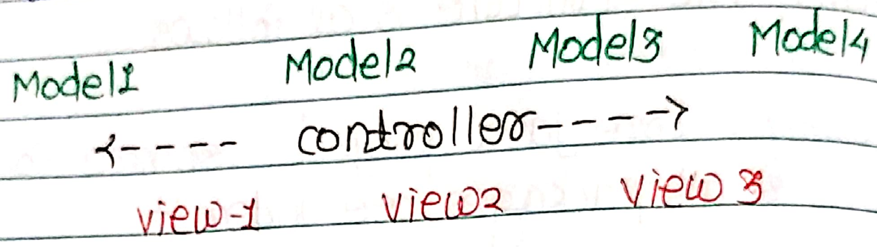
1st July

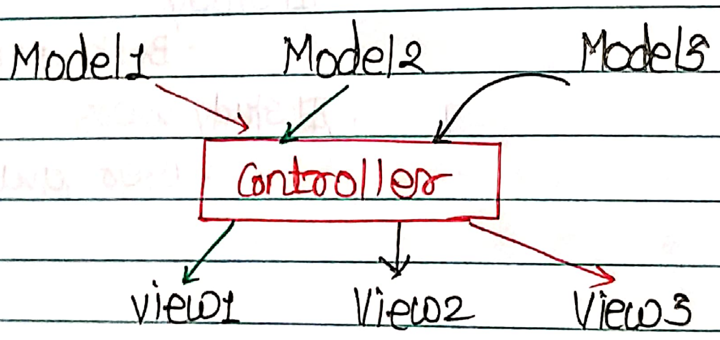
MVC - Model View Controller
Architecture

Model = Data (DO, DAO, Data processing logic
View = Presentation of the data (HTML, JSP
.. usually it is done on browsers.

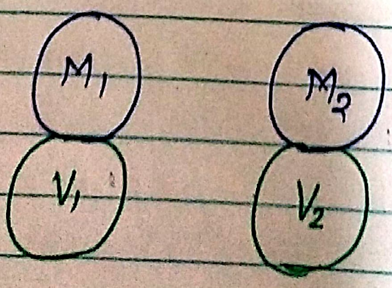
We need to fit the data in HTML/JSP files to show in View so,
To loosely couple Model & View, A controller is used



Controller connects model & view at runtime.



loose Coupling of model & view via controller



Tight coupling of model & view is not desirable apsara

Date: ___.___.___

MVC in servlet JSP

Model	⇒	DAO, DO
servlet	⇒	Controller
view	⇒	JSP.

Spring ~~Et~~ MVC

Model	→	Beans (DAO, DO, BL beans)
controller	⇒	Web MVC controller (It is also a bean)
View	⇒	JSP / Thymeleaf

spring Annotations written for class level

@Configuration ⇒ Spring will think:
This is javaconfig, I will get
@Bean tag insideit

@Component
@service
@Repository
} for spring ⇒ All are same
This is java bean, I have to
manage, instantiate, DI !!!

These diff names are for programmer's readability

@service	⇒	business logic related task
@Repository	⇒	DAO/DB
@ component	⇒	General bean / not specified.

Page No.: .....

@controller : Spring ⇒ This is java class that
∴ I have to manage, Instantiate,
DI. +
spring has to call all the methods of
this class when different requests
comes.

We write a controller
MAP the request to model & view

Request ⇒ GET / POST
Output ⇒ A JSP is generated & goes to
browser.

Phase-1 ⇒ development
1. Write JSP
2. Write configuration
3. write controller

Phase-2 ⇒ deployment
Run the tomcat server
- the embedded tomcat runs & deploys

Production phase / Request phase
client starts sending request from browser.

Flow of a project
1. Create a JSP file   Hello.jsp
2. Application Configuration   suffix - Prefix.
3.
```
spring.mvc.view.prefix = "WEB-INF"/views"
spring.mvc.view.suffix = ".jsp"
```

3. Controller
```
@Controller          ⇒ Annotation.
public String MyWebController {
  @RequestMapping(value = "/Hello", method=RequestMethod.GET)
    public String F1 ()
  {
        return "Hello";
  }
```

Spring Boot : Many Boiler plate codes are wrapped inside.
-- many configuration can be given through key value pair ~~using~~ setting in applications.properties.

flow Execution #
1. Application context is created by spring Boot
2. If web controller class is in the same package as main class then scan is not needed, spring Boot will auto scan it.
3. When spring find controller, it initialised the controller.
4. After that, it takes its request mapping It will map the request.

e.g.
```
@RequestMapping(value = "/hello", method=RequestMethod.GET)
    Public String F1 ()
    {
        return "Hello";
    }
```
This mns that, when requested url is /Hello & method is Get, return hello.

This hello, will go to application properties. (collected by spring framework) prepend & append by prefix & suffix

$$\underbrace{\text{WEB-INF/views/}}_{\text{Prefix.}}\text{hello}\underbrace{\text{.jsp}}_{\text{suffix}}$$