**Spring Boot mail**

**(Java mailing using spring boot)**

**=>To maintain db tables we have DB s/ws (eg: oracle,mysql,...)**

**=>To maintain java objs for global visibiitly we have Jndi registries (eg: rmi registry, cos registry,...)**

**=> To maintain web applications we have web servers (eg: Tomcat, Jetty and etc..)**

**=> To maintain email accounts and email messages we have mail servers. (eg: James, Ms Exchange server,...) eg:: James mail server, Microsoft exhange server, Lotus notes servers and etc.. (mail servers)**

**note: Few Application servers like weblogic, webshpere and etc..**

**giving built-in mail servers. eg: weblogic mail server, wildfly mail server, glassfish mail server and etc.. note:: In java, we have different apis to locate**

**Java App**

-JDBC API--

**DB s/w**

**Java App**

JNDI API

**Jndi registry**

**Java mail api pkgs are**

**Java App**

Java Mail API ———————— Mail server

**(or)**

**and interact with different destinations servers/softwares**

**javax.mail, javax.mail.activation (old names)**

**| jakarta.mail, jakarta.mail.activation( new names)**

**What is the first work u do after going to company?**

**What is the first work u do after logged in to computer (Work from Home)**

**Ans) Checking mail**

**In**

**Q) Can i call java mail api as the JMS?**

**ans) JMS means Java Messaging Service and it is noway related to java mail api.. so feeling JMS as Java Mail Service is the wrong analyzations**

**=>JMS is given for messages based communication across multiple java**

**comps**

**=> JAva mail is given for performing mail operations from the java app**

**PM:: Project Manager**

**PL :: Project Leader**

**TL ::Team Leader**

**How do u chekc ur mails and what account mails u will check?**

**a) We check company mail account using Ms outlook / ms outlook express / Google Suite**

How do u get task?

eg:: siva.bollikonda@cognizant.com Purush.katasani@otsi.com

Very popular

JIRA

Ans) Through mail or JIRA tickets / User Stories /tasks

♡

These company mail accounts

will be configured with

Ms outlook..in u r desktop or laptop

(or) Google Sheets

(or) Esay Redmine

In Agile model PM is like Scrum Master

Two types of protocols

note:: Application level protocols run on the top of network level protocols

(old days)

for maintainance

for Scratch

project do

level developmemt

When leave is required whom u contact and who will approve?

=>Write mail to TL, PL, PM, HR team but will be approved by PM (or) Reporting Manager

(In some places... company portals will be used for all these approvals)

Mail Server Architecture

Internet

application protocols

will be used note:: In Most of the companies the company Leave Management Tool will be for this

network protocols eg:: tcp/ip

Mail server Architecture

Outgoing Server /SMTP Server

Incoming Server (POP3/IMAP Server)

Email Account1

|--->InBox

|--->mail1

|-->mail2

Email Account2

|---> mail1 |--->mail2

Network

**Ready Made mail client (Outlook)**

**Ready Made mail client (Lotus notes)**

**SMTP :: Simple**

**internet**

**eg: http,https, smtp, imap and etc..**

**=>Network Level protocol gives set of rules and guidelines required for**

**machines interaction**

**eg:: TCP/IP, UDP**

**=>Application Level protocol gives set of rules and guidelines required for**

**the applications interaction**

**eg:: http, https, iiop, smtp, pop3 and etc...**

**=>InComing server is for recieving messages =>Outgoing server is for sending messages.**

**Mail Transfer Protocol**

**POP3: Post office**

**Protocol 3**

**IMAP: Internet Message Access Protocol**

**Email Messages**

**note:: Gmail is having its own mail server, So gmail.com is the web application cum mail client connected gmail mail server.**

**Java mail App/spring Mail App/spring boot mail app (gmail.com/hcl.com)**

**(these are programmable mail clients)**

**=>Email Account is an account created in Mail server**

**having the ability to send Email message from one Email Account to another Email account and useful to receive and manage email messages sent by others..**

**are**

**=>Mail clients either ready made clients or programmable apps use**

**POP3 or IMAP protocols to interact with Incoming server**

**are**

**=>Mail clients either ready made clients or programmable apps use SMTP protocol to interact with Outgoing server**

**Mail Clients..**

**on**

**=>Spring mail/spring boot mail provides abstraction java mail api and simplies mailing operations. => plain java mail api supports the following mail opeartions**

**->send mail with /with out attachment**

**->recieve mail**

->delete mail

*

=>mail updation is not possible

->forward mail

->replay mail

=> As of now spring mail/spring boot mail supports only send mail with /with out attachement . so for other opeations we need to use plain java mail api.

=> spring mail /spring boot mail has simplified send mail operation to trigger email message at end

of any business transaction..

eg: triggering email after credit/debit card payment /purachase

=> The moment u add spring-boot-starter-mail dependency to spring boot project ..we get JavaMailSender object through autoconfiguration pointing that mail server whose detais are specified in application.properties.

<dependency>

In pom.xml

<groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-mail</artifactId>

</dependency>

=>gmail is having its own mail server running incoming and outgoing server on

different port numbers...

smtp

server

Imap incoming

server

pop3 incoming

server

outgoing server port no :: 587

outoging server host name :: smtp.gmail.com incoming server port no :: 993

incoming server host name :: imap.gmail.com

incoming server port: 995

incoming server hostname pop.gmail.com

The block diagram of Mail Message

==================================

Cmessage headers

To, From,cc, bcc, sendDate, content Type 'subject

messag body

and etc..

The company's email account is created in company specific mail server

=> Every mail server contains mail accounts having capability to manage the email messages

Spring/spring Boot is very popular for enterprirse Apps development like e-commerce apps or banking apps.. In these apps triggering email messages after each business transaction is quiet common task.. For this spring boot mail is quiet required.

gives inputs required for the AutoConfiguration activity

(we provide mail server details here)

=>In POP3 Incoming mail server once the mail client reads the

mail messages they will be shifted mail clients ... there onwords managing the mail messages is the responsibility of mail client

=>In IMAP Incoming mail server

once the mail client reads the

mail messages they still be maintained by mail server

cc :: carbon copy (Each reciver of multiple recivers knows about other recievers) bcc :: blind carbon copy (Each reciever of multiple reciever does not know about other recievers)

having

=> Generally The company specific mail server who are less storage space use POP3 incoming server

eg: when we purchase domain/space from GoDaddy.com we get some limited email acccounts.. which are created in POP3 Incoming server of their Mail server

=> Gmail, yahoo mail and etc.. kind of commercial/large scale mail operations use Imap Incoming server generally

text content

org.sf.javamail.JavaMailSender(1)

implements

org.sf.javamail.JavaMailSenderImpl(c)

JavaMailSender obj comes spring bean through AutoConfiguration acitivity

attachment1

attachment2

Different MIME types based attachments can be there..

attachment3

Example App

=============

=> Spring mail /Spring boot mail module provides abstraction on java mail api and simplies the process sending email messages with our with out attachment

step1) spring boot starter project adding the Java Mail sender

starter

step2) collect gmail outgoing server details from internet and place them in application.propeties

step3) Inject JavaMailSender object to Service class to construct email message with

or with out attachment and perform send mail operation

service Interface

==============

package com.nt.service;

```
public interface IpurchaseOrder {
}
```

application.properties

```
#java mail properties
spring.mail.port=587
spring.mail.host=smtp.gmail.com
spring.mail.username=jauharisumit0@gmail.com
spring.mail.password=Sumit123@
spring.mail.properties.mail.smtp.starttls.enable=true
spring.mail.properties.mail.smtp.starttls.required=true
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.connectiontimeout=5000
spring.mail.properties.mail.smtp.timeout=10000
spring.mail.properties.mail.smtp.writetimeout=5000
```

```
public String purchase(String []items, double[] prices,String[] emails) throws Exception;
```

service impl class

============

```
package com.nt.service;
import java.util.Arrays;
import java.util.Date;
import javax.mail.internet.MimeMessage;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value; import org.springframework.core.io.ClassPath Resource;
import org.springframework.mail.javamail.JavaMailSender; import org.springframework.mail.javamail.MimeMessageHelper; import org.springframework.stereotype.Service;
@Service("purchaseService")
public class PurchaseOrderImpl implements IpurchaseOrder {
@Autowired
private JavaMailSender sender;
@Value("${spring.mail.username}")
private String fromEmail;
@Override
public String purchase(String[] items, double[] prices, String[] to Emails) throws Exception {
//calculate the bill amount
double billAmt=0.0;
for(double p:prices)
billAmt-billAmt+p;
String msg=Arrays.toString(items)+" with prices"+Arrays.toString(prices)+" are purchsed with BillAmount"+billAmt;
//send mail
String status=sendMail(msg, toEmails);
return msg+"---->"+status;
}
private String sendMail(String msg, String[] toEmails) throws Exception {
MimeMessage message=sender.createMimeMessage(); //empty email message
}
MimeMessageHelper helper-new MimeMessageHelper(message, true);
helper.setFrom(fromEmail);
helper.setCc(toEmails);
helper.setSubject("open it to know it");
helper.setSentDate(new Date());
helper.setText(msg);
helper.addAttachment("nit.jpg", new ClassPathResource("nit.jpg")); //place nit.jpg file src/main/resource folder
sender.send(message);
return "mail sent ";
```

**step5) develop the Client App code in main class**

```
package com.nt;
ClentApp
========
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication; import
org.springframework.context.ApplicationContext;
import org.springframework.context.ConfigurableApplicationContext;
import com.nt.service.IpurchaseOrder;
@SpringBootApplication
public class SpringBootMail01SendMailApplication {
public static void main(String[] args) {
//get IOC container
ApplicationContext ctx=SpringApplication.run(SpringBootMail01Send MailApplication.class, args);
// get Serivce class object ref
IpurchaseOrder order=ctx.getBean("purchaseService",IpurchaseOrder.class);
//invoke method
try {
String msg=order.purchase(new String[] {"shirt","trouser","watch"},
new double[] {5000,6000,7000},
new String[] {"nataraz@gmail.com","sreeni.moto@gmail.com","mranil900@gmail.com",
"prakash.er127@gmail.com","sabita11038@gmail.com"});
```

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

}

**catch(Exception e) {**

**e.printStackTrace();**

}

**//close container**

**((ConfigurableApplicationContext) ctx).close();**

}

**(from email account)**

**step6) Before running the App perform the following operations from senders email account settings**

**Enable**

two step verification

**(use manage google account)**

**c) Run the Application**

**Manage google account ----> security ----> make sure that two step verification is enabled -----> search for "AppPassword" ---> cereate App name ---> create ---> collect the password ---> remove spaces in the password ---> use it in application.properties file**

**note: use the generated password in application.properties by removing the spaces**

**spring.mail.password=nbeknsrbvyuvqmvn**

SpringBootMail01-SendMail [boot]

>Spring Elements

#src/main/java

✓ com.nt

>SpringBootMail01SendMailApplication.java

com.nt.service

>

>

PurchaseOrderImpl.java

IpurchaseOrder.java

#src/main/resources

application.properties

nit.jpg

> src/test/java

> JRE System Library [JavaSE-11]

> Maven Dependencies

>src

> target

w HELP.md

mvnw

mvnw.cmd

M pom.xml