO 인터페이스

```
public interface ReplyDAO {

    public List<ReplyVO> list(Integer bno) throws Exception;

    public void create(ReplyVO vo) throws Exception;

    public void update(ReplyVO vo) throws Exception;

    public void delete(ReplyVO vo) throws Exception;

}
```

2.3.5.2 XML Mapper의 작성

- replyMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="org.zerock.mapper.ReplyMapper">
    <select id = "list" resultType = "ReplyVO">
        SELECT
            *
        FROM
            tbl_reply
        WHERE bno = #{bno}
        ORDER BY rno DESC
    </select>
    <insert id="create">
        INSERT INTO tbl_reply
        (bno, replytext, replyer)
        VALUES
        (#{bno}, #{replytext}, #{replyer})
    </insert>
    <update id="update">
        UPDATE
            tbl_reply
        SET replytext = #{replytext}, updatedate = now()
        WHERE rno = #{rno}
    </update>
    <delete id = "delete">
        delete from tbl_reply
        WHERE rno = #{rno}
    </delete>
</mapper>
```

2.3.5.3 ReplyDAOImpl의 작성

- ReplyDAOImpl

```
@Repository
public class ReplyDAOImpl implements ReplyDAO{

    @Inject
    private SqlSession session;

    private static String namespace = "org.zerock.mapper.ReplyMapper";

    @Override
    public List<ReplyVO> list(Integer bno) throws Exception {

        return session.selectList(namespace + ".list", bno);
```

```
    }

    @Override
    public void create(ReplyVO vo) throws Exception {
        session.insert(namespace + ".create", vo);
    }

    @Override
    public void update(ReplyVO vo) throws Exception {
        session.update(namespace + ".update", vo);
    }

    @Override
    public void delete(Integer rno) throws Exception {
        session.delete(namespace + ".delete", rno);
    }
}
```

2.3.6 ReplyService/ReplyServiceImpl 작성

- ReplyService 인터페이스

```
public interface ReplyService {

    public void addReply(ReplyVO vo) throws Exception;

    public List<ReplyVO> listReply(Integer bno) throws Exception;

    public void modifyReply(ReplyVO vo) throws Exception;

    public void removeReply(Integer rno) throws Exception;
}
```

- ReplyServiceImpl

```
@Service
public class ReplyServiceImpl implements ReplyService {

    @Inject
    private ReplyDAO dao;

    @Override
    public void addReply(ReplyVO vo) throws Exception {
        dao.create(vo);
    }

    @Override
    public List<ReplyVO> listReply(Integer bno) throws Exception {
        return dao.list(bno);
```

```
    }

    @Override
    public void modifyReply(ReplyVO vo) throws Exception {
        dao.update(vo);
    }

    @Override
    public void removeReply(Integer rno) throws Exception {
        dao.delete(rno);
    }
}
```