Built-in events in Spring framework

ContentStartedEvent

```
MyClass obj = new MyClass();

MyClass obj = content.getBean("myClass");

main(String[] args){
  AnotherClass obj
    = new AnotherClass();
}

class AnotherClass{
  private MyClass obj; //field/property dependency
  AnotherClass(MyClass obj){//constructor injection
    this.obj=obj;
  public void setMyClass(MyClass obj){//setter injection
    this.obj=obj;
  }
}
```

Events (triggered when specific things occur)

Built-in or Predefined

(Events related to lifecycle of the content object)

→ ContentStartedEvent
→ ContentStoppedEvent
→ ContentRefreshedEvent
etc.

Custom or User-defined

(must be defined and published)

↳ extends ApplicationEvent
↳ must define a no-arg constructor (calls super()) } define
↳ create an EventPublisher which implements ApplicationEventPublisherAware. (mention this class in config) } publish
↳ define event handler } handle

*EventHandler
→ associated with a specific event
→ get activated automatically when that event is triggered.

boolean withdrawMoney()

ATM

withdraw money process

↳ cash is expensed by the ATM & receipt is printed. (1)

↳ sends you an SMS (sometimes) (2)

What determines a successful withdrawal?

a. Only (1) ✓

b. Only (2)

c. Both (1) & (2)

d. None

New scenario for SMS

New device login

Note - Trigger a custom event called 'CashWithdrawalEvent' after withdrawMoney returns true. The event will activate 'SendSMSHandler'

AOP (Aspect-Oriented Programming) complements OOPs by providing another way of thinking about program structure.

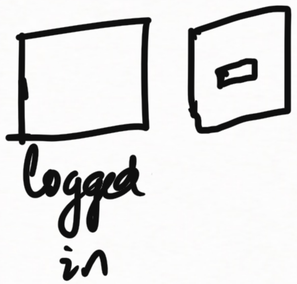
In traditional OOPs, the key unit of modularity is a <u>class</u>.
In AOP OOPs, the key unit of modularity is an <u>aspect</u>.

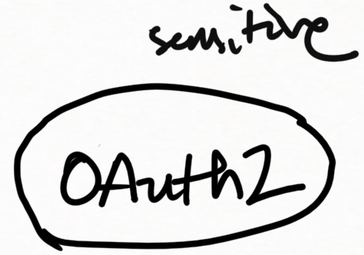
Benefits of aspects is transaction management.

Aspects are a feature of J2EE.

@Aspect

Spring Security





authentication

CORS (Cross-Origin Resource Sharing (Policy))

✓CSRF (Cross-site Request Forgery)

session management

↳ Every application has its own CORS policy. It can be toggled.

(YouTube) cross-origin (Instagram) Wikipedia (YouTube videos)

url source ⟷ url destination

↳ www.sbi.com/send?targetAcc=1234&amt=5000; (social engineering attack)

Scenario: You are logged into in another tab.

And, you click on an instagram ad. You click on a fake button.

source of button = Insta ad. Target = SBI netbanking

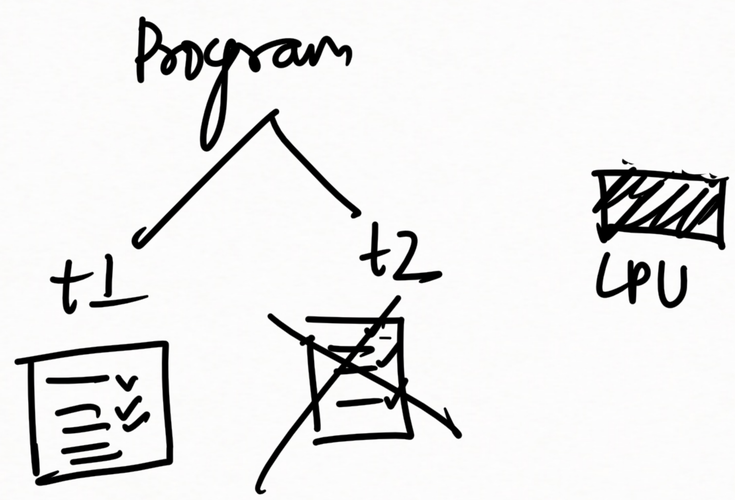
not allow any requests for sensitive transaction from any place other than SBI transfer page

Reactive processing in Spring

Reactive processing lets you build non-blocking, asynchronous, low-latency applications that can smartly handle flow-control. They are message-driven & elastic.

Better utilization of modern processors.

→ WebFlux
→ Project Reactor



t1 | t2
ready queue

Sop ("My name is Akash"); ①
Sop (2+3); ② = Add & print
Sop (3-2+1); ③ Subtract, Add, Print

t1 (2 lines)
t2 (2 lines)
t1 (1 line)

t2 (1 line) ✓
t1 (2 lines) ✓

Spring Actuator

Used for production-ready features

Metrics used to monitor performance of our app (state)

Spring actuator provides you production-grade tools to monitor your web-app without having to implement the logic.

Gives you operational info of your app.

Spring vs. Struts

Both can be used to develop Java web application.

Struts is older & hence has lesser features.

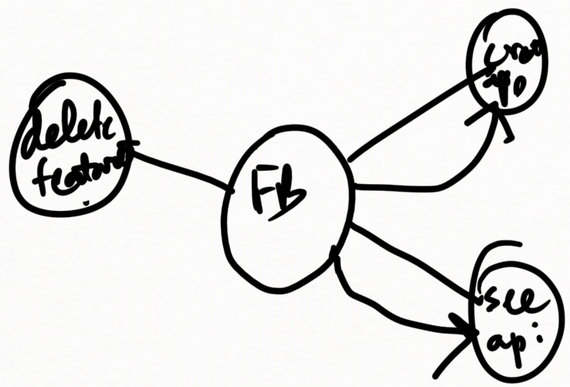
Spring is newer & more popular. Has more features.

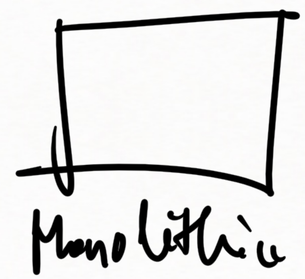
Spring uses MVC architecture & supports ORM.

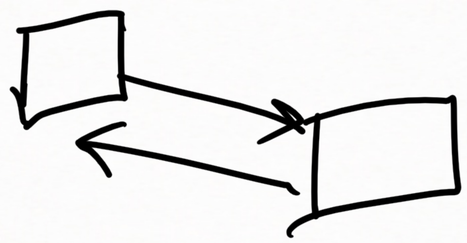
Struts doesn't.

Microservices

REST api can be further broken down into microservices.







Spring Exception Handling

SimpleMappingExceptionResolver

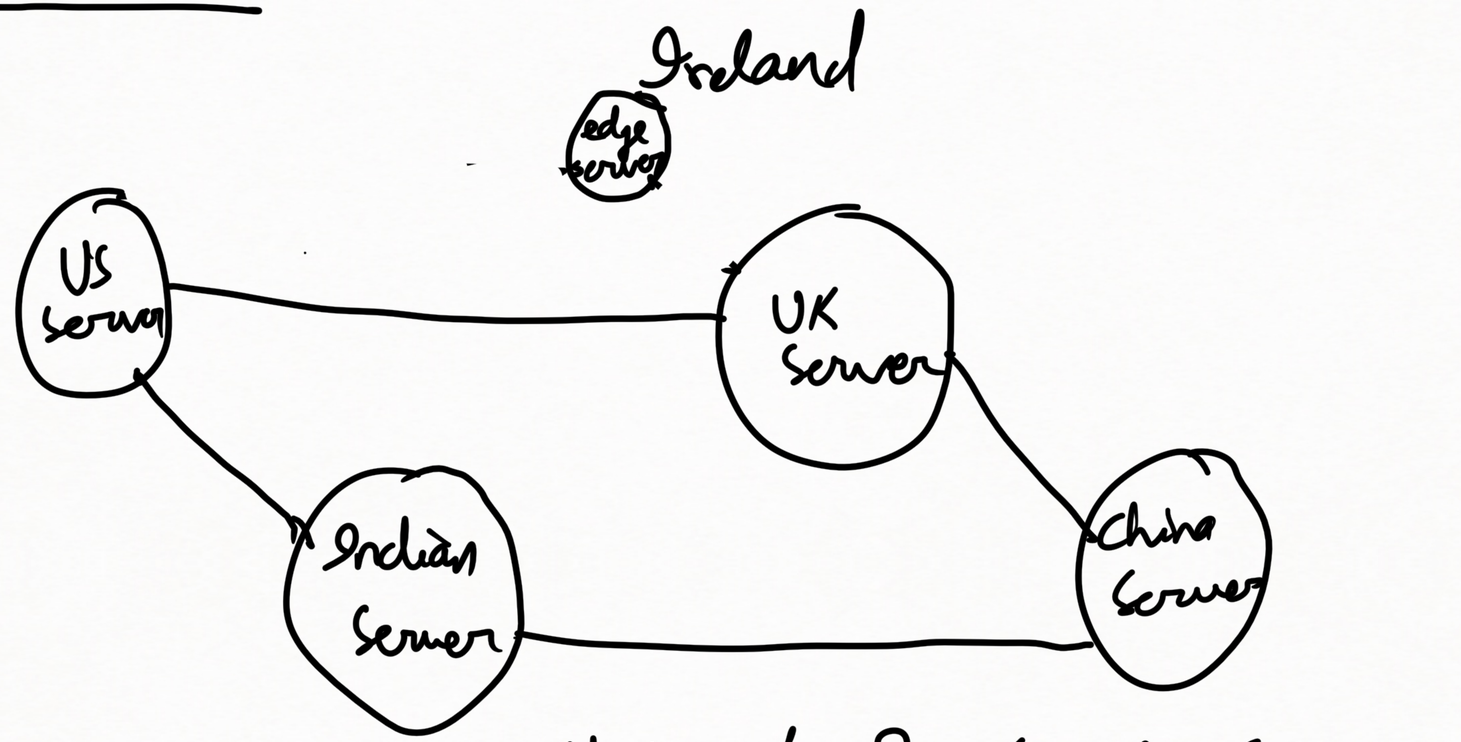
/error

4xx

```
@ExceptionHandler(DataIntegrityViolation.class)
  public void handleError(HttpServletRequest req, Exception ex){

  }
```

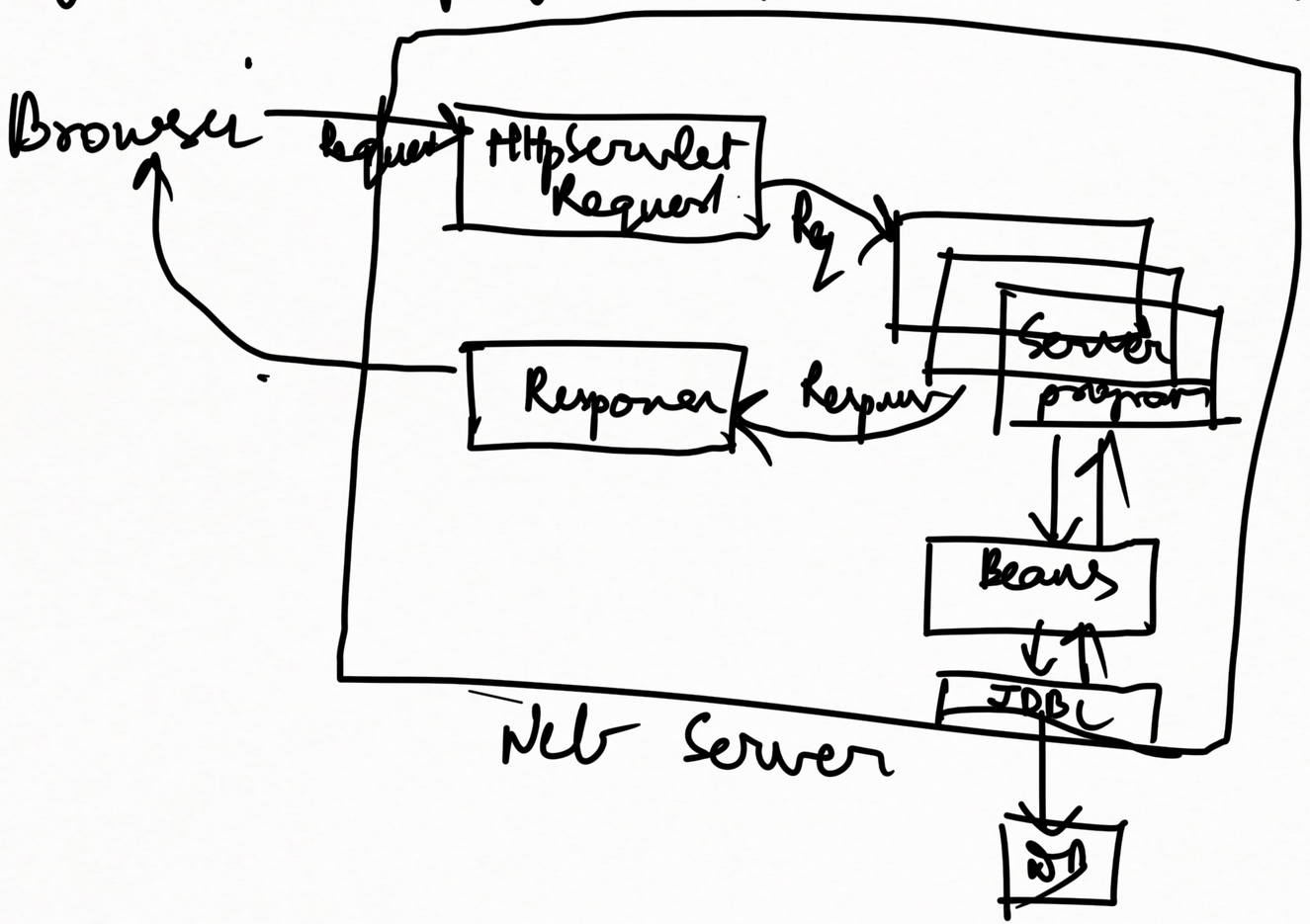
Edge server



Network of Google servers

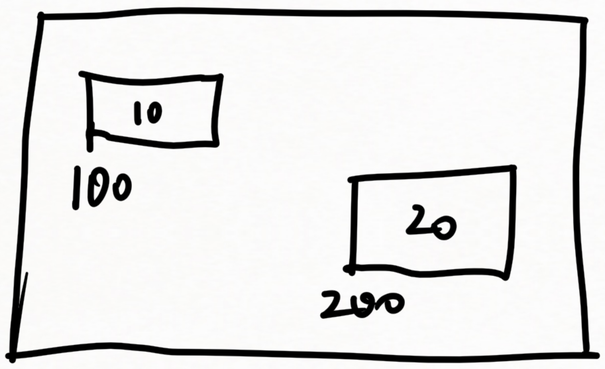
Web Containers

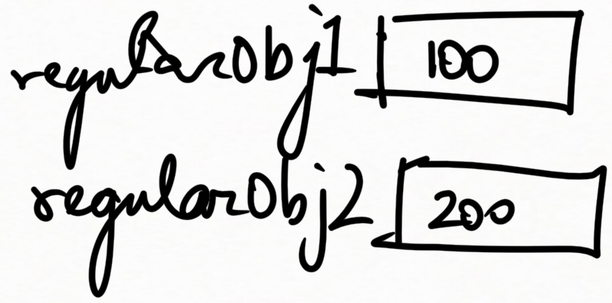
Manage the lifecycle of the web application.



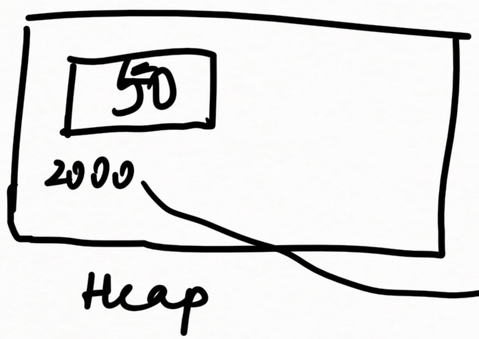
Singleton = only one active object of that class in memory

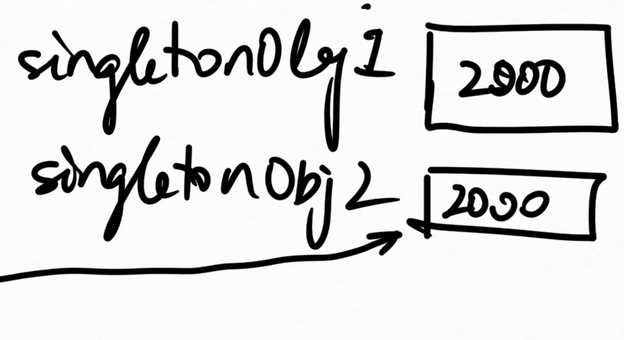
Regular Class

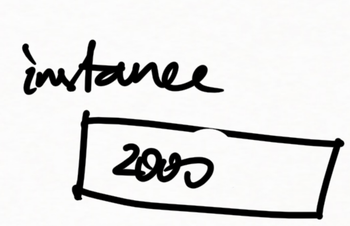




Singleton Class





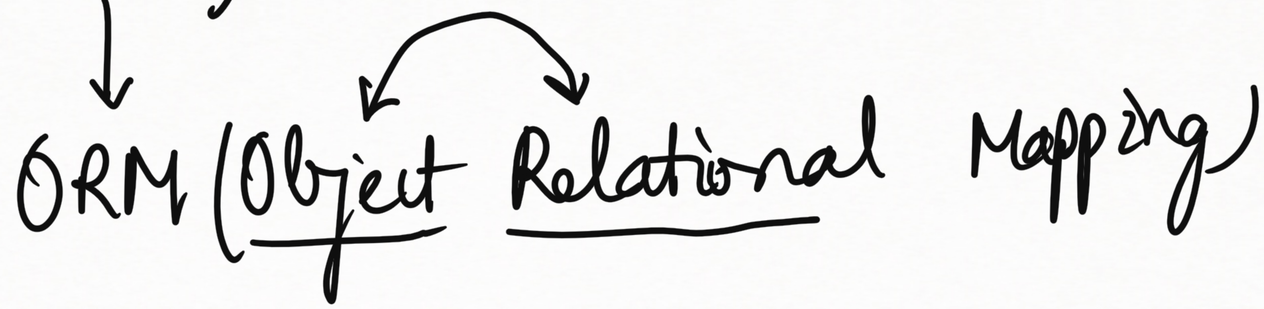


JPA (Java Persistence API)

1) connector + JDBC

2) spring starter JDBC

3) Spring JPA (Hibernate)



ORM

1) Model → one model object = one record of that table
2) Interact with tables using object-oriented programming
3) No need of using JDBL-specific syntax or SQL-specific syntax.
   Table operation can be treated as function calls.
4) ORM query syntax is DB agnostic.

Title {
    Worker worker;
    public void getWorker(worker)
        this.worker = worker;
    }

Example

Without ORM

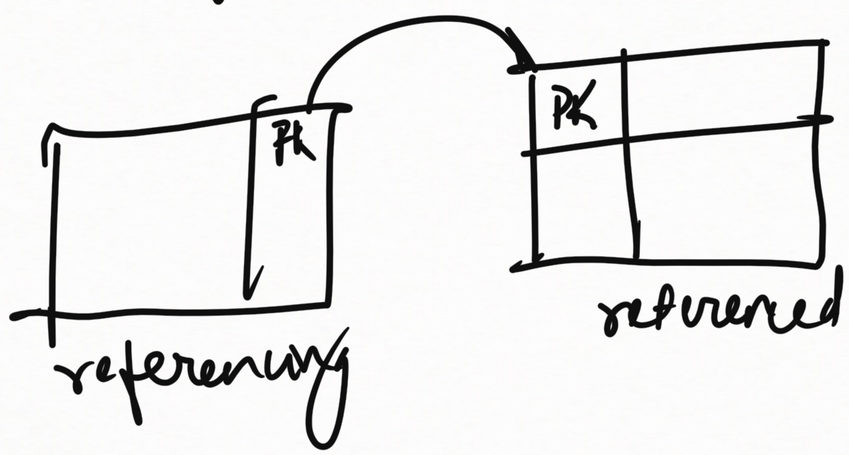
query = "Select * from Worker WHERE id = 5";
rs = stmt.executeQuery(query);
while( ) {
    // set values into model object.
}

Worker worker = new Worker();
worker.findById(5);

worker.findAll();

```
Title title = new Title();
title.findWhere("WORKER_TITLE", "Manager")
     .getWorker().findById();
```

You can treat records of "referenced table" as if they were part of your own table with an ORM.



Hibernate + Lombok

@Getter
@Setter
> @Data

Task

1) Catch-up with all the Spring Boot tasks
2) Read-up on basic syntax of HTML
, , <form>, lists <ul, ol>, table