Sending Email using Spring Boot
===============================
 Steps to develop Spring boot application to send Email

1. Create spring boot application with below dependancies
   - a. spring-boot-starter-web
   - b. spring-boot-starter-mail

2. Configure below properties
   - server.port
   - smtp properties

note: Here we are using gmail SMTP properties for practise purpose,In company they will share SMTP properites for use.

3. Enable Less Secure Apps for the mail which is configure in SMTP properties for Authentication.
   1. Enable 2 step verification
   2. Open 2 step verification page, go to the bottom and Click on App passwords.
   3. Generate app password with type as ‘Others’(copy the password and keep in applicatin.properties)

4. Create EmailService Class with the required methods to send mail(we will use Spring Provided JavaMailSender to send emails)

5. Create RestController method to accept the request(This method will call EmailService class method to send mail)

6. Run boot application and Test it.

application.properties
======================
```
#Properties to tell the mail protocol vendor
spring.mail.host=smtp.gmail.com
spring.mail.port=587

#Actual username,password of sender
spring.mail.username=username
spring.mail.password=(password generated)

#Property to trigger smtp
spring.mail.properties.mail.smtp.auth=false
spring.mail.properties.mail.smtp.starttls.enable=true
spring.mail.properties.mail.smtp.starttls.required = true
```

refer:: SpringBoot-MailApp

Microservices
=============

 Monolith Architecure

In our application,we will have several modules and several components are also available.

```
 Components are
a. Presentation Components
                 -> These are responsible for handling HttpRequests.
                 -> Webapplications means front end pages,Distrubuted applications
means RestClients.
 b. Web Components
                 -> Responsible to handle user requests.(Servlets basically
handles the Requests).
 c. Business Components
                 -> Responsible to handle the buisness logic as per the buisness
deals.
 d. Persistence Components
                 -> DAO is responsible for performing DB operations like
DML,DDL,TCL,....
 e. Integration Components(Webservices,RestFul Services)
                 -> Two projects can talk to each other only when we have
Integration logic(basically RestfulServices,WebServices,...).
 f. Authorization Components
                 -> Responsible for Authorizing the user.
 g. Notification Components
                 -> Responsible for sending email or mobile msgs notifications.

If we develop all these components as a single project then it is called as
"Monolith Architecture".

            refer : MonolithArchitecure.png

Benefits of going for Monolith Architecure
==========================================
  1. Simple for development(becoz all are available in single project): At initial
time it is very easy.
  2. Easy for Testing    : End to End Testing we can perform.
  3. Easy for deployment : One war file only we have to deploy to server.
  4. Easy for Scaling    : multiple server we can spin easily(hortizontal/vertical
scaling)
                               if requests increases then we can make keep
multiple servers through which the scaling can be done to provide
                               good responses to the client.


DisAdvantages of Monolith Architecture
=====================================
  1. Maintainence of the application:
                                   If the application is to large and complex,it is
difficult to understand.
                                         Change Request in the code is very
difficult.
                                         Some changes in the existing application
will have an impact to the other code,
                                         Lot of Impact Anaylsis will occur.

  2. Adapting to NewTechnology is required: Change in java version and adapting to
the version change also will take time.
                                                Application start up will take
more time becoz it is a fatty war file.

  3. Relaibility : If one components has a bug, it leads to application bug and
entire application will go down.
```

  4. If we make some changes to the code, we need to redeploy the entire application to the server which is time consuming.
  5. Adpating to new version of framework and releasing the quick release is very difficult.
  6. New Team members can't understand the project easily because of lack of Impact anaylis and less exposure to project.

To resolve the above mentioned disadvantages we need to opt for an Architecutre called "Microservices".

LoadBalancers and LoadBalancers Algorithms
==

-> When all components are in same applications, all request comes to same servers then burden will increase on the same server.
-> More the requests to the server, then the performance at the server side will be slow so the response time is high for clients.
            eg: youtube,myntra,flipkart,.......(laksh of users will send the request still the response time will be uniform for client)

-> when burden increased on server it will process the request slowly and some times the server might crash also.
-> To reduce the burden on the server, people use LoadBalancer for the applications.
=> LoadBalancer uses LoadBalancing algorithms to balance the load.
=> Our applications will be deployed in multiple servers and those servers will be connected to LoadBalancer.

How LoadBalancer will distribute the load?
  LoadBalancer will use LoadBalancer Algorithms to distrubute incoming request 2 servers.

Algorithms
==========

   a. Round Robin
                  => In RoundRobin fashion the request will be processed in round robin ways(1,2,3, and again it repeats)
  b. Sticky Session
                  => Based on session object details stored in the server the balancing will happen.
   c. IP Hashing
                  => By implementing one formula on the server side, serverNo=hash(IP) the balancing will be done on the LoadBalancer.

What is Microservices?
   MicroServices is not a technology or a framework or not an API.
   It is an Architectural Design pattern.(just like singleton pattern,factory design pattern,strategy design pattern)
   Microservice design pattern came into market to avoid the problems of monolithic architecture.
   If we compare the microservices architecture with the other design patterns we get to know the benefits of Microservices.

```
refer: monolitich(load balancer).png
       microservicearchitecture.png
       monolitich vs mircoservices.png
        SAGE pattern.png
```

What are the challenges which will be faced by the developers if we are following microservices design pattern in our project?

a. Bounded context
b. Lot of configuration
c. Less visibility
d. Pack of Cards Problem