

CS544

# **LESSON 12**

## **SCHEDULING, EVENTS, CONFIGURATION, LOGGING, ACTUATORS**

# **JOB SCHEDULING**

# Job scheduling

---

- JDK Timer: `java.util.Timer`
  - Basic scheduling support
    - Execute at a given time
    - Execute at some fixed frequency
- Quartz scheduling
  - Open source framework
  - Powerful job scheduling engine
  - Cron-based scheduling

# Scheduling basics

---

- Job
  - Unit of work that needs to execute at a specific time or interval
- Trigger
  - The condition (specific time or interval) that causes a job to run
- Schedule
  - A collection of triggers

# JDK Timer example

```
public class HelloWorldTask extends TimerTask{

    public void run() {
        Date date = Calendar.getInstance().getTime();
        DateFormat timeFormatter = DateFormat.getTimeInstance(DateFormat.DEFAULT);
        String currenttime = timeFormatter.format(date);

        System.out.println("This task runs at "+currenttime);
    }
}
```

```
import java.util.Timer;

public class Application {

    public static void main(String[] args) {
        Timer timer = new Timer();
        timer.scheduleAtFixedRate(new HelloWorldTask(), 5000, 5000);
    }
}
```

The job to run

Start the job  
after 5 seconds

Run the job every  
5 seconds

```
This task runs at 10:45:52
This task runs at 10:45:57
This task runs at 10:46:02
This task runs at 10:46:07
```

# Quartz cron scheduling example

```
public class HelloWorldJob implements Job{

    public void execute(JobExecutionContext arg0) throws JobExecutionException {
        Date date = Calendar.getInstance().getTime();
        DateFormat timeFormatter = DateFormat.getInstance(DateFormat.DEFAULT);
        String currenttime = timeFormatter.format(date);

        System.out.println("This task runs at "+currenttime);
    }
}
```

```
public class CronApplication {
    public static void main(String[] args) throws SchedulerException, ParseException {
        Scheduler scheduler =new StdSchedulerFactory().getScheduler();
        scheduler.start();

        JobDetail jobDetail=new
            JobDetail("HelloWorldJob",scheduler.DEFAULT_GROUP,HelloWorldJob.class);

        String cronExpression="0/5 * * * * ?";
        Trigger crontrigger=new
            CronTrigger("crontrigger",scheduler.DEFAULT_GROUP,cronExpression);
        scheduler.scheduleJob(jobDetail, crontrigger);
    }
}
```

The trigger is expressed with a cron expression

```
This task runs at 12:04:25
This task runs at 12:04:30
This task runs at 12:04:35
```

# Quartz cron expressions

- String with 6 or 7 space separated sub-expressions with the following meaning:  
seconds minutes hours dayOfMonth month dayOfWeek year(optional)

- Examples

- "0 0 12 ? \* WED"

- every Wednesday at 12:00 pm

- "0 0/5 \* \* \* ?"

- every 5 minutes

- "10 0/5 \* \* \* ?"

- every 5 minutes, at 10 seconds after the minute (i.e. 10:00:10 am, 10:05:10 am, etc.).

- "0 30 10-13 ? \* WED,FRI"

- 10:30, 11:30, 12:30, and 13:30, on every Wednesday and Friday.

- "0 0/30 8-9 5,20 \* ?"

- every half hour between the hours of 8 am and 10 am on the 5th and 20th of every month.

\*=every

?=no specific value (only for dayOfMonth and dayOfWeek)

# Spring annotation based scheduling

@SpringBootApplication

@EnableScheduling

Enable scheduling

```
public class SpringBootSchedulingApplication {
```

```
    public static void main(String[] args) {  
        SpringApplication.run(SpringBootSchedulingApplication.class, args);  
    }  
}
```

@Component

```
public class WelcomeTask {
```

Run every 5 seconds

```
    @Scheduled(fixedRate = 5000)
```

```
    public void welcome() {
```

```
        Date date = Calendar.getInstance().getTime();
```

```
        DateFormat timeFormatter = DateFormat.getInstance(DateFormat.DEFAULT);
```

```
        String currenttime = timeFormatter.format(date);
```

```
        System.out.println("This task runs at " + currenttime);
```

```
    }  
}
```

```
This task runs at 12:07:50
```

```
This task runs at 12:07:55
```

```
This task runs at 12:08:00
```



# @Scheduled

```
@Scheduled(fixedDelay = 5000)  
public void welcome() {
```

Run every 5 seconds measured from the completion time of the welcome() method

```
@Scheduled(fixedRate = 5000)  
public void welcome() {
```

Run every 5 seconds measured from the start time of the welcome() method

```
@Scheduled(initialDelay=1000, fixedRate=5000)  
public void welcome() {
```

Run every 5 seconds but wait 1 second before the first execution

```
@Scheduled(cron="*/5 * * * * MON-FRI")  
public void welcome() {
```

Cron expression: Run every 5 seconds on Monday till Friday.

# Main point

---

- Spring makes it simple to schedule methods of spring beans.

*Science of Consciousness:* There is order in creation. In creation everything happens according the laws of Nature.

# **EVENTS**

# **ASYNCHRONOUS METHODS**

# Events

---

```
public class AddCustomerEvent {  
    private String message;  
  
    public AddCustomerEvent(String message) {  
        this.message = message;  
    }  
  
    public String getMessage() {  
        return message;  
    }  
}
```

A simple event class

# Event publisher and listener

```
@Service
public class CustomerServiceImpl implements CustomerService {
    @Autowired
    private ApplicationEventPublisher publisher;

    public void addCustomer() {
        publisher.publishEvent(new AddCustomerEvent("New customer is added"));
    }
}
```

Inject a publisher

```
@Service
public class Listener {

    @EventListener
    public void onEvent(AddCustomerEvent event) {
        System.out.println("received event :" + event.getMessage());
    }
}
```

Listen to AddCustomer events

# Asynchronous events



```
@Service
@EnableAsync
public class Listener {

    @Async
    @EventListener
    public void onEvent(AddCustomerEvent event) {
        System.out.println("received event :" + event.getMessage());
    }
}
```

# Asynchronous methods

@EnableAsync

public class MyServiceImpl implements MyService {

@Async

public void welcome() {

Date date = Calendar.getInstance().getTime();

DateFormat timeFormatter = DateFormat.getTimeInstance(DateFormat.DEFAULT);

String currenttime = timeFormatter.format(date);

System.out.println("This task runs at " + currenttime);

}

Enable asynchronous methods

The method call returns immediately

# Main point

---

- Spring events is a powerful technique to implement publish subscribe within the application.

*Science of Consciousness:* When one subscribes daily to the intelligence of nature one automatically receives support of Nature.



# **SPRING BOOT CONFIGURATION**

# @Value

```
@Service
public class EmailServiceImpl implements EmailService{
    @Value("${smtpserver}")
    String outgoingMailServer;

    ...
}
```

Works for small and simple data,  
not for complex data

```
smtpserver=smtp.mydomain.com
```

# @ConfigurationProperties

---

application.properties

**Mapping single properties**

```
myapp.mail.to=frank@hotmail.com  
myapp.mail.host=mail.example.com  
myapp.mail.port=250
```

**#Mapping list or array**

```
myapp.mail.cc=mike@gmail.com,david@gmail.com  
myapp.mail.bcc=john@hotmail.com,admin@acme.com
```

**#Mapping nested POJO class**

```
myapp.mail.credential.user-name=john1234  
myapp.mail.credential.password=xyz@1234
```

application.yml

**myapp:**

**mail:**

```
to: frank@hotmail.com  
host: mail.example.com  
port: 250
```

**cc:**

- mike@gmail.com
- david@gmail.com

**bcc:**

- john@hotmail.com
- admin@acme.com

**credential:**

```
user-name: john1234  
password: xyz@1234
```

# @ConfigurationProperties

```
@ConfigurationProperties(prefix="myapp.mail")
public class MailProperties {

    private String to;
    private String host;
    private int port;
    private String[] cc;
    private List<String> bcc;

    private Credential credential = new Credential();

    //Setter and Getter methods

    public class Credential {
        private String userName;
        private String password;
        //Setter and Getter methods
    }
}
```

## Mapping single properties

```
myapp.mail.to=frank@hotmail.com
myapp.mail.host=mail.example.com
myapp.mail.port=250
```

## #Mapping list or array

```
myapp.mail.cc=mike@gmail.com,david@gmail.com
myapp.mail.bcc=john@hotmail.com,admin@acme.com
```

## #Mapping nested POJO class

```
myapp.mail.credential.user-name=john1234
myapp.mail.credential.password=xyz@1234
```

# Using MailProperties

```
@Component
public class MailService {

    @Autowired
    private MailProperties mailProperties;

    public void print() {
        System.out.println("Mail TO = " + mailProperties.getTo());
        System.out.println("HOST = " + mailProperties.getHost());
        System.out.println("PORT = " + mailProperties.getPort());
        System.out.println();

        //Print list or array
        System.out.println("Mail CC = " + String.join(", ", mailProperties.getCc()));
        System.out.println("Mail BCC = " + mailProperties.getBcc());
        System.out.println();

        //Print nested bean's properties
        System.out.println("User Name = " + mailProperties.getCredential().getUserName());
        System.out.println("Password = " + mailProperties.getCredential().getPassword());
    }
}
```

# Using MailProperties

```
@SpringBootApplication
@EnableConfigurationProperties(MailProperties.class)
public class SpringBootProjectApplication implements CommandLineRunner {
    @Autowired
    private MailService mailService;

    public static void main(String[] args) {
        SpringApplication.run(SpringBootProjectApplication.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        mailService.print();
    }
}
```

@EnableConfigurationProperties

# Advantages of @ConfigurationProperties

---

- Relaxed binding
  - If the property is **db.username**
  - Then these all will work:
    - db.user-name
    - db.user\_name
    - db.UserName
- Property validation
  - Properties can be validated using JSR-303 validation annotation

# Property Validation

```
@ConfigurationProperties(prefix="myapp.mail")
@Validated
public class MailProperties {
    @Email
    private String to;
    @NotBlank
    private String host;
    private int port;
    private String[] cc;
    private List<String> bcc;

    private Credential credential = new Credential();

    //Setter and Getter methods

    @Valid
    public class Credential {
        @NotBlank
        private String userName;
        @Size(max = 15, min = 6)
        private String password;

        //Setter and Getter methods
    }
}
```



# Property Validation

```
myapp:
  mail:
    to: frankhotmail.com
    host:
    port: 250
    cc:
      - mike@gmail.com
      - david@gmail.com
    bcc:
      - john@hotmail.com
      - admin@acme.com
  credential:
    user-name: john1234
    password: xyz@1234
```

```
<dependency>
  <groupId>org.hibernate.validator</groupId>
  <artifactId>hibernate-validator</artifactId>
  <version>6.0.5.Final</version>
</dependency>
```

Binding to target [org.springframework.boot.context.properties.bind.BindException](#):

```
Property: myapp.mail.to
Value: frankhotmail.com
Origin: class path resource [application.yml]:3:9
Reason: must be a well-formed email address
```

```
Property: myapp.mail.host
Value:
Origin: class path resource [application.yml]:4:10
Reason: must not be blank
```

# Configure a different webserver

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

Spring Boot starts Tomcat by default

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
  <exclusions>
    <exclusion>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-tomcat</artifactId>
    </exclusion>
  </exclusions>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-jetty</artifactId>
</dependency>
```

Start the embedded jetty webserver

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

Undertow webserver

# Spring Boot Properties

- <https://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html>

```
# -----  
# CORE PROPERTIES  
# -----  
debug=false # Enable debug Logs.  
trace=false # Enable trace Logs.  
  
# LOGGING  
logging.config= # Location of the logging configuration file. For instance, `classpath:logback.xml` for Logback.  
logging.exception-conversion-word=%wEx # Conversion word used when logging exceptions.  
logging.file= # Log file name (for instance, `myapp.log`). Names can be an exact location or relative to the current directory.  
logging.file.max-history=0 # Maximum of archive log files to keep. Only supported with the default logback setup.  
logging.file.max-size=10MB # Maximum log file size. Only supported with the default logback setup.  
logging.level.*= # Log levels severity mapping. For instance, `logging.level.org.springframework=DEBUG`.  
logging.path= # Location of the log file. For instance, `/var/log`.  
logging.pattern.console= # Appender pattern for output to the console. Supported only with the default Logback setup.  
logging.pattern.dateformat=yyyy-MM-dd HH:mm:ss.SSS # Appender pattern for log date format. Supported only with the default Logback setup.  
logging.pattern.file= # Appender pattern for output to a file. Supported only with the default Logback setup.  
logging.pattern.level=%5p # Appender pattern for log level. Supported only with the default Logback setup.  
logging.register-shutdown-hook=false # Register a shutdown hook for the logging system when it is initialized.  
  
# AOP  
spring.aop.auto=true # Add @EnableAspectJAutoProxy.  
spring.aop.proxy-target-class=true # Whether subclass-based (CGLIB) proxies are to be created (true), as opposed to standard Java interface-based.  
  
# IDENTITY (ContextIdApplicationContextInitializer)  
spring.application.name= # Application name.  
  
# ADMIN (SpringApplicationAdminJmxAutoConfiguration)  
spring.application.admin.enabled=false # Whether to enable admin features for the application.  
spring.application.admin.jmx-name=org.springframework.boot:type=Admin,name=SpringApplication # JMX name of the application admin MBean.  
  
# AUTO-CONFIGURATION  
spring.autoconfigure.exclude= # Auto-configuration classes to exclude.
```

# Main point

---

- @ConfigurationProperties allows to package configuration properties together and in addition provides property validation.

*Science of Consciousness:* In cosmic consciousness one spontaneously handles according the laws of Nature.

# **SPRING BOOT LOGGING**

# Zero configuration logging

---

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

- When you add this dependency, Spring boot automatically uses Commons Logging for logging.

# Using a Logger

@Component

```
public class CustomerService {  
    Logger logger = LoggerFactory.getLogger(CustomerService.class);  
  
    public void addCustomer(){  
        logger.trace("A TRACE Message");  
    }  
  
    public void updateCustomer(){  
        logger.debug("A DEBUG Message");  
    }  
  
    public void removeCustomer(){  
        logger.info("An INFO Message");  
    }  
  
    public void findCustomerById(){  
        logger.warn("A WARN Message");  
    }  
  
    public void findCustomersByName(){  
        logger.error("An ERROR Message");  
    }  
}
```

# The application

**@SpringBootApplication**

```
public class CustomerApplication implements CommandLineRunner {
```

**@Autowired**

```
private CustomerService customerService;
```

```
public static void main(String[] args) {  
    SpringApplication.run(CustomerApplication.class, args);  
}
```

**@Override**

```
public void run(String... args) throws Exception {  
    customerService.addCustomer();  
    customerService.updateCustomer();  
    customerService.removeCustomer();  
    customerService.findCustomerById();  
    customerService.findCustomersByName();  
}  
}
```



```
2022-07-05 13:32:45.706 INFO 1948 --- [main] app.CustomerApplication : Starting CustomerApplication using Java
11.0.1 on DESKTOP-BVHRK6K with PID 1948 (C:\EnterpriseArchiterture\demo code\Lesson13Logging\target\classes started by vedam in
C:\EnterpriseArchiterture\demo code\Lesson13Logging)
2022-07-05 13:32:45.708 INFO 1948 --- [main] app.CustomerApplication : No active profile set, falling back to
default profiles: default
2022-07-05 13:32:46.216 INFO 1948 --- [main] app.CustomerApplication : Started CustomerApplication in 0.936
seconds (JVM running for 1.381)
2022-07-05 13:32:46.218 INFO 1948 --- [main] app.CustomerService : An INFO Message
2022-07-05 13:32:46.218 WARN 1948 --- [main] app.CustomerService : A WARN Message
2022-07-05 13:32:46.218 ERROR 1948 --- [main] app.CustomerService : An ERROR Message
```

33

# Logging level

---

- TRACE: gives detailed information about the code
- DEBUG: gives more specific diagnostic information that you need during debugging
- INFO (default level): gives high level information
- WARN: potential problems that might cause problems
- ERROR: serious issues like exceptions

**ERROR** ← **WARN** ← **INFO** ← **DEBUG** ← **TRACE**

# Logging level

Default logging level is INFO

```
#logging.level.root=DEBUG
```

2022-07-05 12:18:51.737	INFO	29548	---	[	main]	app.CustomerService	:	An INFO Message
2022-07-05 12:18:51.737	WARN	29548	---	[	main]	app.CustomerService	:	A WARN Message
2022-07-05 12:18:51.737	ERROR	29548	---	[	main]	app.CustomerService	:	An ERROR Message

```
logging.level.root=WARN
```

2022-07-05 12:19:52.173	WARN	2692	---	[	main]	app.CustomerService	:	A WARN Message
2022-07-05 12:19:52.175	ERROR	2692	---	[	main]	app.CustomerService	:	An ERROR Message

```
logging.level.root=ERROR
```

2022-07-05 12:20:57.133	ERROR	3560	---	[	main]	app.CustomerService	:	An ERROR Message
-------------------------	-------	------	-----	---	-------	---------------------	---	------------------

# Logging level



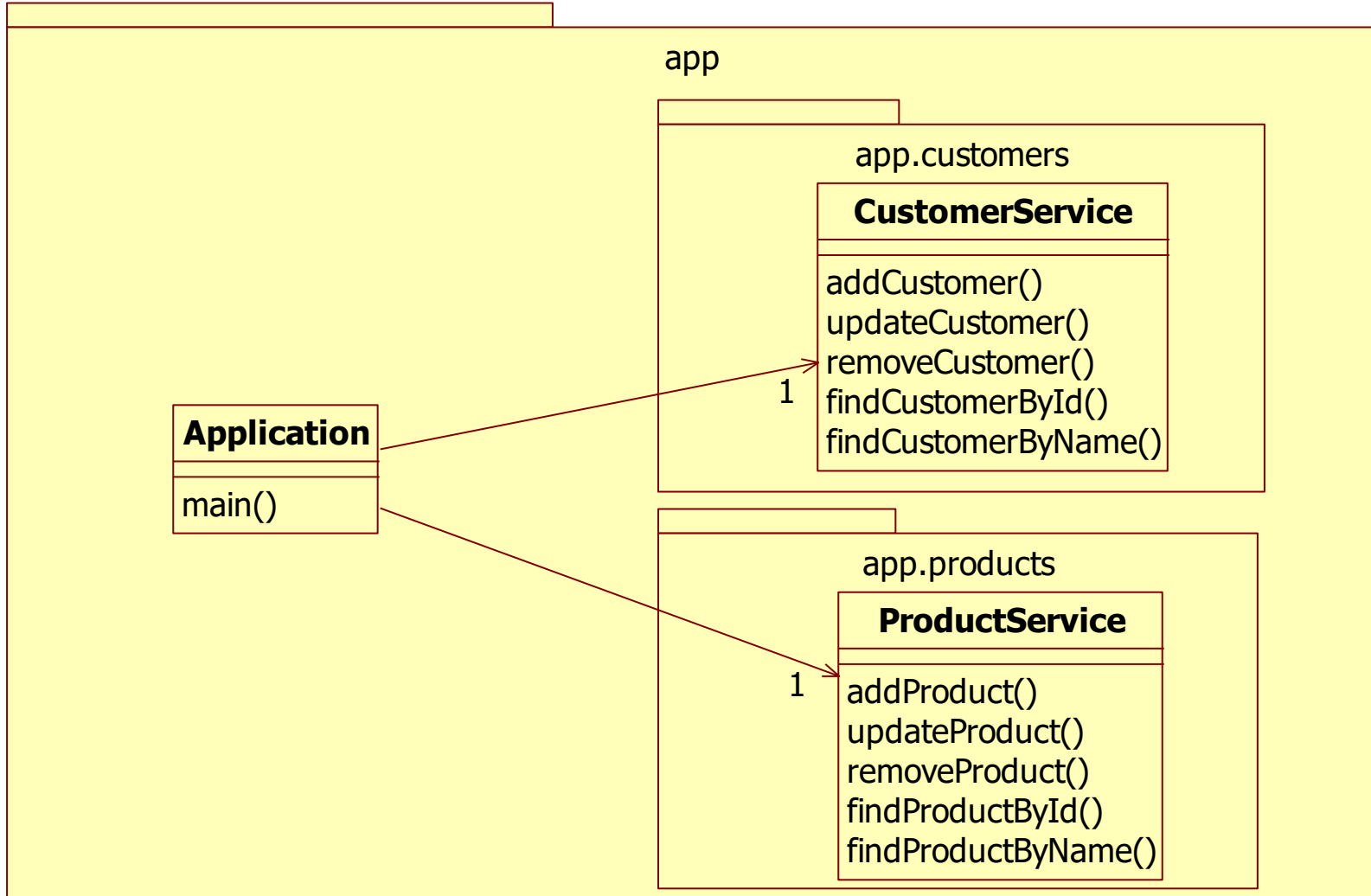
```
logging.level.root=DEBUG
```

```
2022-07-05 12:16:15.581 DEBUG 29812 --- [main] app.CustomerService : A DEBUG Message
2022-07-05 12:16:15.581 INFO 29812 --- [main] app.CustomerService : An INFO Message
2022-07-05 12:16:15.581 WARN 29812 --- [main] app.CustomerService : A WARN Message
2022-07-05 12:16:15.581 ERROR 29812 --- [main] app.CustomerService : An ERROR Message
```

```
logging.level.root=TRACE
```

```
2022-07-05 12:13:47.372 TRACE 8380 --- [main] app.CustomerService : A TRACE Message
2022-07-05 12:13:47.372 DEBUG 8380 --- [main] app.CustomerService : A DEBUG Message
2022-07-05 12:13:47.372 INFO 8380 --- [main] app.CustomerService : An INFO Message
2022-07-05 12:13:47.372 WARN 8380 --- [main] app.CustomerService : A WARN Message
2022-07-05 12:13:47.372 ERROR 8380 --- [main] app.CustomerService : An ERROR Message
```

# Logging level on packages



# The application

```
@SpringBootApplication
public class Application implements CommandLineRunner {
    @Autowired
    private CustomerService customerService;
    @Autowired
    private ProductService productService;

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        Logger logger = LoggerFactory.getLogger(Application.class);
        logger.info("An INFO Message");
        logger.error("An ERROR Message");

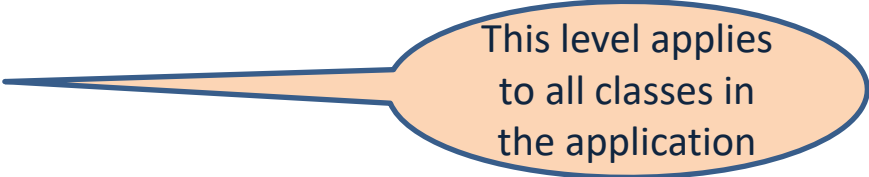
        customerService.addCustomer();
        customerService.updateCustomer();
        customerService.removeCustomer();
        customerService.findCustomerById();
        customerService.findCustomersByName();

        productService.addProduct();
        productService.updateProduct();
        productService.removeProduct();
        productService.findProductById();
        productService.findProductByName();
    }
}
```

# Logging level on packages

---

```
logging.level.root=INFO
```



This level applies  
to all classes in  
the application

2022-07-05 19:10:59.036	INFO	29528	---	[	main]	app.Application	:	An INFO Message
2022-07-05 19:10:59.036	ERROR	29528	---	[	main]	app.Application	:	An ERROR Message
2022-07-05 19:10:59.036	INFO	29528	---	[	main]	app.customers.CustomerService	:	An INFO Message
2022-07-05 19:10:59.036	WARN	29528	---	[	main]	app.customers.CustomerService	:	A WARN Message
2022-07-05 19:10:59.036	ERROR	29528	---	[	main]	app.customers.CustomerService	:	An ERROR Message
2022-07-05 19:10:59.036	INFO	29528	---	[	main]	app.products.ProductService	:	An INFO Message
2022-07-05 19:10:59.036	WARN	29528	---	[	main]	app.products.ProductService	:	A WARN Message
2022-07-05 19:10:59.036	ERROR	29528	---	[	main]	app.products.ProductService	:	An ERROR Message

# Logging level on packages

```
logging.level.root=INFO
logging.level.app.customers=ERROR
logging.level.app.products=TRACE
```

Logging level on  
individual  
packages

```
2022-07-05 19:21:21.497 INFO 30568 --- [main] app.Application : An INFO Message
2022-07-05 19:21:21.497 ERROR 30568 --- [main] app.Application : An ERROR Message
2022-07-05 19:21:21.497 ERROR 30568 --- [main] app.customers.CustomerService : An ERROR Message
2022-07-05 19:21:21.497 TRACE 30568 --- [main] app.products.ProductService : A TRACE Message
2022-07-05 19:21:21.497 DEBUG 30568 --- [main] app.products.ProductService : A DEBUG Message
2022-07-05 19:21:21.497 INFO 30568 --- [main] app.products.ProductService : An INFO Message
2022-07-05 19:21:21.497 WARN 30568 --- [main] app.products.ProductService : A WARN Message
2022-07-05 19:21:21.497 ERROR 30568 --- [main] app.products.ProductService : An ERROR Message
```



# Log format



2022-07-05	12:13:47.372	TRACE	8380	---	[	main]	app.CustomerService	:	A TRACE Message
2022-07-05	12:13:47.372	DEBUG	8380	---	[	main]	app.CustomerService	:	A DEBUG Message
2022-07-05	12:13:47.372	INFO	8380	---	[	main]	app.CustomerService	:	An INFO Message
2022-07-05	12:13:47.372	WARN	8380	---	[	main]	app.CustomerService	:	A WARN Message
2022-07-05	12:13:47.372	ERROR	8380	---	[	main]	app.CustomerService	:	An ERROR Message
1	2	3	4			5	6		7

1. Date and Time
2. Log level
3. Process ID
4. The separator ---
5. Thread name
6. Logger name source class
7. Log message

# Change Log format

```
logging.level.root=INFO
```

```
logging.pattern.console= %d{yyyy-MM-dd HH:mm:ss} - %logger - %msg%n
```

```
logging.pattern.file=%d{yyyy-MM-dd HH:mm:ss} - %logger - %msg%n
```

```
2022-07-05 12:37:13 - app.CustomerService - An INFO Message
```

```
2022-07-05 12:37:13 - app.CustomerService - A WARN Message
```

```
2022-07-05 12:37:13 - app.CustomerService - An ERROR Message
```

# Change Log format

```
logging.level.root=INFO
```

```
logging.pattern.console= %d{yyyy-MM-dd HH:mm:ss} - %logger - %msg%n
```

```
logging.pattern.file=%d{yyyy-MM-dd HH:mm:ss} - %logger - %msg%n
```

```
2022-07-05 12:37:13 - app.CustomerService - An INFO Message
```

```
2022-07-05 12:37:13 - app.CustomerService - A WARN Message
```

```
2022-07-05 12:37:13 - app.CustomerService - An ERROR Message
```

```
logging.pattern.console= %d{yyyy-MM-dd HH:mm:ss} [%thread] %level %logger - %msg%n
```

```
logging.pattern.file= %d{yyyy-MM-dd HH:mm:ss} [%thread] %level %logger - %msg%n
```

```
2022-07-05 12:46:26 [main] INFO app.CustomerService - An INFO Message
```

```
2022-07-05 12:46:26 [main] WARN app.CustomerService - A WARN Message
```

```
2022-07-05 12:46:26 [main] ERROR app.CustomerService - An ERROR Message
```

# Logging to a file

```
logging.file.name=c:/temp/application.log
```

C:\temp\application.log

```
2022-07-05 13:52:49.002 INFO 5640 --- [main] app.CustomerApplication : Starting
CustomerApplication using Java 11.0.1 on DESKTOP-BVHRK6K with PID 5640 (C:\EnterpriseArchiterture\demo
code\Lesson13Logging\target\classes started by vedam in C:\EnterpriseArchiterture\demo code\Lesson13Logging)
2022-07-05 13:52:49.005 INFO 5640 --- [main] app.CustomerApplication : No active profile set,
falling back to default profiles: default
2022-07-05 13:52:49.628 INFO 5640 --- [main] app.CustomerApplication : Started
CustomerApplication in 1.141 seconds (JVM running for 1.569)
2022-07-05 13:52:49.634 INFO 5640 --- [main] app.CustomerService : An INFO Message
2022-07-05 13:52:49.634 WARN 5640 --- [main] app.CustomerService : A WARN Message
2022-07-05 13:52:49.634 ERROR 5640 --- [main] app.CustomerService : An ERROR Message
```

# Logging to a file



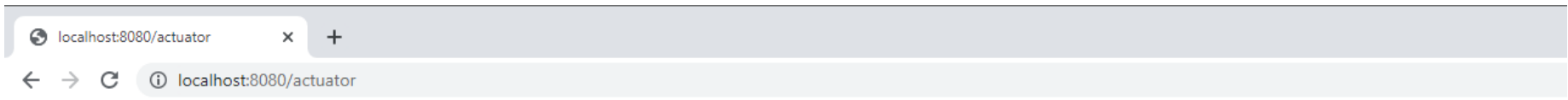
```
logging.file.path=c:/temp/logs
```

C:\temp\logs\spring.log

```
2022-07-05 13:52:49.002 INFO 5640 --- [main] app.CustomerApplication : Starting
CustomerApplication using Java 11.0.1 on DESKTOP-BVHRK6K with PID 5640 (C:\EnterpriseArchiterture\demo
code\Lesson13Logging\target\classes started by vedam in C:\EnterpriseArchiterture\demo code\Lesson13Logging)
2022-07-05 13:52:49.005 INFO 5640 --- [main] app.CustomerApplication : No active profile set,
falling back to default profiles: default
2022-07-05 13:52:49.628 INFO 5640 --- [main] app.CustomerApplication : Started
CustomerApplication in 1.141 seconds (JVM running for 1.569)
2022-07-05 13:52:49.634 INFO 5640 --- [main] app.CustomerService : An INFO Message
2022-07-05 13:52:49.634 WARN 5640 --- [main] app.CustomerService : A WARN Message
2022-07-05 13:52:49.634 ERROR 5640 --- [main] app.CustomerService : An ERROR Message
```

# ACTUATORS

# /actuator



```
{"_links":{"self":{"href":"http://localhost:8080/actuator","templated":false},"beans":{"href":"http://localhost:8080/actuator/beans","templated":false},"caches-cache":{"href":"http://localhost:8080/actuator/caches/{cache}","templated":true},"caches":{"href":"http://localhost:8080/actuator/caches","templated":false},"health":{"href":"http://localhost:8080/actuator/health","templated":false},"health-path":{"href":"http://localhost:8080/actuator/health/{*path}","templated":true},"info":{"href":"http://localhost:8080/actuator/info","templated":false},"conditions":{"href":"http://localhost:8080/actuator/conditions","templated":false},"shutdown":{"href":"http://localhost:8080/actuator/shutdown","templated":false},"configprops":{"href":"http://localhost:8080/actuator/configprops","templated":false},"configprops-prefix":{"href":"http://localhost:8080/actuator/configprops/{prefix}","templated":true},"env":{"href":"http://localhost:8080/actuator/env","templated":false},"env-toMatch":{"href":"http://localhost:8080/actuator/env/{toMatch}","templated":true},"loggers":{"href":"http://localhost:8080/actuator/loggers","templated":false},"loggers-name":{"href":"http://localhost:8080/actuator/loggers/{name}","templated":true},"heapdump":{"href":"http://localhost:8080/actuator/heapdump","templated":false},"threaddump":{"href":"http://localhost:8080/actuator/threaddump","templated":false},"metrics-requiredMetricName":{"href":"http://localhost:8080/actuator/metrics/{requiredMetricName}","templated":true},"metrics":{"href":"http://localhost:8080/actuator/metrics","templated":false},"scheduledtasks":{"href":"http://localhost:8080/actuator/scheduledtasks","templated":false},"mappings":{"href":"http://localhost:8080/actuator/mappings","templated":false}}}
```

# Actuator

---

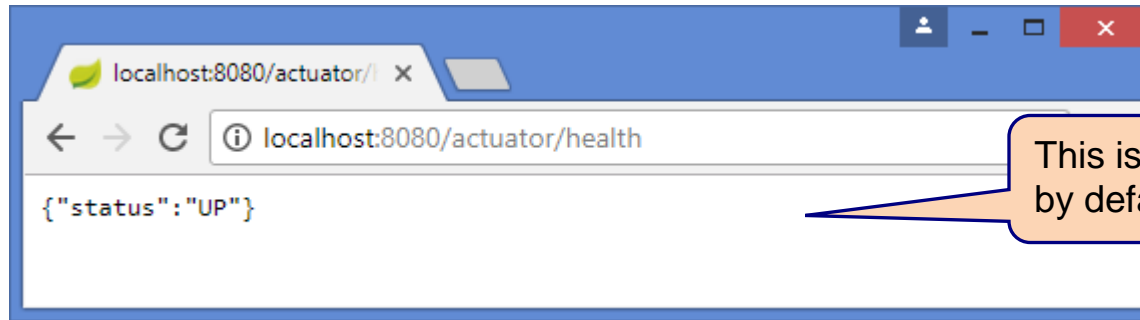
- Actuator brings production-ready features to our application

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

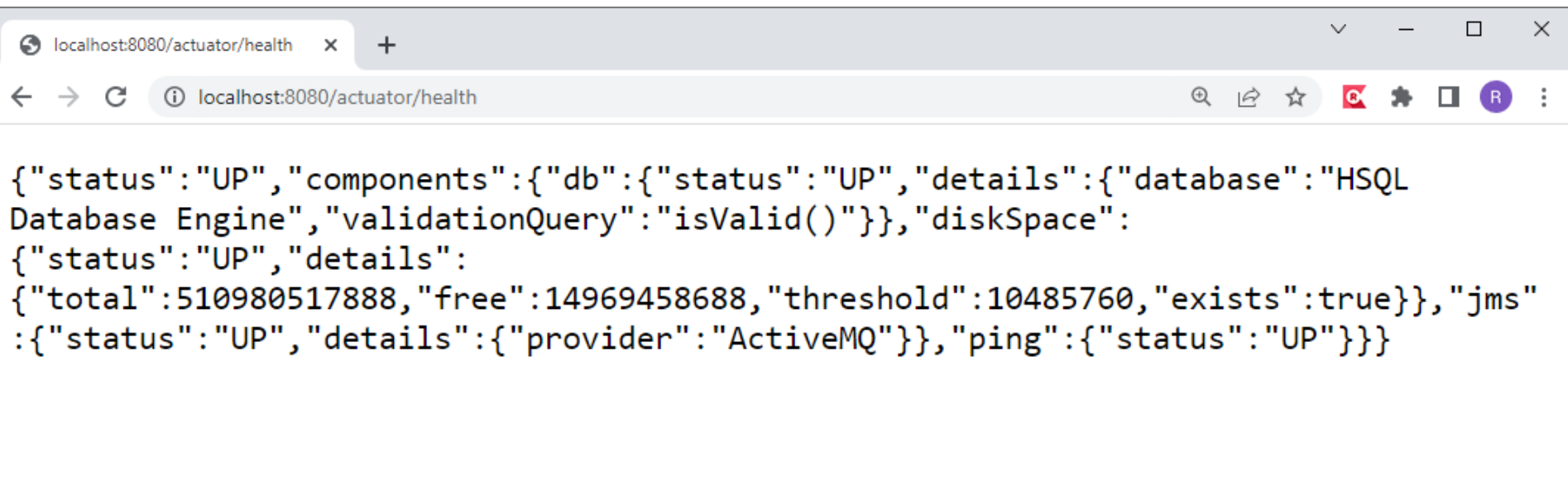
- Once this dependency is on the classpath several endpoints are available for us out of the box.
- You can modify existing actuators and you can write your own actuators



# /actuator/health



This is all that is shown by default



`management.endpoint.health.show-details=ALWAYS`

Show all health information

# Exposing actuators

---

- Only the /health actuator is exposed by default

- Exposing particular actuators

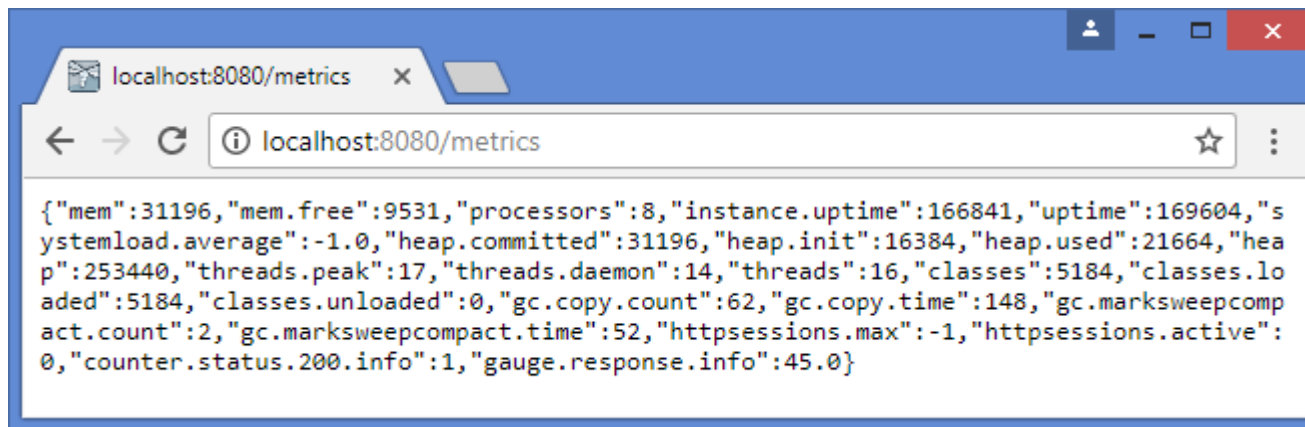
```
management.endpoints.web.exposure.include=beans,mappings
```

- Exposing all actuators

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

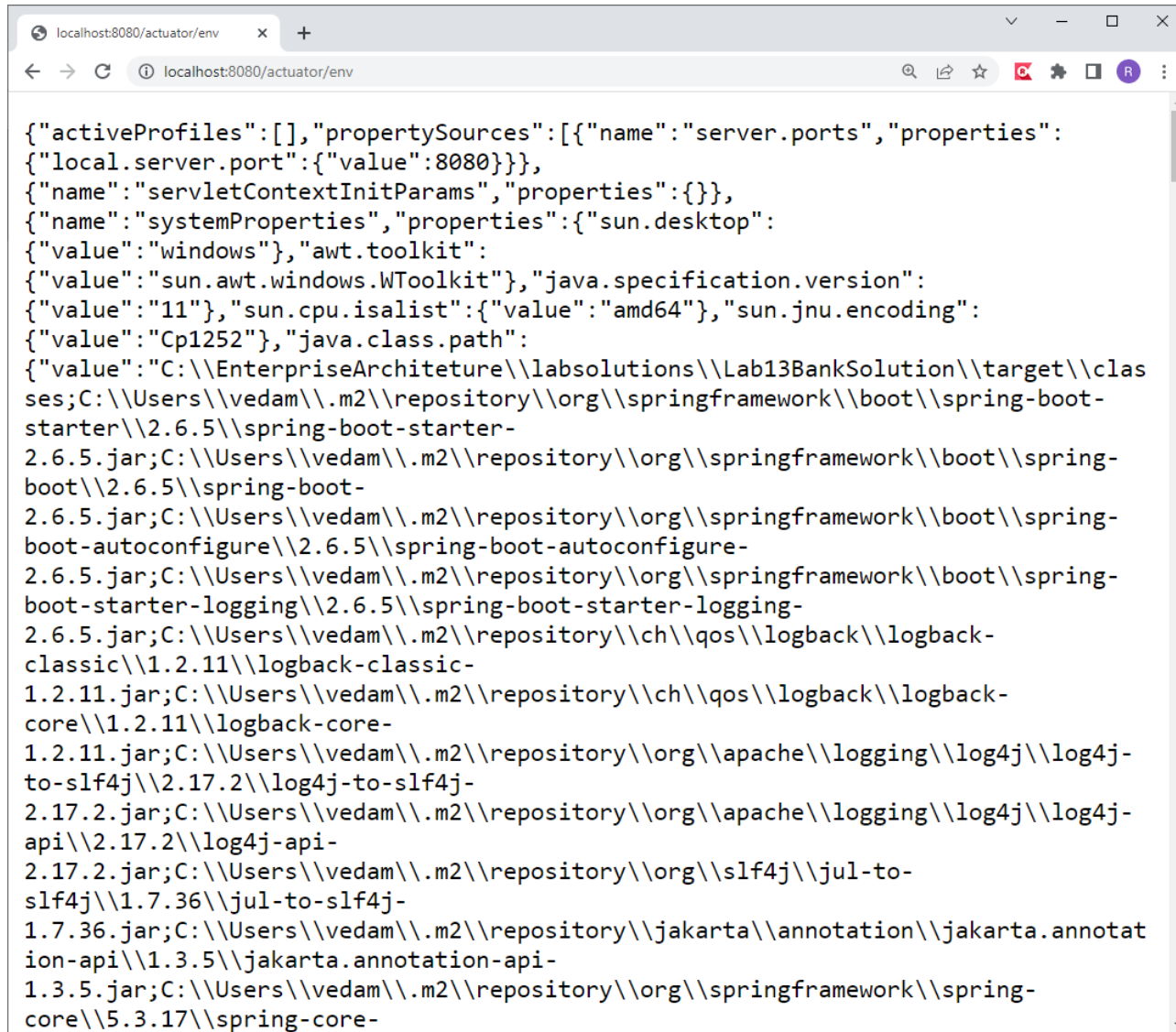
# /actuator/metrics

- Gives information such as memory, heap, processors, threads, classes loaded, classes unloaded, thread pools along with some HTTP metrics as well



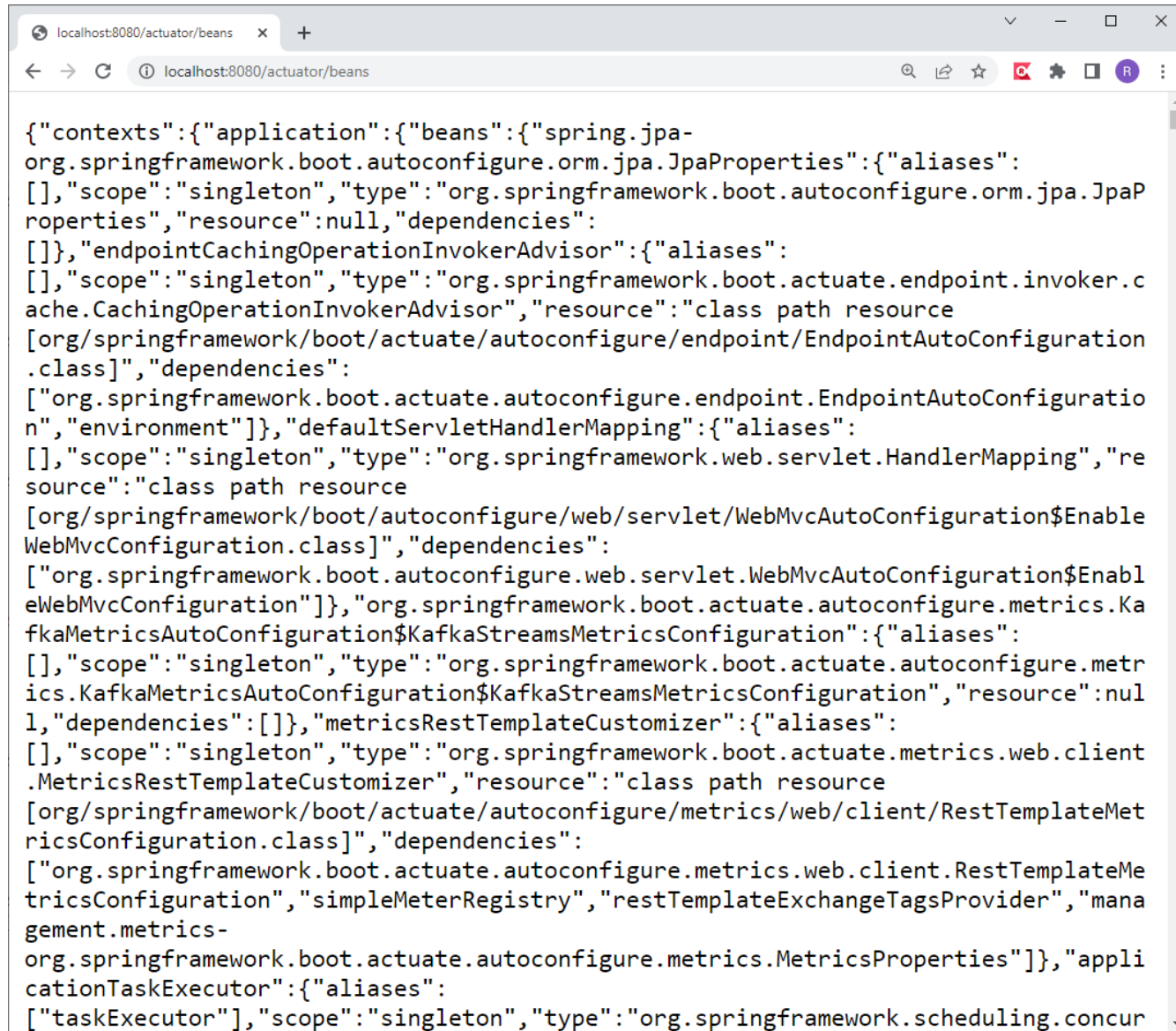
```
{
  "mem": 31196,
  "mem.free": 9531,
  "processors": 8,
  "instance.uptime": 166841,
  "uptime": 169604,
  "systemload.average": -1.0,
  "heap.committed": 31196,
  "heap.init": 16384,
  "heap.used": 21664,
  "heap.max": 253440,
  "threads.peak": 17,
  "threads.daemon": 14,
  "threads": 16,
  "classes": 5184,
  "classes.loaded": 5184,
  "classes.unloaded": 0,
  "gc.copy.count": 62,
  "gc.copy.time": 148,
  "gc.markswEEPcompact.count": 2,
  "gc.markswEEPcompact.time": 52,
  "http.sessions.max": -1,
  "http.sessions.active": 0,
  "counter.status.200.info": 1,
  "gauge.response.info": 45.0
}
```

# /actuator/env



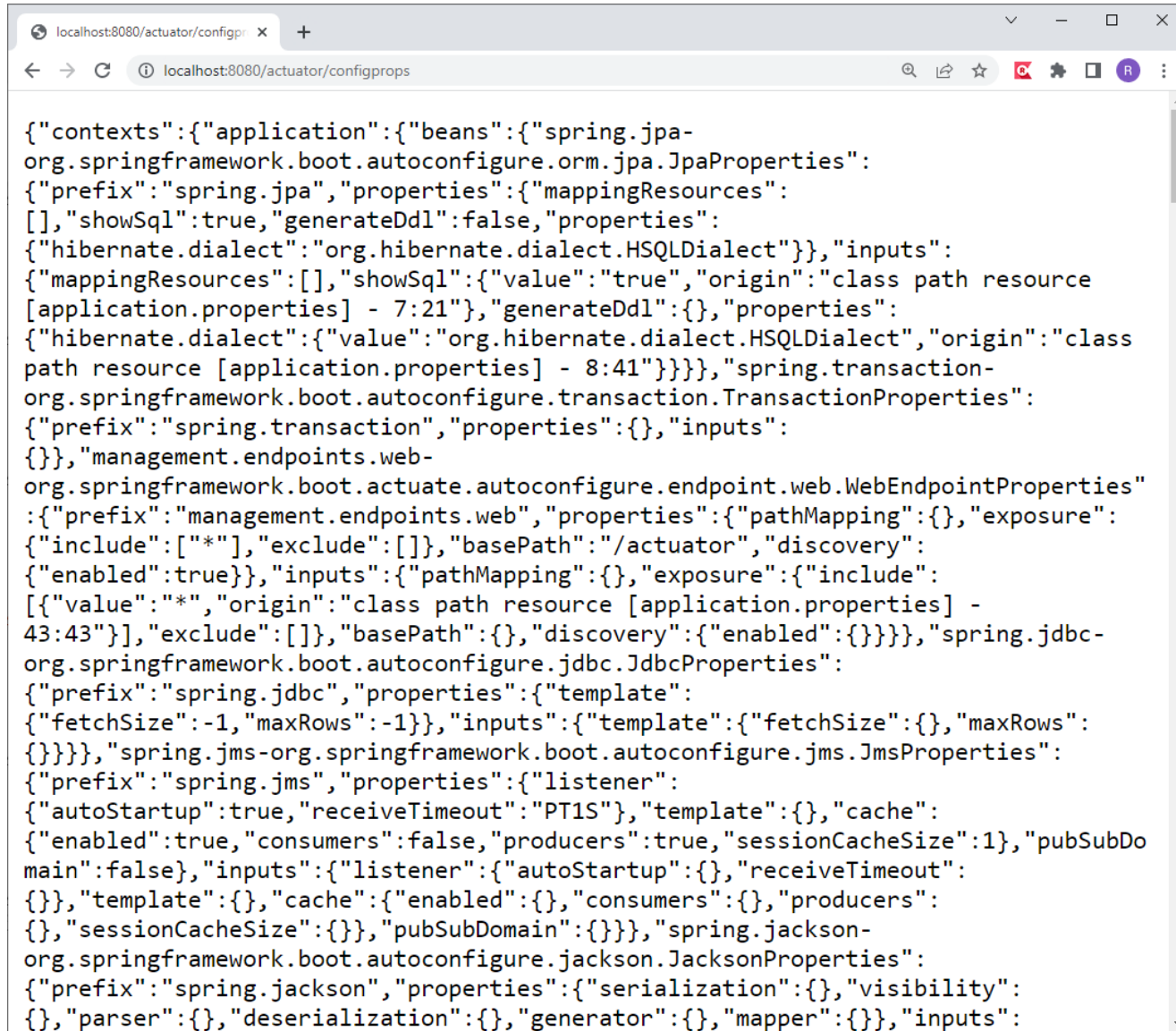
```
{
  "activeProfiles": [],
  "propertySources": [
    {
      "name": "server.ports",
      "properties": {
        "local.server.port": {
          "value": 8080
        }
      }
    },
    {
      "name": "servletContextInitParams",
      "properties": {}
    },
    {
      "name": "systemProperties",
      "properties": {
        "sun.desktop": {
          "value": "windows"
        },
        "awt.toolkit": {
          "value": "sun.awt.windows.WToolkit"
        },
        "java.specification.version": {
          "value": "11"
        },
        "sun.cpu.isalist": {
          "value": "amd64"
        },
        "sun.jnu.encoding": {
          "value": "Cp1252"
        },
        "java.class.path": {
          "value": "C:\\\\EnterpriseArchitecture\\\\labsolutions\\\\Lab13BankSolution\\\\target\\\\classes;C:\\\\Users\\\\vedam\\\\.m2\\\\repository\\\\org\\\\springframework\\\\boot\\\\spring-boot-starter\\\\2.6.5\\\\spring-boot-starter-2.6.5.jar;C:\\\\Users\\\\vedam\\\\.m2\\\\repository\\\\org\\\\springframework\\\\boot\\\\spring-boot\\\\2.6.5\\\\spring-boot-2.6.5.jar;C:\\\\Users\\\\vedam\\\\.m2\\\\repository\\\\org\\\\springframework\\\\boot\\\\spring-boot-autoconfigure\\\\2.6.5\\\\spring-boot-autoconfigure-2.6.5.jar;C:\\\\Users\\\\vedam\\\\.m2\\\\repository\\\\org\\\\springframework\\\\boot\\\\spring-boot-starter-logging\\\\2.6.5\\\\spring-boot-starter-logging-2.6.5.jar;C:\\\\Users\\\\vedam\\\\.m2\\\\repository\\\\ch\\\\qos\\\\logback\\\\logback-classic\\\\1.2.11\\\\logback-classic-1.2.11.jar;C:\\\\Users\\\\vedam\\\\.m2\\\\repository\\\\ch\\\\qos\\\\logback\\\\logback-core\\\\1.2.11\\\\logback-core-1.2.11.jar;C:\\\\Users\\\\vedam\\\\.m2\\\\repository\\\\org\\\\apache\\\\logging\\\\log4j\\\\log4j-to-slf4j\\\\2.17.2\\\\log4j-to-slf4j-2.17.2.jar;C:\\\\Users\\\\vedam\\\\.m2\\\\repository\\\\org\\\\apache\\\\logging\\\\log4j\\\\log4j-api\\\\2.17.2\\\\log4j-api-2.17.2.jar;C:\\\\Users\\\\vedam\\\\.m2\\\\repository\\\\org\\\\slf4j\\\\jul-to-slf4j\\\\1.7.36\\\\jul-to-slf4j-1.7.36.jar;C:\\\\Users\\\\vedam\\\\.m2\\\\repository\\\\jakarta\\\\annotation\\\\jakarta.annotation-api\\\\1.3.5\\\\jakarta.annotation-api-1.3.5.jar;C:\\\\Users\\\\vedam\\\\.m2\\\\repository\\\\org\\\\springframework\\\\spring-core\\\\5.3.17\\\\spring-core-"
        }
      }
    }
  ]
}
```

# /actuator/beans



```
{
  "contexts": {
    "application": {
      "beans": {
        "spring.jpa-org.springframework.boot.autoconfigure.orm.jpa.JpaProperties": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springframework.boot.autoconfigure.orm.jpa.JpaProperties",
          "resource": null,
          "dependencies": []
        },
        "endpointCachingOperationInvokerAdvisor": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springframework.boot.actuate.endpoint.invoker.cache.CachingOperationInvokerAdvisor",
          "resource": "class path resource [org/springframework/boot/actuate/autoconfigure/endpoint/EndpointAutoConfiguration.class]",
          "dependencies": [
            "org.springframework.boot.actuate.autoconfigure.endpoint.EndpointAutoConfiguration",
            "environment"
          ]
        },
        "defaultServletHandlerMapping": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springframework.web.servlet.HandlerMapping",
          "resource": "class path resource [org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoConfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [
            "org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoConfiguration$EnableWebMvcConfiguration"
          ]
        },
        "org.springframework.boot.actuate.autoconfigure.metrics.KafkaMetricsAutoConfiguration$KafkaStreamsMetricsConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springframework.boot.actuate.autoconfigure.metrics.KafkaMetricsAutoConfiguration$KafkaStreamsMetricsConfiguration",
          "resource": null,
          "dependencies": []
        },
        "metricsRestTemplateCustomizer": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springframework.boot.actuate.metrics.web.client.MetricsRestTemplateCustomizer",
          "resource": "class path resource [org/springframework/boot/actuate/autoconfigure/metrics/web/client/RestTemplateMetricsConfiguration.class]",
          "dependencies": [
            "org.springframework.boot.actuate.autoconfigure.metrics.web.client.RestTemplateMetricsConfiguration",
            "simpleMeterRegistry",
            "restTemplateExchangeTagsProvider",
            "management.metrics-org.springframework.boot.actuate.autoconfigure.metrics.MetricsProperties"
          ]
        },
        "applicationTaskExecutor": {
          "aliases": [
            "taskExecutor"
          ],
          "scope": "singleton",
          "type": "org.springframework.scheduling.concurrent"
        }
      }
    }
  }
}
```

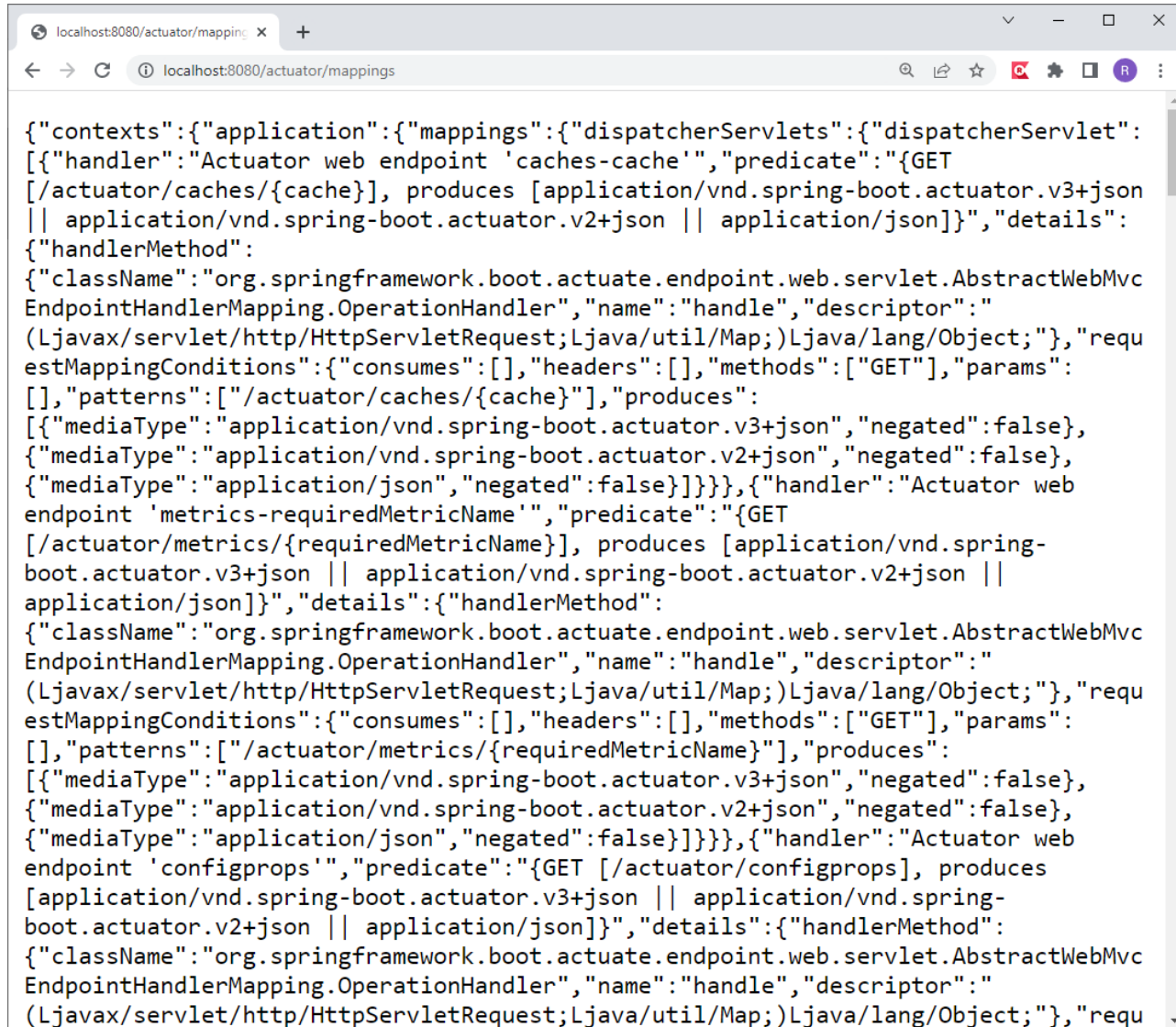
# /actuator/configprops



```
{
  "contexts": {
    "application": {
      "beans": {
        "spring.jpa-
org.springframework.boot.autoconfigure.orm.jpa.JpaProperties": {
          "prefix": "spring.jpa",
          "properties": {
            "mappingResources": [
              ],
            "showSql": true,
            "generateDdl": false,
            "properties": {
              "hibernate.dialect": "org.hibernate.dialect.HSQLDialect"
            }
          },
          "inputs": {
            "mappingResources": [
              ],
            "showSql": {
              "value": "true",
              "origin": "class path resource [application.properties] - 7:21"
            },
            "generateDdl": {
              },
            "properties": {
              "hibernate.dialect": {
                "value": "org.hibernate.dialect.HSQLDialect",
                "origin": "class path resource [application.properties] - 8:41"
              }
            }
          }
        },
        "spring.transaction-
org.springframework.boot.autoconfigure.transaction.TransactionProperties": {
          "prefix": "spring.transaction",
          "properties": {
            },
          "inputs": {
            }
        },
        "management.endpoints.web-
org.springframework.boot.actuate.autoconfigure.endpoint.web.WebEndpointProperties": {
          "prefix": "management.endpoints.web",
          "properties": {
            "pathMapping": {
              },
            "exposure": {
              "include": [
                "*"
              ],
              "exclude": [
                ]
            },
            "basePath": "/actuator",
            "discovery": {
              "enabled": true
            }
          },
          "inputs": {
            "pathMapping": {
              },
            "exposure": {
              "include": [
                {
                  "value": "*",
                  "origin": "class path resource [application.properties] - 43:43"
                }
              ],
              "exclude": [
                ]
            },
            "basePath": {
              },
            "discovery": {
              "enabled": {
                }
            }
          }
        },
        "spring.jdbc-
org.springframework.boot.autoconfigure.jdbc.JdbcProperties": {
          "prefix": "spring.jdbc",
          "properties": {
            "template": {
              "fetchSize": -1,
              "maxRows": -1
            }
          },
          "inputs": {
            "template": {
              "fetchSize": {
                },
              "maxRows": {
                }
            }
          }
        },
        "spring.jms-org.springframework.boot.autoconfigure.jms.JmsProperties": {
          "prefix": "spring.jms",
          "properties": {
            "listener": {
              "autoStartup": true,
              "receiveTimeout": "PT1S",
              "template": {
                },
              "cache": {
                "enabled": true,
                "consumers": false,
                "producers": true,
                "sessionCacheSize": 1
              },
              "pubSubDo
main": false
            },
            "inputs": {
              "listener": {
                "autoStartup": {
                  },
                "receiveTimeout": {
                  }
              },
              "template": {
                },
              "cache": {
                "enabled": {
                  },
                "consumers": {
                  },
                "producers": {
                  },
                "sessionCacheSize": {
                  },
                "pubSubDomain": {
                  }
              }
            }
          },
          "spring.jackson-
org.springframework.boot.autoconfigure.jackson.JacksonProperties": {
          "prefix": "spring.jackson",
          "properties": {
            "serialization": {
              },
            "visibility": {
              },
            "parser": {
              },
            "deserialization": {
              },
            "generator": {
              },
            "mapper": {
              },
            "inputs": {
              }
            }
          }
        }
      }
    }
  }
}
```

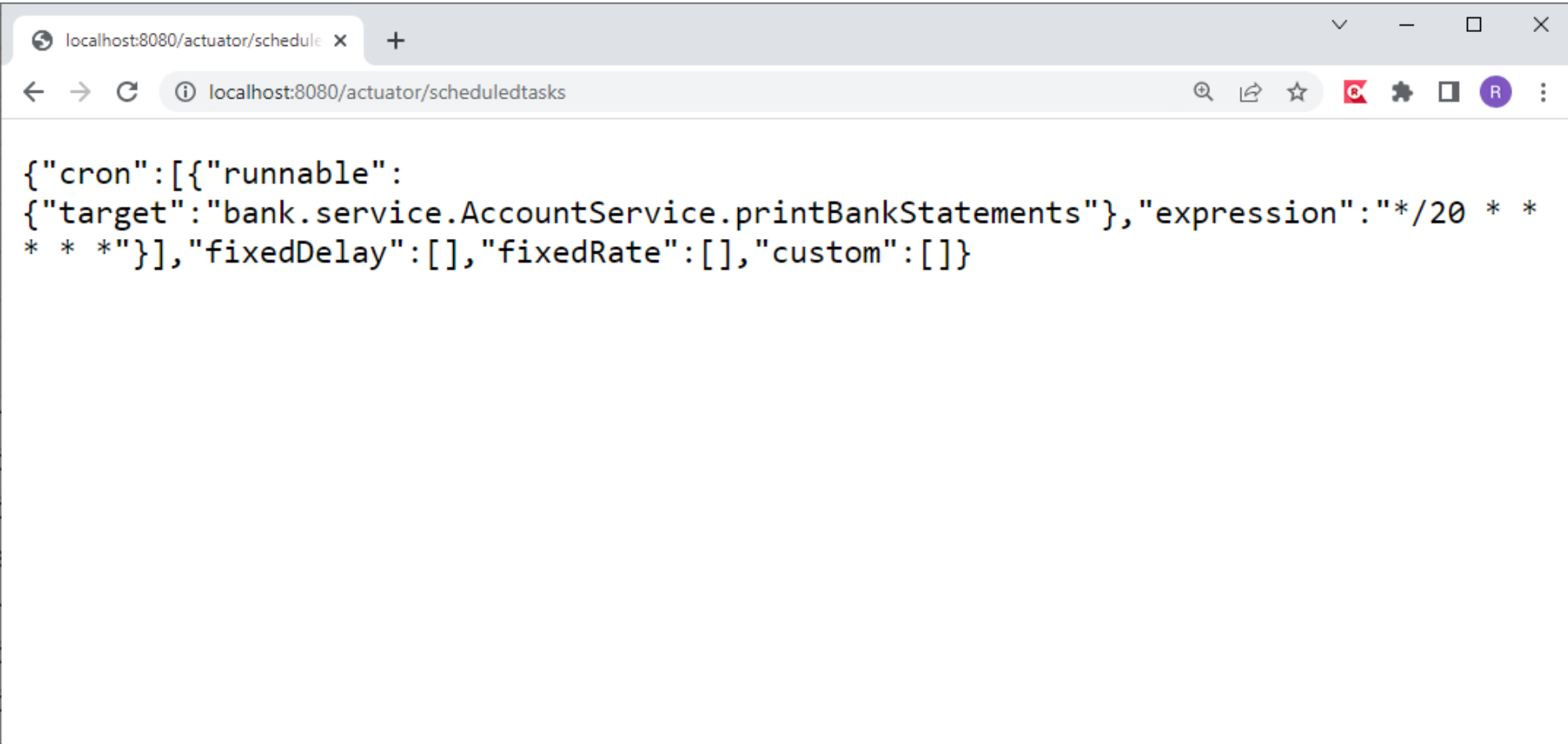


# /actuator/mappings



```
{
  "contexts": {
    "application": {
      "mappings": {
        "dispatcherServlets": {
          "dispatcherServlet": [
            {
              "handler": "Actuator web endpoint 'caches-cache'",
              "predicate": "{GET [/actuator/caches/{cache}], produces [application/vnd.spring-boot.actuator.v3+json || application/vnd.spring-boot.actuator.v2+json || application/json]}",
              "details": {
                "handlerMethod": {
                  "className": "org.springframework.boot.actuate.endpoint.web.servlet.AbstractWebMvcEndpointHandlerMapping.OperationHandler",
                  "name": "handle",
                  "descriptor": "(Ljavax/servlet/http/HttpServletRequest;Ljava/util/Map;)Ljava/lang/Object;",
                  "requestMappingConditions": {
                    "consumes": [],
                    "headers": [],
                    "methods": ["GET"],
                    "params": [],
                    "patterns": ["/actuator/caches/{cache}"],
                    "produces": [
                      {
                        "mediaType": "application/vnd.spring-boot.actuator.v3+json",
                        "negated": false
                      },
                      {
                        "mediaType": "application/vnd.spring-boot.actuator.v2+json",
                        "negated": false
                      },
                      {
                        "mediaType": "application/json",
                        "negated": false
                      }
                    ]
                  }
                }
              }
            }
          ]
        }
      }
    },
    "metrics-requiredMetricName": {
      "mappings": {
        "dispatcherServlets": {
          "dispatcherServlet": [
            {
              "handler": "Actuator web endpoint 'metrics-requiredMetricName'",
              "predicate": "{GET [/actuator/metrics/{requiredMetricName}], produces [application/vnd.spring-boot.actuator.v3+json || application/vnd.spring-boot.actuator.v2+json || application/json]}",
              "details": {
                "handlerMethod": {
                  "className": "org.springframework.boot.actuate.endpoint.web.servlet.AbstractWebMvcEndpointHandlerMapping.OperationHandler",
                  "name": "handle",
                  "descriptor": "(Ljavax/servlet/http/HttpServletRequest;Ljava/util/Map;)Ljava/lang/Object;",
                  "requestMappingConditions": {
                    "consumes": [],
                    "headers": [],
                    "methods": ["GET"],
                    "params": [],
                    "patterns": ["/actuator/metrics/{requiredMetricName}"],
                    "produces": [
                      {
                        "mediaType": "application/vnd.spring-boot.actuator.v3+json",
                        "negated": false
                      },
                      {
                        "mediaType": "application/vnd.spring-boot.actuator.v2+json",
                        "negated": false
                      },
                      {
                        "mediaType": "application/json",
                        "negated": false
                      }
                    ]
                  }
                }
              }
            }
          ]
        }
      }
    },
    "configprops": {
      "mappings": {
        "dispatcherServlets": {
          "dispatcherServlet": [
            {
              "handler": "Actuator web endpoint 'configprops'",
              "predicate": "{GET [/actuator/configprops], produces [application/vnd.spring-boot.actuator.v3+json || application/vnd.spring-boot.actuator.v2+json || application/json]}",
              "details": {
                "handlerMethod": {
                  "className": "org.springframework.boot.actuate.endpoint.web.servlet.AbstractWebMvcEndpointHandlerMapping.OperationHandler",
                  "name": "handle",
                  "descriptor": "(Ljavax/servlet/http/HttpServletRequest;Ljava/util/Map;)Ljava/lang/Object;",
                  "requestMappingConditions": {
                    "consumes": [],
                    "headers": [],
                    "methods": ["GET"],
                    "params": [],
                    "patterns": ["/actuator/configprops"],
                    "produces": [
                      {
                        "mediaType": "application/vnd.spring-boot.actuator.v3+json",
                        "negated": false
                      },
                      {
                        "mediaType": "application/vnd.spring-boot.actuator.v2+json",
                        "negated": false
                      },
                      {
                        "mediaType": "application/json",
                        "negated": false
                      }
                    ]
                  }
                }
              }
            }
          ]
        }
      }
    }
  }
}
```

# /actuator/scheduledtasks



```
{
  "cron": [
    {
      "runnable": {
        "target": "bank.service.AccountService.printBankStatements",
        "expression": "* / 20 * * * * *"
      },
      "fixedDelay": [],
      "fixedRate": [],
      "custom": []
    }
  ]
}
```



# Available actuators

GET	/autoconfig	Provides an auto-configuration report describing what auto-configuration conditions passed and failed.
GET	/configprops	Describes how beans have been injected with configuration properties (including default values).
GET	/beans	Describes all beans in the application context and their relationship to each other.
GET	/dump	Retrieves a snapshot dump of thread activity.
GET	/env	Retrieves all environment properties.

# Available actuators

GET	/env/{name}	Retrieves a specific environment value by name.
GET	/health	Reports health metrics for the application, as provided by HealthIndicator implementations.
GET	/info	Retrieves custom information about the application, as provided by any properties prefixed with info.
GET	/mappings	Describes all URI paths and how they're mapped to controllers (including Actuator endpoints).
GET	/metrics	Reports various application metrics such as memory usage and HTTP request counters.

# Available actuators

GET	/metrics/{name}	Reports an individual application metric by name.
POST	/shutdown	Shuts down the application; requires that endpoints.shutdown.enabled be set to true.
GET	/trace	Provides basic trace information (timestamp, headers, and so on) for HTTP requests.

# shutdown

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

POST



http://localhost:8080/actuator/shutdown

Params

Authorization

Headers (9)

Body ●

Pre-request Script

Tests

Settings

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON



```
1 {  
2   "message": "Shutting down, bye..."  
3 }
```