SPRING ACTUATOR

Spring Actuator : (to monitoring the application)

Actuator: (gives no of endpoints to provide metrics like memory, caching, session, disk space, jvm details etc)
Actuator is mainly used to expose operational information about the running application — health, metrics, info, dump, env, etc. It uses HTTP endpoints or JMX beans to enable us to interact with it. Once this dependency is on the classpath, several endpoints are available for us to interact with.
It brings production-ready features to our application.

To Enable Spring Boot Actuator add this dependency in pom.xml
```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

In Spring Boot 5, enable the actuators in the application.properties file.
```
management.endpoints.web.exposure.include=*
```

In the browser or postman, the URL must be
Example : localhost:8080/actuator/health

Custom Endpoints:
The custom endpoints also can be created, by implementing the interfaces and overriding the methods in it as per the project requirement.

We can also easily extend the behavior of a predefined or existing endpoint using the *@EndpointExtension* annotations

Some endpoints have been added, some removed and some have been restructured:

- */auditevents* lists security audit-related events such as user login/logout. Also, we can filter by principal or type among other fields.
- */beans* returns all available beans in our *BeanFactory*. Unlike */auditevents*, it doesn't support filtering.
- */conditions*, formerly known as /*autoconfig*, builds a report of conditions around autoconfiguration.
- */configprops* allows us to fetch all *@ConfigurationProperties* beans.
- /***env*** returns the current environment properties. Additionally, we can retrieve single properties.(server.port, classpath,application.properties file details etc)
- */flyway* provides details about our Flyway database migrations.

- */**health*** summarizes the health status of our application. (status,disk space, total, free etc)
- */heapdump* builds and returns a heap dump from the JVM used by our application.
- ***/info*** returns general information. It might be custom data, build information or details about the latest commit.
- */liquibase* behaves like */flyway* but for Liquibase.
- */logfile* returns ordinary application logs.
- */loggers* enables us to query and modify the logging level of our application.
- */**metrics*** details metrics of our application. This might include generic metrics as well as custom ones. (jvm.memory, jvm.gc, jvm.buffer, jdbc.connections,diskfree etc)
- ***/prometheus*** returns metrics like the previous one, but formatted to work with a Prometheus server.
- */scheduledtasks* provides details about every scheduled task within our application.
- */sessions* lists HTTP sessions given we are using Spring Session.
- */shutdown* performs a graceful shutdown of the application.
- */threaddump* dumps the thread information of the underlying JVM.

TPS: Transactions per second or
OPS : Operations per second
Number of requests a service can handle in a second

Connection timeout: The total time, a service waits for connecting with the other service, If not connected in this given time, the service times out ,
Ex: Service A tries to connect with Service B with connection timeout 500ms. Service A waits till then and stops trying
Read timeout: Service A waits for the response from Service B and times out after the readtimeout

— End points can be exposed in 2 ways.   https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#appendix.application-properties.actuator

- using REST endpoints

  management.endpoints.beans.enabled = true

  http://localhost:9090/actuator/

- using JMX

— heapdump/ thread dump

  heaphero.io

— Profiling using Visual VM

- https://visualvm.github.io/download.html
- To check the performance pf the objects and methods present in the classes

— How to set security to the endpoints?