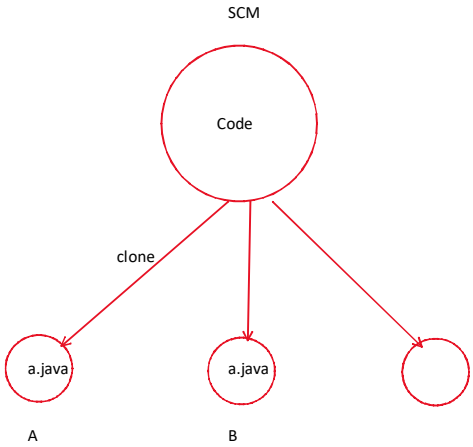
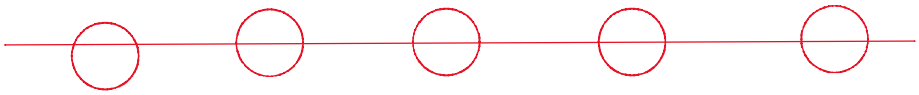
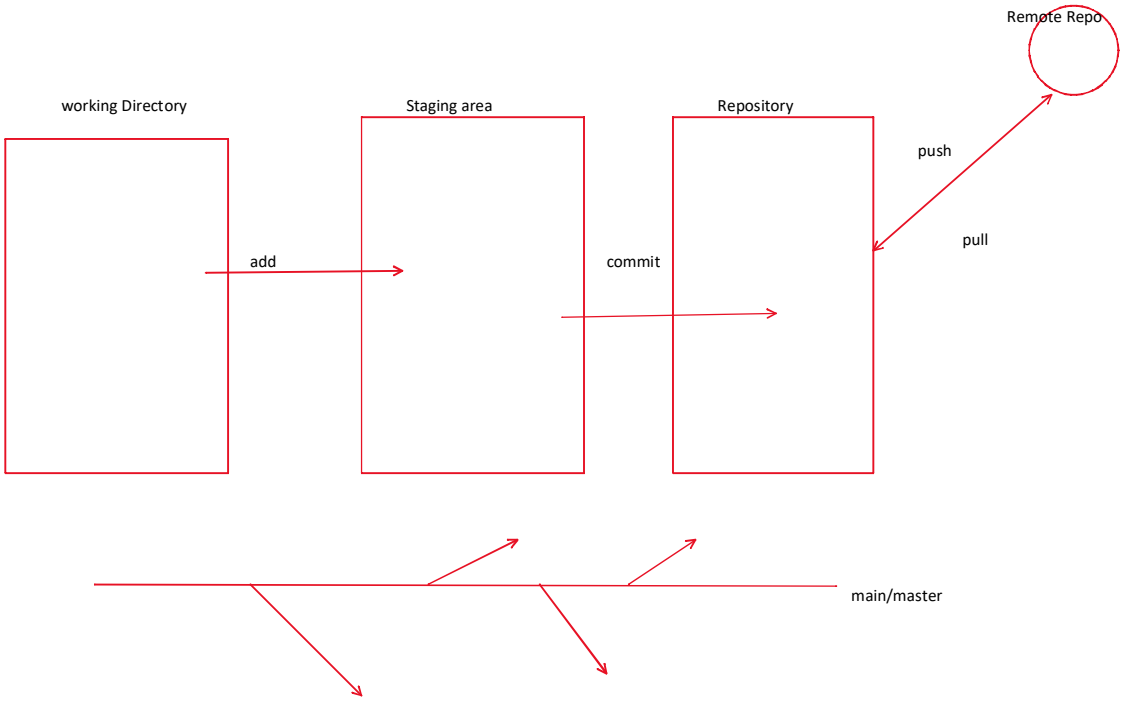
Sapient Java Spring

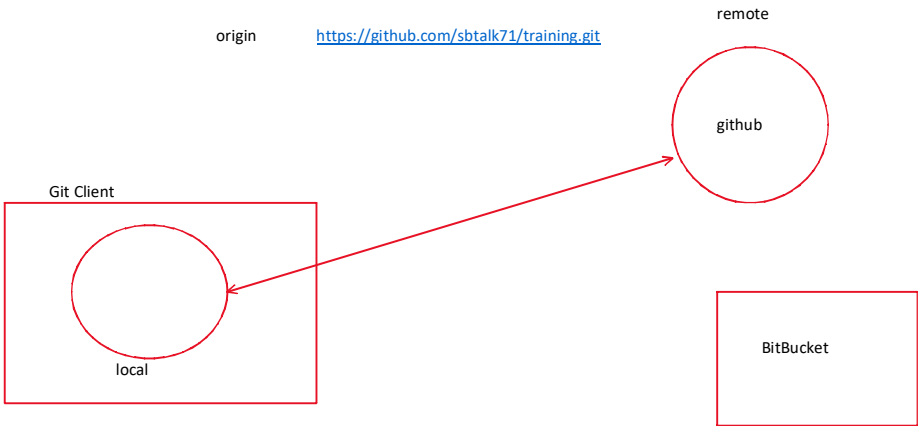
06 June 2022 09:16







mkdir --make directory
ls list contents of a directory



git init
git status
git commit
git add
git remote
git push

1. Create a repository "training" on GitHub
2. clone the repository locally in 2 locations user "bob" and "alice"
3. With bob
   a. create a file "file1.txt" add some content
   b. commit and push
4. With Alice
   a. create a file "file2.txt" add some content
   b. commit and push
5. Pull the changes into Bob and bob adds one line to file2.txt and commits it to remote
6. Pull the changes in Alice .....

Steps:
clone repository : git clone <clone uri>

Add files : git add ./ git add <file_name>/git add --all

Commit : git commit -m"commit message"

Push : git push <remote_name> git_branch

e.g. git push origin master

example

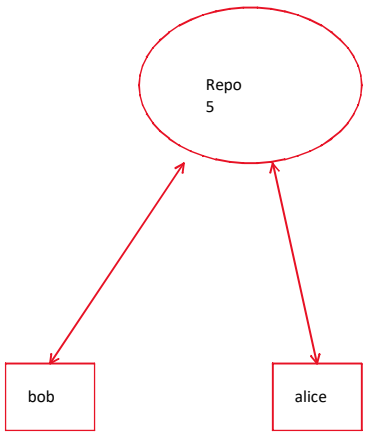
git clone https://github.com/sbtalk71/training.git

git push origin master

git push https://github.com/sbtalk71/training.git master

myrepo=https://github.com/sbtalk71/training.git

origin = https://github.com/sbtalk71/training.git



JAVA_HOME=c:\opt\java\jdk-17.0.3.1 $JAVA_HOME

PATH=%JAVA_HOME%\bin

name='tiger'

print(name)

NAME=Tiger

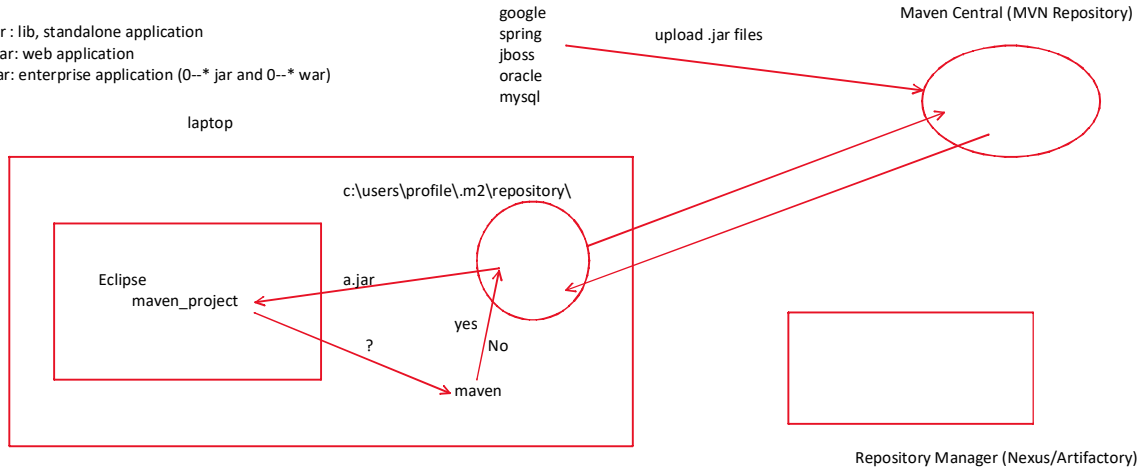
%NAME%

Integrated Development Environment (IDE)

Eclipse
VSCode
Intellij IDEA

Java

jar : lib, standalone application
war: web application
ear: enterprise application (0--* jar and 0--* war)



company domain: play.com
project: game-slot-1
version: 2.0.1
Packaging: jar

com.play:game-slot-1:2.0.1:jar

.jar ---> Base Folder
|
META-INF
\
com/demo/java
demo.class

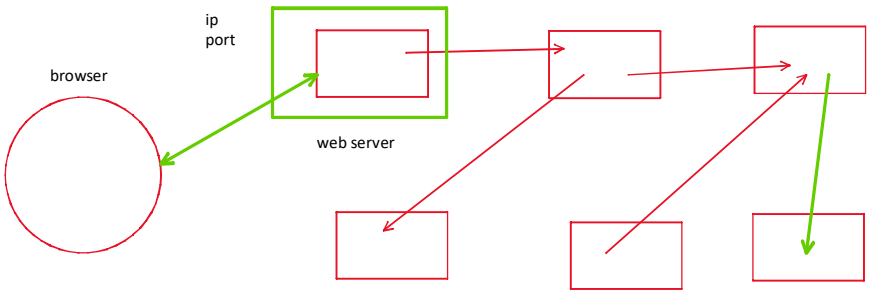
.war file structure

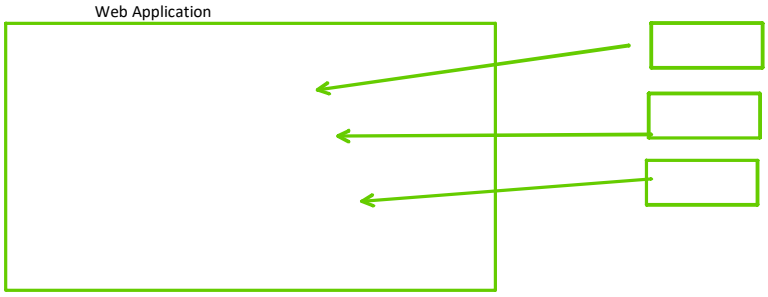
web projects---->context_root (webapp)
\
pages
|
demo.jsp
|
META-INF
|
WEB-INF
|
classes
|
com/demo/servlet.class
|
lib
|
web.xml

Apache Web Server
Nginx

java==> Apache Tomcat

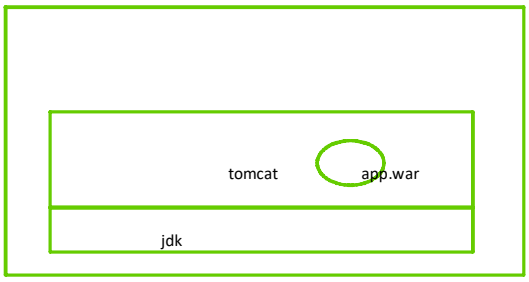
www

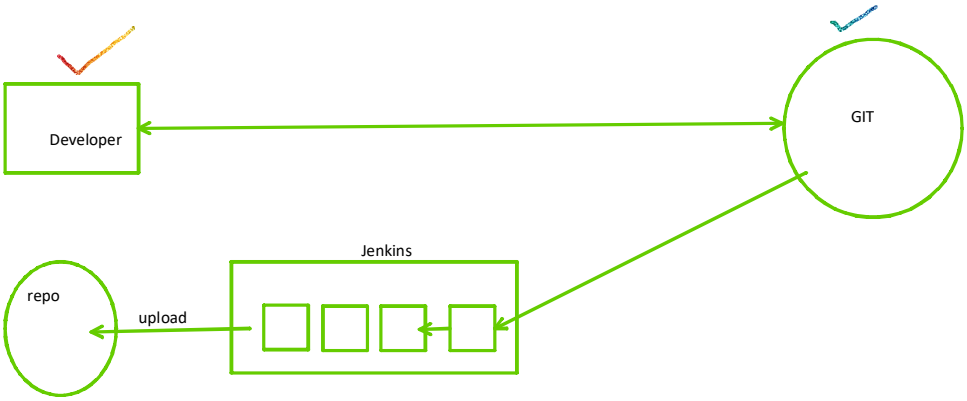


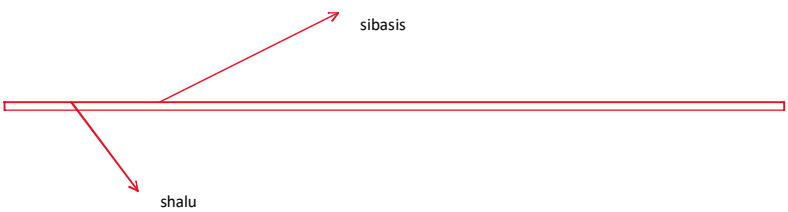


scopes in pom.xml

compile : available always
runtime: available only at runtime
provided: will be provided by target server/runtime
test: the dependency available only in test phase
system:







git push origin shalu

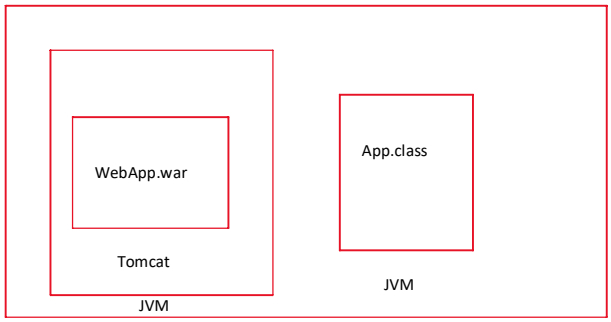
git push origin sibasis

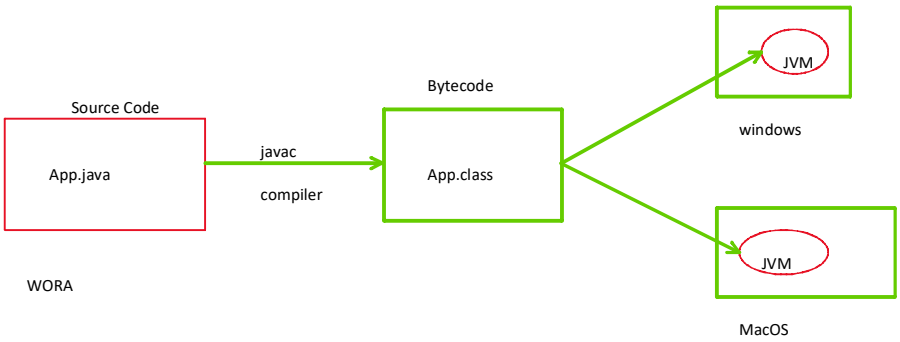
com.demo.java

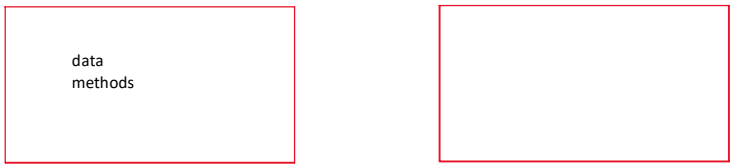
Windows: com\demo\java

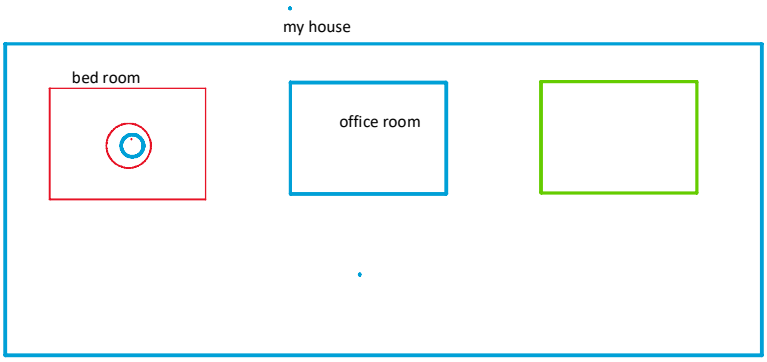
*X: com/demo/java

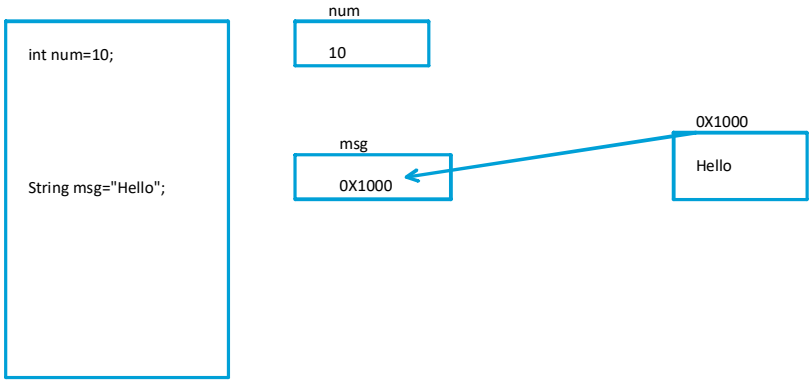
platform

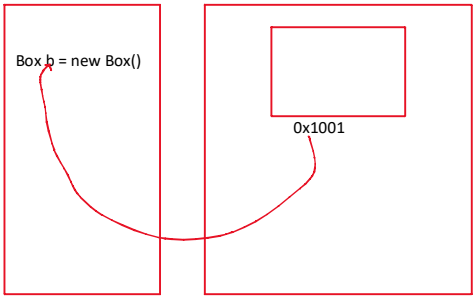

WebApp.war
Tomcat
JVM
App.class
JVM

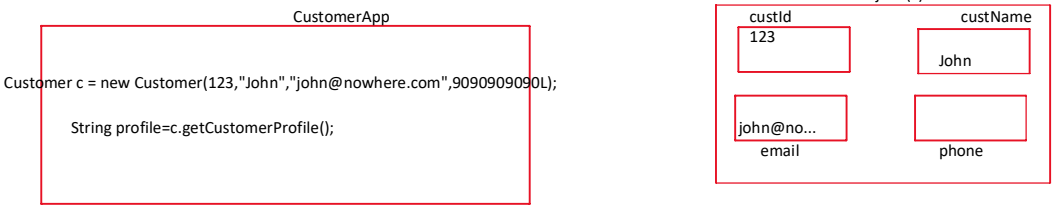

Source Code
App.java
javac
compiler
Bytecode
App.class
JVM
windows
JVM
MacOS
WORA

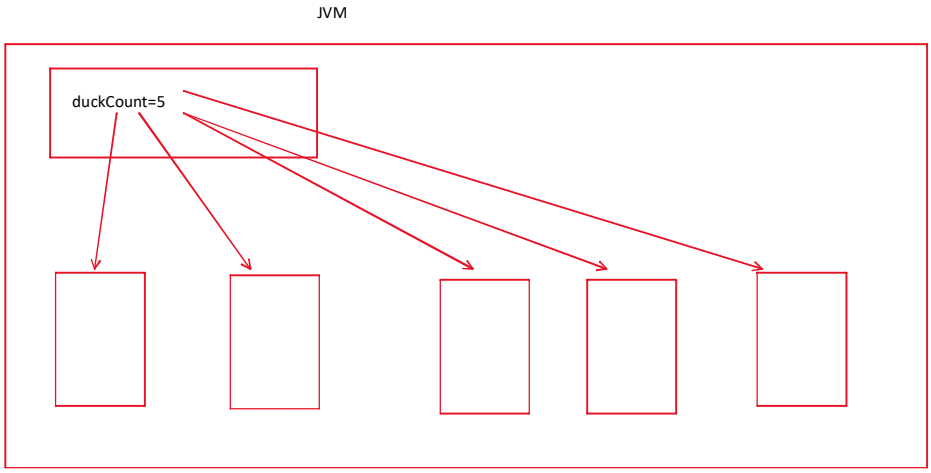

data
methods


my house
bed room
office room

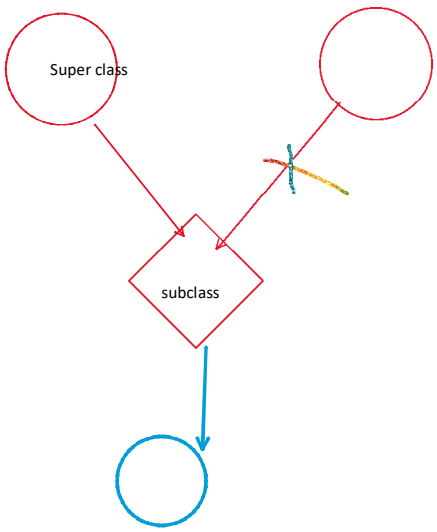

int num=10;
String msg="Hello";
num
10
msg
0X1000
0X1000
Hello
heap

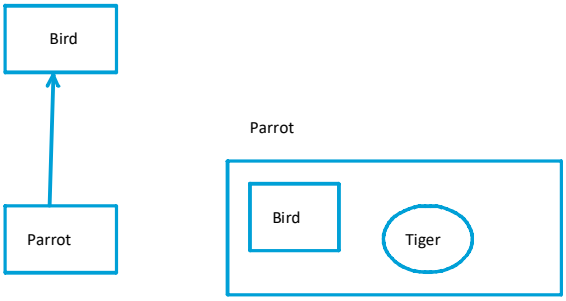






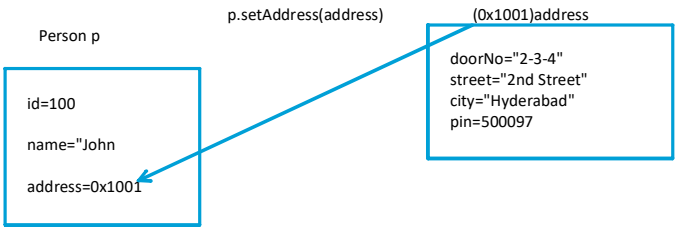
Encapsulation: access specifiers and packages
Polymorphism: method overloading(???),method Overriding, Dynamic Nature of Java (RTTI)





Bird Animal Fruit Vehicle

final class
final method
final field (variable)



Mail

String toAddress;
String fromAddress;
Message message;

public void compose(String body, String heading){

Message m= new Message(body, heading);
this.message=m;

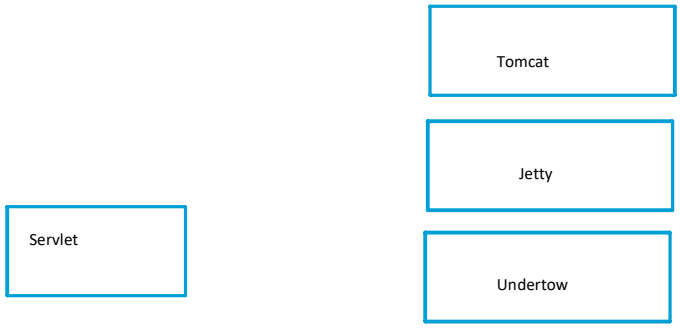
}

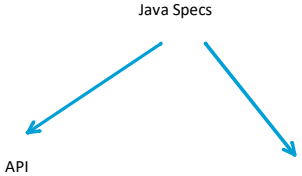
public void send(){

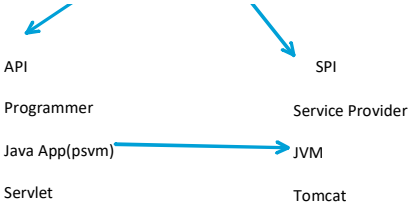
Sysout(toAddress+" "+from Address+" "+message;
//print mail details

}

```
class Message{
private String body;
private String heading;


}
```



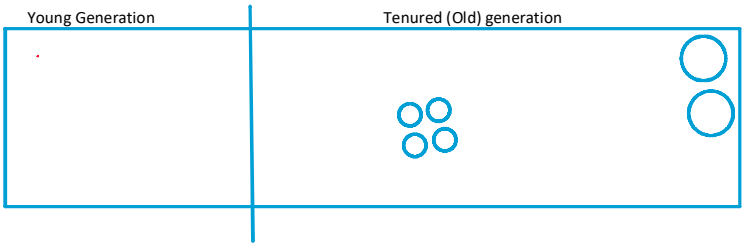


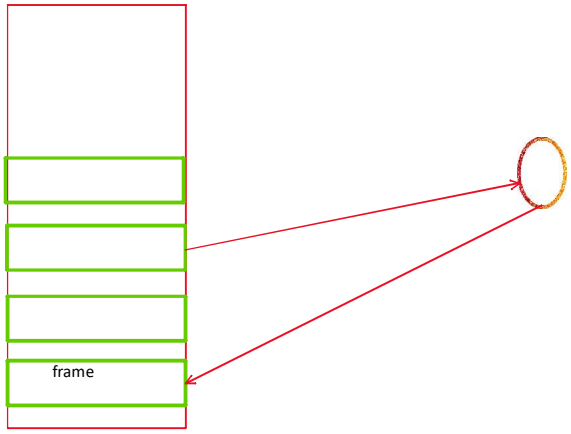


Java Memory Model

Heap---> Where we store java Objects

Stack--> Where Our threads execute (each thread has their own stack)





Person p = new Person();

int x= 10;

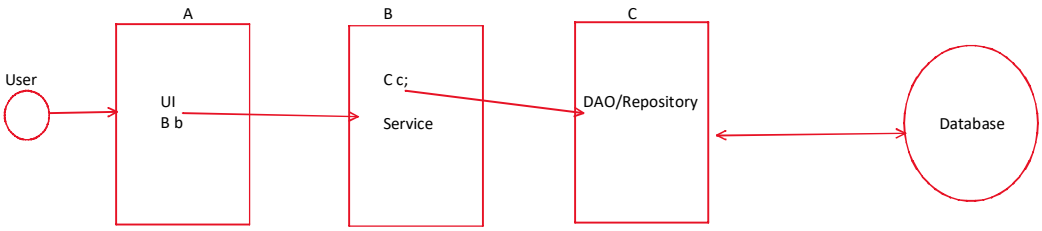
int x;
Person p2;

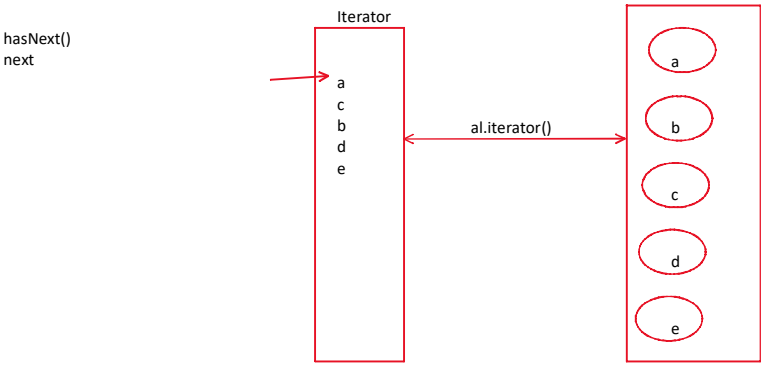
IF checked exception: class MyException extends Exception{

}

if Unchecked Exception :
class MyException extends RuntimeException{

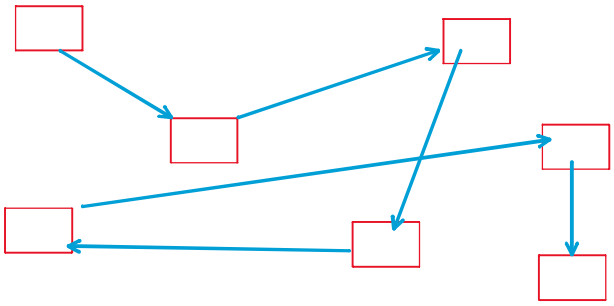
}

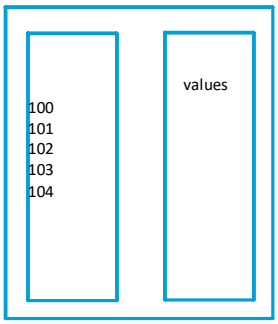


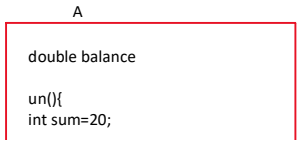


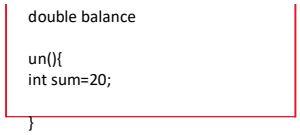
ArrayList

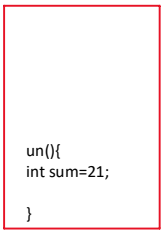
LInkedList

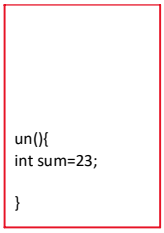




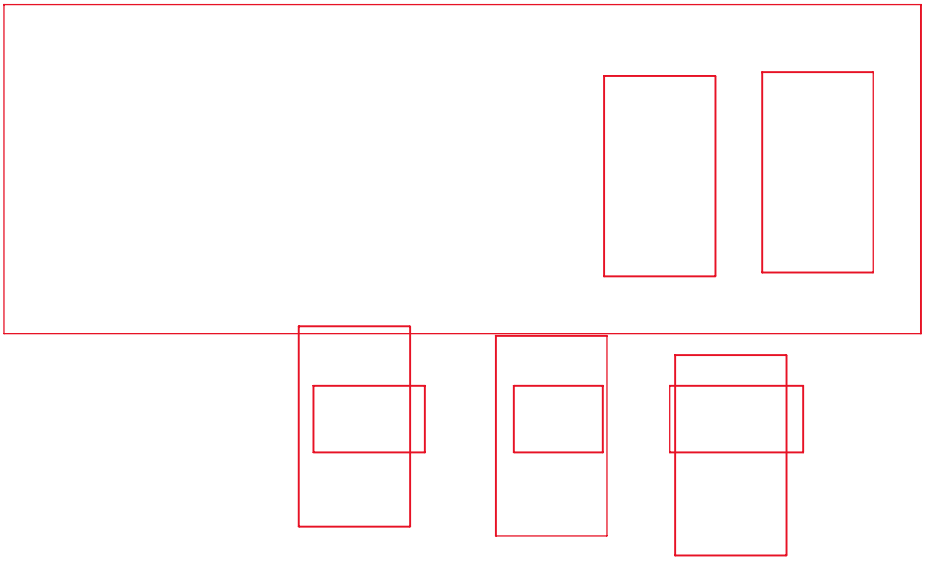


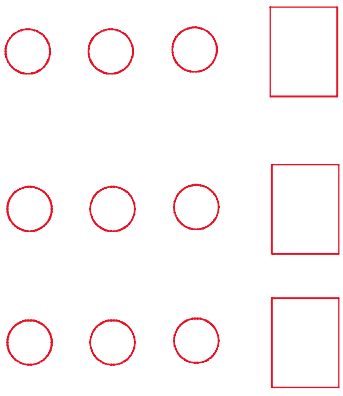






network based code
Database related code
IO Based code
Your code (Optional)



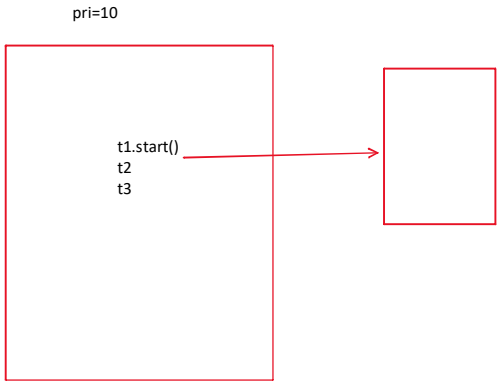


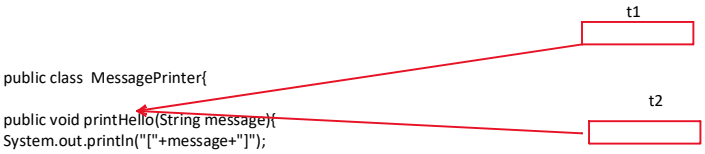
java.lang.Thread (class)

public void run()

java.lang.Runnable (interface)

public void run()






```
public class  MessagePrinter{

public void printHello(String message){
System.out.println("["+message+"]");
}

}
```

```
class A{
B b;

A(B b){
this.b=b;

}


}
```

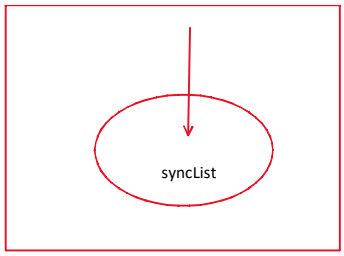
```
class B{

}
```

```
    B b = new B();
```

Synchronization

wait
notify

CA Hore



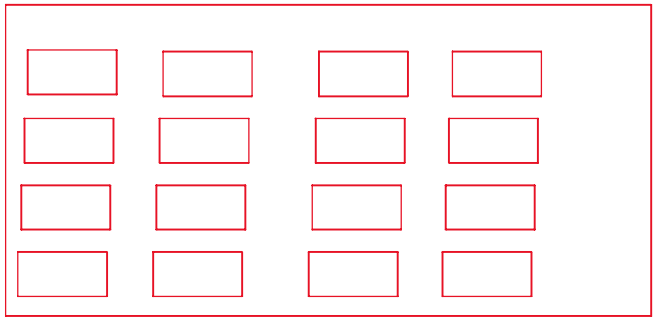
lock

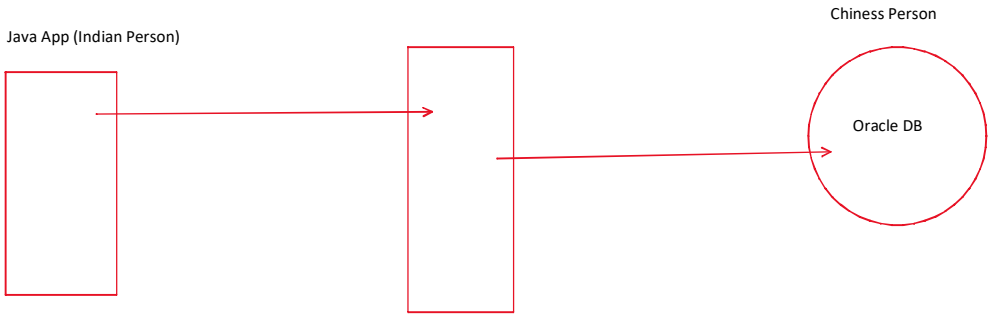
monitor

Thread safe

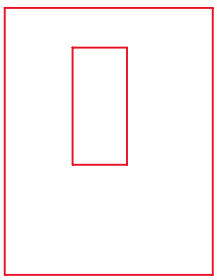
Nested Class:

non-static : inner class



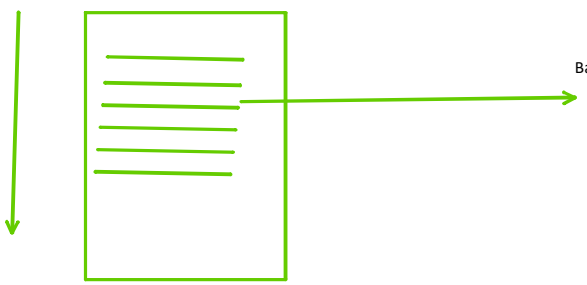


Thread Pool



Executor Framework

Synchronous
Asynchronous



Thread status
Response from Thread

Future<?>

Mathematics:

f(x)=x+5

f(2)=2+5=7
f(5)=5+5=10


```
public int ad(int a, int b){

int c =a+b;

return  c;

}
```

```
public int ad(int a, int b){

return a+b;

}
```

```
public int addTo1(int a){

a=a+1;
return a;

}
```

```
for(int x=0;i<10;i++){

}
```

f(x)=x+5

```
public int addTo5(int x){

return x+5;

}
```

functionalInterface

```
public int add(int a, int b){

return a+b;
}
```

add(int a, int b)=a+b

lambda

(a,b)->a+b

Runnable:

()->SYsout

```
public interface Adder{

public int add(int a, int b);
}
```

f()

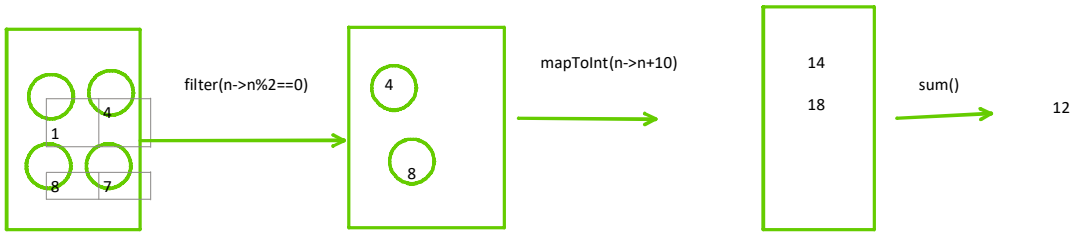
public T f() --Supplier ()->"Hello"

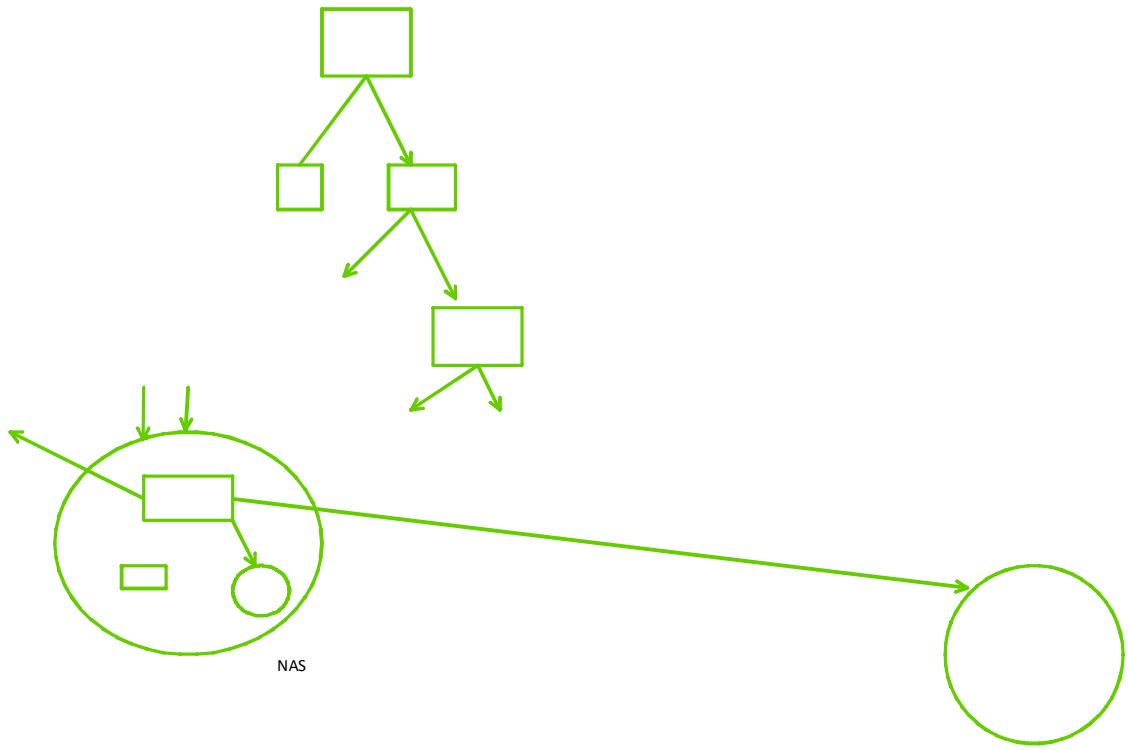
public T f(V) -- Function

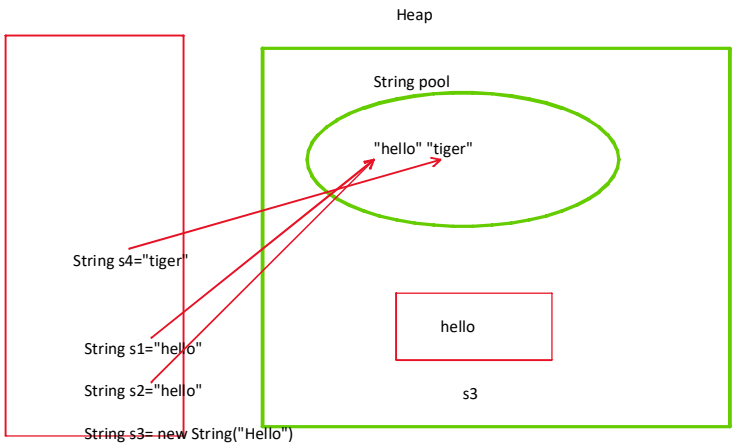
public Boolean f(v) --Predicate

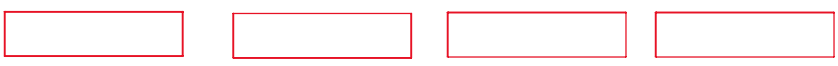
public void f(V) --Consumer

Stream


filter(n->n%2==0)
mapToInt(n->n+10)
sum()
12
14
18
4
1
8
7
4
8


NAS


Heap
String pool
"hello" "tiger"
String s4="tiger"
hello
String s1="hello"
String s2="hello"
s3
String s3= new String("Hello")

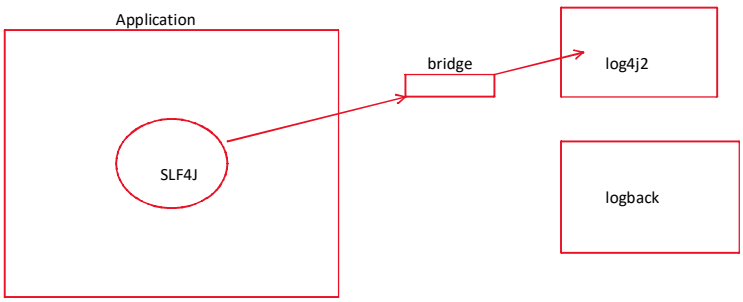


Logging Requirement:
Logger/Logging Implementation
The Corresponding Appender File Console and File

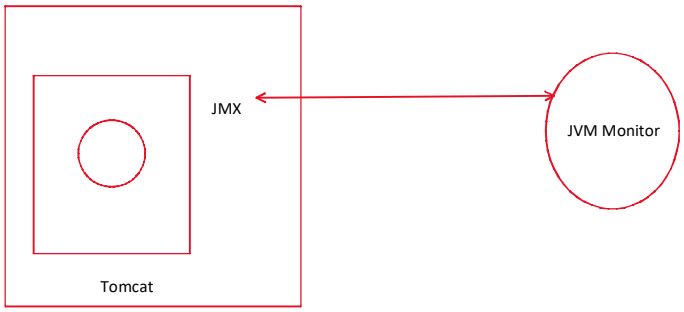
1. Log4j
2. Log4j2
3. Logback
4. SLF4J
5. java.util.logging

Logging Levels:

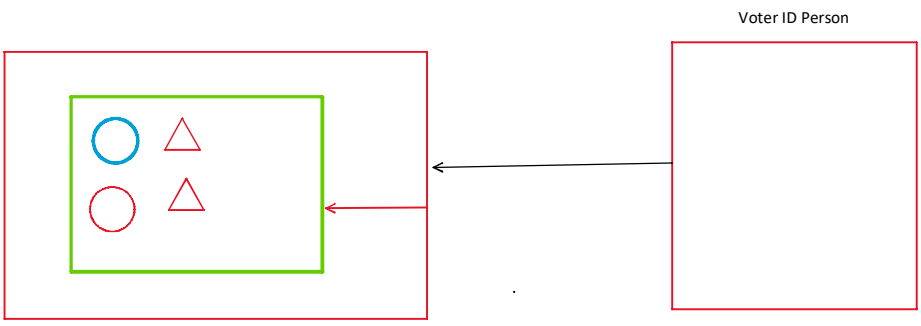
INFO
DEBUG
SEVERE



Weblogic
Websphere
Apache Tomcat
Jetty



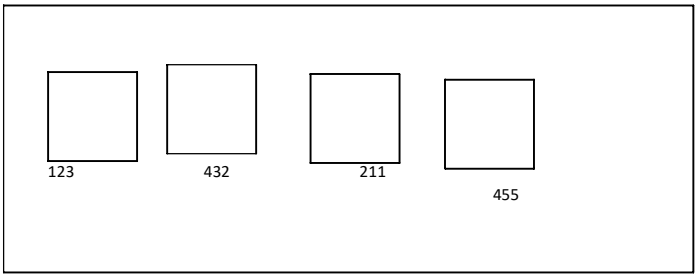
JVM



Monitoring and Profiling Tool for JVM

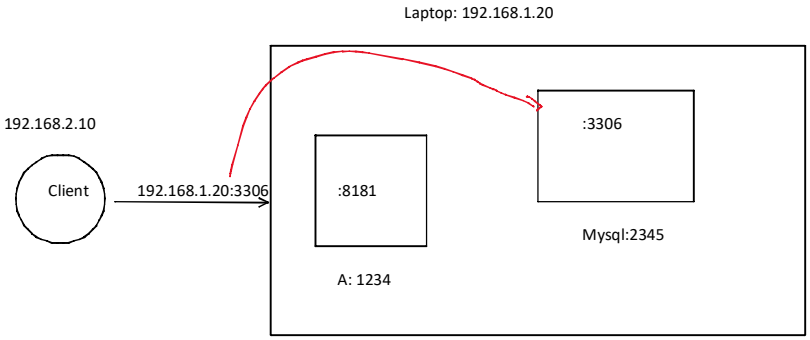
1. Jprofiler
2. Yourkit
3. JMC (Java Mission Control)
4. VisulaVM (Free)

Heap ---Objects
Stack -->Thread--->method calls--CPU Load

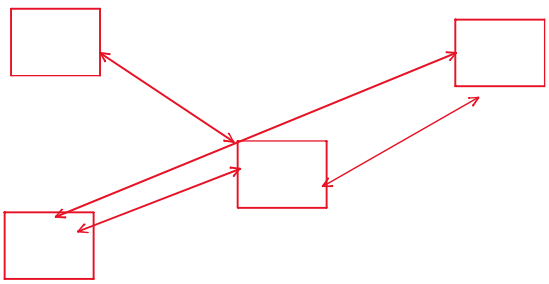


JMX
Heap Profiling
CPU Profiling
Profiling tools for Java app

IP address IPV4 X.X.X.X 255.255.255.255



localhost 127.0.0.1



8-12

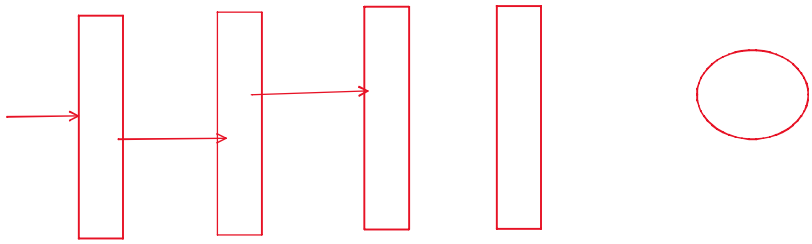
Pattern

Cycle Factory

Memory
Processor
PowerSupply
Motherboard
HDD

```
applyLoan(){
    checkCredit()
    CheckBackgroud();
    approve()
    f3()
    f4()
    }
```

Optional<T>



```
class A{
B b;
f1(){
b=new B();
b.f2()
}
}
```

```
class B{
C c;

f2(){
c=new C();
c.f3();
}


}
```

```
class C{

f3(){

}


}
```

```
class Invoker{
Command cmd;

on(){
cmd.execute();
}


}
```

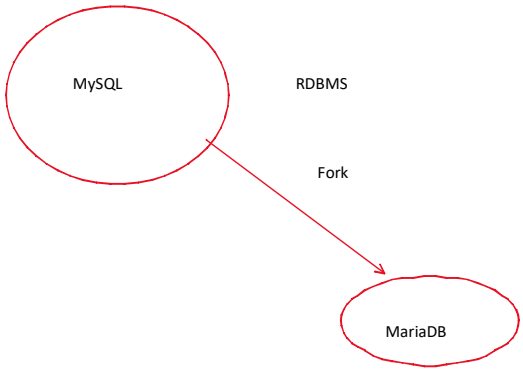
```
class Command{
Receiver rec

execute(){
rec.switchOn();
}


}
```

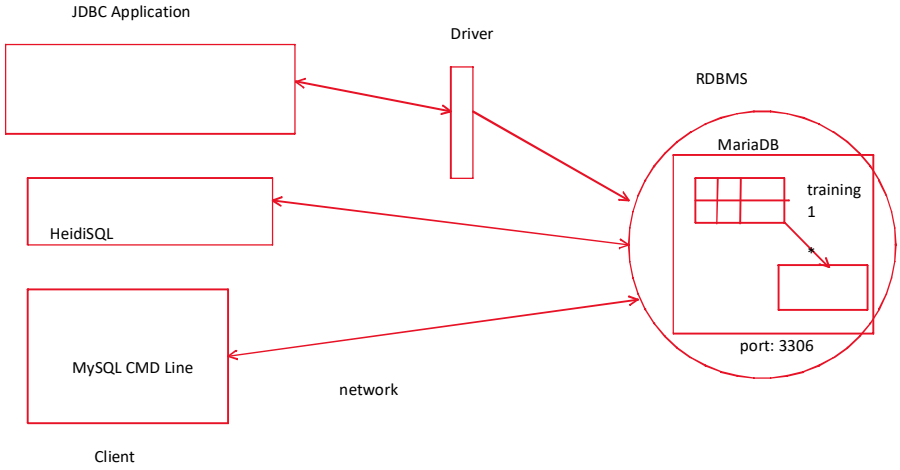
```
class Receiver{

switchOn()


}
```

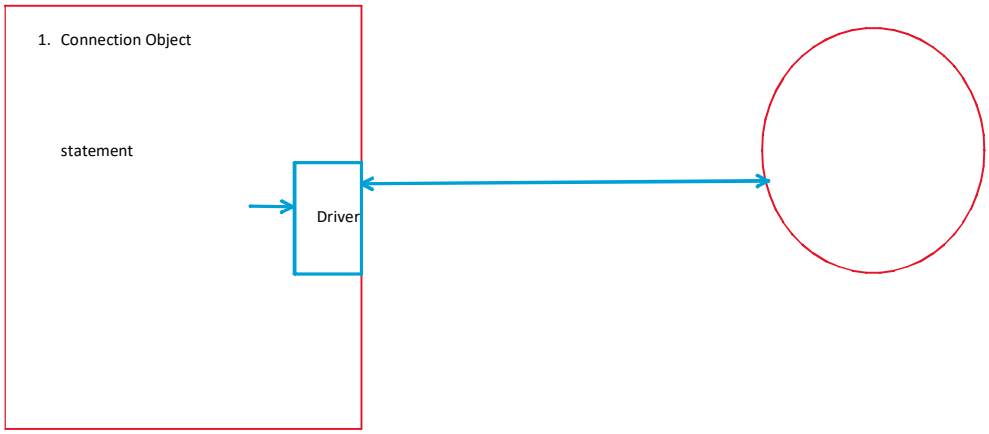
Test Fixture
Test Suite
set Up — beforeEach/beforeAll
Tear down — afterEach/afterAll

```
public class Counter{

private int counter=0;

public void increment(){
count++;
}
```

```
public void decrement(){

count--;
}

public int getCounter(){
return counter;
}
}
```



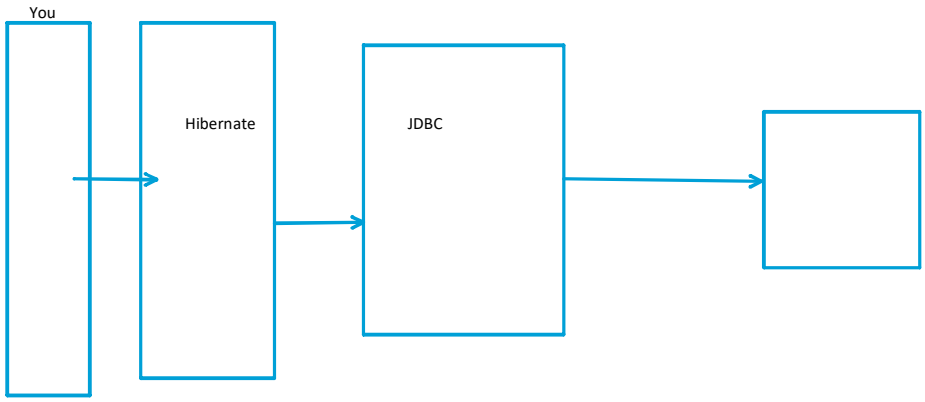


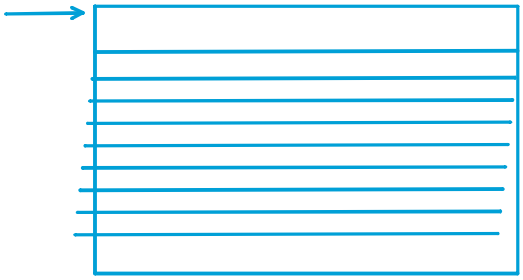


Transactions

Local --> What ever strategy the DB supports locally--In Java it is JDBC Transaction

Global--> spans across more than one resource, distributed





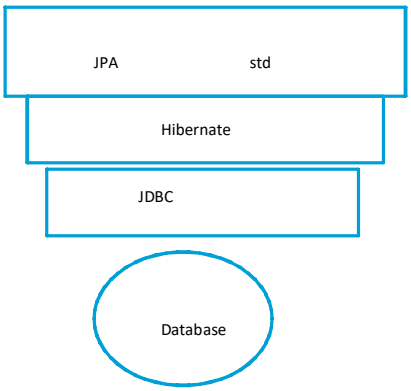
javax and jakarta

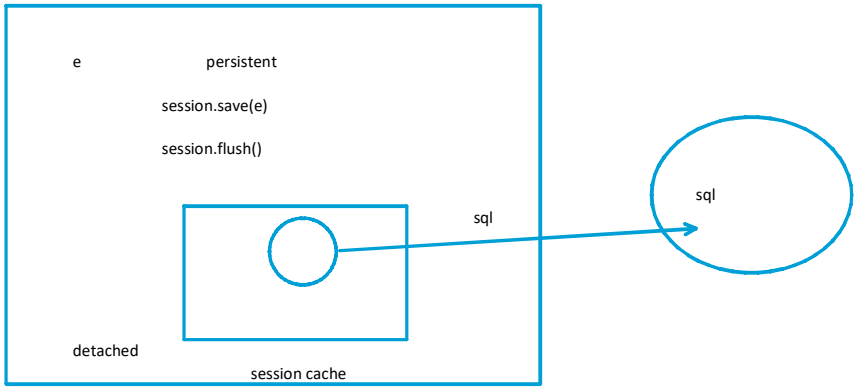
Sun Microsystems <--- Oracle

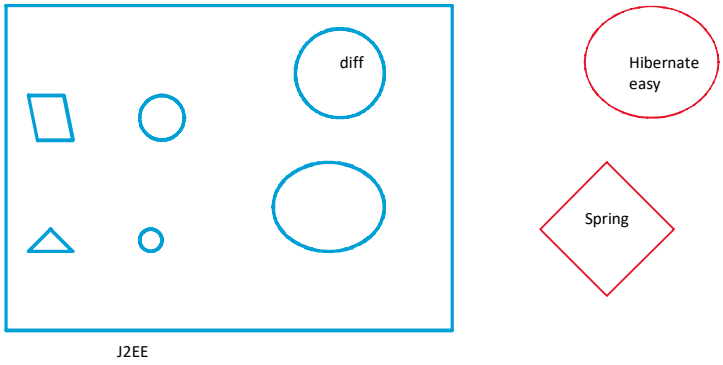
j2EE			J2EE and Java EE 8

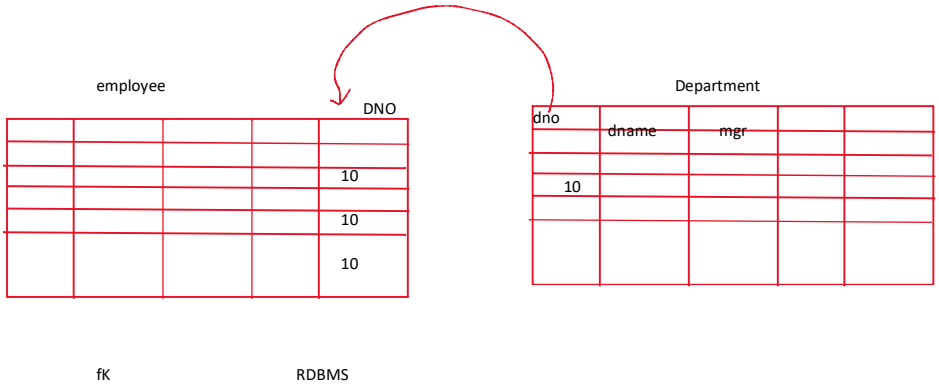
---> Eclipse Foundation (JakartaEE)

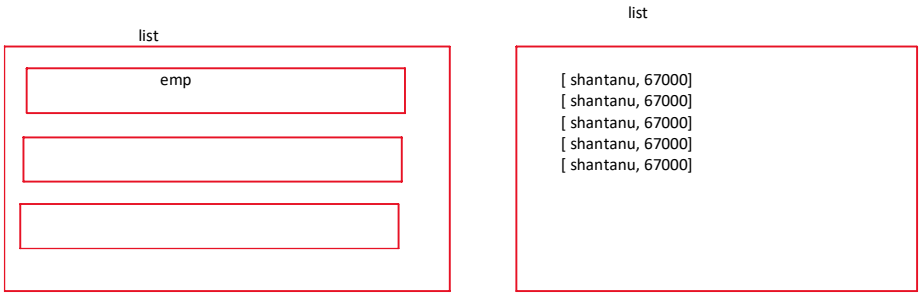
javax --->jakarta




e
persistent
session.save(e)
session.flush()
sql
sql
detached
session cache


diff
Hibernate
easy
Spring
J2EE


employee
DNO
10
10
10
Department
dno
dname
mgr
10
fK
RDBMS


list
emp
list
[shantanu, 67000]
[shantanu, 67000]
[shantanu, 67000]
[shantanu, 67000]
[shantanu, 67000]

save

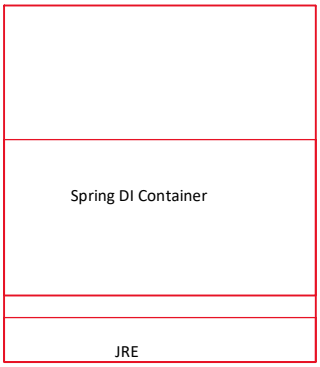
JVM+Runtime Library is the container

Java App with main method --> Application Container

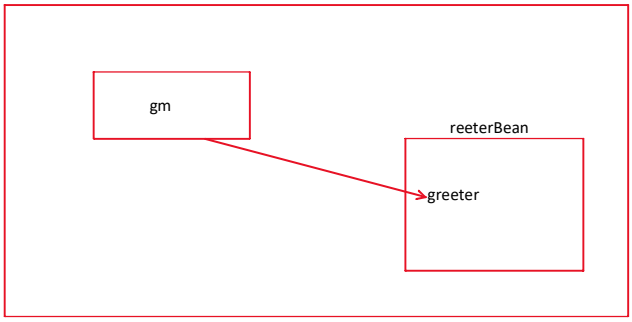
Servlets/JSP (Web Component) ---> Web Container

EJB (Business Component)--> EJB Container X not present in Spring Framework

Applets ------> Applet Container



JVM




```
public class Greeter {

	private Greet greet;

	public Greeter() {

	}

	public void setGreet(Greet greet) {
		System.out.println("Setter called with"+greet);
		this.greet = greet;
	}


	public Greeter(Greet greet) {
		this.greet = greet;
	}
	}
```

```
<bean id="greet1" class="com.demo.spring.GoodMorning" />

	<bean id="greeterBean" class="com.demo.spring.Greeter">
		<property name="greet" ref="greet" />
	</bean>

	<bean id="greeterBean1" class="com.demo.spring.Greeter">
		<constructor-arg name="greet" ref="greet" />
	</bean>
```

Annotaions

<bean id="greet1" class="com.demo.spring.GoodMorning" />

Annotaions

Type Level
@Component
- @Service
- @Controller
- @Repository
- @Configuration

Method Level
@Bean

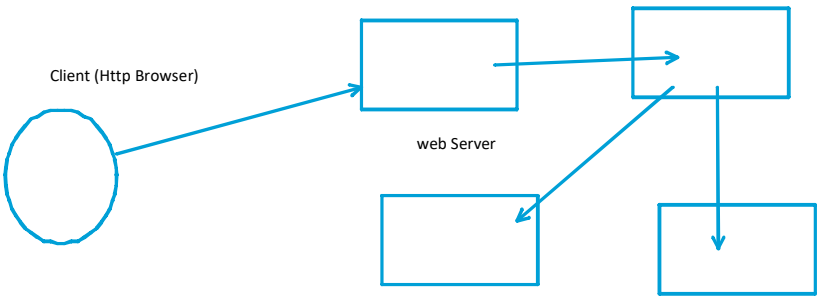
@Autowired

```
<bean id="greet1" class="com.demo.spring.GoodMorning" />

    <bean id="greeterBean" class="com.demo.spring.Greeter">
        <property name="greet" ref="greet" />
    </bean>

    <bean id="greeterBean1" class="com.demo.spring.Greeter">
        <constructor-arg name="greet" ref="greet" />
    </bean>
```

```
Greet g1= new GoodMorning();

Greet g2= new  GoodEvening();
```

singleton
prototype

Webapplication
- request
- session
- application




```
#!/usr/local/bin/perl
# hello.pl -- my first perl script!
print "Content-type: text/html\n\n";
print <<"EOF";
<HTML>
<HEAD>
<TITLE>Hello, world!</TITLE>
</HEAD>
<BODY>
<H1>Hello, world!</H1>
</BODY>
</HTML>
EOF
```

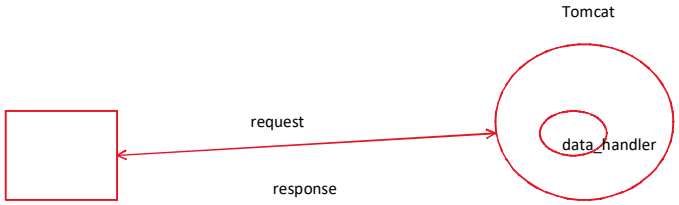
From <http://www.csun.edu/~andrzej/begperl/hello_cgi.html>

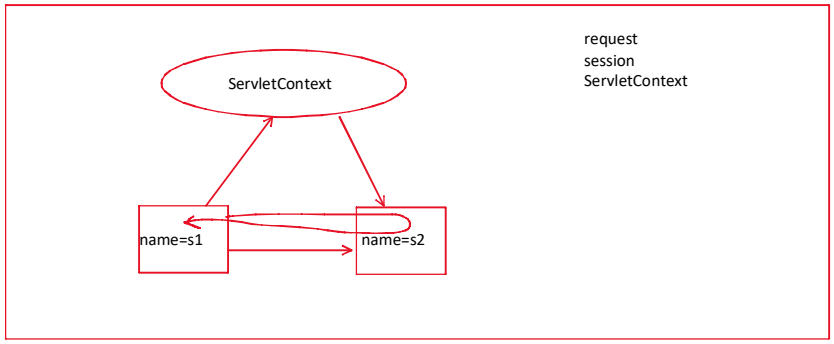
client	HttpServlet	
Http Methods		
GET	doGet()	@GetMapping
POST	doPost()	@PostMapping
PUT	doPut()	
DELETE	doDelete()	
HEAD		

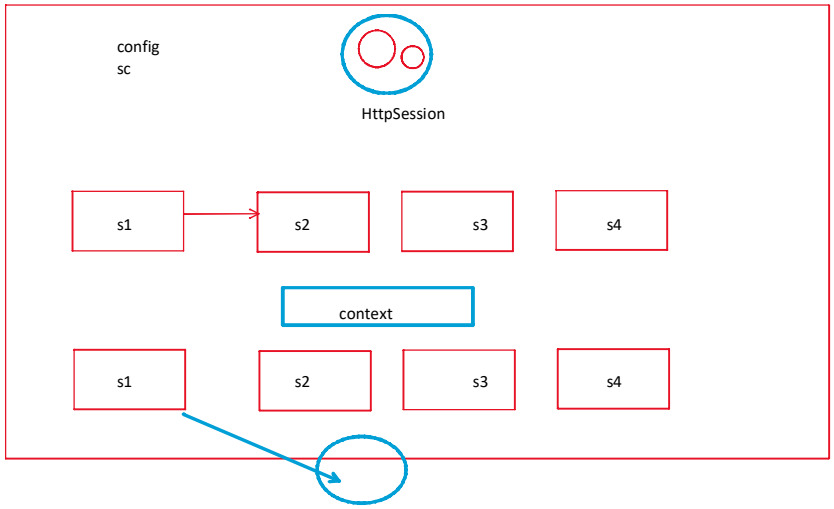
TRACE

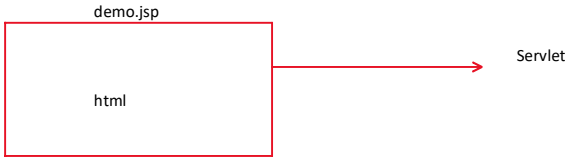
OPTION

etc etc etc..









class Hello_jsp .....{

_jspService(){

}

}

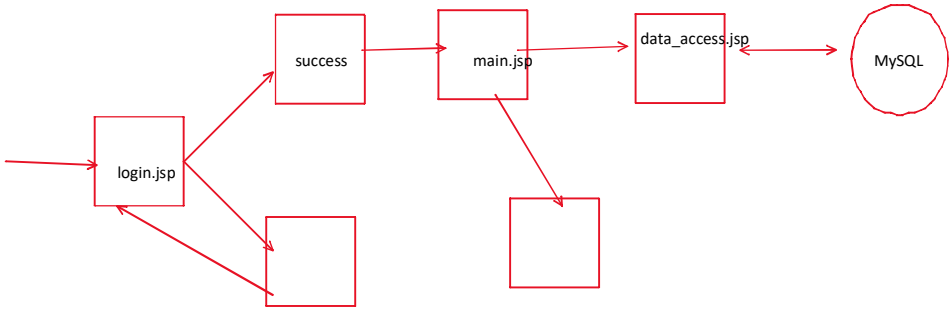
home.jsp

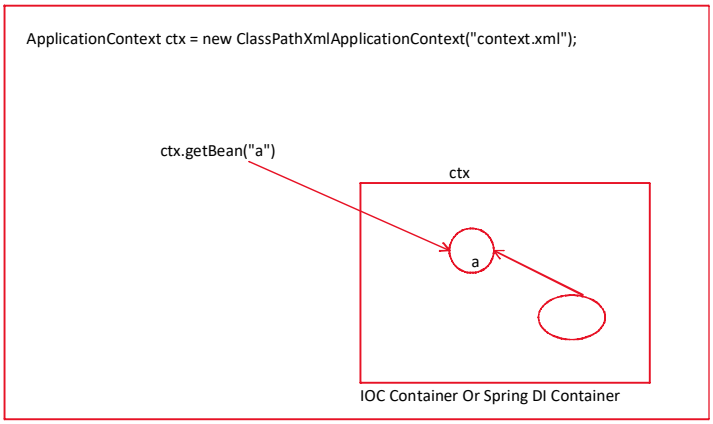
```
<form action="" method="get">
    <input type="text" name="user"><br> <input type="submit"
        value="check">
</form>
```

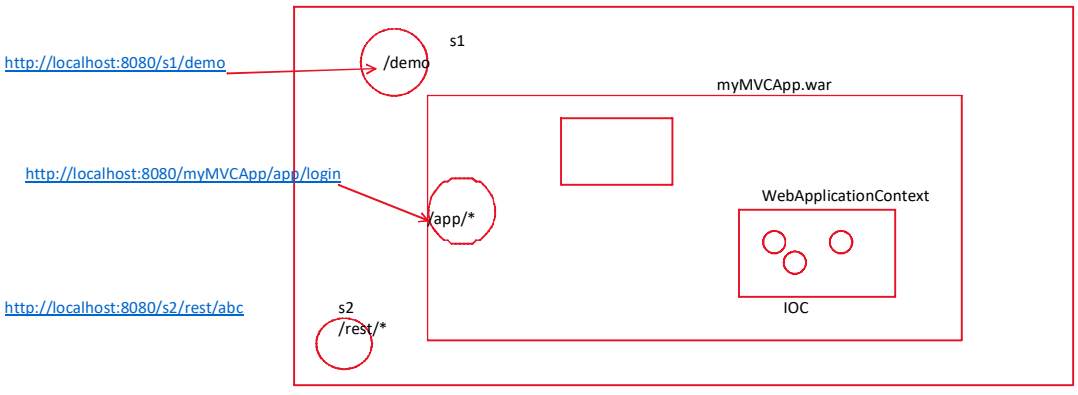
<jsp:forward page="page location"/>

Tiger --> tiger.jsp
Lion-->lion.jsp
Fox->fox.jsp

include

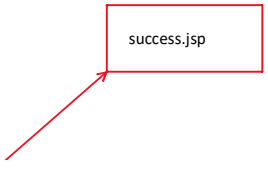






Apache Tomcat

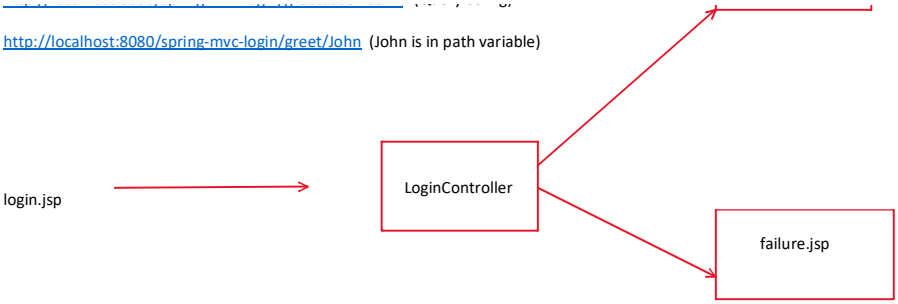
/WEB-INF/pages/| hello| .jsp

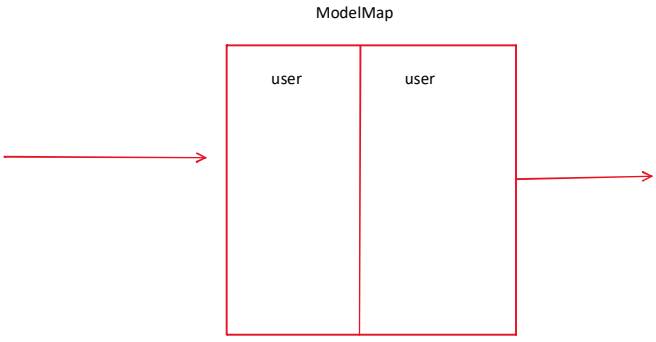


http://localhost:8080/spring-mvc-login/greet?user=John (Query String)

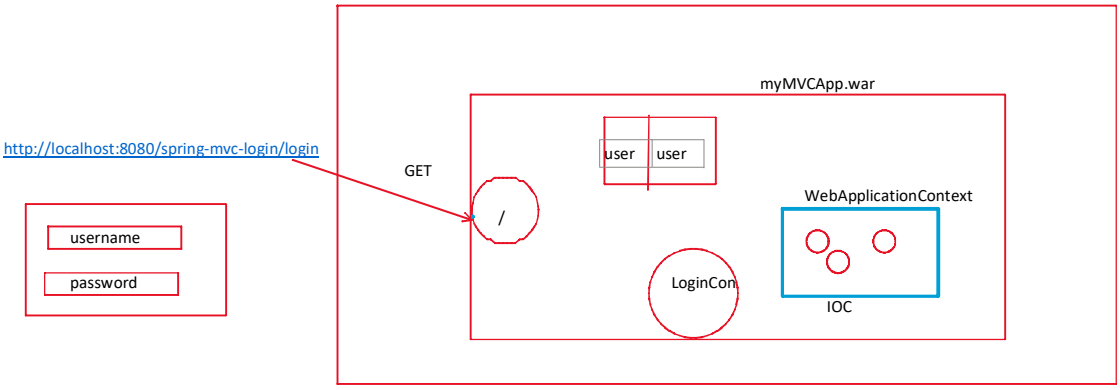
http://localhost:8080/spring-mvc-login/greet/John (John is in path variable)

http://localhost:8080/spring-mvc-login/greet/John (John is in path variable)

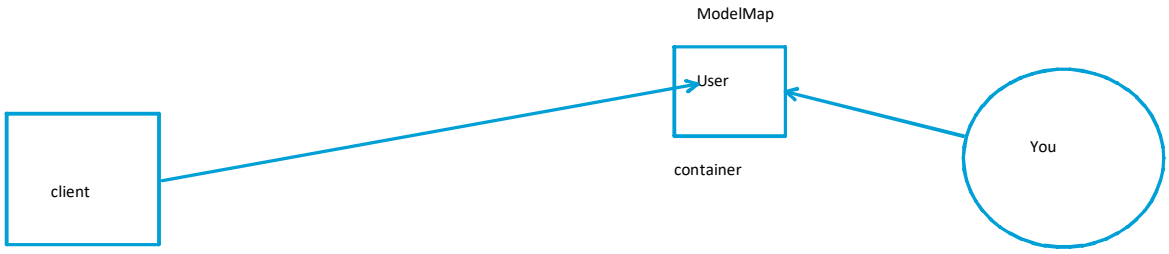




request
session
application



Apache Tomcat



container

Client

Spring MVC App

GET http://localhost:8080/findEmp?id=102

```
@GetMapping(value="/findEmp")
 public String findOne(@RequestParam("id") int id) {

}
```

POST http://localhost:8080/add-emp

```
@PostMapping()
```

PUT http://localhost:8080/update

@RequestMapping(method=PUT, value="/update") --> @PutMapping();

DELETE http://localhost:8080/emp/102

@DeleteMapping(value="/emp/{id}"

REST

http://www.google.co.in

Service

Web Service
Restful Web Service

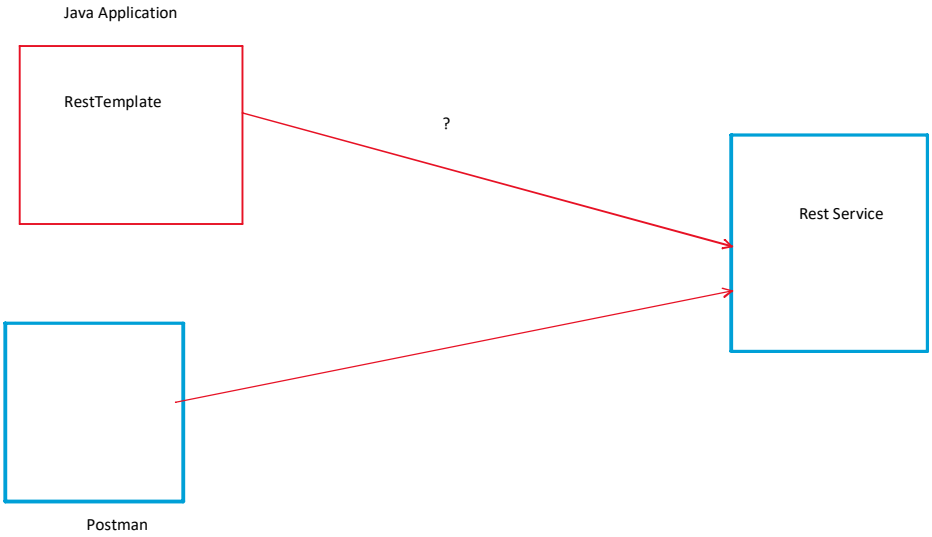
UI Layer

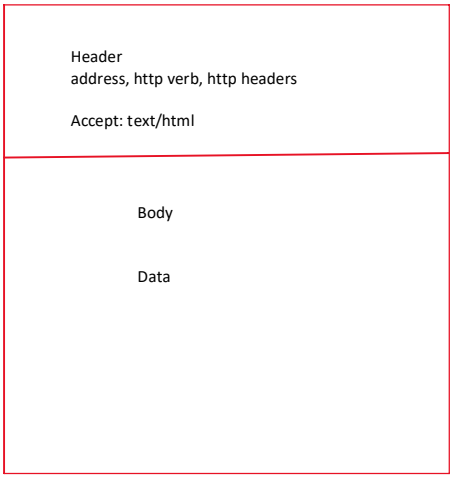
Service Layer (Business Logic Layer)

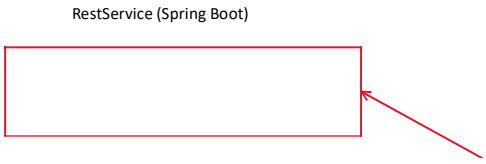
Data layer

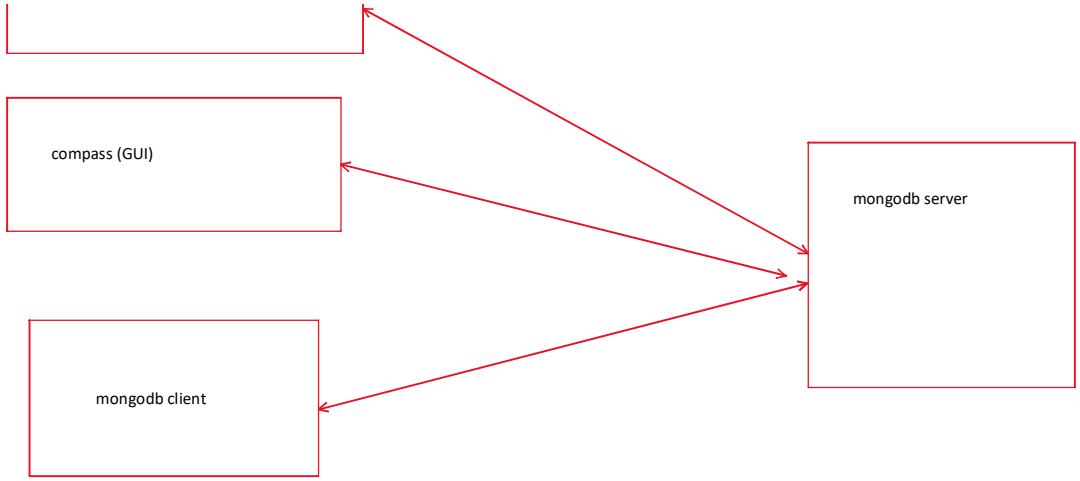
TaxCalculator
Currency Converters
WeatherReport
Stco Data

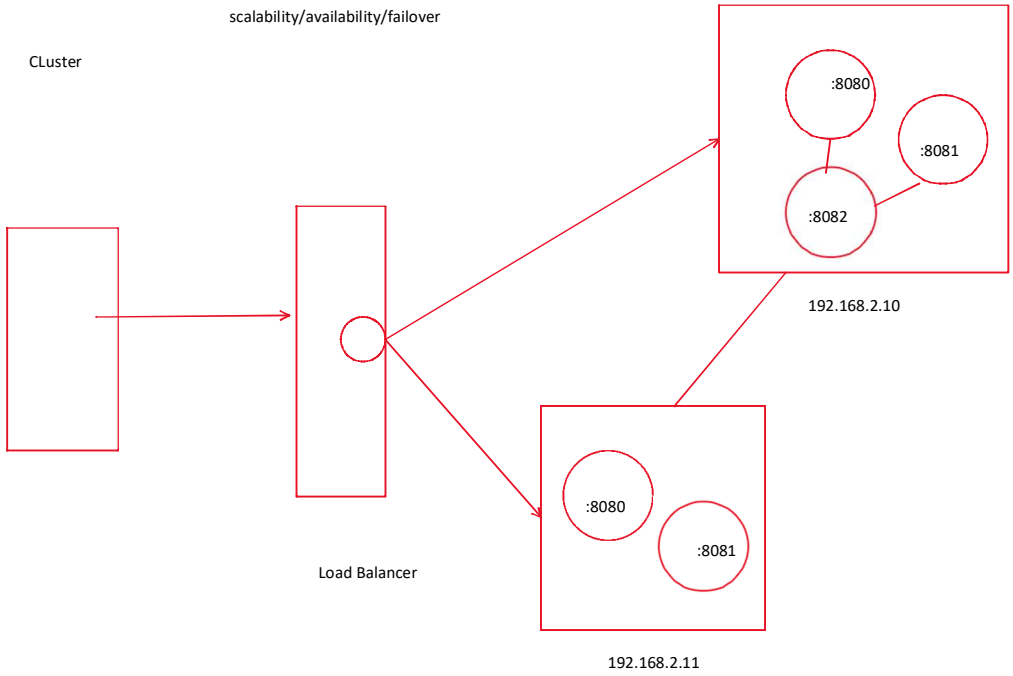
SOAP

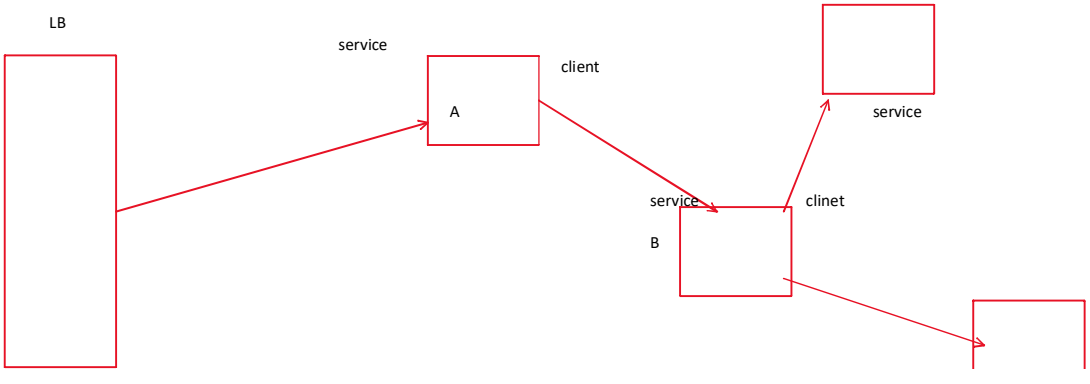







compass (GUI)
mongodb server
mongodb client


scalability/availability/failover
CLuster
:8080
:8081
:8082
192.168.2.10
:8080
:8081
Load Balancer
192.168.2.11


LB
service
client
A
service
service
clinet
B

JWT
Oauth2

Hyper-V (Old)

WSL2 (needs Updated Linux Kernel)

Authentication: who you are? Are you a valid user? (username/password)

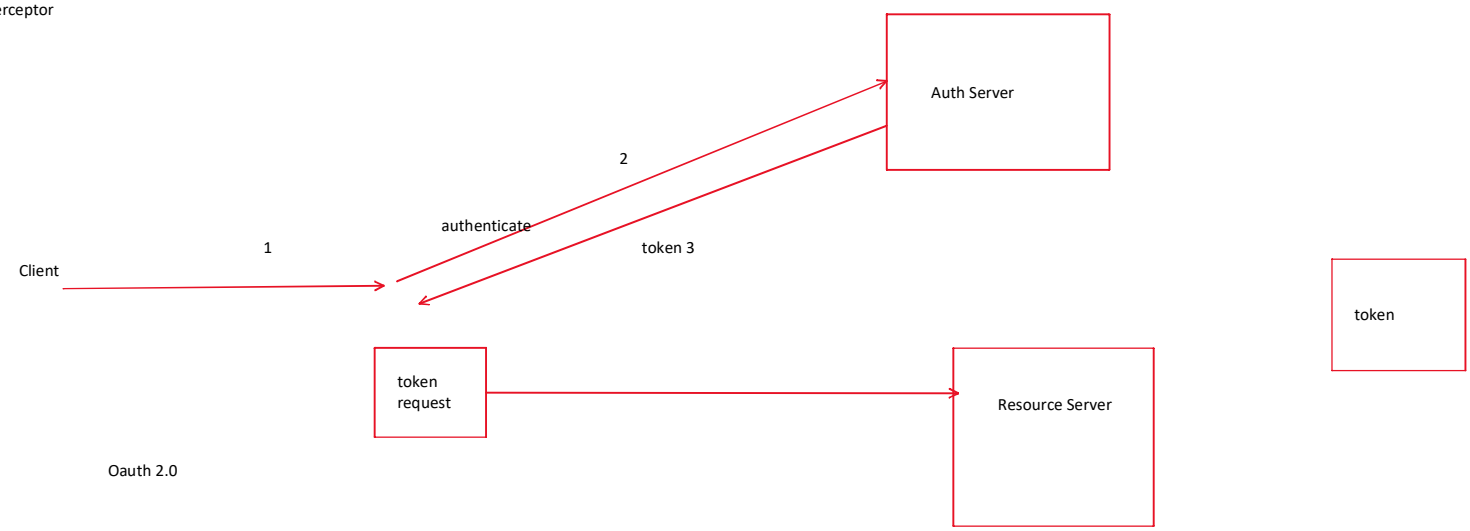
Authorization: if you are a valid user, what permissions do you have?

Role: decides the policy and assigned to a single user or a Group

Group : is a group of users

Principal : The User
Credentials : username/password/Tokens
Tokens
Credential Store: the storage where the credentials are kept (Memory, File, Database, LDAP Server)

Interceptor



Oauth 2.0