Spring

1) What is Spring?

It is a lightweight, loosely coupled and integrated framework for developing enterprise applications in java.

2) What are the modules of spring framework?

1) Core Container 2) Data Access/Integration 3) Web (**MVC**) 4) AOP (Aspect Oriented Programming) 5) Instrumentation 6) Test

4) What are the advantages of spring framework?

1) Predefined Templates 2) Loose Coupling 3) Easy to test 4) Lightweight 5) Fast Development 6) Powerful Abstraction 7) Declarative support ~~8) Predefined Templates 9) Loose Coupling 10) Easy to test 11) Lightweight 12) Fast Development 14) Powerful Abstraction 15) Declarative support~~

6) What is IOC (Inversion of control)?

IOC is an object oriented practices where object coupling is bound at **run time**.

IOC behaves like factory pattern (Collection of an object is called factory pattern).

IOC container is having different object for xml file or managed bean.

It is responsible to instantiate, configure and assembled the object.

There are 2 type of IOC container:-

1. Application Context. 2. Bean Factory

7) What is the Differentiate between BeanFactory and ApplicationContext in spring?

No.	Application Context	Bean Factory
1)	ApplicationContext is the **advanced container**	BeanFactory is the **basic container**
2)	It extends the BeanFactory interface	It bundle of object
3)	It provides more facilities than BeanFactory such as integration with spring AOP	It does not provide
4)	It can have more than 1 config file	Only 1 config file or .xml file is possible with
5)	It has application layer specific context	It doesn't have
6)	It support life cycle events and validation	It does not support

8) What are the benefits of IOC?

The main benefits of IOC or dependency injections are:

1. It minimizes the amount of code in your application.
2. It makes your application easy to test as it doesn't require any singletons or JNDI lookup mechanisms in your unit test cases.
3. Loose coupling is promoted with minimal effort and least intrusive mechanism.
4. IOC containers support eager instantiation and lazy loading of services.

Q: How to call cache in spring ?

→

Scanned by CamScanner

9) What is Dependency Injection?
- It is a design pattern that removes the dependency from programing to xml file, so that it can be easily to manage

- It is make our programming to loosely couple

10) What are the different types of dependency injection?
- There are 3 types of dependency injection:

- Spring supports only first two categories of Injection

1. By Constructor Injection: - Here dependencies are provided as constructor parameters.
2. By Setter Injection: Dependencies are assigned through JavaBeans properties.
3. Interface Injection: Injection is performed through an interface and not supported in spring framework.

11) What is the difference between constructor injection and setter injection?

No.	Constructor Injection	Setter Injection
1)	No Partial Injection	Partial Injection
2)	Doesn't override the setter property	Overrides the constructor property if both are defined.
3)	Creates new instance if any modification occurs	Doesn't create new instance if you change the property value
4)	Better for too many properties	Better for few properties.

12) When to use Dependency Injections?
There are different scenarios where you Dependency Injections are useful:-

- When we need to inject configuration data into one or more component.
- When we need to inject the same dependency into multiple components.
- When we need to inject different implementation of the same dependency.
- When we need to inject the same implementation in different configuration.
- When we need some of the services provided by container.

13) When you should not use Dependency Injection?
There were scenarios where you don't need dependency injections e.g:-
- You will never need a different implementation.
- You will never need different configurations.

14) What is Bean Factory in Spring?
- A Bean Factory is like a factory class that contains collections of beans. The Bean Factory holds be definition of multiple beans within itself and then instantiates the bean when asked by client.

- Bean Factory is actual representation of the Spring IOC container that is responsible for contain and managing the configured beans.

15) What is Bean Life Cycle in Spring? (IPS-SPI-PBD)
1) Instantiate 2) Populate Properties 3) Set Bean Name 4) Set Beans Factory 5) Pre_Initialization 6) Initialization 7) Post_Initialization 8) Bean is ready to use 9) Destroy Bean

1) Instantiate: - Spring container instantiate the bean

Scanned by CamScanner

2) Populate Properties: - Spring Container inject the bean properties

3) Set Bean Name: - This method is used to set the bean name of bean spring container

4) Set Beans Factory: - This method is used to inject the bean factory object

5) Pre_Initialization: - It is called to give new bean instance before initialization

6) Initialize bean: - Init method is invoked by spring container to initialize bean

7) Post_Initialization: - It is called to give new bean instance after any bean initialization call back

8) Bean is ready to use: - Bean will be ready to use

9) Destroy Bean: - Destroy method is called to destroy the bean

16) What is Bean wiring (autowiring) in spring?
- Combining Bean with spring container is called bean wiring (**autowiring**).
- Autowiring enables the programmer to inject the bean automatically. We don't need to write explicit injection logic.

Let's see the code to inject bean using dependency injection.

```
1.   <bean id="emp" class="com.javatpoint.Employee" autowire="byName" />
```

There are (4-types or modes) of Auto-wiring supported:-

The "autodetect" mode is deprecated since spring 3.

1) no – Default 2) byName 3) byType 4) Constructor

No.	Mode	Description
1)	no – Default	- This is the default mode, it means autowiring is not enabled. no auto wiring, set it manually via "ref" attribute. Ex:- <bean id="customer" class="com.test.autowire.Customer"> <property name="person" ref="person" /> </bean> <bean id="person" class="com.test.autowire.Person" />
2)	byName	- Injects the bean based on the property name. It uses setter method. If the name of a bean is same as the name of other bean property, auto wire it. Ex:- <bean id="customer" class="com.test.autowire.Customer" autowire="byName"/> <bean id="person" class="com.test.autowire.Person" />
3)	byType	- Injects the bean based on the property type. It uses setter method. Ex:- <bean id="customer" class="com.test.autowire.Customer" autowire="byType"/> <bean id="person" class="com.test.autowire.Person" />
4)	Constructor	- It injects the bean using constructor Ex:- <bean id="customer" class="com.test.autowire.Customer" autowire="autodetect"/> <bean id="person" class="com.test.autowire.Person" />

Scanned by CamScanner

17) List the limitations of auto wiring.

Autowiring has the following limitations:
1. Overriding and auto wiring are caused due to dependency in property and constructor argument setting.
2. Less precise as compared to explicit wiring.
3. Tools that generate documentation using a spring might not have access to wiring information.
4. There is a possibility of clash between bean definitions and argument or method to be wired.
5. The problem does not occur much in case of maps, arrays and collection and cannot be resolved randomly for dependencies which expect one value.

18) In scenarios where you have to avoid using auto wiring, what are the other options that can be explored to achieve the same?

In scenarios where using autowiring is prohibited, the following replacements can achive the same :
1. Abandon autowiring for favor of explicit wiring.
2. Avoid autowiring for a bean definition by setting the autowire-candidate for attributes to false as described in the next section.
3. Set a single bean definition as the primary candidate by setting the primary attribute of its <bean/> element to true.
4. When Java 5 or later is used, implementation needs more fine-grained control available with annotation-based configuration.

19) What are the different bean scopes in spring?

- There are 5 bean scopes in spring framework. (SSP-GR)

No.	Scope	Description
1)	singleton	The bean instance will be only once and same instance will be returned by the IOC container. It is the default scope.
2)	prototype	The bean instance will be created each time when requested.
3)	request	The bean instance will be created per HTTP request.
4)	session	The bean instance will be created per HTTP session.
5)	globalsession	The bean instance will be created per HTTP global session. It can be used in portlet context only.

20) In which scenario, you will use singleton and prototype scope?

- You should use the prototype scope for all beans that are stateful and the singleton scope should be used for stateless beans

Scanned by CamScanner

21) What is the difference between singleton and prototype bean?

- Mainly it is the scope of a beans which defines their existence on the application
  Singleton: It means single bean definition to a single object instance per Spring IOC container.
  Prototype: It means a single bean definition to any number of object instances.

22) How do beans become 'singleton' or prototype?

- There exists an attribute in bean tag, called 'singleton'.
- If it is marked 'true', the bean becomes 'singleton'.
- If it is marked 'false', the bean becomes 'prototype'

23) What are the transaction management supports provided by spring?

- Spring framework provides two type of transaction management supports:

1. **Programmatic Transaction Management**: should be used for few transaction operations.
2. **Declarative Transaction Management**: should be used for many transaction operations.

24) How do you access Hibernate using Spring?

There are two ways to Spring's Hibernate integration:

1. By Inversion of Control with a HibernateTemplate and Callback.
2. By extending HibernateDaoSupport and Applying an AOP Interceptor.

25) How would you integrate Spring and Hibernate using HibernateDaoSupport?

This can be done through Spring's SessionFactory called LocalSessionFactory. The steps in integration process are:

1. Configure the Hibernate SessionFactory.
2. Extend your DAO Implementation from HibernateDaoSupport.
3. Wire in Transaction Support with AOP.

26) What is JdbcTemplate in Spring? And how to use it?

Less code:- The JdbcTemplate class is the main class of the JDBC Core package. The JdbcTemplate (The class internally use JDBC API) helps to eliminate lot of code you write with simple JDBC API (Creating connection, closing connection, releasing resources, handling JDB Exceptions, handle transaction etc.). The JdbcTemplate handles the creation and release of resources, which helps you to avoid common error like forgetting to close connection.

You can execute the query directly.

- Examples :-

1. Getting row count from database.

int rowCount = this.jdbcTemplate.queryForObject("select count(*) from t_employee", int.class);

2. Querying for a String.

Scanned by CamScanner

```
String lastName = this.jdbcTemplate.queryForObject( "select last_name from t_employee where
Emp_Id = ?", new Object[]{377604L}, String.class);
```

3. Querying for Object

```
Employee employee = this.jdbcTemplate.queryForObject( "select first_name, last_name from
t_employee where Emp_Id = ?", new Object[]{3778604L}, new RowMapper()

{

public Employee mapRow(ResultSet rs, int rowNum) throws SQLException {

Employee employee = new Employee();

employee.setFirstName(rs.getString("first_name"));

employee.setLastName(rs.getString("last_name"));

return employee;

}

});
```

4. Querying for N number of objects.

```
List<Employee> employeeList = this.jdbcTemplate.query("select first_name, last_name from
t_employee", new RowMapper<Employee>() {

public Employee mapRow(ResultSet rs, int rowNum) throws SQLException

{

  Employee employee = new Employee();

  employee.setFirstName(rs.getString("first_name"));

  employee.setLastName(rs.getString("last_name"));

  return employee;

}

});
```

27) What are classes for spring JDBC API?

1. JdbcTemplate
2. SimpleJdbcTemplate
3. NamedParameterJdbcTemplate
4. SimpleJdbcInsert
5. SimpleJdbcCall

Scanned by CamScanner

28) How can you fetch records by spring JdbcTemplate?

You can fetch records from the database by the **query method of JdbcTemplate**. There are two interfaces to do this:-

1. ResultSetExtractor
2. RowMapper

29) What is the advantage of NamedParameterJdbcTemplate?

NamedParameterJdbcTemplate class is used to pass value to the named parameter. A named parameter is better than? (question mark of PreparedStatement).

It is **better to remember**.

30) What is the advantage of SimpleJdbcTemplate?

The **SimpleJdbcTemplate** supports the feature of var-args and autoboxing

» Spring AOP Interview Questions:-

31) What is AOP?

- It is a programming technique that allow developers to modularize cross cutting concern_cuts across the typical division of responsibilities.

Example:-

1) Logging (Difference in different logging more then 1)
2) Transaction management
3) Authentication {a) IP check b) user check c) data check}
4) Security

* AOP break the programming logic into distinct part (**called concern**)

-It is use to increase modularity by cross cutting concern:-

Example:-

If we have **"5 method"** of similar functionality and we want to maintained **log** then we have to write Log programing in each method but with AOP will write additional method for Log and maintained the relationship in XML file. Which can be managed easily.

32) What are the AOP terminology?

AOP terminologies or concepts are as follows: - **(IIT AAP WAJ)**

Scanned by CamScanner

1) Introduction 2) Interceptor 3) Target Object 4) Advice 5) Aspect 6) Pointcut 7) Weaving 8) AOP Proxy 9) JoinPoint

1) Introduction: - Introduction represents introduction of new fields and methods for a type.

2) Interceptor: - Interceptor is a class like aspect that contains one advice only.

3) Target Object: - Target Object is a proxy object that is advised by one or more aspects.

4) Advice: - Advice is a code implementation of aspect at particular join point.
Type of Advice: - (**It is a code or implementation**)
a) Before Advice b) After Advice c) throw Advice d) Around Advice.

5) Aspect: - Aspect is a class in spring AOP that contains advices and joinpoints.

6) Pointcut: - Pointcut is expression language of Spring AOP.

7) Weaving: - It is a process of linking aspect to other application type or object to create advice object at run time

8) AOP Proxy: - An object which is created by AOP framework for implementing the aspect contracts.

9) JoinPoint: - It is a program like method execution, exception handling, data access etc.

33) Does spring perform weaving at compile time?

-No, spring framework performs weaving at runtime.

» Spring MVC Interview Questions:-

34) What is the front controller class of Spring MVC?

The **DispatcherServlet** class works as the front controller in Spring MVC.

35) What does @Controller annotation?

The **@Controller** annotation marks the class as controller class. It is applied on the class.

36) What does @RequestMapping annotation?

Scanned by CamScanner

The **@RequestMapping** annotation maps the request with the method. It is applied on the method.

37) What does the ViewResolver class?

The **View Resolver** class resolves the view component to be invoked for the request. It defines prefix and suffix properties to resolve the view component.

38) Which ViewResolver class is widely used?

The **org.springframework.web.servlet.view.InternalResourceViewResolver** class is widely used.

39) Does spring MVC provide validation support?

Yes.

40) How to control current session on java web application? Using spring Security.

We can use spring security to control number of Active session in java web Application. Spring security provides **"Out of Box"** and **"When Enabled"**. A user can only have only once Active Session at a time.

41) What is limitation of Autowiring?

Limitations of autowiring are:

Overriding possibility: You can still specify dependencies using <constructor-arg> and <property> settings which will always override autowiring.

Primitive data types: You cannot autowire so-called simple properties such as primitives, Strings, and Classes.

Confusing nature: Autowiring is less exact than explicit wiring, so if possible prefer using explicit wiring.

42) What is Spring MVC?

Example with diagram:-

Scanned by CamScanner

No.	Annotation	USE	Description
1)	@Autowired	Constructor, Field, Method	The @Autowired annotation provides more fine-grained control over where and how autowiring should be accomplished. It can be used to autowire bean on the setter method just like @Required annotation, on the constructor, on a property or pn methods with arbitrary names and/or multiple arguments.
2)	@Configurable	Type	The bean instance will be created each time when requested.
3)	@Order	Type , Method, Field	Defines ordering, as an alternative to implementing the org. springframework.core.Ordered interface.
4)	@Qualifier	Field, Parameter, Type, Annotation Type	When there are more than one beans of the same type and only one is needed to be wired with a property, the @Qualifier annotation is used along with @Autowired annotation to remove the confusion by specifying which exact bean will be wired.
5)	@Required	Method(Setters)	This annotation simply indicates that the affected bean property must be populated at configuration time, through an explicit property value in a bean definition or through autowiring. The container throws BeanInitializationException if the affected bean property has not been populated.
6)	@Scope	Type	Specifies the scope of a bean, either singleton, prototype, request, session, or some custom scope.

Scanned by CamScanner