RESTful API

RESTful API

REST(Representational State Transfer)의 의미

- 기존 HTTP 프로토콜을 위반하지 않고 호환되며 HTTP를 개선시킬 방법으로 나온 객체
    - HTTP Object Model -> REST로 부르기 시작
- 분산 하이퍼미디어 시스템(예:web)을 위한 아키텍쳐 스타일
    - 아키텍쳐 스타일 : 제약조건의 집합

REST를 구성하는 스타일

---

1. client - server
2. stateless
3. cacheable
4. uniform interface
    - indentification of resources
    - manipulation of resources through representations
        - self-descriptive messages (**자신을 설명하는 메시지**)
            - 목적지 Host가 어딘지
            - Body가 어떤 언어로 되어 있는지
            - Body를 어떻게 해석해야할 지
        - 언제나 메세지만 가지고 해석이 가능하기 때문에 서버나 클라이언트가 변경되어도 확장이 가능함
        - Ex) Json일 때 self-descriptive 적용 방법
            - Media type
                - Media Type과 Media Schema를 정의하고 Schema를 IANA에 등록하는 방법
                - 단점) 매번 Media type 문서를 등록해야함
            - Profile
                - json body에 관련된 key에 대한 의미를 *명세*에 작성
                - Link 헤더에 profile relation으로 해당 명세를 링크 json
                Link: <https://example.org/docs/todos>; rel="profile"

            - 이제 메세지를 보는 사람은 명세를 찾아갈 수 있으므로 문서의 의미를 온전히 해석 가능해짐
    - HATEOAS (hypermedia as the engine of application state) *[application의 상태는 Hyperlink를 이용해 전이되어야함]*
        - json의 Link 헤더
            - html의 <a> tag
            - 링크를 타고 다른 상태(페이지)로 이동한 후에 다음 링크와 바인딩이 되기 때문에 링크가 동적으로 바뀌어도 애플리케이션에 영향을 주지 않음 -> 이는 독립적인 발전을 가능케 함
            - Ex) Json일 때 HATEOAS 적용 방법
                - data로 json
                { "links":{ "todo":"https://example.org/todos/{id}" } }
                - HTTP 헤더로 json Location: /todos/1 Link: </todos/>; rel="collection"
                - 정의된 relation만 활용한다면 표현에 한계가 있음
                - data로 하는 방법을 통해 User가 customizing하는 것과 HTTP 헤더로 이미 정해진 링크를 쓰는 것을 적절히 섞어 쓰는 것이 옳음
5. layered system
6. code-on-demand(optional)
    - server가 client에게 code를 보낼 수 있음을 의미

REST API 설계 규칙

---

1. 슬래시 구분자(/)는 계층 관계를 나타내는데 사용
    - Ex) http://restapi.exmaple.com/houses/apartments
2. URI 마지막 문자로 슬래시(/)를 포함하지 않는다.
    - URI에 포함되는 모든 글자는 Resource의 유일한 식별자로 사용되어야 하며 URI가 다르다는 것은 Resource가 다르다는 것이고, 역으로 Resource가 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/aprtments/ (x)
3. 하이픈(-)은 URI 가독성을 높이는데 사용
    - URI가 불가피하게 긴 경우 하이픈을 사용해 가독성을 높임
4. 언더바(_)는 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 header 사용
    - 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. Resource간에 연관 관계가 있는 경우
   - /리소스명/리소스ID/관계가 있는 다른 리소스명
   - Ex) `GET : /users/{userid}/devices

Spring에서 REST API 제공방법

Spring 버전별

1. Spring3 은 @ResponseBody 어노테이션을 지원하면서 REST 방식의 처리 지원
2. `Spring4` 부터 `@RestController` 로 지원
   - @RestController = @ResponseBody + @Controller + ... + ...

```
@RequestMapping("/login-test")
@ResponseBody
public Map<String,Object> loginTestController(UserDTO userDTO, Model model){

  UserVO userVO = userService.login(userDTO);
  Map<String,Object> loginUser = new HashMap<String, Object>();
  if(userVO != null){
        loginUser.put("user",userVO);
  }
  return loginUser;
}
```

- @ResponseBody를 메소드에 적으므로써 해당 메소드의 반환값을 Json 형태로 변형시켜줌

Json으로 데이터 주고 받기

Json

JavaScript Object Notation

네트워크를 통해 데이터를 주고받는 데 자주 사용되는 경량의 데이터 형식

name-value의 쌍으로 이루어진 집합

```
{"users":[
   {
      "firstName":"Ray",
      "lastName":"Villalobos",
      "joined":{
         "month":"January",
         "day":12,
         "year":2012
      }
   },
   {
      "firstName":"John",
      "lastName":"Jones",
      "joined":{
         "month":"April",
         "day":28,
         "year":2010
      }
   }
]}
```

- 중괄호를 통해 객체에 대한 정보를 구분
- "name":value
    - name type = String
    - value type = 기본 자료형, 배열, 객체
- 대괄호 []는 배열을 의미

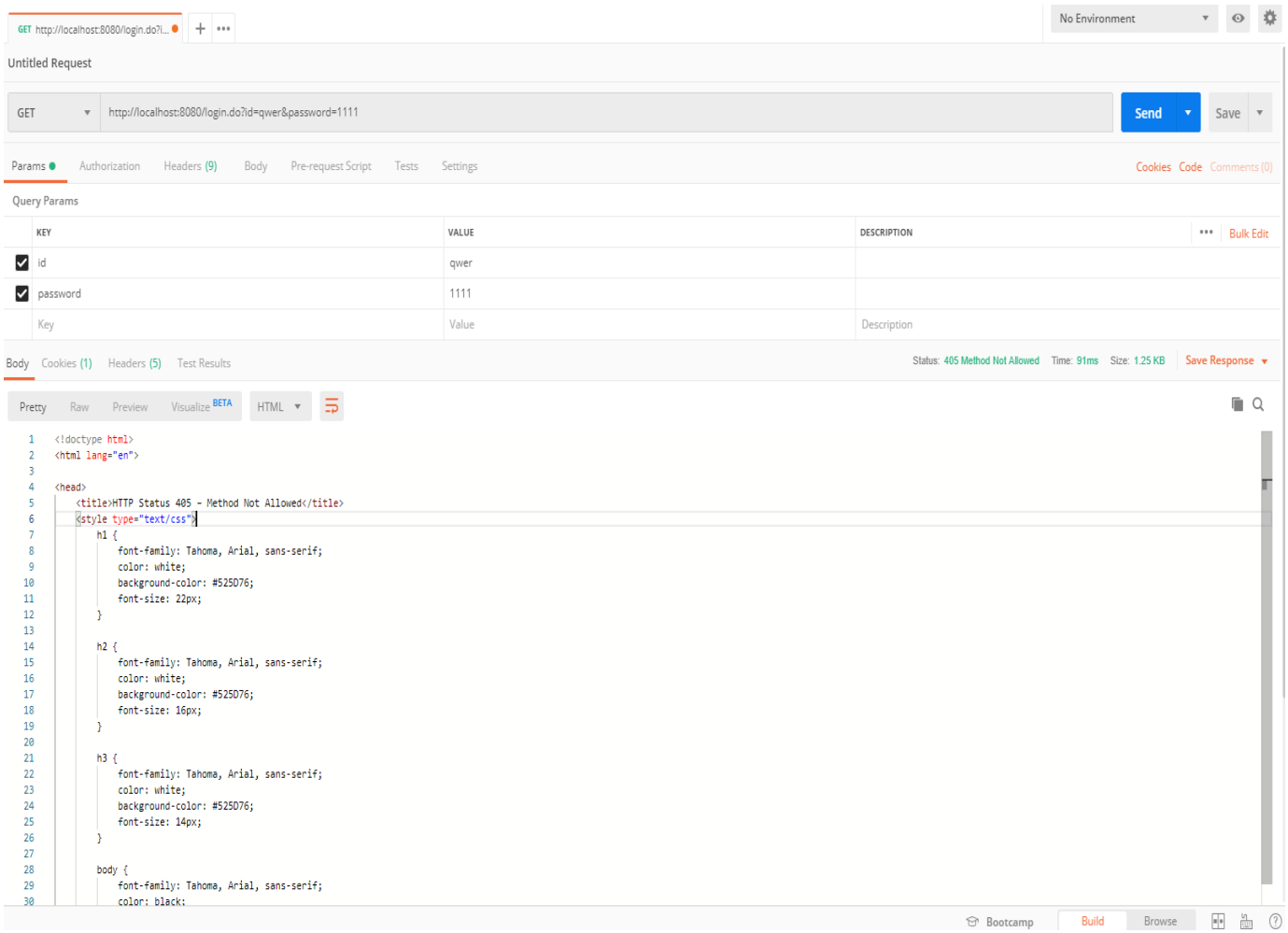
Postman

개발한 API를 테스트하고, 테스트 결과를 공유하여 API 개발의 생산성을 높여주는 플랫폼

Postman downloads

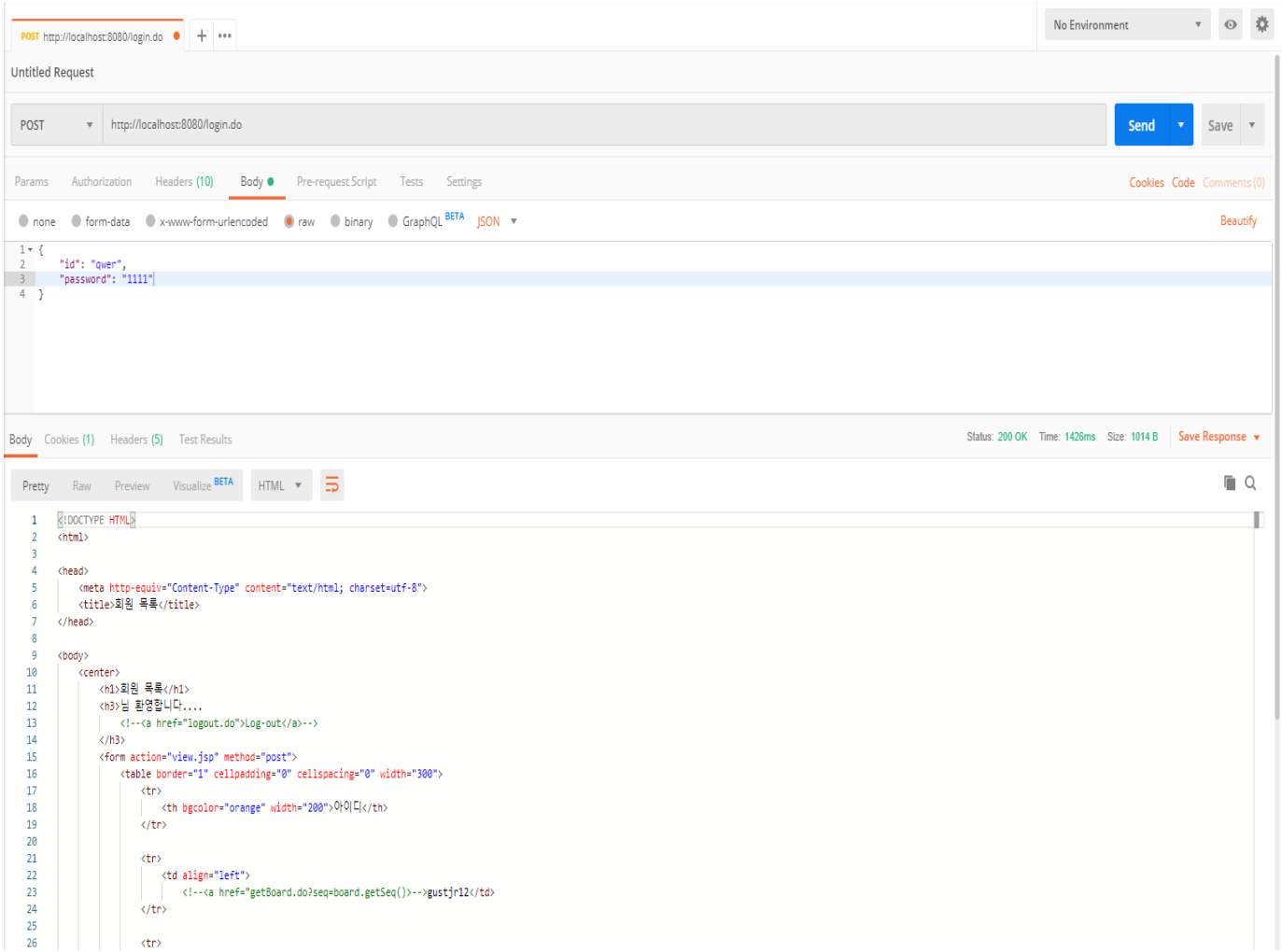
활용

- Query String이 포함된 GET 방식의 호출
    - URI에 query string을 입력(Ex : ?id=1&name=")하여 호출하면
      query string에 있는 key와 value값이 URI 입력칸 아래 [key]와 [value]로 추가되고,
      API를 호출한 결과는 하단의 [Body]영역에 출력



- 현재 Controller에 RequestMapping에 적용되는 method를 'POST'로 설정해서 HTTP Status 405 반환

- JSON이 사용된 POST방식의 호출
    - 요청할 URI를 입력 후, Body에서 raw-JSON(application/json) 을 선택하고 JSON Data를 입력
    - Header의 Content-Type이 application/json이 자동으로 추가됨
    - 그 밖의 HTTP Header 정보를 [Header]에 추가



- Collections
    - 개발한 API를 그룹화하는 기능
- Code generate
    - 우측의 code 버튼을 누르면 다양한 언어별로 해당 API를 호출하는 Source code 생성
    - HTTP, Java, Ruby, C, cURL, C#, Go, JavaScript, PHP, Python 등
- Manage environments
    - 환경 (local, development, production) 마다 다르게 호출해야 하는 경우를 관리해주는 기능
        - hostname
        - parameter
    - 우측 상단의 톱니바퀴 버튼 클릭으로 사용 가능
- Interceptor
    - chrome 브라우저를 이용하여 브라우저 내에서 발생한 Request를 자동으로 Postman History에 등록해주는 기능
    - 특정 APi를 분석, 이용하고 싶을 때 개발자 도구를 통해 값을 일일이 복사하지 않아도 됨

- Ex) 로그인 기능부터 원하는 데이터를 얻을 때까지 전체 flow를 브라우저를 통해 한 바퀴 돌아본 뒤, 기록되어 있는 history를 분석하여 활용가능

Tomcat + Spring을 활용한 Server / Client 개발

(DAO모델 - JSP화면 - Controller)