1. Spring IOC

IoC represent for Inversion of Control
It is a principle in software which transfers the control of creating objects manually to a container or framework.

How to achieve IOC
dependency injection, factory design pattern, strategy design pattern

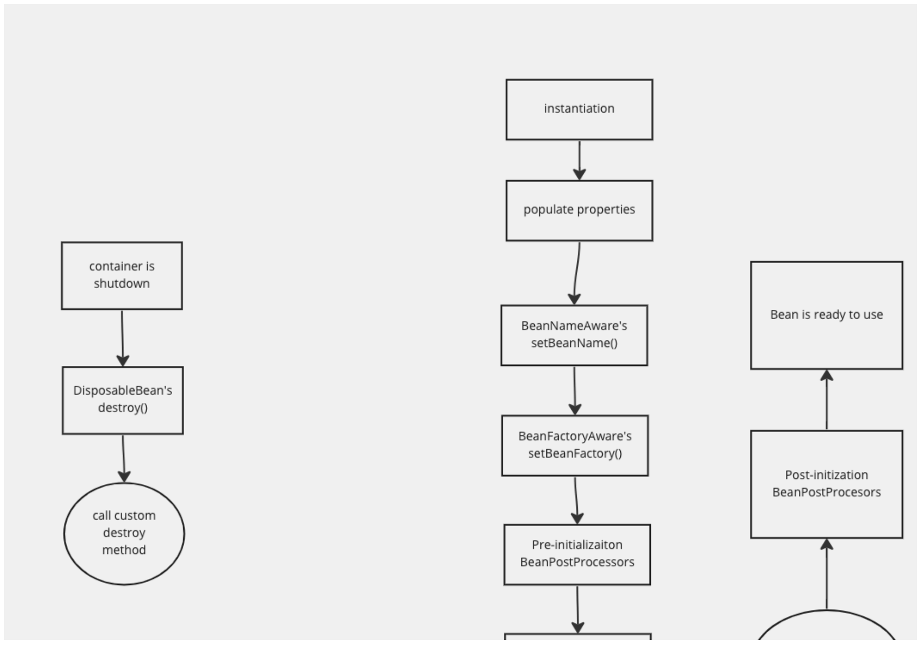
What is dependency injection?
The dependency between objects is managed by a container, and the object injection is completed by a container.

Three way to implement dependency injection:
constructor injection: is the process of using the constructor to pass in the dependencies of a class.
setter injection: injects the dependency object using the setter method.
field injection: set value object as dependency to the field of an object.

What is Bean?
objects managed by containers.

Bean scope:
singleton (default): a single bean object instance in the ioc containerprototype: it produces a new instance each and every time a bean is requestedrequest: it produces a new bean for each http requestsession: it produces a new bean for each sessionApplication: it produces a new bean for each servletContextwebSocket: it produces a new bean for each webSocketThread: it produces a new bean for each Thread

Bean lifecycle:


instantiation
populate properties
BeanNameAware's setBeanName()
BeanFactoryAware's setBeanFactory()
Pre-initializaiton BeanPostProcessors
Bean is ready to use
Post-initization BeanPostProcesors
container is shutdown
DisposableBean's destroy()
call custom destroy method

The bean container finds the definition of the Spring bean in the configuration file.
The bean container creates an instance of a bean using the Java Reflection API.
If some attribute values are involved, use the set() method to set some attribute values.
If the Bean implements the BeanNameAware interface, call the setBeanName() method and pass in the name of the Bean.
If the Bean implements the BeanFactoryAware interface, call the setBeanFactory() method and pass in an instance of the BeanFactory object.
Similar to the above, if other *.Aware interfaces are implemented, the corresponding method is called.
If there is a BeanPostProcessor object related to the Spring container that loads this bean, execute the postProcessBeforeInitialization() method
If the Bean implements the InitializingBean interface, execute the afterPropertiesSet() method.
If the bean definition in the configuration file contains the init-method attribute, execute the specified method.
If there is a BeanPostProcessor object related to the Spring container that loads this bean, execute the postProcessAfterInitialization() method

When the Bean is to be destroyed, if the Bean implements the DisposableBean interface, execute the destroy() method.
When the Bean is to be destroyed, if the Bean definition in the configuration file contains the destroy-method attribute, the specified method is executed.

2. Spring AOP

What is AOP?
Aspect oriented programming.
Horizontal separation to separate cross cutting concerns code and business logic code

Aspect:
An Aspect is the concern (cross cutting concern) which you want to implement in the application such as logging, performance monitoring, transactional handing etc.

Advice:
An Advice is the actual implementation of the aspect. Aspect is a concept and Advice is the concrete implementation of the concept.

JoinPoint:
A JoinPoint is a point in the execution of the program where an aspect can be applied. It could be before/after executing the method, before throwing an exception, before/after modifying an instance variable etc. Keep in mind that it is not necessary and also not required to apply an aspect at all the available join points. Spring AOP only supports method execution join points.

PointCut:
PointCuts tell on which join points the aspect will be applied. An advice is associated with a point cut expression and is applied to a join point which matches the point cut expression.

Target:
Target is the application object on which the advice will be applied.

Transactional:
Transaction is a way to achieve declarative transaction management. It cane used on Class and methods.
When it is used on a class, that means all public method in the class is annotate by @Transaction.

To implement this, AOP will be used, it is like intercept the method before and after, then create or join a transaction before the target method starts, and commit or roll back according to the execution situation after the target method is executed.

We don't need to write complicate codes to control Transactions. The Transaction annotation can define propagation, isolation, how transactions roll back.

3. Spring MVC

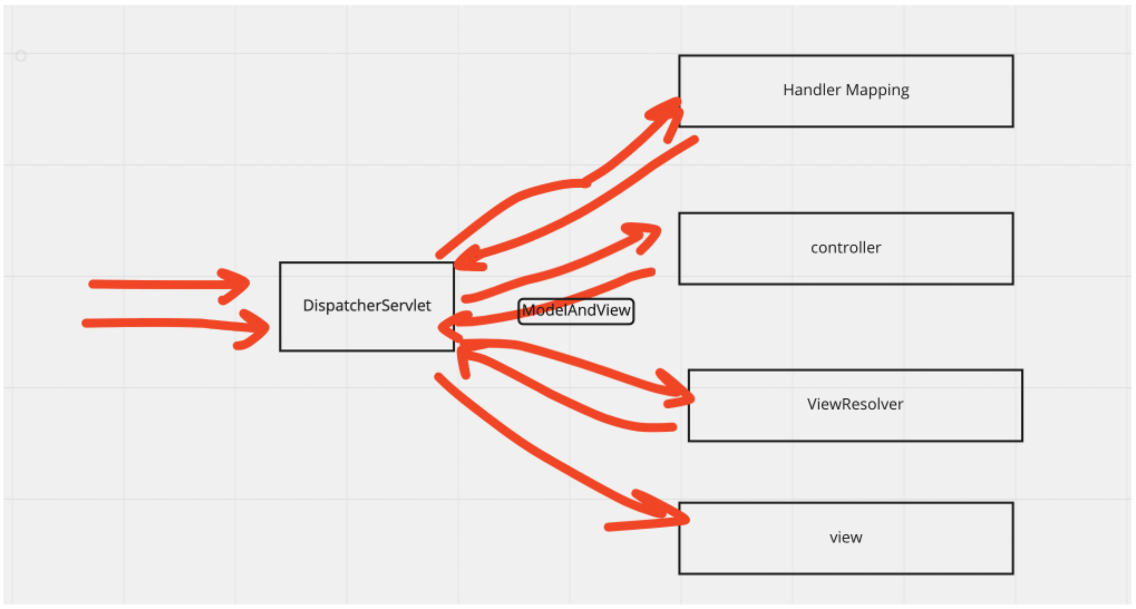
What is Spring MVC?
Spring MVC is a module in Spring, which enable Spring to quickly build a web program for MVC architecture.
MVC represents for models, views, and controllers. It is a core idea to organize code by separating business logic, data, and presentation logic.

Model: contains the data of the application, the data can be a single object or a collection of objects

Front controller: will intercept all the request, in spring mvc, the DispatchServerlet class works as the front controller
Controller: business logic of an application

View: display the information/data to user

Spring MVC workflow**?**



1) The client (browser) sends a request, and DispatcherServlet intercepts the request.
2) DispatcherServlet calls HandlerMapping based on request information. HandlerMapping matches the uri to find the Handler that can be processed (that is, what we usually call the Controller controller), and encapsulates the interceptor involved in the request with the Handler.
3) DispatcherServlet calls HandlerAdapter to adapt and execute Handler.
4) After the Handler finishes processing the user request, it will return a ModelAndView object to the DispatcherServlet. ModelAndView, as the name suggests, contains information about the data model and the corresponding view. Model is the returned data object, and View is a logical View.
5) ViewResolver will find the actual View based on the logical View.
6) DispaterServlet passes the returned Model to View (view rendering).
7) Return the View to the requester (browser)

4. Spring Boot

advantages:
it provides a flexible way to configure java beans, xml configuration, and database transaction.
it provides a powerful batch processing and manages rest endpoints.in spring boot, everything is auto configured, no manual configuration are needit offers annotation based spring applicationEase dependency managementInclude embedded servlet container -> Tomcat

What is Spring boot Starter?
Spring Boot Starters are dependency descriptors that can be added under

the <dependencies> section in pom.xml. There are around 50+ Spring Boot Starters for different Spring and related technologies. Spring Boot Starters were introduced to solve this problem so that the developers can spend more time on actual code than dependencies.

Auto Configuration:
Spring Boot auto-configuration attempts to automatically configure your Spring application based on the jar dependencies that you have added.

Rest API Design:
Rest is an architectural style for distributed hypermedia systems. REST has its guiding principles and constraints. These principles must be satisfied if a service interface needs to be referred to as RESTful.

How REST APIs work:
REST APIs communicate via HTTP requests to perform standard database functions like creating, reading, updating, and deleting records (also known as CRUD) within a resource.

HTTP methods:
http method: CRUD operations
create: postread: getupdate: postdelete: delete

idempotent: get, put, delete
safe: get
cacheable: get

http status code:
1xx information2xx success3xx redirect4xx client side error5xx server side error

200	ok
201	created
202	accepted
204	no content

400	bad request
401	unauthorized
403	forbidden
404	not found
405	method not allowed

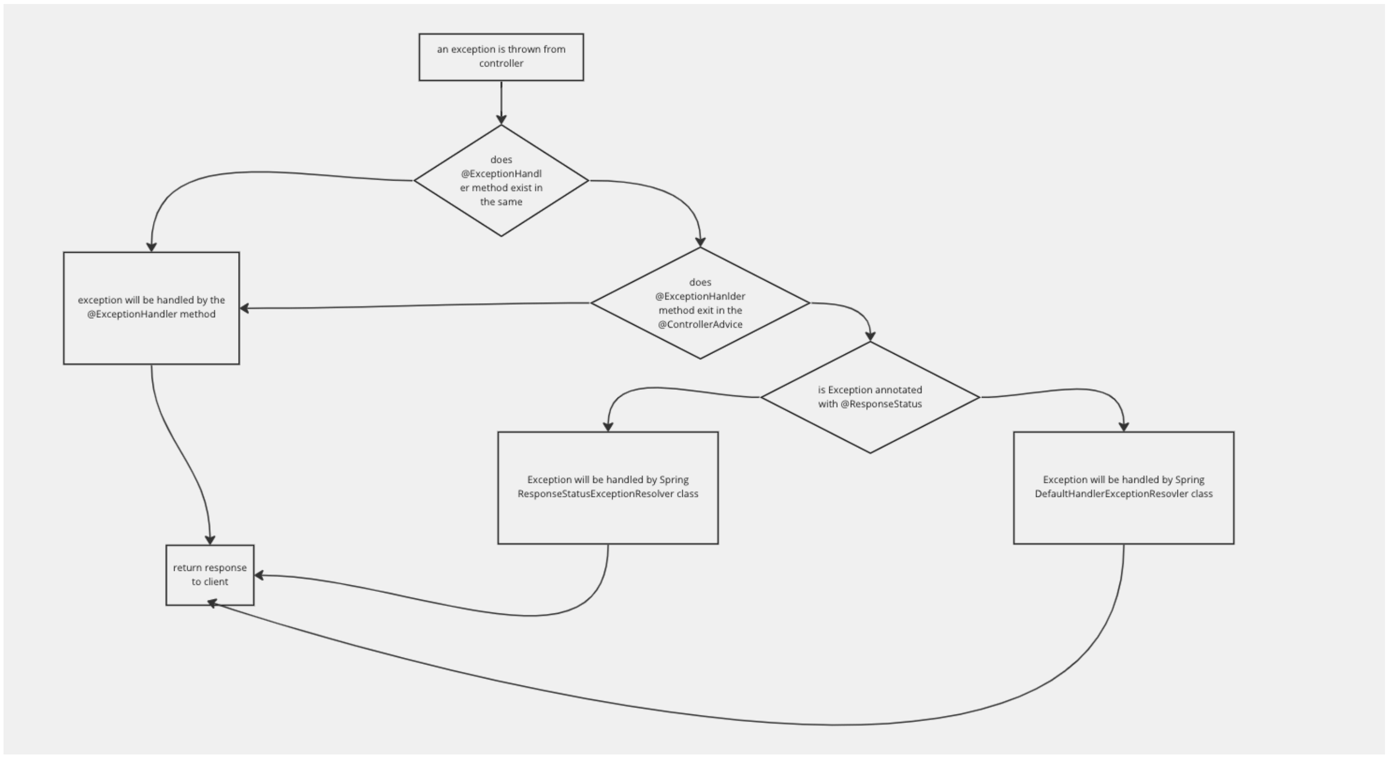
500	internal server error

How to design Restful API in Spring?

Use the following annotations.
RequestMapping, GetMapping, PutMapping, PostMapping, DeleteMappingRequestParam, PathVariableRequestBody (json -> java object), ResponseBody (java object -> json)Controller/RestController, Service, Repository

@RequestMapping is used to define header, URI, method type, parameters, path, consume type, etc.
@GetMapping ensures that HTTP GET requests to the URI are mapped to the method.
@PutMapping is used to handle PUT request.
@PostMapping is used to handle POST request.
@DeleteMapping is used to handle DELETE request.
@RequestParam is used to collect all parameters expecting from request.
@PathVariable is used to collect value from request URI
@RequestBody maps data coming as part of request with an Object.
@ResponseBody will generate response in predefined format mentioned as part of @PostMapping annotation.

5. Exception Handling Process


an exception is thrown from controller
does @ExceptionHandler method exist in the same
exception will be handled by the @ExceptionHandler method
does @ExceptionHanlder method exit in the @ControllerAdvice
is Exception annotated with @ResponseStatus
Exception will be handled by Spring ResponseStatusExceptionResolver class
Exception will be handled by Spring DefaultHandlerExceptionResovler class
return response to client

define exception handler methods in controllers:

The @ExceptionHandler is an annotation used to handle the specific exceptions and sending the custom responses to the client.

How to handle exceptions globally?
@ControllerAdvice annotation that we can use with any class to define our global exception handler.

Notice that we can use @ResponseStatus annotation with exception classes to define the HTTP code that will be sent by our application when this type of exception is thrown by our application and handled by our exception handling implementations.

6. Validation

Validate form input for applications. It takes user input and checks the input by using standard validation annotations.

7. Swagger

Swagger helps users build, document, test and consume RESTful web services. It generates an interactive API for the users so that they can understand about the API more quickly.