

# 질문/답변 게시판

—



SLiPP

Search



Home ▾

Posts

로그인

회원가입

로그아웃

개인정보수정

8

국내에서 Ruby on Rails와 Play가 활성화되기 힘든 이유는 뭘까?

2016-01-15 18:47 자바지기

12

runtime 에 reflect 발동 주체 객체가 뭔지 알 방법이 있을까요?

2016-01-05 18:47 김문수

«

1

2

3

4

5

»

질문하기

SLiPP

Search



Home ▾

Posts

로그인

회원가입

로그아웃

개인정보수정

사용자 아이디

User ID

비밀번호

Password

이름

Name

이메일

Email

회원가입

SLiPP

Search



Home ▾

Posts

로그인

회원가입

로그아웃

개인정보수정

사용자 아이디

User ID

비밀번호

Password

로그인

## InitializingBean implements afterPropertiesSet() 호출되지 않는 문제.

**kimmunsu**

2015-12-30 01:47

A 에 의존성을 가지는 B라는 클래스가 있습니다.

B라는 클래스는 InitializingBean 을 상속하고 afterPropertiesSet을 구현하고 있습니다. 서버가 가동되면서 bean들이 초기화되는 시점에 B라는 클래스의 afterPropertiesSet 메소드는

A라는 클래스의 특정 메소드인 afunc()를 호출하고 있습니다.

[수정](#) • [삭제](#) • [목록](#)

### 2개의 의견

**자바지기**

2016-01-12 14:06

이 글만으로는 원인 파악하기 힘들겠다. 소스 코드와 설정을 단순화해서 공유해 주면 같이 디버깅해줄 수도 있겠다.

[수정](#) • [삭제](#)**자바지기**

2016-01-12 14:06

이 글만으로는 원인 파악하기 힘들겠다. 소스 코드와 설정을 단순화해서 공유해 주면 같이 디버깅해줄 수도 있겠다.

[수정](#) • [삭제](#)

Post

# HTTP 이해

—



## 개발 환경 설정 - git clone 및 maven 빌드

- <https://github.com/slipp/web-application-server> 프로젝트를 자신의 계정으로 Fork한다.  
Github 우측 상단의 Fork 버튼을 클릭하면 자신의 계정으로 Fork된다.
- Fork한 프로젝트를 eclipse 또는 터미널에서 clone 한다.
- Fork한 프로젝트를 eclipse로 import한 후에 Maven 빌드 도구를 활용해 eclipse 프로젝트로 변환한다.(mvn eclipse:clean eclipse:eclipse)
- 빌드가 성공하면 반드시 refresh(fn + f5)를 실행해야 한다.

## 웹 서버 시작 및 테스트

- `webserver.WebServer` 는 사용자의 요청을 받아 `RequestHandler` 에 작업을 위임하는 클래스이다.
- 사용자 요청에 대한 모든 처리는 `RequestHandler` 클래스의 `run()` 메서드가 담당한다.
- `WebServer`를 실행한 후 브라우저에서 <http://localhost:8080>으로 접속해 “Hello World” 메시지가 출력되는지 확인한다.



# 요구사항 1

<http://localhost:8080/index.html> 로 접속했을 때 webapp 디렉토리의 index.html 파일을 읽어 클라이언트에 응답한다.

http header 예

```
GET /index.html HTTP/1.1  
Host: localhost:8080  
Connection: keep-alive  
Accept: */*
```

- InputStream => InputStreamReader => BufferedReader
  - 구글에서 “java inputstream bufferedreader “로 검색 후 문제 해결
- BufferedReader.readLine() 메소드 활용해 라인별로 http header 읽는다.
- http header 전체를 출력한다.
  - header 마지막은 while (!"".equals(line)) } 로 확인 가능하다.
  - line이 null 값인 경우에 대한 예외 처리도 해야 한다. 그렇지 않을 경우 무한 루프에 빠진다.(if (line == null) { return;})

http header 예

**GET /index.html HTTP/1.1**

Host: localhost:8080

Connection: keep-alive

Accept: \*/\*

- Header의 첫 번째 라인에서 요청 URL(위 예의 경우 /index.html 이다.)을 추출한다.
  - `String[] tokens = line.split(" ");` 활용해 문자열을 분리할 수 있다.
- 구현은 별도의 유틸 클래스를 만들고 단위 테스트를 만들어 진행하면 편하다.

http header 예

```
GET /index.html HTTP/1.1  
Host: localhost:8080  
Connection: keep-alive  
Accept: */*
```

- 요청 URL에 해당하는 파일을 webapp 디렉토리에서 읽어 전달하면 된다.
- 구글에서 “java files readallbytes”로 검색해 파일 데이터를 byte[]로 읽는다.

```
byte[] body = Files.readAllBytes(new File("./webapp" + url).toPath());
```

# 잠깐 휴식

- 요구사항 구현을 통해 새롭게 알게 된 내용을 README.md 파일에 기록한다.
- 구현 중 모르는 상태에서 구현에 집중한 내용 또는 용어 등을 README.md 파일에 기록한다.

## 요구사항 2

“회원가입” 메뉴를 클릭하면 <http://localhost:8080/user/form.html> 으로 이동하면서 회원가입할 수 있다. 회원가입한다.

회원가입을 하면 다음과 같은 형태로 사용자가 입력한 값이 서버에 전달된다.

`/create?userId=javajigi&password=password&name=%EB%B0%95%EC%9E%AC%EC%84%B1&email=javajigi%40slipp.net`

HTML과 URL을 비교해 보고 사용자가 입력한 값을 파싱해 `model.User` 클래스에 저장한다.

## http header 예

GET /create?userId=javajigi&password=password&name=%EB%B0%95%EC%9E%AC%EC%84%B1&email=javajigi%40slipp.net HTTP/1.1

- Header의 첫 번째 라인에서 요청 URL을 추출한다.
- 요청 URL에서 접근 경로와 이름/값을 추출해 User 클래스에 담는다.
- 구현은 가능하면 junit을 활용해 단위 테스트를 진행하면서 하면 좀 더 효과적으로 개발 가능하다.
- 이름/값 파싱은 util.HttpRequestUtils 클래스의 parseQueryString() 메서드를 활용한다.

```
String url = "/?data=234";  
int index = url.indexOf("?");  
String requestPath = url.substring(0, index);  
String params = url.substring(index+1);
```

# 잠깐 휴식

- 요구사항 구현을 통해 새롭게 알게 된 내용을 README.md 파일에 기록한다.
- 구현 중 모르는 상태에서 구현에 집중한 내용 또는 용어 등을 README.md 파일에 기록한다.



# 요구사항 3

<http://localhost:8080/user/form.html> 파일의 form 태그 method를 get에서 post로 수정한 후 회원가입 기능이 정상적으로 동작하도록 구현한다.

## http header와 body 예

```
POST /create HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Content-Length: 59
Content-Type: application/x-www-form-urlencoded
Accept: */*
```

```
userId=javajigi&password=password&name=%EB%B0%95%EC%9E%AC%EC%84%B1&email=javajigi%40slipp.net
```

- POST method로 데이터를 전달할 경우 전달하는 데이터는 HTTP Body에 담긴다.
- HTTP Body는 HTTP header 이후 빈 공백을 가지는 한 줄(line) 다음부터 시작한다.
- HTTP Body에 전달되는 데이터는 GET method의 이름/값과 같다.
- BufferedReader에서 본문 데이터는 util.IOUtils 클래스의 readData() 메서드를 활용한다. 본문의 길이는 http header의 Content-Length의 값이다.
- 회원가입시 입력한 모든 데이터를 추출해 User 객체를 생성한다.

# 잠깐 휴식

- 요구사항 구현을 통해 새롭게 알게 된 내용을 README.md 파일에 기록한다.
- 구현 중 모르는 상태에서 구현에 집중한 내용 또는 용어 등을 README.md 파일에 기록한다.

## 요구사항 4

“회원가입”을 완료하면 /index.html 페이지로 이동하고 싶다. 현재는 URL이 /create 로 유지되는 상태로 읽어서 전달할 파일이 없다. 따라서 서블릿의 redirect 방식처럼 회원가입을 완료한 후 “index.html”로 이동해야 한다. 즉, 브라우저의 URL이 /index.html로 변경해야 한다.

- HTTP 응답 헤더의 status code를 200이 아니라 302 code를 사용한다.
  - [http://en.wikipedia.org/wiki/HTTP\\_302](http://en.wikipedia.org/wiki/HTTP_302) 문서 참고

# 잠깐 휴식

- 요구사항 구현을 통해 새롭게 알게 된 내용을 README.md 파일에 기록한다.
- 구현 중 모르는 상태에서 구현에 집중한 내용 또는 용어 등을 README.md 파일에 기록한다.

# 요구사항 5

“로그인” 메뉴를 클릭하면 <http://localhost:8080/user/login.html> 으로 이동해 로그인할 수 있다. 로그인이 성공하면 index.html로 이동하고, 로그인이 실패하면 /user/login\_failed.html로 이동해야 한다.

앞에서 회원가입한 사용자로 로그인할 수 있어야 한다. 로그인이 성공하면 cookie를 활용해 로그인 상태를 유지할 수 있어야 한다. 로그인이 성공할 경우 요청 header의 Cookie header 값이 logged=true, 로그인이 실패하면 Cookie header 값이 logged=false로 전달되어야 한다.

- HTTP 응답 헤더(response header)에 Set-Cookie 를 추가해 로그인 성공 여부를 전달한다.

http response header 예

```
HTTP/1.1 200 OK  
Content-Type: text/html  
Set-Cookie: logged=true
```

- HTTP 요청 헤더의 Cookie 헤더 값을 활용해 응답 헤더로 전달한 값을 추출한다.

http request header 예

```
GET /index.html HTTP/1.1  
Host: localhost:8080  
Connection: keep-alive  
Accept: */*  
Cookie: logged=true
```



- 정상적으로 로그인 되었는지 확인하려면 앞 단계에서 회원가입한 데이터를 유지해야 한다.
  - 앞 단계에서 회원가입할 때 생성한 User 객체를 DataBase.addUser() 메서드를 활용해 RAM 메모리에 저장한다.
- 아이디와 비밀번호가 같은지를 확인해 로그인이 성공하면 응답 header의 Set-Cookie 값을 logged=true, 로그인이 실패할 경우 Set-Cookie 값을 logged=false로 설정한다.
- 응답 header에 Set-Cookie 값을 설정한 후 요청 header에 Cookie이 전달되는지 확인한다.

# 잠깐 휴식

- 요구사항 구현을 통해 새롭게 알게 된 내용을 README.md 파일에 기록한다.
- 구현 중 모르는 상태에서 구현에 집중한 내용 또는 용어 등을 README.md 파일에 기록한다.

## 요구사항 6

지금까지 구현한 소스 코드는 stylesheet 파일을 지원하지 못하고 있다.  
Stylesheet 파일을 지원하도록 구현하도록 한다.

http header 예

```
GET ./css/style.css HTTP/1.1
Host: localhost:8080
Accept: text/css, */*;q=0.1
Connection: keep-alive
```

- 응답 헤더의 Content-Type을 text/html로 보내면 브라우저는 html 파일로 인식하기 때문에 css가 정상적으로 동작하지 않는다.
- Stylesheet인 경우 응답 헤더의 Content-Type을 text/css로 전송한다.  
Content-Type은 확장자를 통해 구분할 수도 있으며, 요청 헤더의 Accept를 활용할 수도 있다.

# 잠깐 휴식

- 요구사항 구현을 통해 새롭게 알게 된 내용을 README.md 파일에 기록한다.
- 구현 중 모르는 상태에서 구현에 집중한 내용 또는 용어 등을 README.md 파일에 기록한다.

## 요구사항 7

Javascript도 정확한 Content Type으로 응답하도록 구현한다. Javascript의 Content Type은 application/javascript 이다.

# HTTP와 웹에 대한 추가 학습

- “프로가 되기 위한 웹 기술 입문” 책 8 페이지 ~ 84 페이지까지 학습

