1) IOC Container

a) Find Beans → It will look for @Component
b) Wire dependencies → @Autowired
c) Manage lifecycle of the bean
↳ Create, process & destroy

Find Beans

1) Find the beans by searching @Component, find their dependies and wire in.

```
@Component
public class ComplexAlgorithmImpl {

    @Autowired
    private SortAlgorithm sortAlgorithm;

}
```

dependent — spring will look for bean

```
@Component
public class QuickSortAlgorithm implements
        SortAlgorithm {

}
```

↳ It doesnot depends on other bean

Spring Framework create bean object

Class inject

↓

After injecting

II Terms related to IOC Container

Bean Factory

1) It is provided by Spring framework in order to perform basic functionality of IOC container

Application Context can be defined using
- xml
- java @Configuration

Application Context (IOC Container)
→ It has 4 functionality
1) Bean Factory features
2) Spring AOP features
3) I18n capabilities → Internationalization
4) WebApplicationContext for web applications etc
   - request scope
   - session scope

1) How does Spring Know where to search for Components or Beans?

```
@Configuration
@ComponentScan ( basePackage = {"com.org.repo",
                                "com.org.dto"} )
```

In spring boot, you don't need to write @ComponentScan, Why?

Main class

```
package com.org.todo;
```

com.org.todo.repo; com.org.todo.dto; ← Scan

This is base package

@SpringBootApplication → It triggers ComponentScan

```
public class TodoApplication
```

Note:

@Component ⇒ Spring should manage the bean
@Autowired ⇒ Spring should find the matching bean and wire the dependency in.

Note: If 1 JVM has multiple ApplicationContext then it will have multiple instance.

@Component → Generic Component,
It can be used anywhere.

@Controller – controller in MVC pattern
It is defined in web layer
→ Business Layer
@Service → Bussiness Service Facade logic
@Repository → dealing with database

Scope of Bean

Q3 Other Scopes available in Spring ?

Ans
a) Singleton :- One instance per Spring Context (Application Context (IOC
b) Prototype :- New bean whenever requested
c) request :- One bean per HTTP request.
Web-aware SpringApplicationContext.
d) session - One bean per HTTP session.
Web-aware SpringApplicationContext.

Note:
Q1) What is default scope of a bean
Ans singleton scope is the default scope in spring
Meaning → One bean per SpringApplicationContext.
Q2) Spring beans are thread safe ?
Ans By default, it is not thread safe.

Singleton
- Java → One instance per ClassLoader
- Spring → One instance per ApplicationContext (IOC container)

Constructor Injection → Creates immutable beans

Q) What are different types of Dependency Injections?
a) Setter injection
b) Constructor injection

Setter injection → Use this Optional Dependencies
→ happen through setter method

```
@Component
public class BussinessService {
    private DataService dataservice;

    @Autowired
    public void setDataService(DataService d)
    {  this.dataservice = d;
```

use the setter method to wire the dataService in

setter

@Autowired → Spring say use setDataService method to wire the dataservice in.

Constructor injection → Use this for Mandatory Dependencies
→ happen through constructor

```
@Component
public class BussinessService {
    private DataService dataservice;

    @Autowired
    public BussinessService(DataService data){
        super();
        this.dataService = data;
    }
```

When there is NO setter & No constructor Injection

Reflection → setter injection
→ Spring Framework uses Reflection

```
@Autowired
private DataService service;
```

1) How does Spring do Autowiring?
2) What are the difference kinds of matching used by spring for Autowiring.

Autowiring can be achieve by 3 way :—

→ byType
→ byName
→ constructor — similar to by Type, but through constructor Injection

constructor @Autowired
private SortAlgorithm sortAlgorithm.

1) byType - class or Interface
↳ classType ↳ implementation Type
search karega

```
@Component
public class ComplexAlgorithm {
    @Autowired
    private SortAlgorithm sortAlgorithm;
}
```

Note: Spring starts searching for implementation of SortAlgorithm interface, once it get then it autowired to ComplexAlgorithm

2) byName

```
@Autowired
private SortAlgorithm QuickSortAlgorithm;
```

Spring will look for the bean but if @component is missing here then → NoSuchBeanException
matches
Definition

```
@Component
public class QuickSortAlgorithm implements SortAlgorithm;
@Component
public class BubbleSortAlgorithm implements SortAlg-
```

Q When one interface has more than one implementation then

↳ NoUniqueBeanDefinitionException

```
@Component
public class CalculateService {
   @Autowired
   private FixedDeposit deposit;
```

or private FixedDeposit ~~SBI~~ BOI;

priority will be given

direct call dyega

```
@Component
@Primary
public class SBI implements FixedDeposit {
}

@Component
public class BOI implements FixedDeposit {
}
```

Qualifier

```
@Component
public class CalculateService {
  @Autowired
  @Qualifier("mainKey")
  private FixedDeposit deposit;


@Component
@Qualifier("mainKey")
public class BOI implements FixedDeposit
```

1) Spring 2.5 → Annotation released e.g @Component
2) Spring 3.0 → Java 5, Expression Language
3) Spring 4.0 → Java 8, Mini Java 6, @RestController, JCache

Context and Dependency Injection (CDI)

1) CDI is Java EE Dependency Injection Standard

CDI
@Inject
@Named
@Singleton

support by Spring Boost

@Autowired
@Component & @Qualifier
Define a scope of singleton

Design Patterns in Spring

1) Front Controller → Dispatcher Servlet
2) Prototype - Bean
3) Dependency Injection
4) Factory Pattern - Bean Factory & App^n Context
5) Template Method

Model and ModelAndView notes written end of the notes

@ModelAttribute
→ It invokes before @RequestMapping methods

@SessionAttributes

@InitBinder

@ControllerAdvice → It is for all ~~controller~~ common logic across ~~projects~~ like Exception handling

~~@ExceptionHandler~~

@ExceptionHandler → It is used to handle specific Exception

Spring MVC

Dispatcher Servlet
View Resolver
View
Model

Spring Boot

1) Why do we need Spring Boot

Ans Spring MVC have lots of configuration like Component Scan, dispatcher servlet, view resolver, hibernate configuration Entity Manager Factory, Message Converter

Spring Boot features

1) Auto Configuration — Different types of App → Web Application, Web services
2) Spring boot Starter projects
3) Embeded Server → tomcat
4) It works on microservices (independent)
5) Spring Boot Actuator → gives details of whole application

JPA, Like

@SpringBootApplication
→ @SpringBootConfiguration
→ This is configuration Spring boot Appl^n
→ This is going to create spring context

2850 → PANCard, 2 Photo, Lease Aggr...

> 78050 (B1) 82600 Lifecare Vasai general st...

@EnableAutoConfiguration
→ Whatever framework available at Classpath it enables by @SpringBootApplication

@ComponentScan
→ It automatically turn on the component scan to the sub packages of spring boot Application.

Spring JDBC

Regular JDBC	JDBC Template
Create Connection, PrepareStatement, also provide try catch in order to handle Checked exception, then close con, ps object	jdbcTemplate.update("query", dto parameter

2) RowMapper
If database column name is same as dto then, new BeanPropertyRowMapper(Todo.class)

else

```
Class TodoMapper implements RowMapper<Todo>
{
    @Override
    public Todo getMapped(ResultSet rs, int rowNum)
      throws SQLException {
        Todo todo = new Todo();
        todo.setId(rs.getInt("id"));
        todo.setUser(rs.getString("user"));

    } return todo;
}
```

Advantage of JPA :-

1) No need to write Query because JPA implementation (Hibernate) will write the query for us.

JPA (Java Persistance API)

→ JPA defines the mapping of Java objects to a table

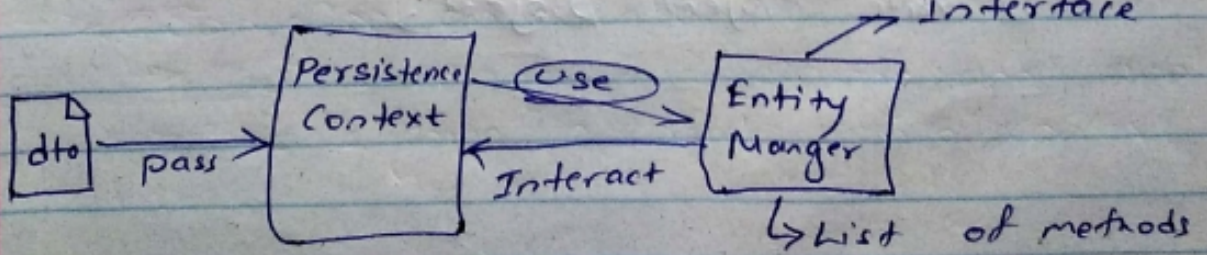
→ Hiberate is a JPA implementation
- Once the mapping of Java objects to the table is done,
  → JPA implementation will create the query for us.

Q) How to define entity in a JPA by using @Entity, then bean becomes entity and it is managed by JPA

Q) It is mandatory to @Table annotation ?
Ans No, if bean name and table name is same but if bean name is TodoDto and table name is TODO_MASTER the use @Table

Q How does Hibernate generate auto id
An @GeneratedValue (strategy = GenerationType.IDENTITY)

Q It is mandatory to use @Column ?
Ans No, if variable name is same as colums name of the table.



@PersistenceContext

One to One Relationship

```
@Entity
@Table (name = "Student")
public class Student {
    @OneToOne
    private Passport pasport;

}
@Entity
public class passport {
    @OneToOne (fetch = FetchType.Lazy, mappedBy = "pasport")
    private Student student;
}
```

One to Many ~~Retio~~ Relationship

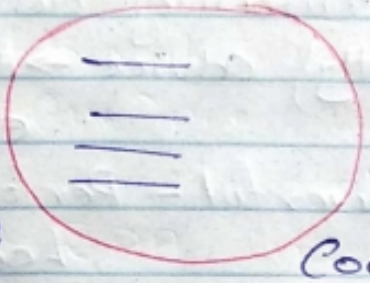
```
@Entity
public class Project {
    @OneToMany (mappedBy = "project")
    private List<Task> tasks;

@Entity
public class Task {
    @ManyToOne
    @JoinColumn(name = "PROJECT_ID")
    private Project project;
```

AOP Concepts

PointCut => What type of method to intercept.

```
@Around (value = "execution(* HelloService.*(..))")
public void around (ProceedingJoinPoint joinPoint)
                    throws Throwable {
    ---
    ---
    ---
    ---
}
```

Around Advice



→ Code is present here is Advice

Code that you want to execute is known as Advice

Aspect = Advice + PointCut } both combine

Joinpoint => specific result of one execution
=> Specific execution point of the aspect.
→ If it is called 100 times, it will have 100 different join points.

Weaving => It is a process of confirmation of Aspect being executed at right time.

Weaver => It is a AOP framework
→ Spring AOP is basic weaving
→ Spring AOP
→ AspectJ => It has feature more than method calling.

Advice Types

1) @Before → Before Advice
2) @AfterReturning → After returning Advice
→ After throwing advice
→ After (finally) advice → Alway executed

→ Around Advice → Most Powerful = Performance Logging

AspectJ vs Spring AOP

1) AspectJ is a full fledged AOP framework

Restfull (REST APIs)
REprestational State Transfer

1) Data Exchange Format
No Restriction - JSON is popular

2) Transport Layer
only HTTP

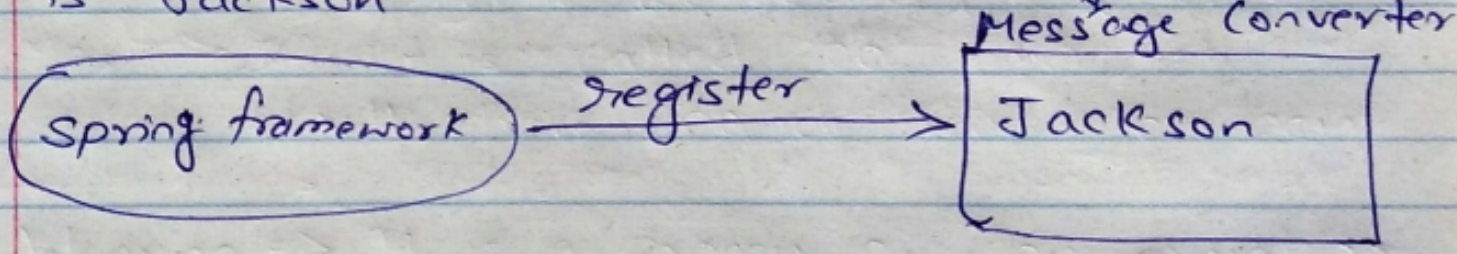
3) Service defination
Swagger.

Response Status

200 – Success

201 – Created → Object is created @ Server side
400 – Bad Request → Validation Error @ Server Side
404 – Resource Not Found
401 – UnAuthorised
500 – Server Error → Server side Exception throw kiya

1) How does we get data in JSON
→ Spring Framework register Message converter by default.
→ Default message converter for JSON format is Jackson



1) Model

```
e.g public String showLoginPage (ModelMap model)
{   model.put ("name", "CRM"); → adding data
    return "LoginPage"; → returning View name
}
```

2) ModelAndView

```
public ModelAndView showLoginPage( ) {
   ModelAndView model = new ModelAndView( );
  model.addObject ("name", "CRM");
  model.setViewName("LoginPage");
  return model;
}
```