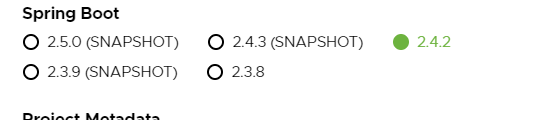
**Spring 5**

1. Download spring 5 jar and add lib to java projects : <https://repo.spring.io/release/org/springframework/spring/>
2. Copy paste ApplicationContext.xml
3. Create bean with bean id and class

**Spring Boot**

****

Dependancy :

1. Actuator
2. Devtools
3. Spring web
4. Spring jpa
5. Mysql
6. H2 database
7. Email
8. Spring data Jdbc api
9. Security
10. spring-boot-starter-validation
11. spring-boot-starter-cache
12. Lombok
13. spring-cloud-starter-config (clinet)
14. spring-cloud-starter-netflix-eureka-client
15. sweger

**Dependancy Injection:**

1. Propertiy injection
2. Setter Injection
3. Construtor injection

Try without spring with junit test and using spring

**Qualifier**

@Qualifier(“beanName”) first letter should be small

1. Use this for property ,setter,constructor

**Primary**

@Primary

**Profile**

Create one Envirment interface with tow implementation one for testing another for production . then add @component(“SameName”)

And then annotate with @Profile(“PROD”) and @Profile(“TEST”)

spring.profiles.active=PROD

which profile you want to use mention here.

**Default profile**

~~spring.profiles.active=PROD~~  remove this from property file

@Profile({"PROD","default"})

**@PostConstruct and @PreDestroy**

@PostConstruct annotation on method will run that method after bean creation

@PreDestroy method called when bean destroy

**Bean Type**

@Scope("prototype")

**Application properties**

logging.level.org.hibernate.stat=debug # enabling debug logger for specific package

spring.jackson.serialization.write-dates-as-timestamps**=**false # by setting this false it will not convert date into Timestamp long value

Logback.xml

logging:

file:

name: /log/mail/sysout.log

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<configuration>

<property name=*"path"* value=*"log"*/>

<property name=*"archivedPath"* value=*"log/archived"*/>

<property name=*"pattern"* value=*"%d [%t:%p] [%C:%L] : %m%n"*/>

<property name=*"archivedFileNamePattern"* value=*"%d{yyyy-MM-dd}.%i.log.gz"*/>

<property name=*"maxfilesize"* value=*"10MB"*/>

<property name=*"totalSizeCap"* value=*"1GB"*/>

<property name=*"maxHistory"* value=*"5"*/>

<appender name=*"consoleAppender"* class=*"ch.qos.logback.core.ConsoleAppender"*>

<encoder>

<pattern>

${pattern}

</pattern>

</encoder>

</appender>

<appender name=*"consoleFileAppender"* class=*"ch.qos.logback.core.rolling.RollingFileAppender"*>

<file>${path}/sysout.log</file>

<rollingPolicy class=*"ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy"*>

<fileNamePattern>${archivedPath}/sysout.${archivedFileNamePattern}</fileNamePattern>

<maxFileSize>${maxfilesize}</maxFileSize>

<totalSizeCap>${totalSizeCap}</totalSizeCap>

<maxHistory>${*maxHistory*}</maxHistory>

</rollingPolicy>

<encoder>

<pattern>

${pattern}

</pattern>

</encoder>

</appender>

<root level=*"INFO"*>

<appender-ref ref=*"consoleAppender"*/>

<appender-ref ref=*"consoleFileAppender"*/>

</root>

</configuration>

%d [%t-%p] [%C : %L] - %m%n

%d -- date

%t --- thread

%p --- level

%C ---- class path

%L ---- line number

%m --- msg and %n --- new line

%M ---- Method Name

Email using spring boot

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-mail</artifactId>

</dependency>

spring.mail.host=smtp.gmail.com

spring.mail.port=587

spring.mail.username=daryabusiness@gmail.com

spring.mail.password=\*\*\*\*\*\*\*\*

spring.mail.properties.mail.smtp.starttls.enable=true

@Autowired

JavaMailSender mailsender;

**public** **void** sendFile(String To,String Text,String Report) **throws** MessagingException {

MimeMessage mimeMessage=mailsender.createMimeMessage();

MimeMessageHelper helper=**new** MimeMessageHelper(mimeMessage,**true**);

helper.setFrom(from);

helper.setTo(To);

helper.setSubject(subject);

helper.setText(Text);

FileSystemResource rs=**new** FileSystemResource(**new** File(Report));

helper.addAttachment("bill.pdf", rs);

mailsender.send(mimeMessage);

}

Send Html Email :

helper.setText(msgBody,**true**);

msgBody 🡪 must be HTML String all other conten same as perviuse just send true;

HTML in line connect

String msgBody= "<html><body>"+dto.getMailContent()+

"<br>"+

"<img src='cid:identifier1234'></body></html>"

+ "<br><br><br>"+(dto.getSignature()!=**null** && !dto.getSignature().trim().equals("")?dto.getSignature():signature);

<code other same like privous

>

File file=**new** File("E:\\Learning\\Projects\\Resturant Billing System\\bws\\BillingSystemBackend\\src\\main\\resources\\invoice\_logo.png");

FileSystemResource res=**new** FileSystemResource(file);

helper.addInline("identifier1234", res);

need to pass same content id in helper and template like identifier1234

Async Functionality Method

@EnableAsync

**public** **class** LoginApplication **implements** CommandLineRunner{

@Component

**public** **class** AsyncService {

@Async("test1")

**public** Future<String> testAsyncMethod() **throws** InterruptedException {

System.***out***.println("testAsyncMethod start "+Thread.*currentThread*().getName());

Thread.*sleep*(10000);

System.***out***.println("testAsyncMethod complete "+Thread.*currentThread*().getName());

**return** **new** AsyncResult<String>("Chal gya");

}

@Async("test2")

**public** **void** testAsyncMethod2() **throws** InterruptedException {

System.***out***.println("testAsyncMethod2start "+Thread.*currentThread*().getName());

Thread.*sleep*(10000);

System.***out***.println("testAsyncMethod2 complete "+Thread.*currentThread*().getName());

}

}

@Configuration

**public** **class** AyncConfig {

@Bean(name="test1")

**public** Executor test1() {

ThreadPoolTaskExecutor th=**new** ThreadPoolTaskExecutor();

th.setCorePoolSize(5);

th.setMaxPoolSize(5);

th.setThreadNamePrefix("Test ");

th.initialize();

**return** th;

}

@Bean(name="test2")

**public** Executor test2() {

ThreadPoolTaskExecutor th=**new** ThreadPoolTaskExecutor();

th.setCorePoolSize(2);

th.setMaxPoolSize(2);

th.setThreadNamePrefix("Prod :");

th.initialize();

**return** th;

}

}

----------function shuld be only use when when you need to perform task once all thread process

@CrossOrigin

@GetMapping("test")

**public** String test() **throws** Exception {

Future<String> obj=asyncservice.testAsyncMethod();

**while**(**true**) {

**if**(obj.isDone()) {

System.***out***.println(obj.get());

**break**;

}

}

**return** "Hello world";

}

**Unit Testing in spring boot**

@SpringBootTest

This is Example of equals

@TestMethodOrder(OrderAnnotation.**class**)

**class** BillingSystemBackendApplicationTests {

@Test

@Order(1)

**public** **void** adminUserExist() {

UserRequestDTO user=**new** UserRequestDTO();

user.setUsername("Admin");

user.setPassword("Admin");

ResponseDTO respons=**this**.billingProcess.getUserDetails(user);

Assertions.*assertNotEquals*(**null**, respons.getData());

}

This is Example of throw

@Test

@Order(6)



**public** **void** whenUserNameEmpty() {

MandatoryFieldException expt=

Assertions.*assertThrows*(MandatoryFieldException.**class**,()->userBusinessLogic.getUserDetails("", "Arya$786"));

}

Filter

@Component

@Order(1)

**public** **class** LoginFilterOne **implements** Filter{

@Override

**public** **void** doFilter(ServletRequest request, ServletResponse response, FilterChain chain)

**throws** IOException, ServletException {

HttpServletRequest httprequest=(HttpServletRequest)request;

System.***out***.println(httprequest.getRequestURI());

chain.doFilter(request, response);

}

}

**@ConfigurationProperties**

@Component

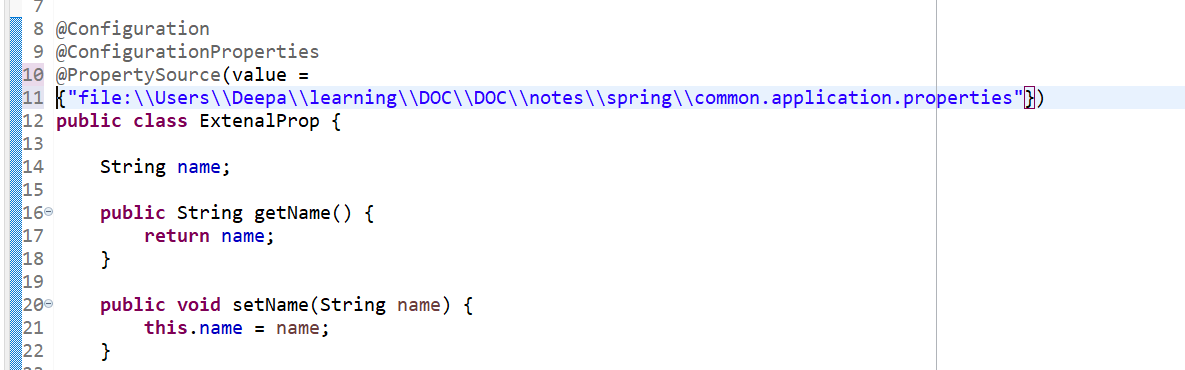
@ConfigurationProperties("paytm.payment.sandbox")

**public** **class** PaytmPropConfig {



--

External property file reading.



**Cache**

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-cache</artifactId>

</dependency>

@EnableCaching ------------- most important annotion

**import** org.springframework.cache.annotation.CacheEvict;

**import** org.springframework.cache.annotation.Cacheable;

@GetMapping("getData2")

@Cacheable("getData2")

**public** List<String> getData2() **throws** InterruptedException{

Thread.*sleep*(5000);

**return** Arrays.*asList*("HIII","by");

}

@GetMapping("clearAll")

@CacheEvict({"getData","getData2"})

**public** String clearData1() {

**return** "clear 1";

}

**Rest API validation**

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-validation</artifactId>

</dependency>

@PostMapping("sendData5")

**public** **void** testValidationAPI(@Valid @RequestBody ValidationTestDO validdo) {

System.***out***.println(validdo);

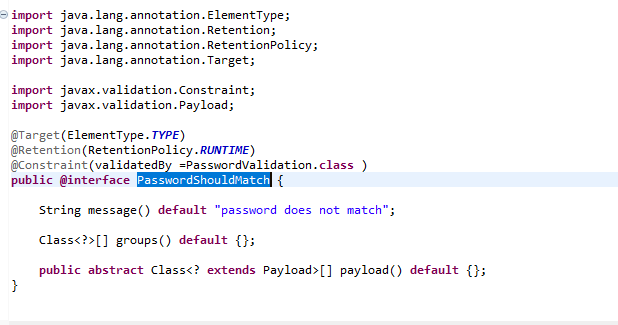
}



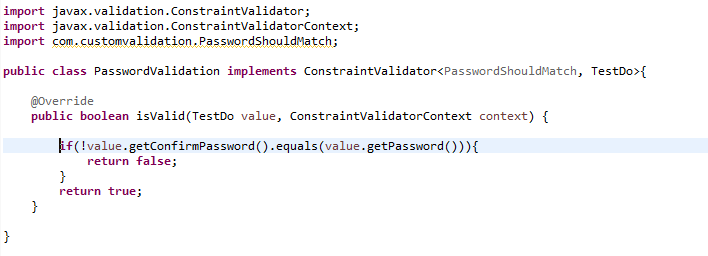
**Custom validation**

****

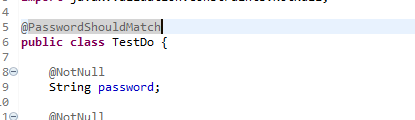
**Step 1.Create annotation**

****

**Step 2 . create validation**

****

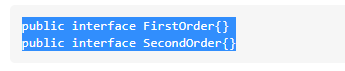
**Step 3 apply validation.**

****

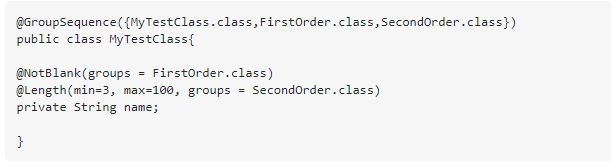
**How to create ORDER of validation.**

@costomValidation(groups=”FirstOrder.clss”)

1. Create Interface



2.



Pathparam vs request param

http://localhost:8081/N/readProperties/A

// path param

@GetMapping("readProperties/{key}")

**public** String readProperties(@PathVariable("key")String key) {

System.***out***.println("readProperties1");

**return** key;

}

http://localhost:8081/N/readProperties?key=11

// @RequestParm

@GetMapping("readProperties")

**public** String readProperties2(@RequestParam("key")String key) {

System.***out***.println("readProperties2");

**return** key;

}

Output:

**Schedular**

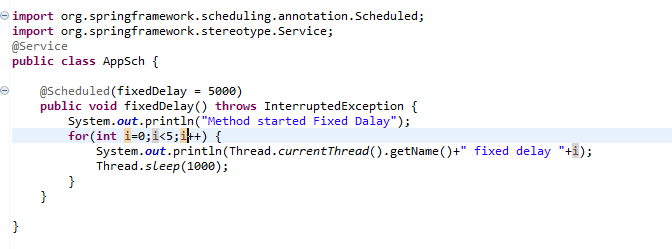
Step 1:Enable scheudlar

@EnableScheduling

**public** **class** CurrencyExchangeApplication {

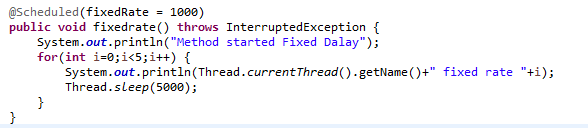
**Type 1 . Fixed delay**

* **Wait for fixed time after completion of first execution.**

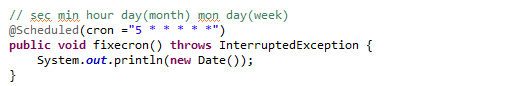
****

**Type 2 . Fixed rate**

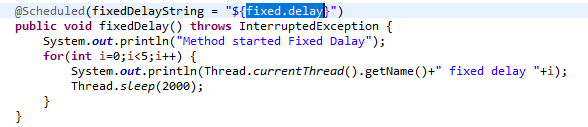
* **Run at fixed rate but wait for pevious execution complete**

****

**Type 2 . cron job.**

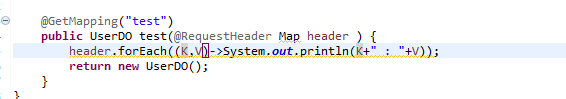


**Type 3 . refer from peerpert file**

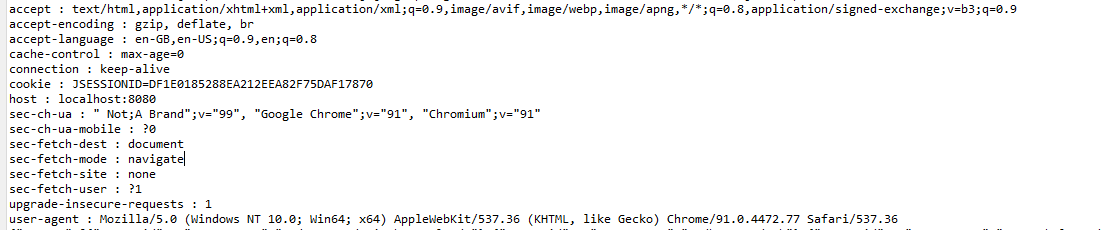
****

****

**How to Recive Header Parameter in rest Request**



String common header



**Code to download File / download code.**

****

spring.servlet.multipart.max-file-size=1GB

spring.servlet.multipart.max-request-size=1GB

AOP :

Step 1 :

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-aop</artifactId>

</dependency>

**Step 2 : before intercepter with printing method name:**

@Configuration

@Aspect

**public** **class** AopConfig {

//What kind of method calls I would intercept

//execution(\* PACKAGE.\*.\*(..)) // point cut

@Before(value="com.example.demo.PointCutConf.ff()")

**public** **void** logBefore(JoinPoint joinpoint) {

**for**(Object obj:joinpoint.getArgs()) {

System.***out***.println("param :"+obj);

}

System.***out***.println("Intercept before "+joinpoint);

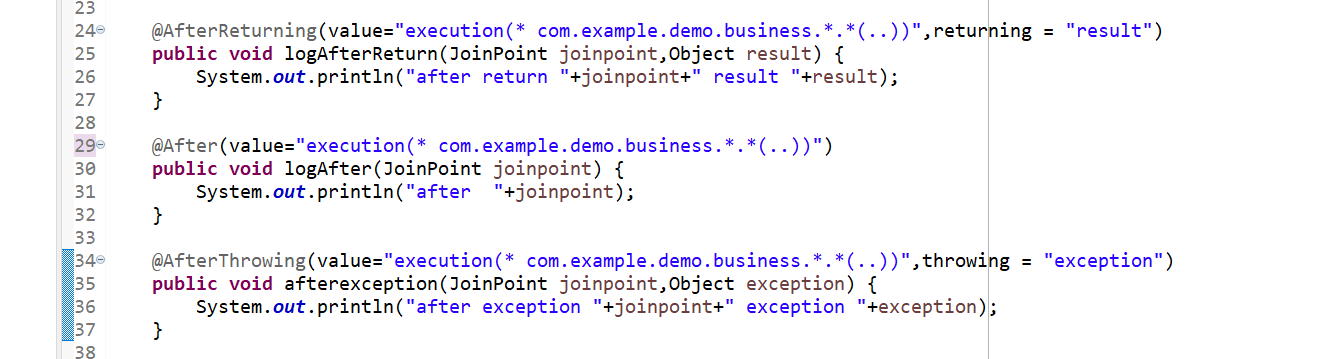
}

}

Point cut:

1. execution(\* com.example.demo..\*.\*(..)) -> intercept all subpackage under com.example.demo

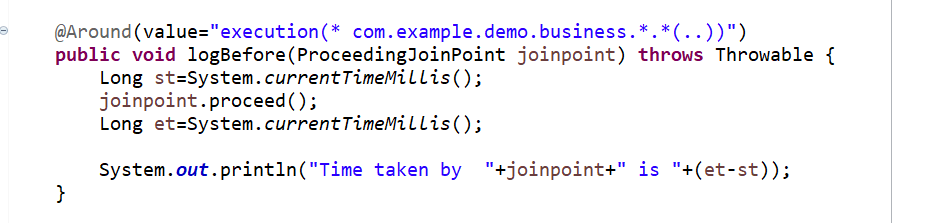
AfterReturn and after and afterthrow



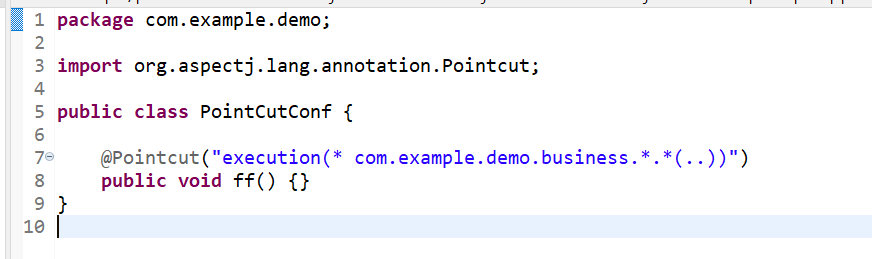
Note after and after return both call when method return with or without argument

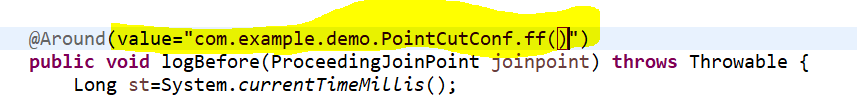
If return type is null then dispay null;

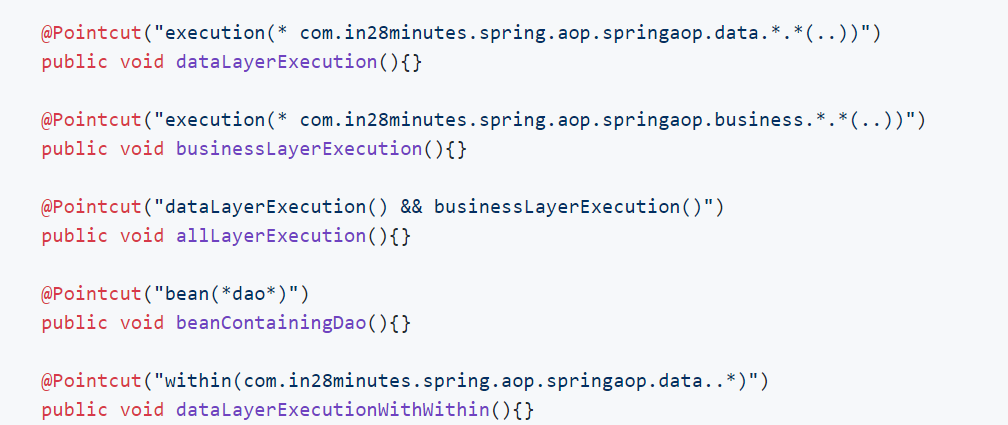
After throwing only call when exception occure



Best practice to define point cut

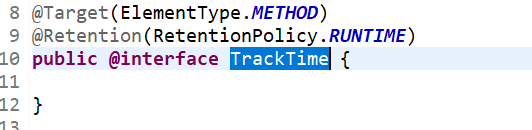




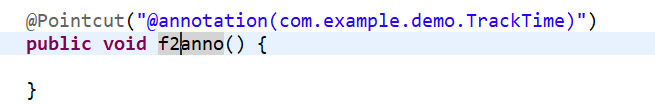


How to create custom annotation for around;

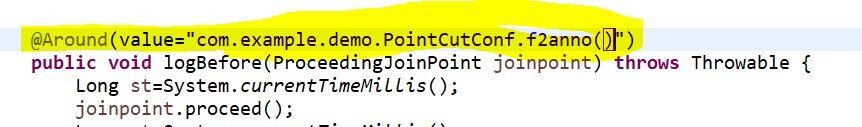
Step 1

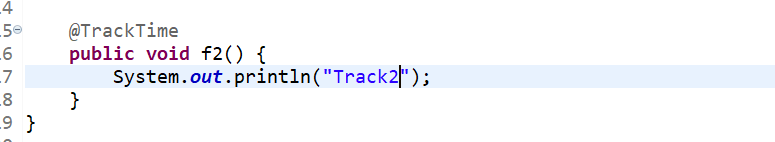


Step 2



Step 3



Step 4: use annotation 

Rest template to upload files :



Steps

1.add all dependency

2. create in memeory user first and test