1. Spring IOC

IOC: Inversion of Control is a principle in software which transfers the control of object or portions of program to the container or framework.

DI: Dependency Injection is a design pattern in which an object or function receives other objects or functions that it depends on from the framework. Fundamentally, dependency injection consists of passing parameters to a method.

Dependency Injection Approaches: different ways to achieve DI.

Constructor based: Inject dependencies at the constructor. Should be used for mandatory dependencies.

Setter based: Inject dependencies at the setter of the related field. Should be used for optional dependencies.

Field based: Inject dependencies at the field where it's declared directly. It would simplify the amount of arguments in the constructor and the amount of setters, but it's costlier than constructor and setter based because it uses reflection API. It's also discouraged because it may potentially hide the mandatory dependencies from the constructor.

Bean scope: the scope for a bean using the @Scope annotation. By default it's singleton which means a single bean object instance per IOC container. Other options are:
Prototype: create a new instance every time a bean is requested.
Request: create a new instance for each http request.
Session: for each session.
Application: for each servletContext.
WebSocket: for each WebSocket session.
Thread: for each thread.

Bean Life Cycle: the life cycle of a bean. Custom init and cleanup methods are called during instantiation and shutdown of the container.
@PostConstruct: annotated method will be invoked after the bean has been constructed.
@PreDestroy: annotated method will be invoked just before the bean is destroyed.

2. Spring AOP

AOP: Aspect Oriented Programming is a programming paradigm that aims to increase modularity by allowing the separation of cross-cutting concerns. It does so by adding behavior to existing code (an advice) without modifying the code itself, instead

separately specify which code is modified via a “pointcut” specification.

Aspect: an aspect is a class that implements enterprise application concerns that cut across multiple classes. Such as transaction management, logging, security, etc. The annotation is @Aspect.

Advice: additional behavior to be executed at multiple sections of multiple classes.
@Before: before the joinPoint.
@AfterReturning: after the return of the joinPoint.
@AfterThrowing: after the exception handling of the joinPoint.
@Around: before and after the joinPoint.

JoinPoint: a point during the execution of the program, such as the execution of a method or handling an exception.

PointCut: a predicate that matches the joinPoint. Advice is associated with a PointCut expression and runs at any joinPoint matched by the PointCut.

Target: object to be advised.

@Transactional: Spring creates proxies for all classes or methods annotated with @Transactional. The proxy allows the framework to inject transactional logic before and after the running method, mainly for starting and commiting the transaction.
Further configurations are:
- propagation
- isolation
- timeout
- readOnly
- RollBack

Propagation attribute indicates if any component or service will participate in the transaction and how it will behave if the calling component or service already has or does not have an existing transaction. The default value is REQUIRED which means it will either use the existing transaction of the caller or it will create a new transaction.
SUPPORTS: either use the existing transaction of the caller or runs without transaction.
NOT_SUPPORTED: runs without transaction. Caller’s existing transaction will be suspended.
REQUIRES_NEW: always creates a new transaction. Caller’s existing transaction will be suspended.
NEVER: runs without transaction. Throws an exception if the caller has an existing transaction.
MANDATORY: always use the caller’s existing transaction. Throws an exception if the caller does not have an existing transaction.

Isolation: same concept with the isolation level of the database. The default value is the same as the database, and others are also the same: READ_UNCOMMITED, READ_COMMITTED, REPEATABLE READ and SERIALIZABLE.
Timeout attribute indicates the time frame our transaction should complete in (milliseconds). Otherwise we will get a TransactionException. The default value is -1 which means no timeout at all.
ReadOnly: a hint boolean flag used to indicate if the transaction is read-only. Default is false.
Rollback: the default rollback behavior will rollback on throwed runtime unchecked exceptions. (The checked exceptions does not trigger a rollback)
This can also be controlled by using the attributes rollbackFor and rollbackForClassName, and noRollbackFor and noRollbackForClassName to avoid rollback on listed exceptions.

3. Spring MVC

MVC: stands for Model View Controller.
Model: contains the data of the application, can be a single object or a collection.
Front controller: intercept all the request. The DispatcherServlet class works as the front controller in the spring mvc.
Controller: business logic of an application.
View: display the information/data to user.

Workflow: All the request is intercepted by the DispatcherServlet.
The DispatcherServlet gets an entry of the hander mapping from the xml file and forwards the request to the controller
The controller returns an object of ModelAndView
The DispatcherServlet checks the entry of the view resolver in the xml file and invokes the specified view component

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 need.
It offers annotation based spring application.
Ease dependency management.
Include embedded servlet container -> Tomcat.

Spring boot starter: helps manage dependency.

Auto Configuration: automatically configures application based on the Jar

dependencies you added in the project.

Rest API design: application programming interface that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services.
Spring Restful API:
@RequestMapping: used for mapping http requests to methods.
Also @GetMapping, @PutMapping, @PostMapping, @DeleteMapping
@RequestParam: used for getting parameters in a http url after the question mark.
@PathVariable: used for getting parameters in a http url before the question mark.
@RequestBody: transfer json to java object.
@ResponseBody: transfer java object to json.
@Controller/RestController, @Service, @Repository: stereotypical aliases of @Component for different uses of different layers.
@Controller + @ResponseBody = @RestController.

5. Exception Handling Process

Local exceptions will be handled by the @ExceptionHandler annotation.
Global exceptions will be handled by the @ExceptionHandler inside the @ControllerAdvice class.
@ResponseStatus: last resort to check if a exception is annotated. HandlerExceptionResolvers will be evoked.

6. Validation

Constraints for certain fields in the record-object mapping such as @NotNull, @Max, @Min, @Pattern, @Email, etc.

7. Swagger

A debugging tool to test the user APIs.
.