**Spring Boot – Actuator**

In this **Spring boot actuator tutorial**, learn about in-built HTTP endpoints available for any boot application for different **monitoring and management purposes**. Before spring framework, if we had to introduce this type of monitoring functionality in our applications then we had to manually develop all those components and that too were very specific to our need. But with spring boot we have Actuator module which makes it very easy.

We just need to configure few things and we are done – all the management and monitoring related information is easily available. Let’s learn to configure **Spring boot actuator endpoints**.

**1. Spring Boot Actuator Module**

Spring boot’s module Actuator allows you to monitor and manage application usages in production environment, without coding and configuration for any of them. These monitoring and management information is exposed via [REST](https://restfulapi.net/) like endpoint URLs.

**1.1. Actuator Maven Dependency**

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

**1.2. Important Actuator Endpoints**

Some of important and widely used actuator endpoints are given below:

|  |  |
| --- | --- |
| **ENDPOINT** | **USAGE** |
| /env | Returns list of properties in current environment |
| /health | Returns application health information. |
| /auditevents | Returns all auto-configuration candidates and the reason why they ‘were’ or ‘were not’ applied. |
| /beans | Returns a complete list of all the Spring beans in your application. |
| /trace | Returns trace logs (by default the last 100 HTTP requests). |
| /dump | It performs a thread dump. |
| /metrics | It shows several useful metrics information like JVM memory used, system CPU usage, open files, and much more. |

**1.3. Security related properties**

|  |
| --- |
| management.security.enabled = true  management.security.roles = ADMIN    security.basic.enabled = true  security.user.name = admin  security.user.password = admin |

Please note that by default, to access the actuator-restricted endpoints, you have to have the ACTUATOR role. You need to override this configuration by management.security.roles property.

**1.4. Actuator Security with WebSecurityConfigurerAdapter**

|  |
| --- |
| import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;  import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;    @Configuration  public class SpringSecurityConfig extends WebSecurityConfigurerAdapter {        @Autowired      public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {          auth.inMemoryAuthentication().withUser("admin").password("admin").roles("ADMIN");        }  } |

**2. Spring Boot Actuator Endpoint Example**

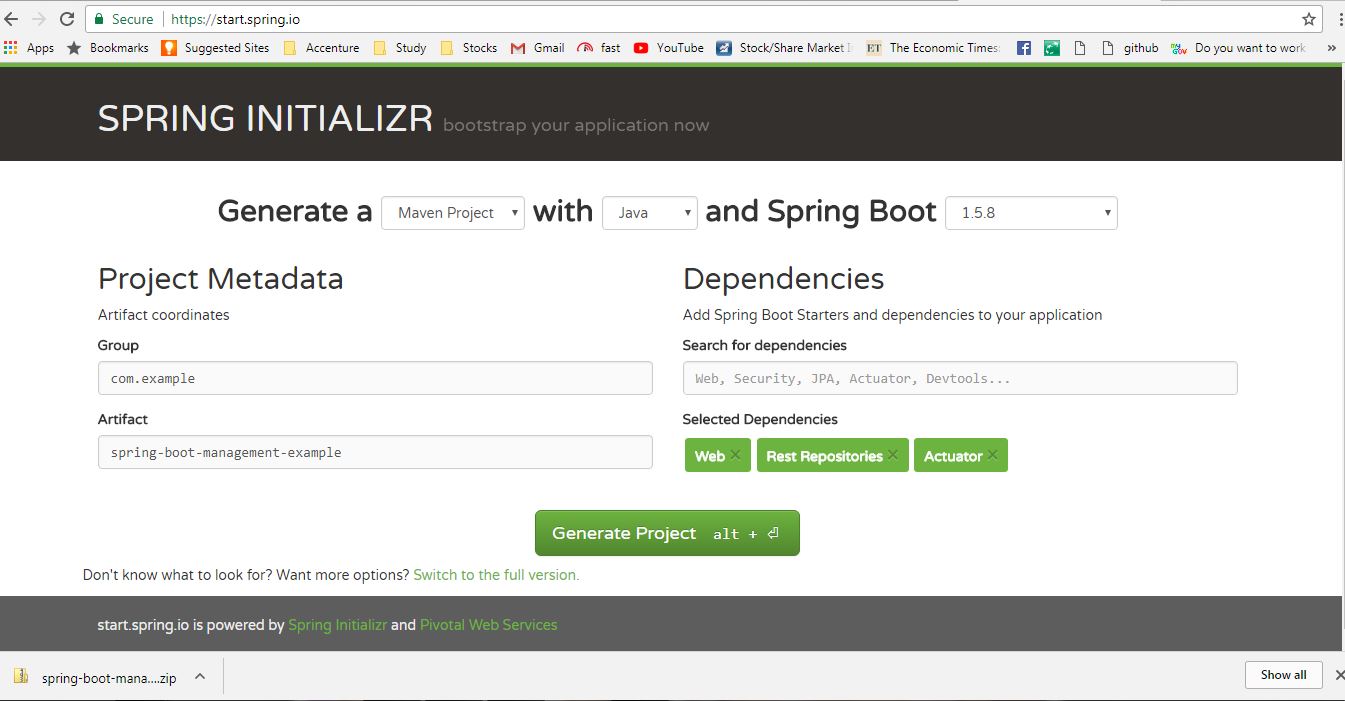
In this example, we will create a simple string boot application and the access the actuator endpoints to know more about them.

**2.1. Development environment**

* JDK 1.8, Eclipse, Maven – Development environment
* Spring-boot – Underlying application framework
* Spring-boot Actuator – Management endpoints

**2.2. Create Maven Project**

Start with creating one spring boot project from [Spring Initializer](https://start.spring.io/) site with Web, Rest Repositories and Actuator dependencies. Download project in zipped format. Unzip and then import project in eclipse as maven project.



Spring boot actuator project

**2.3. Add simple Rest endpoint**

Now add one simple Rest endpoint /example to the application.

|  |
| --- |
| package com.example.springbootmanagementexample;    import java.util.Date;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.RestController;    @RestController  public class SimpleRestController {      @GetMapping("/example")      public String example() {          return "Hello User !! " + new Date();      }  } |

**3. Spring Boot Actuator Endpoints Demo**

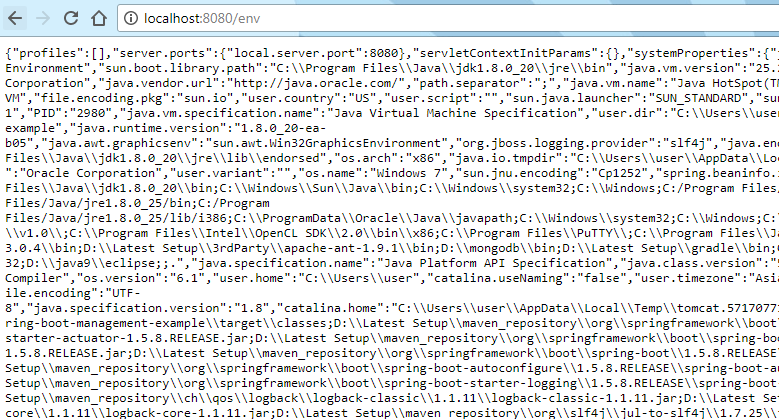
I have added management.security.enabled=false entry to the **application.properties** file to disable actuator security. Here I am more interested in actuator endpoints responses.

Do maven build using mvn clean install and start the application using java -jar target\spring-boot-management-example-0.0.1-SNAPSHOT.jar command. This will bring up one tomcat server in default port 8080 and application will be deployed in it.

Access /example API in browser to generate few monitoring information on server.

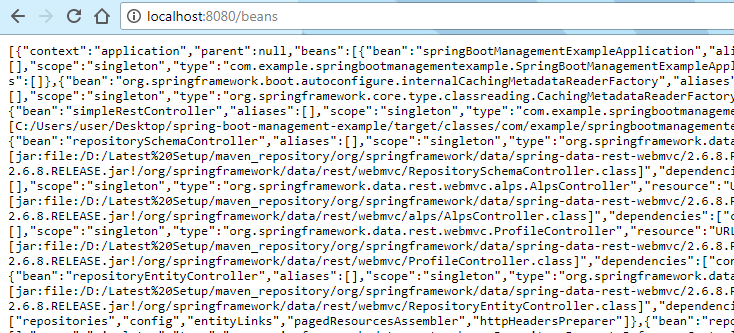
* **http://localhost:8080/env**

This will give all the environmental configuration about the server.

Endpoint env Output

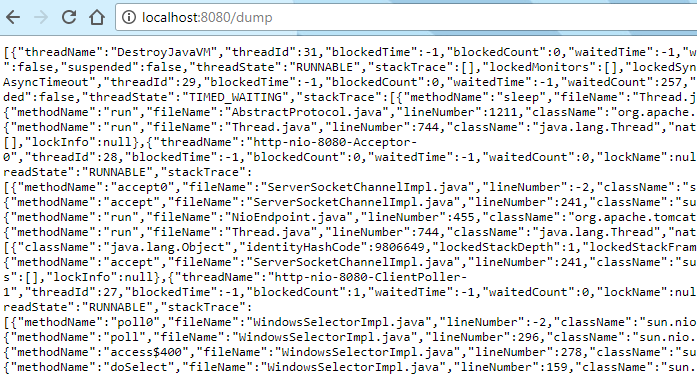
* **http://localhost:8080/beans**

This will give all the spring beans loaded in the context.

Endpoint beans Output

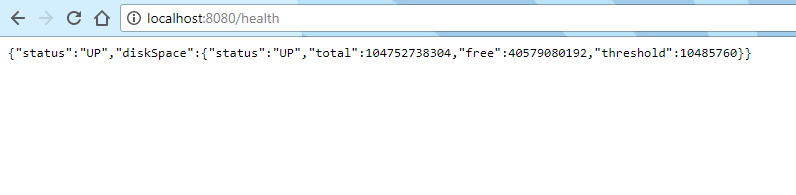
* **http://localhost:8080/dump**

This will give the current server thread dump.

Endpoint dump Output

* **http://localhost:8080/health**

This will give generic health of the application and server.

Endpoint health Output

* **http://localhost:8080/metrics**

The /metrics endpoint lists all the metrics that are available for you to track.

|  |
| --- |
| {      "mem": 316656,      "mem.free": 169495,      "processors": 4,      "instance.uptime": 1449726,      "uptime": 1463662,      "systemload.average": -1.0,      "heap.committed": 263168,      "heap.init": 131072,      "heap.used": 93672,      "heap": 1846272,      "nonheap.committed": 54400,      ........  } |

Those endpoints will give standard information in the browser. These are the basic important endpoints we generally refer, but spring boot provides many more endpoints as mentioned in