Boot Spring Cloud

(working with MicroServices)

Monolithic Apps Vs

SOA Apps Vs MicrorServices Apps

Monolithic Apps /Monolith Apps

==========

====

that

In spring Boot the Micro service archicture Apps

will be implemented with support of

spring Cloud Module and multiple other facilities

=> Generally we do not prefer using spring framework for MicroService implemenations .. we prefer spring boot a lot in the angel of spring boot cloud to implement MicroServices..

=>The Application packs multiple services as multiple modules in a single unit (nothing but project) either as war file or jar file or ear file is called Monolithic Application/Project.

1 application = 1 Project contains multiple services as muitiple module s inside the project/Application

=>So far the spring MVC Apps, spring data jpa Apps, spring core Apps and etc.. we developed are called Monolithic Apps

=> Even spring MVC with sprign data JPA /spring ORM/spirng JDBC that we have developed so far (except Spring Rest Apps) are called Monolithic Apps..

Monolithic App /Project

service/module#1

service/module#2

eg::

The popular architectures in java domain are a) Monolith Arch

b) SOA (Service Oriented Arch) c) MicroServices Arch

we can place MVC layers

in all these architetures based Projects development

The apps that we have developed so far having MVC architecture are called as Monolith Architecture Apps (All the mini Projects that we have developed as Layered App is called Monolith Arch Project)

Nareshlt.com, Flipkart.com, Banking Apps and etc.. Nareshlt.com (Monolithic App:)

service/module.3

service/module#4

Admisions service/module Batches module

(.war/.jar / ear)

Accounts module

payroll module

getting

=>Monolith Apps are like meal as single unit/packet having multiple itemsin it

=>each feature is called onera /.ear)

=>One module can have 1 or more services

The MVC architecture Apps whose multiple services are combined to single jar /war file is called Monolith Architecture Apps

Build Process/Building the Application

==========&F====

========

.war ==>web application archieve

.jar ===> standalone App (Non-web applicaiton) / standalone web application

(best) ear file = jar file + war file +..... (or)

ear file => jar file + jar file +jar file .... (or) ear file => war file +war file +war file

=>The process keeping the app ready fo'execution in certain env.. is called build process

*.java ----> *.class

.properties files

other files

Liparies (jar files) (apis)

war file/jar file

development called

(Using Embedded Server)

ear file :: enterprise App

archieve

(now ear files not popular becoz web services comps also coming as war files.. When EJB was there

ear files bit popluar to represent web applicaiton(war) + ejb comp (jar) togather as ear file)

The code that completes the process is stable code (no compile time errors) and very much ready for execution

of

=> In buid process, developmentn of source code, compilaation of source code, development helper operations resources like properties files and other files, adding api libaries and etc.. will be there .. At last the war file or jar file will be created reprsenting the whole project.

between

Java

What is the difference App /Service or API?

=>API --> application Programming Interface .. It is base for programmer to devleop certain language or technology or framework Apps. In Java api comes in the form of pkgs having classes,interfaces, enums and annotations.. These apis will be released /available in the form jar files (libaries/dependencies) (These are

also called as Progamming apis)

3 types of APIs :: a) pre-defifned apis (given by technology/language/framework vendor) b) user-defined apis (given by developer)

c) third party apis (given by third party vendor)

=>Application/Service is the outcome of build process either in the form of jar file or war file having certain to task to perform.

of

note:: In the development App /Services in certain language/technology /framework we take the support APIs. (Libraries)

MVC layers will be there in all the 3 architecture projects

a) Monolith Arch --> the layers of all the services/modules

b) SOA ---->

will be combined togather in to single unit

3 comps will be there in the app (a) service provider b) Service Consumer c) Service Registry note:: The MVC layers will be there in Service Consumer and Service Provider

c) MicroServices arch --> Every Service is a separate Project having the layers all these projects will be integrated

These are APIs are language/technology/framework supplied apis acting as base to develope the Apps /services (These are also called as Libraries)

note:: The existing APIS/libraries can be used to create new APIs/Libaries or projects/Apps/sevices

Example Java APIs

lang api utility api jdbc api

servlet api jsp api

=> servlet api is used to develop servlet comps (web comps generating web pages)

and etc...

eg: we can use hibernate api directly to develop hibernate persinstece logic /hibernate App the spring creators have used the same hibernate api to develop spring ORM apis and spring data jpa apis

=>Jar file purpose will change context to context..

jar repsents apis (Libraries) (eg:: servlet-api.jar)

jar represents jdbc driver s/w (ojdbc6.jar)

=> jdbc api is used to develop jdbc code/Apps (persistence logic)

API can be used to create the apps /services/projects and the same API can be used to develop the other APIs

to

note:: The Language/technology/framework APIs are no way related Restful web services APIs note :: Restful web service APIS represent Distributed Apps that developed using Restful web services

jar represents standalone web application (that uses Embedded server) jar represents ejb comp

(app1.jar)

jar represents web application (war file) (war file is extension of jar file) jar represents standalone App/project

and etc...

Deployment/Installation

==========

note:: The language/techlogoy/framework APIs are given as the libraries to develop the software apps/projects/services

note:: Restful web services APIs are Distributed Apps using which we can go for app to app interaction or the service provider apps can be accessed from different types of Local/remote Consumer Apps

=>The process of keeping web application (.war file) in the server (either that is running or later started) is called deployment..

web server/Application server (Like Tomcat)

=> JAVA apis are libraries given by java language/technology/framework eg:: JDBC API, Servlet API, JNDI API and etc.. (Programming APIs)

=> Restful webservice APIs are Distributed Comps having business operations as the Endpoints eg: Weather Comp, ICC Score comp, PhonePe Comp and etc.. (Rest APIs)

=>The process of removing the app from the server is called undeployment

web application .war file

Deployment/Installation..

=>if we deploy the web application when the server is in running mode then it is called HOT deployment

=>if we deploy the web application when the server is in stopped mode then it is called COLD deployment

note:: In the devleopment /Testing env.. deployment takes place in Local servers or s/w companies cloud account (aws/gcp/azure and etc..) or in the embedded servers

note: In the UAT and Production the deployment takes place in the Client Org Servers or Client org's Cloud Account

How build Process takes place in Realtime companies?

Ans) using Maven/gradle Tool In combination with Jenkins CI/CD configration

UAT:: Use-Acceptence Test

(The test that happens at client organization side after release of the Project)

CI: Continous intergration

CD :: Continous Deployment /Delivery

How do we pack app/service/project in to single unit having entire env.. for execution?

=>Only Code packing -----> war /jar file => Envirormment packing ---> docker image

enviroment

(code(war/jar) + server + DB + OS + ...)

Using Docker tool we can pack code +env..

Service Instance / App Instance /instance

=====

note:: using maven/gradle tool the jar/war file will be generated where as the using Jenkins tools jar /war file generation and its deployment

to the server can be automated for every

change that is done in source code or at regular intervals

Using Docker our app code becomes more portable to execute in different machines becoz docker maintains both app code + env.. (containerization)

=>A runnig Application inside the server giving services to Clients (either for end users or for other Apps)

is callled Service Instance /instance/App instance..

=> Each copy of app(jar or war) that is deployed in server successfully to render services for clients is called

is called service Instance..

App/Project ---> is like class

each deployed App/Project giving services to clients is like instance

note:: One class can have multiple instances.. simpliarly one app/project

can deployed in multiple servers i.e can have multiple instances/Service instances

nareshIT web application (App /Project) => (complted devleopment/testing) (nit.war)

client1

request1

WebServer/Application Server

(tomcat of machine1)

(instance1 of applicaiton)

response1 request2

nit.war (deloyed)

we can deploy the

the multiple instances /sevice instances of

one App/Project

in different servers of same machine or different meachiens

client2

response2

WebServer/Application Server

client1

request1

(weblogic of machine2)

(instance2 of applicaiton)

nit.war

response1 request2

(deloyed)

client2

response2

instance at a time

is

In Every server we can specify max no.f requests that each service can allow, In most of the server this max

request count 200 by default

In spring boot MVC or spring Rest Apps we can change /control this max request count using the following properties

of application.properties file

In application.properties

are

server.tomcat.threads.max= 150 (default is 200) server.jetty.threads.max= 300 (default is 200)

(Both these propperties related to Embedded servers of spring boot app)

Load Count/Current Load

==========

=> The no.of requests that are currently under proccessing by service instance is called Load Count/Current Load.

=> if the App1 instance/service instance is currently processing 10 requests then

the Load count /Current Load is 10.

Load Factor

=>

it is current Load/Max Load

=> Load factor courrent Load/max Load

to

⇒ if App instance with respect server max load is 150 and that app

instance is processing only 100 request currently then 100/150 is the Load Factor (0.666)

Load Factor is always >=0 to <=1

(between 0 to 1)

maxLoad = max requests that server can take for each instance of the Application at a time we can specify this using server settings or using application.properties while working with embedded servers of spring boot.

Scaling

==========

the

In extenal tomcat server

=> Increasing the service capacity of App/Service /Project is called Scaling..

=> if the App is having facility to increase its service capacity as needed then it called scalable App.. (scalability feture)

Examples :: eg1:: increasing shop capacity from 1000 square feet to 10000 square feet (vertical scaling)

eg2:: keeping same shop of 1000square feet in 10 places. (Horizontal scaling) →

eg3:: increasing no.of ATM machies in different locations (Horizontal scaling)

eg4:: Increasing more no.of ATM machines in a single room/location (Vertial scaling) /

Two types of Scaling

==========

a) Horizontal Scaling (good)

b) Vertical Scaling (not recomanded)

a) Horizontal Scaling

<Tomcat_home>\conf\server.xml file and use

<Executor name="tomcatThreadPool" namePrefix="catalina-exec-"

maxThreads="150" minSpareThreads="4"/>

of

In Horizontal scaling we take the multiple copies same App i.e multiple service Instances to give services to Clients.. and we control these multiple instances with the support of Load Balancer Service (LBS)

client1

request1 (thread1)

LoadBalancerServer

(LBS)

WebServer /AppServer (we can take muliple webservers or App servers)

ID Load Factor 1122 80/150

nit.war (instance1)

Id: 1122

note:: most of times, we keep

113390/150

nit.war (instance2) Id: 1133

the mulitple service instances of

one app in multiple different copies of

same server software running on different machines

Client?

These instances

nit.war (instance3)

can be there either

request2 (thread2)

1144 100/150

Id: 1144

in same server different domains

or different copies of same server or different servers residing in the same machine or different machines,

(eg:: HttpLoad Balacer)

=> The given request goes to that instance/service instance of the App whose LoadFactor is less (whose value is nearer 0) .. In our example the both req1,req2 will go to instance beoz it is having less load factor (80/150). if all the instances are having same load factor then

the LBS will pick up the instance randomly.. |-->Load Banlancer Server/Service

Examples for different LoadBalancers

==========

• HAProxy - A TCP load balancer.

• NGINX - A http load balancer with SSL termination

support....

mod_athena Apache based http load balancer.

• Varnish - A reverse proxy based load balancer.

• Balance - Open source TCP load balancer.

• LVS - Linux virtual server offering layer 4 load balancing.

and etc..

Keeping gmail App in different zones of world map

falls under horizontal scaling.

1 instance in singapore, 1 instance in dubai, 1 instance sydney

and etc..

Either DevOps team or Infrastucture team

like Linux admin and etc.. will take care

of this kind Load Balancer..

b) Vertical Scaling

=> Here we add more software or hardware infrastructure for the existing App env..

to make it ready for taking more requests from more clients.

client/browser

of

Machine1 webServer

nit.war

existing setup

16GBRAM

new setup

32GB RAM

1 TB HDD

dual coreCPU

Tomcat

mysql

10TB HDD quard core CPU Wildfly oracle DB

In Cloud env.. like AWS

max of 720 GB RAM

and unlimited HDD suport

can be taken..

(moving to new setup from the

existing setup is called vertical scaling)

Pros and cons Monolithic architecture based Applicaiton Development

pros (advantages)

=====

===

=> Provides simple and easy env.. to develop Projects/Apps as the Layered Apps having

layers like presnetation layer, controller layer, service Layer, Persistence Layer(DAO), Integration layer and etc..

note: Most of the current maintainance projects are Monolithic projects

=> Easy to deploy (all togather single jar/war file), easy to manage (single war/jar file), easy to scale (increasting or decresing

instances)

=> Performing unit Testing is very easy becoz all layers and all services are avaiable togather

=> Less possiblity of getting network related issues becoz all services /layers mostly there togather.

=> Performing logging and dubegging operations is very easy.. (Even Adutiing activities are easy)

cons (DisAvatages)

====

to

=> For small or tiny changes in one or two services /modules/layers we need redeploy the whole application by alterting the endusers when App in the production. (App will not for thosemany hours)

=> one bug or issue in one module/service of Project/App may effect other services /modules ..this indicates

these Apps are not reliable. (Sometimes Apps will shutdown providing no services to clients)

(To be continued ....)