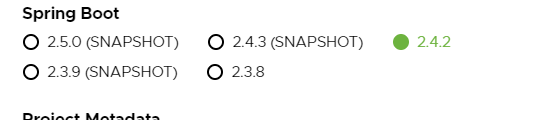
**Spring 5**

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. Devtools
2. Spring web
3. Spring jpa
4. Mysql
5. H2 database
6. Lombok
7. Spring boot actuator
8. Thymeleaf
9. Email
10. Jdbc api

**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")

Logback.xml

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

<property name=*"archivedpath"* value=*"${path}/archive"*/>

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

<configuration>

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

<layout class=*"ch.qos.logback.classic.PatternLayout"*>

<pattern>

%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n

</pattern>

</layout>

</appender>

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

<file>Log/sysout.log</file>

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

<fileNamePattern>Log/archived/sysout.%d{yyyy-MM-dd}.%i.log.gz </fileNamePattern>

<maxFileSize>1MB</maxFileSize>

<totalSizeCap>2GB</totalSizeCap>

<maxHistory>5</maxHistory>

</rollingPolicy>

<encoder>

<pattern>[%d] %p %c{1.} [%t] %m%n</pattern>

</encoder>

</appender>

<root level=*"INFO"*>

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

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

</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

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

@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"));

}