NHN Academy

8. Spring Boot Actuator

Spring Boot Actuator

- 상용화 준비(Production-Ready)기능을 위한 Spring Boot 모듈
- 실행 중인 애플리케이션을 관리하고 정보를 수집하고 상태를 점검하는 진입점 제공
- HTTP 또는 JMX 를 사용할 수 있음.

Actuator 설치

· maven

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

· gradle

```
dependencies {
    compile("org.springframework.boot:spring-boot-starter-actuator")
}
```

Endpoints(1)

- Actuator 엔드포인트로 spring boot 애플리케이션의 모니터링 및 상호작용 가능
- 스프링 부트는 다양한 빌트인 엔드포인트를 제공

ID	설명
auditevent	응용시스템의 모든 감사 이벤트 목록을 제공, AuditEventRepository 빈 필요
beans	애플리케이션의 모든 빈의 목록을 제공
caches	가능한 캐시를 노출
conditions	설정 및 자동설정 클래스를 평가한 조건의 목록과 조건의 부합 여부에 대한 이유를 제공
configprops	값이 설정된 모든 @ConfigurationProperties 의 목록을 제공

Endpoints(2)

ID	설명
env	스프링의 ConfigurableEnvironment 의 속성을 제공
health	애플리케이션의 health 정보를 제공
httptrace	http 의 요청,응답 내용을 표시, (기본 설정으로 100개 까지만 제공, HttpTraceRepository 빈 필요)
info	애플리케이션의 정보 제공
shutdown	애플리케이션의 셧다운 명령
startup	startup 단계 데이터를 제공 (SpringApplication 을 BufferingApplicationStartup으로 설정 필요)
threaddump	쓰레드 덤프를 실행

Spring Boot Actuator 설정

Endpoint 활성화

- 기본설정으로 shutdown 을 제외한 모든 end point 는 활성화
- management.endpoint.<id>.enabled 속성으로 활성화/비활성화 설정

```
management.endpoint.shutdown.enabled=true
```

```
management:
 endpoint:
   shutdown:
     enabled: true
```

Spring Boot Actuator 설정

Endpoint 활성화

· opt-in 방식으로 설정하고자 한다면...

```
management.endpoints.enabled-by-default=false ## 모두 비활성화
management.endpoint.info.enabled=true         ## info만 활성화
```

```
management:
  endpoints:
    enabled-by-default: false       ## 모두 비활성화
  endpoint:
    info:
      enabled: true                 ## info만 활성화
```

Spring Boot Actuator 설정

Endpoint 노출방식(JMX, Web) 설정

· acutator 는 민감한 정보를 노출하기 때문에 노출방식을 신중하게 설정해야 함

· Web은 health Endpoint 만 제공함

ID	JMX	Web
auditevents	Yes	No
beans	Yes	No
caches	Yes	No
conditions	Yes	No
configprops	Yes	No
env	Yes	No
flyway	Yes	No
health	**Yes**	**Yes**
heapdump	N/A	No
httptrace	Yes	No
info	Yes	No
integrationgraph	Yes	No
jolokia	N/A	No
logfile	N/A	No
loggers	Yes	No
liquibase	Yes	No
metrics	Yes	No
mappings	Yes	No
prometheus	**N/A**	**No**
quartz	Yes	No
scheduledtasks	Yes	No
sessions	Yes	No
shutdown	Yes	No
startup	Yes	No
threaddump	Yes	No

Spring Boot Actuator 설정

Endpoint 노출방식(JMX, Web) 설정

· JMX는 모든 Endpoint 를 노출하고, Web은 health 만 노출하는 것이 기본 설정
· include, exclude 프로퍼티로 노출방식을 활성화 할 수 있음

Property	기본설정
management.endpoints.jmx.exposure.exclude	
management.endpoints.jmx.exposure.include	*
management.endpoints.web.exposure.exclude	
management.endpoints.web.exposure.include	health

Spring Boot Actuator 설정

Endpoint 노출방식(JMX, Web) 설정

· exclude 설정은 include 설정보다 우선한다.

· 예) health, info 만 JMX에서 노출

```
management.endpoints.jmx.exposure.include=health,info
```

· 예) env, bean 를 제외한 모든 Endpoint를 web에서 노출

```
management.endpoints.web.exposure.include=*
management.endpoints.web.exposure.exclude=env,bean
```

Spring Boot Actuator 보안

Spring Security 설정

· spring-security가 클래스패스에 존재하면 health를 제외한 모든 Endpoint는 기본 자동설정기능에 의해 보호된다.

· WebSecurityConfigurerAdapter 또는 SecurityFilterChain 빈을 설정하여 기본 자동설정을 제거하고 보안설정을 정의할 수 있다.

```
@Configuration(proxyBeanMethods = false)
public class MySecurityConfiguration {

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http.requestMatcher(EndpointRequest.toAnyEndpoint())
                .authorizeRequests((requests) -> requests.anyRequest().hasRole("ENDPOINT_ADMIN"));
        http.httpBasic();
        return http.build();
    }

}
```

Spring Boot Actuator 보안

Spring Security 설정

· spring-security를 사용하지만 방화벽 내에서 동작하는 Actuator라 보안이 필요 없는 경우는 다음 설정으로 전체 개방한다.

```
management.endpoints.web.exposure.include=*
```

```
@Configuration(proxyBeanMethods = false)
public class MySecurityConfiguration {

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http.requestMatcher(EndpointRequest.toAnyEndpoint())
                .authorizeRequests((requests) -> requests.anyRequest().permitAll());
        return http.build();
    }

}
```

EndPoint 사용자 정의

- **@Endpoint: Endpoint로 노출할 빈에 설정**
  - @WebEndpoint: HTTP Endpoint로만 노출할 때
  - @JmxEndpoint: JMX Endpoint로만 노출할 때

- **@ReadOperation, @WriteOperation, @DeleteOperation**
  - HTTP 의 GET, POST, DELETE 메소드

- **@EndpointWebExtension, @EndpointJmxExtension**
  - 이미 존재하는 Endpoint에 기술 전용 오퍼레이션을 추가할 때 사용

EndPoint 사용자 정의

· Counter를 관리하는 Endpoint 예

```
@Component
@Endpoint(id = "counter")
public class CounterEndpoint {
    private final AtomicLong counter = new AtomicLong();

    // curl -XGET http://localhost:8080/actuator/counter
    @ReadOperation
    public Long read() {
        return counter.get();
    }

    // curl -X POST -H"Content-Type: application/json" -d'{"delta":100}'  http://localhost:8080/actuator/counter
    @WriteOperation
    public Long increment(@Nullable Long delta) {
        if (delta == null) {
            return counter.incrementAndGet();
        }
        return counter.addAndGet(delta);
    }

    // curl -X DELETE http://localhost:8080/actuator/counter
    @DeleteOperation
    public Long reset() {
```

EndPoint 사용자 정의

· 이미 존재하는 Endpoint에 특정 기술에서 동작하는 Endpoint를 추가하고 싶으면 @EndpointWebExtension, @EndpointJmxExtension을 사용한다.

```
@EndpointWebExtension(endpoint = CounterEndpoint.class)
@Component
public class CounterWebEndPoint {
  private final CounterEndpoint target;

  public CounterWebEndPoint(CounterEndpoint target) {
    this.target = target;
  }

  @WriteOperation
  public WebEndpointResponse<Long> increment(@Nullable Long delta) {
    return new WebEndpointResponse<>(target.increment(delta));
  }
```

Spring Boot Actuator Health Endpoint

health Endpoint

- 애플리케이션의 정상동작 정보를 제공한다.
- ApplicationContext 내의 HealthContributor 타입의 빈을 모두 활용해서 정보를 제공한다.
- HealthContributor 는 HealthIndicator 나 CompositeHealthContributor의 형태로 사용
  - HealthIndicator : 실제 Health 정보 제공
  - CompositeHealthIndicator : HealthContributor 들의 조합정보를 제공

- management.endpoint.health.show-details=ALWAYS 를 설정하면 각각의 HealthContributor 상세 정보를 볼 수 있다.

http://localhost:8080/actuator/health

Spring Boot Actuator Health Endpoint

Spring Boot의 기본 HealthIndicators

· Auto Configuration에 의해서 동작여부 결정
· Spring Boot Reference Documentation

이름	기본설정
CassandraDriverHealthIndicator	카산드라 데이터베이스 상태 체크
CouchbaseHealthIndicator	카우치베이스 클러스터 상태 체크
DiskSpaceHealthIndicator	디스크 공간 체크
DataSourceHealthIndicator	DataSource에서 커넥션을 얻을 수 있는 지 체크
RedisHealthIndicator	레디스 서버의 상태 체크

Spring Boot Actuator Info Endpoint

info Endpoint

- 애플리케이션의 정보를 제공한다.
- ApplicationContext 내의 InfoContributor 타입의 빈을 모두 활용해서 정보를 제공한다.

http://localhost:8080/actuator/info

Spring Boot Actuator Info Endpoint

EnvironmentInfoContributor

- info.* 형식의 모든 환경변수 정보 제공

```
info.edu.springboot.version=2.5.5
info.edu.springboot.instructor=manty
```

```
{
  "edu" : {
    "springboot" : {
      "version" : "2.5.5",
      "instructor" : "manty"
    }
  },
  "app" : {
    "java" : {
      "source" : "11"
    }
  }
}
```

Spring Boot Actuator Info Endpoint

GitInfoContributor

- 클래스 패스상의 git.properties 정보 제공, 실행 중인 서비스의 git 정보 확인용
- maven, gradle 설정 필요

```
<build>
  <plugins>
    ...
    <plugin>
        <groupId>pl.project13.maven</groupId>
        <artifactId>git-commit-id-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

```
plugins {
      id "com.gorylenko.gradle-git-properties" version "1.5.1"
}
```

Spring Boot Actuator Info Endpoint

GitInfoContributor

· info endpoint 응답 예

```
{
  "git": {
    "branch": "master",
    "commit": {
      "id": "077a397",
      "time": "2022-02-01T05:12:05Z"
    }
  }
}
```

Spring Boot Actuator Info Endpoint

BuildInfoContributor

- 클래스 패스의 META-INF/build-info.properties 파일 정보 제공
- maven, gradle 설정 필요

```
<plugin>
  <groupid>org.springframework.boot</groupid>
  <artifactid>spring-boot-maven-plugin</artifactid>
  <executions>
    <execution>
      <goals>
        <goal>build-info</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

```
springBoot {
     buildInfo()
}
```

Spring Boot Actuator Info Endpoint

BuildInfoContributor

· info endpoint 응답 예

```
{
  "build": {
    "artifact": "student",
    "name": "student",
    "time": "2022-02-01T07:07:41.030Z",
    "version": "0.0.1-SNAPSHOT",
    "group": "com.nhn.edu.springboot"
  }
}
```

Spring Boot Actuator Info Endpoint

InfoContributor 사용자 정의

· InfoContributor 인터페이스의 구현체 빈을 생성하여 원하는 정보를 Info Endpoint 에 추가

```
@Component
public class ExampleInfoContributor
                                implements InfoContributor {

    @Override
    public void contribute(Info.Builder builder) {
        builder.withDetail("example",
            Map.of("key", "value"));
    }
}
```

```
{
 "example": {
    "key": "value"
  }
}
```

Spring Boot Actuator

Endpoint 경로변경

· Spring Boot Actuator 의 기본 경로는 /actuator 이다.
· management.endpoints.web.base-path 속성을 변경하여 경로를 변경할 수 있다.

```
management.endpoints.web.base-path=/actuator2          # 2.x
management.context-path=/actuator2                     # 1.x : Set /actuator
```

Endpoint Port 변경

· Spring Boot Actuator 의 기본 포트는 서비스 포트와 동일하다.
· management.server.port 속성을 변경하여 포트를 변경할 수 있다.

```
management.server.port=8888
```

Spring Boot Actuator

Prometheus

· prometheus(https://prometheus.io) 라는 시계열 데이터베이스에 데이터를 제공
· micrometer-registry-prometheus 라이브러리 의존성을 추가해야 함
. http://localhost:8080/actuator/prometheus

```
<dependency>
    <groupId>io.micrometer</groupId>
    <artifactId>micrometer-registry-prometheus</artifactId>
</dependency>
```

Prometheus

· Grafana 로 prometeus 정보를 시각화
· 성능 추이를 추적하여 인프라 scale up 또는 scale out 지표로 사용