

## 4. Section

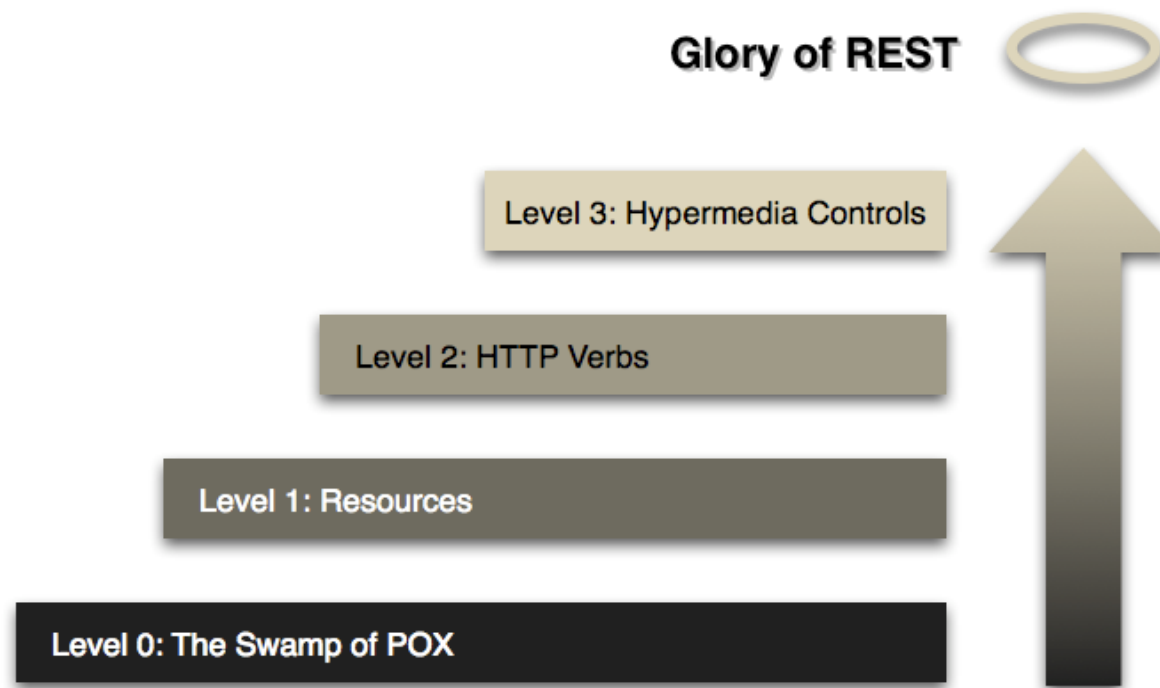
# Spring Boot API 사용

- REST API Level3을 위한 HATEOAS 설정
- REST API Documentation을 위한 Swagger 설정
- RSET API Monitoring을 위한 Actuator 설정
- Spring Security

# step21- Implementing HATEOAS for RESTful Services

- HATEOAS – Hypermedia As the Engine Of Application State

- 현재 리소스와 연관된(호출 가능한) 자원 상태 정보를 제공



- pom.xml

- add dependency

```
<dependency>  
  <groupId>org.springframework.boot</groupId>  
  <artifactId>spring-boot-starter-hateoas</artifactId>  
</dependency>
```

# step21- Implementing HATEOAS for RESTful Services

## ■ spring 2.1.8.RELEASE 일 경우

Resource

ControllerLinkBuilder

```
@GetMapping("/users/{id}")
public Resource<User> retrieveUser(@PathVariable int id) {
    ...
    // HATEOAS
    // "all-users", SERVER_PATH + "/users"
    // retrieveAllUsers
    Resource<User> resource = new Resource<>(user);
    ControllerLinkBuilder linkTo = linkTo(methodOn(this.getClass()).retrieveAllUsers());
    resource.add(linkTo.withRel("all-users"));

    return resource;
}
```

## ■ spring 2.2 일 경우

Resource → EntityModel

ControllerLinkBuilder → WebMvcLinkBuilder

```
@GetMapping("/users/{id}")
public EntityModel<User> retrieveUser(@PathVariable int id) {
    ...
    // HATEOAS
    // "all-users", SERVER_PATH + "/users"
    // retrieveAllUsers
    EntityModel<User> model = new EntityModel<>(user);
    WebMvcLinkBuilder linkTo = linkTo(methodOn(this.getClass()).retrieveAllUsers());
    model.add(linkTo.withRel("all-users"));

    return model;
}
```

# step21- Implementing HATEOAS for RESTful Services

Body Cookies (1) Headers (5) Test Results

Pretty Raw Preview Visualize JSON ↺

```
1 {
2   "id": 1,
3   "name": "Adam",
4   "birthDate": "2020-03-21T05:34:08.040+0000",
5   "_links": {
6     "all-users": {
7       "href": "http://127.0.0.1:8088/users"
8     }
9   }
10 }
```

GET http://127.0.0.1:8088/users/1

Body Cookies (1) Headers (3) Test Results

Pretty Raw Preview Visualize XML ↺

```
1 <Resource>
2   <id>1</id>
3   <name>Kenneth</name>
4   <joinDate>2020-03-23T02:14:44.831+0000</joinDate>
5   <links>
6     <links>
7       <rel>all-users</rel>
8       <href>http://127.0.0.1:8088/users</href>
9       <hreflang/>
10      <media/>
11      <title/>
12      <type/>
13      <deprecation/>
14    </links>
15  </links>
16 </Resource>
```

# step22- Configuring Auto Generation of Swagger Documentation

- pom.xml
  - add dependencies

```
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger2</artifactId>
  <version>2.9.2</version>
</dependency>
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger-ui</artifactId>
  <version>2.9.2</version>
</dependency>
```

```
@Configuration
@EnableSwagger2
// Enable Swagger
public class SwaggerConfig {

    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2);
    }
    // Swagger 2
    // All the paths
    // All the apis
}
```

# step22- Configuring Auto Generation of Swagger Documentation

## ■ Swagger 오류 발생 시

Description:

Parameter 0 of method linkDiscoverers in org.springframework.hateoas.config.HateoasConfiguration required a single bean, but 17 were found:

- modelBuilderPluginRegistry: defined in null
- modelPropertyBuilderPluginRegistry: defined in null
- typeNameProviderPluginRegistry: defined in null
- syntheticModelProviderPluginRegistry: defined in null
- documentationPluginRegistry: defined in null
- apiListingBuilderPluginRegistry: defined in null
- operationBuilderPluginRegistry: defined in null
- parameterBuilderPluginRegistry: defined in null
- expandedParameterBuilderPluginRegistry: defined in null
- resourceGroupingStrategyRegistry: defined in null
- operationModelsProviderPluginRegistry: defined in null
- defaultsProviderPluginRegistry: defined in null
- pathDecoratorRegistry: defined in null
- apiListingScannerPluginRegistry: defined in null
- relProviderPluginRegistry: defined by method 'relProviderPluginRegistry' in class path resource [org.springframework.hateoas.config.HateoasConfiguration.class]
- linkDiscovererRegistry: defined in null
- entityLinksPluginRegistry: defined by method 'entityLinksPluginRegistry' in class path resource [org.springframework.hateoas.config.WebMvcEntityLinksConfiguration.class]

Action:

Consider marking one of the beans as @Primary, updating the consumer to accept multiple beans, or using @Qualifier to identify the bean that should be consumed

해결 방법 1) public class SwaggerConfig {

- Bean 추가

@Bean


```
public LinkDiscoverers discoverers() {  
    List<LinkDiscoverer> plugins = new ArrayList<>();  
    plugins.add(new CollectionJsonLinkDiscoverer());  
    return new LinkDiscoverers(SimplePluginRegistry.create(plugins));  
}
```

해결 방법 2)

- Spring boot 버전 2.1.8.RELEASE로 변경

# step22- Configuring Auto Generation of Swagger Documentation

← → ↻ ⓘ localhost:8088/swagger-ui.html

 **swagger**

## Api Documentation <sup>1.0</sup>

[ Base URL: localhost:8088/ ]  
<http://localhost:8088/v2/api-docs>

Api Documentation  
[Terms of service](#)  
[Apache 2.0](#)

---

**basic-error-controller** Basic Error Controller

---

**hello-world-controller** Hello World Controller

---

**user-resource** User Resource

---

**Models**

← → ↻ ⓘ localhost:8088/v2/api-docs

```
{
  swagger: "2.0",
  - info: {
    description: "Api Documentation",
    version: "1.0",
    title: "Api Documentation",
    termsOfService: "urn:tos",
    contact: { },
    - license: {
      name: "Apache 2.0",
      url: "http://www.apache.org/licenses/LICENSE-2.0"
    }
  },
  host: "localhost:8088",
  basePath: "/",
  - tags: [
    - {
      name: "basic-error-controller",
      description: "Basic Error Controller"
    },
    - {
      name: "hello-world-controller",
      description: "Hello World Controller"
    },
    - {
      name: "user-resource",
      description: "User Resource"
    }
  ],
  - paths: {
    - /error: {
      - get: {
        - tags: [
          "basic-error-controller"
        ],
        summary: "error",
        operationId: "errorUsingGET",
        - produces: [
          "*"/*"
        ],
        responses: {
          200: {
            description: "OK"
          },
          400: {
            description: "Bad Request"
          },
          500: {
            description: "Internal Server Error"
          }
        }
      }
    }
  }
}
```

# step23- Introduction to Swagger Documentation Format

```
localhost:8088/v2/api-docs
{
  swagger: "2.0",
+  info: { ... },
  host: "localhost:8088",
  basePath: "/",
+  tags: [ ... ],
+  paths: { ... },
+  definitions: { ... }
}
```

```
- info: {
  description: "Api Documentation",
  version: "1.0",
  title: "Api Documentation",
  termsOfService: "urn:tos",
  contact: { },
-  license: {
    name: "Apache 2.0",
    url: "http://www.apache.org/licenses/LICENSE-2.0"
  },
},
```

```
- /admin/users/{id}: {
  - get: {
    - tags: [
      "user-controller"
    ],
    summary: "retrieveUser4AdminV1",
    operationId: "retrieveUser4AdminV1UsingGET",
    - produces: [
      "application/vnd.company.appv2+json",
      "application/vnd.company.appv1+json"
    ],
    - parameters: [
      - {
        name: "id",
        in: "path",
        description: "id",
        required: true,
        type: "integer",
        format: "int32"
      }
    ],
    - responses: {
      - 200: {
        description: "OK",
        - schema: {
          $ref: "#/definitions/MappingJacksonValue"
        }
      }
    }
  },
}
```



# step24- Enhancing Swagger Documentation with Custom Annotations

@Configuration

@EnableSwagger2

// Enable Swagger

public class SwaggerConfig {

```
private static final Contact DEFAULT_CONTACT = new Contact("Kenneth Lee",
    "http://www.joneculsting.co.kr", "edowon@joneconsluting.co.kr");
private static final ApiInfo DEFAULT_API_INFO = new ApiInfo("Awesome API Title",
    "Awesome API Documentation", "1.0", "urn:tos", DEFAULT_CONTACT, "Apache 2.0",
    "http://www.apache.org/licenses/LICENSE-2.0", new ArrayList());
private static final Set<String> DEFAULT_PRODUCES_AND_CONSUMES
    = new HashSet<>(Arrays.asList("application/json", "application/xml"));
```

@Bean

public Docket api() {

```
    return new Docket(DocumentationType.SWAGGER_2)
        .apiInfo(DEFAULT_API_INFO)
        .produces(DEFAULT_PRODUCES_AND_CONSUMES)
        .consumes(DEFAULT_PRODUCES_AND_CONSUMES)
        ;
```

}

```
{
  "swagger": "2.0",
  "info": {
    "description": "Awesome API Documentation",
    "version": "1.0",
    "title": "Awesome API Title",
    "termsOfService": "urn:tos",
    "contact": {
      "name": "Kenneth Lee",
      "url": "http://www.joneculsting.co.kr",
      "email": "edowon@joneconsluting.co.kr"
    },
    "license": {
      "name": "Apache 2.0",
      "url": "http://www.apache.org/licenses/LICENSE-2.0"
    }
  },
  "host": "localhost:8088",
  "basePath": "/",
  "tags": [ ... ], // 3 items
  "consumes": [
    "application/xml",
    "application/json"
  ],
  "produces": [
    "application/xml",
    "application/json"
  ],
  "paths": {
    "/admin/users": {
      "get": {
        "tags": [
          "user-controller"
        ],
        "summary": "retrieveUsers4Admin",
        "operationId": "retrieveUsers4AdminUsingGET",
```

# step24- Enhancing Swagger Documentation with Custom Annotations

```
@Data
@AllArgsConstructor
@JsonFilter("UserInfo")
@NoArgsConstructor
@ApiModel(description = "사용자 상세 정보")
public class User {
    private Integer id;

    @Size(min=2, message="Name은 2글자 이상 입력해 주세요.")
    @ApiModelProperty(notes = "사용자 이름을 입력해 주세요.")
    private String name;

    @Past
    @ApiModelProperty(notes = "등록일을 입력해 주세요.")
    private Date joinDate;

    @ApiModelProperty(notes = "비밀번호를 입력해 주세요.")
    private String password;

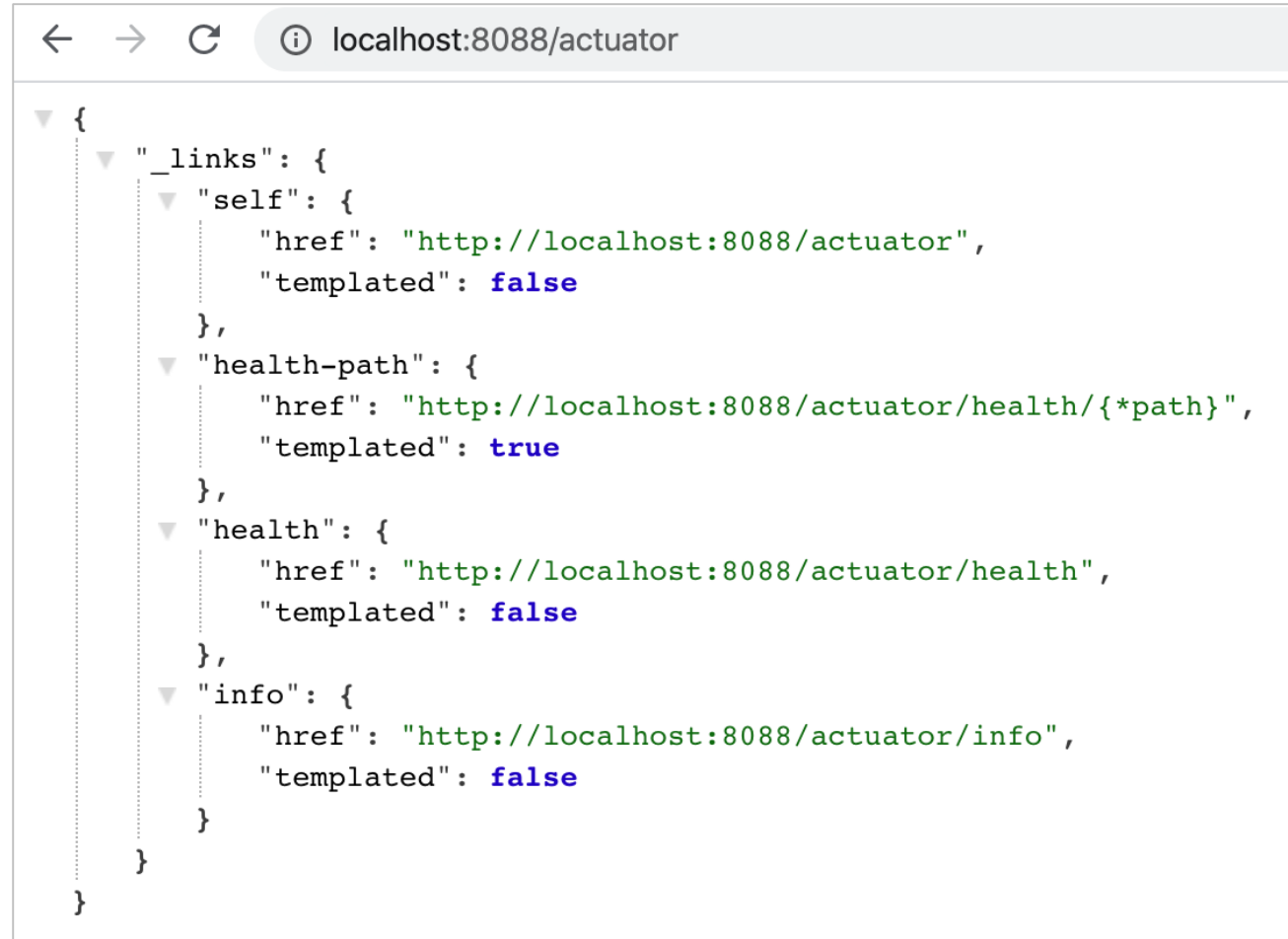
    @ApiModelProperty(notes = "주민등록번호를 입력해 주세요.")
    private String ssn;
}
```

```
- definitions: {
  + FilterProvider: { ... },
  + HelloWorldBean: { ... },
  + Link: { ... },
  + MappingJacksonValue: { ... },
  + ModelAndView: { ... },
  + Resource«User»: { ... },
  - User: {
    type: "object",
    - properties: {
      - id: {
        type: "integer",
        format: "int32"
      },
      - joinDate: {
        type: "string",
        format: "date-time",
        description: "등록일을 입력해 주세요."
      },
      - name: {
        type: "string",
        description: "사용자 이름을 입력해 주세요."
      },
      - password: {
        type: "string",
        description: "비밀번호를 입력해 주세요."
      },
      - ssn: {
        type: "string",
        description: "주민등록번호를 입력해 주세요."
      }
    },
    title: "User",
    description: "사용자 상세 정보"
  },
}
```

# step25- Monitoring APIs with Spring Boot Actuator

- pom.xml
  - add dependency

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```



```
{
  "_links": {
    "self": {
      "href": "http://localhost:8088/actuator",
      "templated": false
    },
    "health-path": {
      "href": "http://localhost:8088/actuator/health/{*path}",
      "templated": true
    },
    "health": {
      "href": "http://localhost:8088/actuator/health",
      "templated": false
    },
    "info": {
      "href": "http://localhost:8088/actuator/info",
      "templated": false
    }
  }
}
```

# step25- Monitoring APIs with Spring Boot Actuator

## ■ application.yml

**management:**  
**endpoints:**  
**web:**  
**exposure:**  
**include: "\*"**

```
..  
▼ "configprops": {  
  "href": "http://localhost:8088/actuator/configprops",  
  "templated": false  
},  
▼ "env": {  
  "href": "http://localhost:8088/actuator/env",  
  "templated": false  
},  
▼ "env-toMatch": {  
  "href": "http://localhost:8088/actuator/env/{toMatch}",  
  "templated": true  
},  
▼ "info": {  
  "href": "http://localhost:8088/actuator/info",  
  "templated": false  
},
```

← → ↻ ⓘ localhost:8088/actuator

```
▼ {  
  ▼ "_links": {  
    ▼ "self": {  
      "href": "http://localhost:8088/actuator",  
      "templated": false  
    },  
    ▼ "auditevents": {  
      "href": "http://localhost:8088/actuator/auditevents",  
      "templated": false  
    },  
    ▼ "beans": {  
      "href": "http://localhost:8088/actuator/beans",  
      "templated": false  
    },  
    ▼ "caches-cache": {  
      "href": "http://localhost:8088/actuator/caches/{cache}",  
      "templated": true  
    },  
    ▼ "caches": {  
      "href": "http://localhost:8088/actuator/caches",  
      "templated": false  
    },  
    ▼ "health-component": {  
      "href": "http://localhost:8088/actuator/health/{component}",  
      "templated": true  
    },  
  },  
}
```

# step25- Monitoring APIs with Spring Boot Actuator

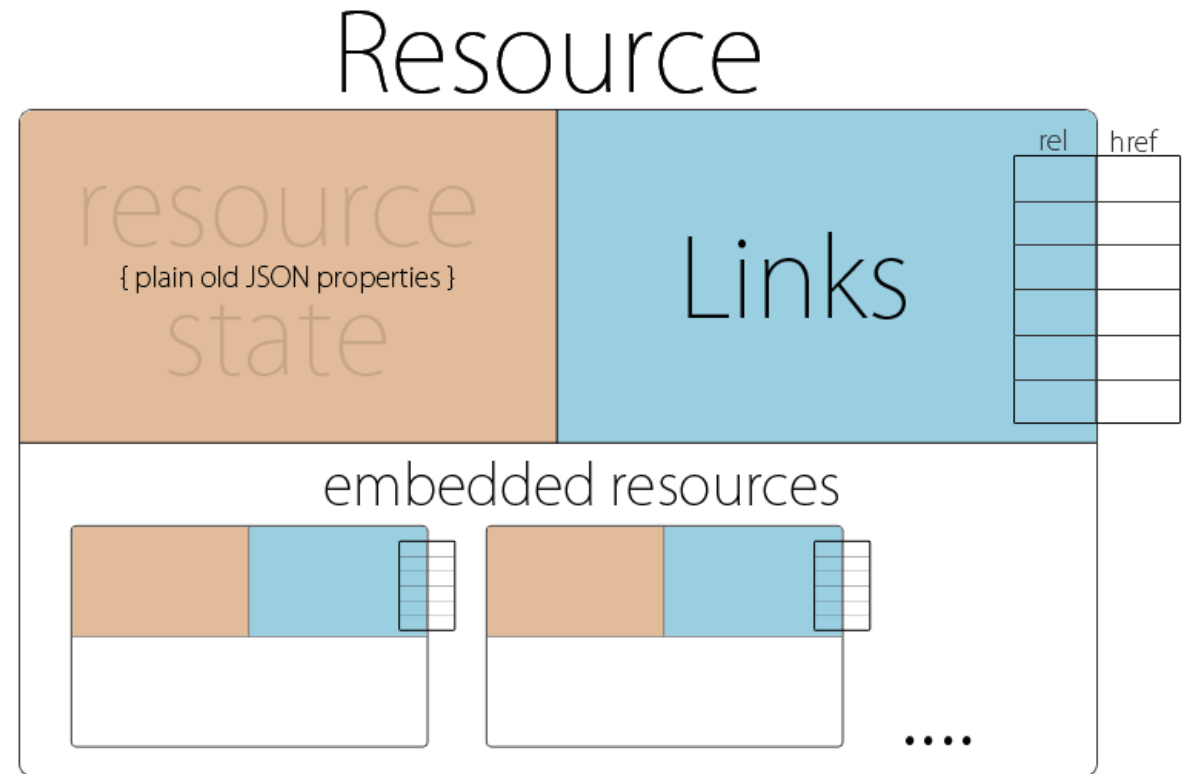
## ■ HAL Browser

- Hypertext Application Language
- “HAL is a *simple format* that gives a consistent and *easy way* to *hyperlink* between *resources* in your API.”

## ■ pom.xml

- add dependency

```
<dependency>
  <groupId>org.springframework.data</groupId>
  <artifactId>spring-data-rest-hal-browser</artifactId>
</dependency>
```



# step25- Monitoring APIs with Spring Boot Actuator

The screenshot shows the Spring Boot Actuator HAL Browser interface. The browser address bar shows `localhost:8088/browser/index.html#/actuator`. The page title is "The HAL Browser (for Spring Data REST)".

**Explorer**

Input field: `/actuator` [Go!]

**Custom Request Headers**

Input field: [ ]

**Properties**

Input field: `{ }`

**Links**

rel	title	name / index	docs	GET	NON-GET
self				→	!
auditevents				→	!
beans				→	!
caches-cache				→	!
caches				→	!
health				→	!

**Inspector**

**Response Headers**

200 success

content-type: application/json;charset=UTF-8  
date: Sat, 21 Mar 2020 12:15:36 GMT  
transfer-encoding: chunked

**Response Body**

```
{
  "_links": {
    "self": {
      "href": "http://localhost:8088/actuator",
      "templated": false
    },
    "auditevents": {
      "href": "http://localhost:8088/actuator/auditevents",
      "templated": false
    },
    "beans": {
      "href": "http://localhost:8088/actuator/beans",
      "templated": false
    },
    "caches-cache": {
      "href": "http://localhost:8088/actuator/caches/{cache}",
      "templated": true
    }
  },
  "auditevents": {
    "href": "http://localhost:8088/actuator/auditevents",
    "templated": false
  },
  "beans": {
    "href": "http://localhost:8088/actuator/beans",
    "templated": false
  },
  "caches-cache": {
    "href": "http://localhost:8088/actuator/caches/{cache}",
    "templated": true
  },
  "caches": {
    "href": "http://localhost:8088/actuator/caches",
    "templated": false
  },
  "health": {
    "href": "http://localhost:8088/actuator/health",
    "templated": false
  }
}
```

`http://localhost:8088/actuator/`

`http://localhost:8088/actuator/metrics/`

`http://localhost:8088/actuator/metrics/jvm.memory.max`

# step26- Implementing Basic Authentication with Spring Security

- pom.xml

- add dependency

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
  <version>2.1.7.RELEASE</version>
</dependency>
```

- automatically configure basic security

```
2020-03-21 22:13:07.161 DEBUG 60741 --- [          main] o.s.b.f.s.DefaultSecurityFilterChain :
2020-03-21 22:13:07.166 INFO 60741 --- [          main] .s.s.UserDetailsServiceAdapter :
Using generated security password: 02924524-cc99-47fe-9e1a-7fbe9f5031a4
```

# step26- Implementing Basic Authentication with Spring Security

GET http://127.0.0.1:8088/users

Params Authorization Headers (9) Body Pre-request Script Tests Settings

▼ Headers (1)

KEY	VALUE	DESCRIPTION
Accept-Language	application/json	
Key	Value	Description

► Temporary Headers (8) ⓘ

Body Cookies (2) Headers (11) Test Results

Status: 401 Unauthorized

Pretty Raw Preview Visualize JSON ↕

```
1 {
2   "timestamp": "2020-03-23T05:38:08.892+0000",
3   "status": 401,
4   "error": "Unauthorized",
5   "message": "Unauthorized",
6   "path": "/users"
7 }
```

GET http://127.0.0.1:8088/users

Params **Authorization** Headers (12) Body Pre-request Script Tests Settings

TYPE

Basic Auth

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Preview Request

Heads up! These parameters hold sensitive data. To keep this data secure while working in recommend using variables. [Learn more about variables](#)

Username user

Password .....

Show Password

Body Cookies (2) Headers (10) Test Results

Status: 200 OK

Pretty Raw Preview Visualize JSON ↕

```
1 [
2   {
3     "id": 1,
4     "name": "Adam",
5     "birthDate": "2020-03-21T13:22:58.053+0000"
6   },
7   {
8     "id": 2,
9     "name": "Eve",
10    "birthDate": "2020-03-21T13:22:58.053+0000"
11  },
12  {
13    "id": 3,
14    "name": "Jack",
15    "birthDate": "2020-03-21T13:22:58.053+0000"
16  }
17 ]
```



# step26- Implementing Basic Authentication with Spring Security

## ■ application.yml

### spring:

messages:

  basename: messages

devtools:

  livereload:

    enabled: true

### security:

  user:

    name: username

    password: passw0rd

GET http://127.0.0.1:8088/users

Params Authorization Headers (12) Body Pre-request Script Tests Settings

TYPE: Basic Auth

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Preview Request

Username: username

Password: .....

☐ Show Password

Status: 200 OK

Body Cookies (2) Headers (10) Test Results

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 1,
4     "name": "Adam",
5     "birthDate": "2020-03-21T13:24:59.250+0000"
6   },
7   {
8     "id": 2,
9     "name": "Eve",
10    "birthDate": "2020-03-21T13:24:59.250+0000"
11  },
12  {
13    "id": 3,
14    "name": "Jack",
15    "birthDate": "2020-03-21T13:24:59.250+0000"
16  }
17 ]
```

# step26- Implementing Basic Authentication with Spring Security

@Configuration

```
public class SecurityConfig extends WebSecurityConfigurerAdapter {  
    @Override  
    protected void configure(HttpSecurity http) throws Exception {  
        http.csrf().disable().authorizeRequests().anyRequest().authenticated().and().httpBasic();  
    }  
}
```

@Autowired

```
public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {  
    auth.inMemoryAuthentication()  
        .withUser("kenneth")  
        .password("{noop}test1234")  
        .roles("USER");  
}  
}
```

SecurityConfig.java

# step26- Implementing Basic Authentication with Spring Security

GET http://127.0.0.1:8088/users

Params Authorization Headers (12) Body Pre-request Script Tests Settings

TYPE: Basic Auth

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Preview Request

Username: kenneth

Password: .....

☐ Show Password

Body Cookies (2) Headers (10) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 1,
4     "name": "Adam",
5     "birthDate": "2020-03-21T13:34:16.963+0000"
6   },
7   {
8     "id": 2,
9     "name": "Eve",
10    "birthDate": "2020-03-21T13:34:16.964+0000"
11  },
12  {
13    "id": 3,
14    "name": "Jack",
15    "birthDate": "2020-03-21T13:34:16.964+0000"
16  }
17 ]
```

GET http://127.0.0.1:8088/users

Params Authorization Headers (12) Body Pre-request Script Tests Settings

Heads up! These parameters hold sensitive data. To keep this data secure while working in Postman, we recommend using variables. [Learn more about variables](#)

Username: kenneth

Password: .....

☐ Show Password

Headers (2) Headers (11) Test Results Status: 401 Unauthorized

Raw Preview Visualize Text

인증 실패

인증 성공