

ANGULAR

What are services in Angular, and why are they used?

1. What is AOT compilation in Angular?
2. Talk about the key differences between components and templates in Angular.
3. What is the use of Angular CLI?
4. Is it possible to use an HTML template in Angular?
5. What are the benefits of using unit tests in Angular, and how is it done?
6. What's the difference between automatic and manual bootstrapping?
7. What's the difference between one-way and two-way data binding in Angular?

1. Why was Angular introduced as a client-side framework?
2. What is the purpose of a filter in Angular?
3. What is the role of decorators in AngularJS?
4. Describe different types of filters in Angular?
5. How are DOM elements different from BOM?
6. Describe the advantages that Angular has over other frameworks.
7. What is a root component, and what role does the component play in Angular?
8. What is Angular Material, and why is it necessary for building web applications?
9. Does Angular 6 support the conversion of angular components into web components? What are some of the other new aspects of Angular 6?
10. Explain string interpolation in Angular. What is added within double curly braces?

1. Talk about the building blocks of Angular 5.
2. What is the use of viewEncapsulation?
3. What is routing in Angular 5?

4. What are the ways in which data binding can be done?
5. What is an AsyncPipe?
6. What is Redux?
7. Provide proper generation logic of Component, Pipe, Class, Directive, Service, and Module for Angular 5.
8. Explain the proper lifecycle hooks for the application development in Angular 5?
9. How do you differentiate Constructors from OnInit?
10. Are there any differences between Activated Route and Router Route in Angular 5?

Angular 4 Interview Questions

See the interview questions for Angular 4 mentioned below:

1. Explain what the digest cycle is in Angular?
2. What does lean component mean in Angular?
3. Explain the differences between scan() and reduce () in Angular?
4. How will you handle HTTP error responses in Angular 4?
5. What is the use of a subscribe method in Angular 4?
6. What are the different directives in Angular 4?
7. Define unit testing in Angular 4?
8. What is ElementRef in angular 4?
9. Why is deep linking important in Angular 4?
10. Explain the process of sending and setting cookies in Angular 4.

Note: Angular 4 was announced in 2016, skipping version 3 to avoid confusion due to the misalignment of the router package's version, which was already distributed as v3.3.0.

Angular 5 Interview Questions

1. Talk about the building blocks of Angular 5.
2. What is the use of viewEncapsulation?

3. What is routing in Angular 5?
4. What are the ways in which data binding can be done?
5. What is an AsyncPipe?
6. What is Redux?
7. Provide proper generation logic of Component, Pipe, Class, Directive, Service, and Module for Angular 5.
8. Explain the proper lifecycle hooks for the application development in Angular 5?
9. How do you differentiate Constructors from OnInit?
10. Are there any differences between Activated Route and Router Route in Angular 5?

Angular 6 Interview Questions

1. Tell us about the various types of filters in Angular 6?
2. Explain what a safe navigation operator in Angular 6 is?
3. Why do we have @input and @output in Angular?
4. What is Route Guards in Angular 6?
5. What is template expression in Angular 6? Explain using syntax.
6. What is bazel and closure compiler in Angular 6?
7. Explain what is traceur compiler in Angular 6.
8. Explain the differences between angular service and factory.
9. What is a zone in Angular 6?
10. Explain the differences between constructor and ngOnInit in Angular 6?

For more Angular 6 interview questions, read our article on [Angular 6 Interview Questions](#) for a comprehensive list of sample questions.

Angular 7 and Angular 8 Interview Questions and Answers

Browse through the interview questions and answers to get a better sense of what to expect:

1. Explain subscribing and multicasting in Angular?

When someone subscribes, at the moment, the observable instance aids in the development of the values. In contrast, broadcasting the list of multiple subscribers in a single execution is known as multicasting. The only observable here is that there are no registered listeners on the document. Whatever happens, the values can be sent to each subscriber.

2. What are the utility functions of RxJS?

The functions include:

- `map()` : Used to map values of different data types
- `filter()` : Used for filtering streams
- `concat()` : Used to concatenate multiple strings
- `merge()`: Used to recursively descend into object properties in the source copy while forming a deep copy of the same.

3. Explain the difference between Parameterized Pipes and Chaining Pipes.

In Angular, any number of parameters can be passed to the pipe using a colon (:), this method is known as Angular Parameterized Pipes. Whereas Chaining Pipes involves chaining multiple pipes together by associating more than one pipe and then transforming the final output with all the pipes applied.

4. What are the types of Forms supported by Angular 7?

There are two types of Forms available in Angular 7.

- **Reactive Forms:** It uses an explicit and immutable approach to managing the state of a form at a given point in time.
- **Template-driven Forms:** All logics, validations, controls are written in the template part.

5. What is NgUpgrade?

A library in Angular that allows us to upgrade Angularjs (1.X) application to Angular gradually. It lets you run Angular side-by-side along with AngularJS without breaking the application. NgUpgrade can be installed using the npm command `npm install @angular/upgrade -- save`.

6. **List out the advantages of Bazel in Angular 8?**

- You can use the same tool for both frontend and backend.
- There are incremental build and test options.
- You can carry out more customization and on Bazel.

7. **What is the reason for using template-driven forms in Angular 8?**

This is to support the addition of simple forms to applications where scalability is not an issue.

8. **How will you check the type of value assigned to a given variable in Angular 8?**

In Angular 8, the type of checks can be used for checking the kind of value assigned to a given variable.

9. **Explain the use of NgUpgrade in Angular 8?**

It is one of the upgrade libraries of Angular 8, mostly used to integrate both Angular and AngularJS components in an application. It helps bridge the gap between the Dependency Injection Systems in AngularJS and Angular.

10. **What is Ivy in Angular 8?**

It is the render engine of Angular 8. It makes applications simpler, smaller, and faster and also reduces the bundle size by at least 30%.

Check out our article on Angular 7 interview questions for more sample questions.

Angular Interview Questions for Beginners

Beginners can expect to be grilled about the basics of the Angular framework. Some of the sample questions include:

1. What is data binding? Which type of data binding does Angular deploy?

2. Differentiate between Angular and AngularJS
3. What are annotations in Angular?
4. What is the use of templates in Angular?
5. What is an AOT compilation, and what are its advantages?
6. Why are there Pipes in Angular?
7. What is an ngModule?
8. What is the meaning of scope in Angular?
9. What are Single Page Applications (SPA)?
10. What is Typescript?

Angular Security Interview Questions

1. How to prevent Cross-Site Scripting (XSS) in Angular?
2. How Angular Protects from XSS Attacks?
3. From a security standpoint, what are the key points to bear in mind when you are developing Angular apps?
4. What is Transpiling?
5. Describe the Angular Security best practices.
6. Why should we implement CSP?
7. How will you prevent Cross-site request forgery on Angular?
8. Why should you avoid modifying angular copy?
9. Do you recommend using DOM's APIs directly?
10. What are reducers in Angular 4?

Angular Interview Questions for a Senior Developer

1. Describe what happens when a double click event takes place.
2. What is the difference between prefix \$ and \$\$?
3. What is a SPA in AngularJS?
4. Explain scope object in AngularJS
5. How can you optimize Angular performance?

6. Talk about the differences between Constructor and ngOnInit?
7. How will you boot AngularJS?
8. Define "Page not found" route
9. Design a component with a button click to show and hide text by clicking.
10. Define JIT and AOT.

Angular Design Patterns Interview Questions

1. Why should someone use design patterns?
2. What is an observable pattern?
3. Define Proxy pattern.
4. Describe View Pattern.
5. What are the Web Design Patterns in Angular 8?
6. Is Angular an MVC design pattern?
7. Describe dependency injection DI.
8. What is the difference between a factory and abstract factory patterns?
9. What is a Prototype Pattern?
10. What is a Singleton Pattern?

Angular vs. React Interview Questions

1. What is Angular?
2. What is React?
3. When it comes to DOM, explain the differences between Angular and React.
4. What are the 5 main advantages of React?
5. What are the shortcomings of Angular?
6. Is Angular better than React?
7. Why do we consider React to be faster than Angular?

UI Developer Angular Interview Questions

1. Explain the basic steps to unit test an AngularJS filter?

2. How is data shared between controllers?
3. Explain the importance of a digest cycle.
4. What directive would you recommend to hide elements from the HTML DOM by removing them from that DOM not changing their styling?
5. What are ngIf and ngFor?
6. How is metadata represented in Angular?

Advanced Angular Interview Questions

1. How do you write unit test cases in Angular?
2. What is the difference between Scan and Reduce in RxJS?
3. What is the difference between flatmap and concatmap in Angular?
4. Name the security principles in Angular development.
5. How does Node.js overcome the problem of blocking of I/O operations?
6. What is Angular Ivy?
7. What is NGXS?
8. Describe AsyncPipe in Angular?
9. Describe the Core Dependencies of Angular 7?

Angular Architecture Interview Questions

1. Describe the building blocks of Angular applications.
2. Explain the need for factory function in AngularJS.
3. Create a pictorial representation of Angular architecture.
4. How is Dependency Hierarchy created?
5. What is HttpClient and its benefits?
6. What is the purpose of the base href tag?
7. What are Single Page Applications, and how do they function in Angular?
8. What function is called when an object is created in TypeScript?
9. How similar is AngularJS to Angular 2?
10. How to perform error handling?

Angular Framework Interview Questions

1. Why have client-side frameworks like Angular been introduced?
2. Which framework should you choose if you want to design mobile applications on AngularJS?
3. What is Angular Expression? Explain the key difference between angular expressions and JavaScript expressions.
4. What is a bootstrapping module?
5. What is the purpose of module.exports?
6. How is Node.js better than other frameworks most popularly used?
7. Explain injector in AngularJS.
8. Explain host view in Angular.
9. Explain property binding in Angular.
10. Explain the way of sharing data between components.

AngularJS and Node.js Interview Questions

1. List the advantages and disadvantages of AngularJS
2. Is AngularJS dependent on JQuery?
3. What do you mean by directives in AngularJS?
4. What role do controllers play in AngularJS?
5. What is a module in AngularJS?
6. What is the reason for Node.js to be single-threaded?
7. Describe event-loop in Node.js?
8. Explain the manner in which Node.js overcomes the problem of blocking of I/O operations?
9. What is the main reason behind separating express app and server?
10. Explain the exit codes of Node.js?

Angular Routing Interview Questions

1. Explain how the Angular router works.
2. How do Angular JS routes work?

3. Describe how you will initialize a select box with options on page load?
4. What is an Angular router?
5. Talk about the directive used for rendering a matched component in angular routing?
6. Demonstrate navigating between different routes in an Angular application.
7. Point out the differences between ActivatedRoute and RouterState, with reference to Angular.
8. What is an activated route?
9. What are router events?
10. What is the reason for using the Wildcard route?

RxJS Angular Interview Questions

1. Talk about the most outstanding features of RxJS
2. Explain the biggest advantages of Reactive Programming
3. What are the core principles of Redux?
4. Explain if there are any similarities between Redux and RxJS?
5. Explain the differences between BehaviorSubject and Observable in RxJS?
6. Define an Observable in RxJS
7. What are the advantages of RxJS Observables over RxJS Promises?
8. What does one mean by Asynchronous when the term is used in the context of RxJS or Reactive programming?
9. What is the role of a subject in RxJS?
10. What is RxJS Map? Explain the use of Higher-Order Observable Mapping.

Difficult Angular Interview Questions

If you are an advanced software developer, expect the questions to get a lot harder. Based on the feedback received from our alumni, here are some questions previously asked by interviewers:

1. What is ngOnInit? Can you define it?
2. What is Bootstrap, and how will you embed it in Angular?

3. Tell us the type of DOM that Angular implements?
4. Explain the difference between Reactive and Template forms?
5. What is the role of Promises and Observables in Angular?
6. Explain the @Component Decorator.
7. Explain the common modules and ngmodules automatically added in angular apps? What errors will occur if these are not added?
8. How will you minify Angular applications?
9. What is :host property in CSS in Angular application?
10. Explain the benefits of statics class.

How to Prepare for Angular Interview Questions

Mentioned below are a few tips on how you can prepare for your Angular interview:

- Working on Angular requires proficiency in different languages. Hence, be thorough with various programming languages, including JavaScript, TypeScript, and HTML coding.
- Thoroughly research the company you are interviewing to understand the different applications that would need the angular framework.
- Schedule as many mock coding interviews as you can to brush up on your programming abilities.
- Since Angular has various iterations, you need to prepare thoroughly for each of them. As a **software developer**, you should be able to show the recruiters that you understand the fine differences between each iteration and are up-to-date with the latest developments. Remember that even though the old iterations will not be developed any further, they still receive support. That's why the interviewer will be keen to know how thorough you are with various versions that **Angular supports**.
- If you are applying for front-end developer roles, you also need to be familiar with basic data structure concepts.

Q1. What is a FormBuilder? How will you import the

FormBuilder into your project?

The FormBuilder provides a syntactic sugar that speeds up FormControl, FormGroup, and FormArray objects' creation. It shortens creating instances and reduces the requirement of boilerplate code to build complex forms. Moreover, it also provides easy-to-use methods for control generation.

You can take the following steps to use the FormBuilder service:

- Import the FormBuilder into your project by using the following command: **import { FormBuilder } from '@angular/forms';**
- Once you inject the FormBuilder service, you can create the form's contents.

Q2. How will you update the view if your model data is updated outside the 'Zone'?

You can use the following options to update your view:

- **ApplicationRef.prototype.tick():** It performs change detection on the entire component tree.
- **NgZone.prototype.run():** It also performs change detection on the complete component tree. Moreover, the run() calls the tick itself. The parameter takes the function before tick and executes it.
- **ChangeDetectorRef.prototype.detectChanges():** It launches the change detection on the current component, including its children.

Q3. How are observables different from promises?

The following table enumerates the key differences between observables and promises.

Observables	Promises
They handle multiple asynchronous events over a period of time.	They deal with a single asynchronous event at a time.
They are lazy, i.e., they are not executed until we use the subscribe() method.	They are not lazy, i.e., they are executed immediately after creation.
Cancellable subscriptions using the unsubscribe() method.	Promises are not cancellable.
They deliver errors to the subscribers.	They push the errors to child promises.
Observables provide operations such as reduce, filter, retry, and retryWhen.	They do not provide operations.

Q4. Why is there a need for compilation? What are the two types of compilation in Angular?

Angular applications need to be compiled as every application has components and templates that are incomprehensible to the browser. So, it is essential to compile them before running them inside the browser.

The two types of compilations are:

- **Just-in-Time or JIT compilation:** The application compiles inside the browser during runtime.
- **Ahead-of-Time or AOT compilation:** The application compiles during the build time.

Q5. What are the advantages of using AOT compilation?

There are several advantages of using AOT compilation. Some advantages are as follows:

- The application compilation occurs before running inside the browser, allowing the browser to load the executable code and immediately render the application. Thus, AOT provides faster rendering.

- AOT needs fewer ajax requests since the compiler sends the external HTML and CSS files with the application; it eliminates the requirement of separate AJAX requests for those source files.
- It minimizes errors as they can be detected and handled during the building phase.
- It provides better security to the applications as the compiler adds HTML and templates into the JS files beforehand, and there are no extra HTML files to be read.

Q6. What is dependency injection?

Dependency injection is one of the primary Angular concepts. It is an application design pattern. Dependencies are services that have the functionality required by various components in an application. Angular provides a seamless mechanism for making dependencies that are injectable across the various components of an application.

Q7. What do you understand about Angular CLI?

Angular CLI stands for Angular command-line interface. It lets you initialize, develop, scaffold, and maintain web applications directly from a command shell. You can also use CLI to create components, services, pipes, and directives. It assists in building, testing, and serving, thus making the Angular development workflow quicker and much easier.

Q8. What is HttpClient?

HttpClient is a popular Angular module. You can use it for communicating with a backend service through the HTTP protocol. Although promises are useful, they lack certain functionalities that observables offer. The use of HttpClient in Angular returns the data as an observable, thus allowing us to subscribe, unsubscribe, and perform operations.

Q9. What is multicasting?

Using the HttpClient module, you can communicate with a backend service and fetch data. Multitasking allows you to broadcast this fetched data to multiple subscribers in one execution. It is beneficial when we have multiple application parts waiting for some data. You require an RxJS subject to use the multicasting feature.

Q10. How will you set, get, and clear cookies in Angular?

You will have to include the module: ngCookies angular-cookies.js in Angular for using cookies.

- You can set cookies in a key-value format using the 'put' method.

cookie.set('nameOfCookie','cookieValue');

- You can retrieve the cookies using the 'get' method.

cookie.get('nameOfCookie');

- You can clear Cookies using the 'remove' method.

cookie.delete('nameOfCookie');

Recommended Reading: [Angular interview questions you must prepare for in 2022.](#)

Latest Angular Interview Questions for Experienced Professionals

You must prepare the latest Angular interview questions and answers for experienced developers. Here is a list of some interview questions for the latest versions of Angular.

Angular 9 Interview Questions

1. How is Constructor different from ngOnInit?
2. What's new about Angular 9?
3. What were the improvements in the tree shaking in Angular 9?
4. What is new in Angular ivy in Angular 9?
5. Explain the new options for 'providedIn' in the 9th version.

Angular 10 Interview Questions

1. Why was bazel deprecated in version 10?
2. How is Angular 10 better than the previous version?
3. How does Angular 10 boost ngcc performance?

4. What are the advantages of choosing Angular 10?

Angular 11 Interview Questions

1. How does Angular 11 offer support for Hot Module Replacement?
2. What is an operation byelog?
3. What is automatic font inlining in Angular 11?
4. What is the use of the parallel function?
5. How does the manualChangeDetection function impact change detection in Angular 11?
6. Why did Angular 11 promote ESLint over TSLint?

Angular 12 and 13 Interview Questions

1. What is the component test harness in Angular 12?
2. What is a nullish coalescing operator?
3. What should you migrate to Angular 12?
4. Why is the View Engine no longer available in Angular 13?
5. Why has IE 11 support been removed in Angular 13?

SPRING/SPRINGBOOT

General Questions – Spring Interview Questions

1. What are the major features in different versions of Spring Framework?

Features of Spring Framework

Version	Logo	Feature
Spring 2.5		This version was released in 2007. It was the first version which supported annotations.
Spring 3.0		This version was released in 2009. It made full-fledged use of improvements in Java5 and also provided support to JEE6.
Spring 4.0		This version was released in 2013. This was the first version to provide full support to Java 8.

2. What is a Spring Framework?

- Spring is a powerful open source, application framework created to reduce the complexity of enterprise application development.
- It is light-weighted and loosely coupled.
- It has layered architecture, which allows you to select the components to use, while also providing a cohesive framework for J2EE application development.
- Spring framework is also called the framework of frameworks as it provides support to various other frameworks such as Struts, Hibernate, Tapestry, EJB, JSF etc.

3. List the advantages of Spring Framework.

- Because of Spring Frameworks layered architecture, you can use what you need and leave which you don't.
- Spring Framework enables POJO (Plain Old Java Object) Programming which in turn enables continuous integration and testability.
- JDBC is simplified due to Dependency Injection and Inversion of Control.
- It is open-source and has no vendor lock-in.

4. What are the different features of Spring Framework?

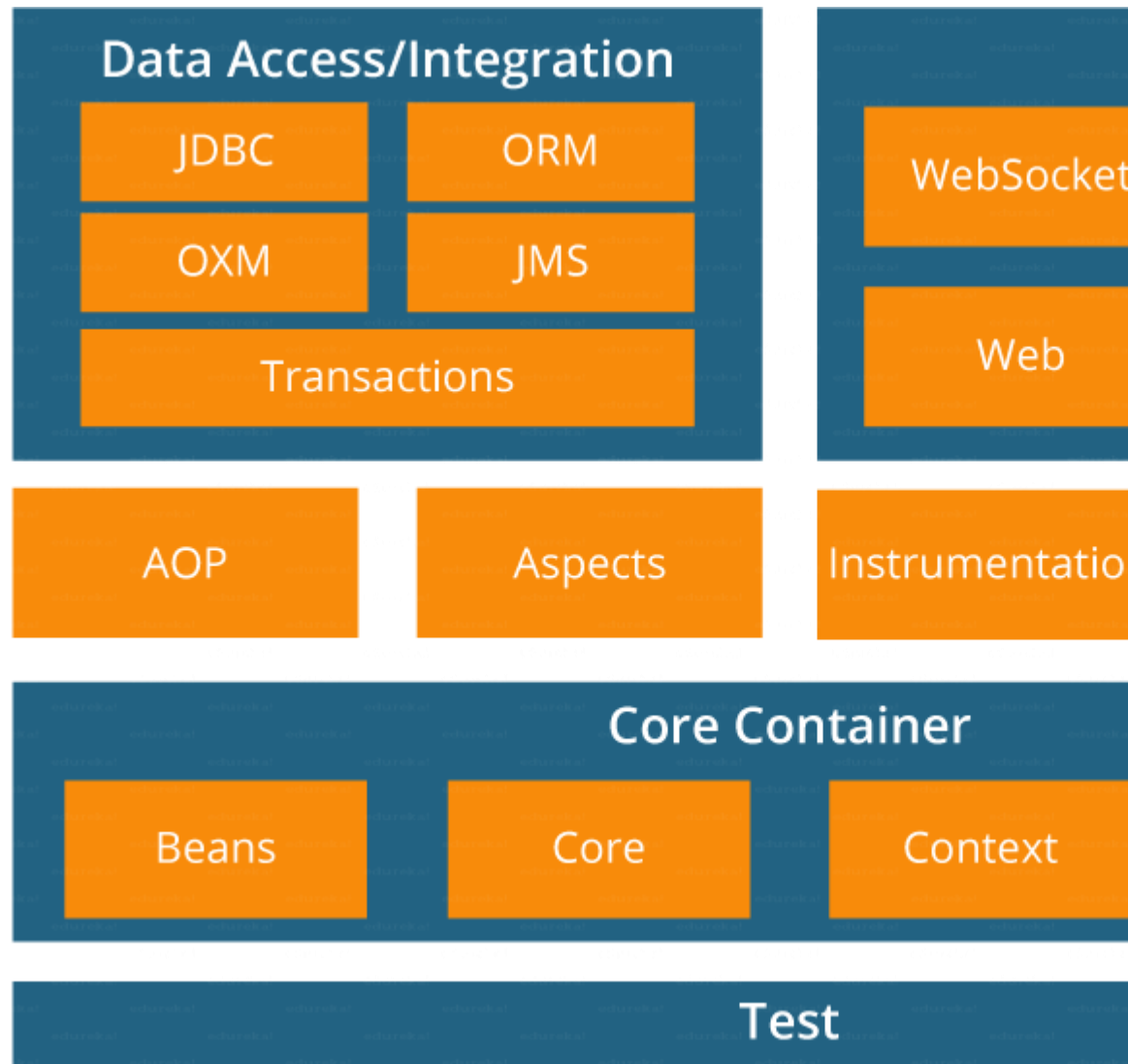
Following are some of the major features of Spring Framework :

- **Lightweight:** Spring is lightweight when it comes to size and transparency.
- **Inversion of control (IOC):** The objects give their dependencies instead of creating or looking for dependent objects. This is called Inversion Of Control.
- **Aspect oriented Programming (AOP):** Aspect oriented programming in Spring supports cohesive development by separating application business logic from system services.
- **Container:** Spring Framework creates and manages the life cycle and configuration of the application objects.
- **MVC Framework:** Spring Framework's MVC web application framework is highly configurable. Other frameworks can also be used easily instead of Spring MVC Framework.
- **Transaction Management:** Generic abstraction layer for transaction management is provided by the Spring Framework. Spring's transaction support can be also used in container less environments.
- **JDBC Exception Handling:** The JDBC abstraction layer of the Spring offers an exception hierarchy, which simplifies the error handling strategy.

5. How many modules are there in Spring Framework and what are they?

There are around 20 modules which are generalized into Core Container, Data

Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation and Test.



- **Spring Core Container** - This layer is basically the core of Spring Framework. It contains the following modules :

1. Spring Core
2. Spring Bean
3. SpEL (Spring Expression Language)
4. Spring Context

- **Data Access/Integration** – This layer provides support to interact with the database. It contains the following modules :
 1. JDBC (Java DataBase Connectivity)
 2. ORM (Object Relational Mapping)
 3. OXM (Object XML Mappers)
 4. JMS (Java Messaging Service)
 5. Transaction
- **Web** – This layer provides support to create web application. It contains the following modules :
 1. Web
 2. Web – MVC
 3. Web – Socket
 4. Web – Portlet
- **Aspect Oriented Programming (AOP)** – In this layer you can use Advices, Pointcuts etc., to decouple the code.
- **Instrumentation** – This layer provides support to class instrumentation and classloader implementations.
- **Test** – This layer provides support to testing with JUnit and TestNG.

Few Miscellaneous modules are given below:

- **Messaging** – This module provides support for STOMP. It also supports an annotation programming model that is used for routing and processing STOMP messages from WebSocket clients.
- **Aspects** – This module provides support to integration with AspectJ.

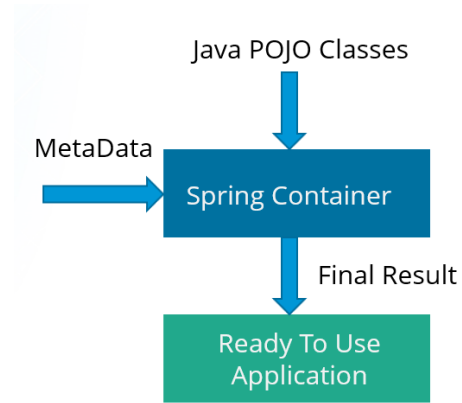
6. What is a Spring configuration file?

A Spring configuration file is an XML file. This file mainly contains the classes information. It describes how those classes are configured as well as introduced to each other. The XML configuration files, however, are verbose and more clean. If it's not planned and written correctly, it becomes very difficult to manage in big projects.

7. What are the different components of a Spring application?

A Spring application, generally consists of following components:

- Interface: It defines the functions.
- Bean class: It contains properties, its setter and getter methods, functions etc.
- Spring Aspect Oriented Programming (AOP): Provides the functionality of cross-cutting concerns.
- Bean Configuration File: Contains the information of classes and how to configure them.
- User program: It uses the function.



8. What are the various ways of using Spring Framework?

Spring Framework can be used in various ways. They are listed as follows:

1. As a Full-fledged Spring web application.
2. As a third-party web framework, using Spring Frameworks middle-tier.
3. For remote usage.
4. As Enterprise Java Bean which can wrap existing POJOs (Plain Old Java Objects).

The next section of Spring Interview Questions is on Dependency Injection and IoC container.

Dependency Injection/ IoC Container – Spring Interview Questions

9. What is Spring IOC Container?

At the core of the Spring Framework, lies the Spring container. The container creates the object, wires them together, configures them and manages their complete life cycle. The Spring container makes use of Dependency Injection to manage the components that

make up an application. The container receives instructions for which objects to instantiate, configure, and assemble by reading the configuration metadata provided. This metadata can be provided either by XML, Java annotations or Java code.

10. What do you mean by Dependency Injection?

In Dependency Injection, you do not have to create your objects but have to describe how they should be created. You don't connect your components and services together in the code directly, but describe which services are needed by which components in the configuration file. The IoC container will wire them up together.

11. In how many ways can Dependency Injection be done?

In general, dependency injection can be done in three ways, namely :

- Constructor Injection
- Setter Injection
- Interface Injection

In Spring Framework, only constructor and setter injections are used.

12. Differentiate between constructor injection and setter injection.

Constructor Injection vs Setter Injection

Constructor Injection	Setter Injection
There is no partial injection.	There can be partial injection.
It doesn't override the setter property.	It overrides the constructor property.
It will create a new instance if any modification is done.	It will not create new instance if any modification is done.
It works better for many properties.	It works better for few properties.

13. How many types of IOC containers are there in spring?

1. **BeanFactory:** BeanFactory is like a factory class that contains a collection of beans. It instantiates the bean whenever asked for by clients.
2. **ApplicationContext:** The ApplicationContext interface is built on top of the BeanFactory interface. It provides some extra functionality on top BeanFactory.

14. Differentiate between BeanFactory and ApplicationContext.

BeanFactory vs ApplicationContext

BeanFactory	ApplicationContext
It is an interface defined in org.springframework.beans.factory. BeanFactory	It is an interface defined in org.springframework.context. ApplicationContext
It uses Lazy initialization	It uses Eager/ Aggressive initialization
It explicitly provides a resource object using the syntax	It creates and manages resource objects on its own
It doesn't supports internationalization	It supports internationalization
It doesn't supports annotation based dependency	It supports annotation based dependency

15. List some of the benefits of IoC.

Some of the benefits of IoC are:

- It will minimize the amount of code in your application.
- It will make your application easy to test because it doesn't require any singletons or JNDI lookup mechanisms in your unit test cases.
- It promotes loose coupling with minimal effort and least intrusive mechanism.
- It supports eager instantiation and lazy loading of the services.

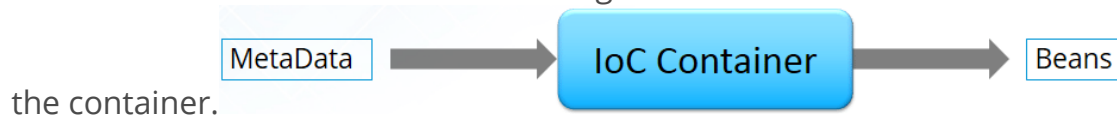
Let's move on to the next section of Spring Interview Questions, that is Spring Beans

Interview Questions.

Spring Beans – Spring Interview Questions

16. Explain Spring Beans?

- They are the objects that form the backbone of the user's application.
- Beans are managed by the Spring IoC container.
- They are instantiated, configured, wired and managed by a Spring IoC container
- Beans are created with the configuration metadata that the users supply to



17. How configuration metadata is provided to the Spring container?

Configuration metadata can be provided to Spring container in following ways:

- **XML-Based configuration:** In Spring Framework, the dependencies and the services needed by beans are specified in configuration files which are in XML format. These configuration files usually contain a lot of bean definitions and application specific configuration options. They generally start with a bean tag. For example:

```
1 <bean id="studentbean" class="org.edureka.firstSpring.StudentBean">  
2   <property name="name" value="Edureka"></property>  
3 </bean>
```

- **Annotation-Based configuration:** Instead of using XML to describe a bean wiring, you can configure the bean into the component class itself by using annotations on the relevant class, method, or field declaration. By default, annotation wiring is not turned on in the Spring container. So, you need to enable it in your Spring configuration file before using it. For example:

```
1 <beans>  
2 <context:annotation-config/>  
3 <!-- bean definitions go here -->  
4 </beans>
```

- **Java-based configuration:** The key features in Spring Framework's new Java-configuration support are @Configuration annotated classes and @Bean annotated methods.

1. @Bean annotation plays the same role as the <bean/> element.

2. @Configuration classes allows to define inter-bean dependencies by simply calling other @Bean methods in the same class.

For example:

```
1 @Configuration  
2 public class StudentConfig  
3 {  
4   @Bean  
5   public StudentBean myStudent()  
6   { return new StudentBean(); }  
7 }
```

18. How many bean scopes are supported by Spring?

The Spring Framework supports five scopes. They are:

- **Singleton:** This provides scope for the bean definition to single instance per Spring IoC container.
- **Prototype:** This provides scope for a single bean definition to have any number of object instances.
- **Request:** This provides scope for a bean definition to an HTTP-request.
- **Session:** This provides scope for a bean definition to an HTTP-session.
- **Global-session:** This provides scope for a bean definition to an Global HTTP-session.

The last three are available only if the users use a web-aware ApplicationContext.

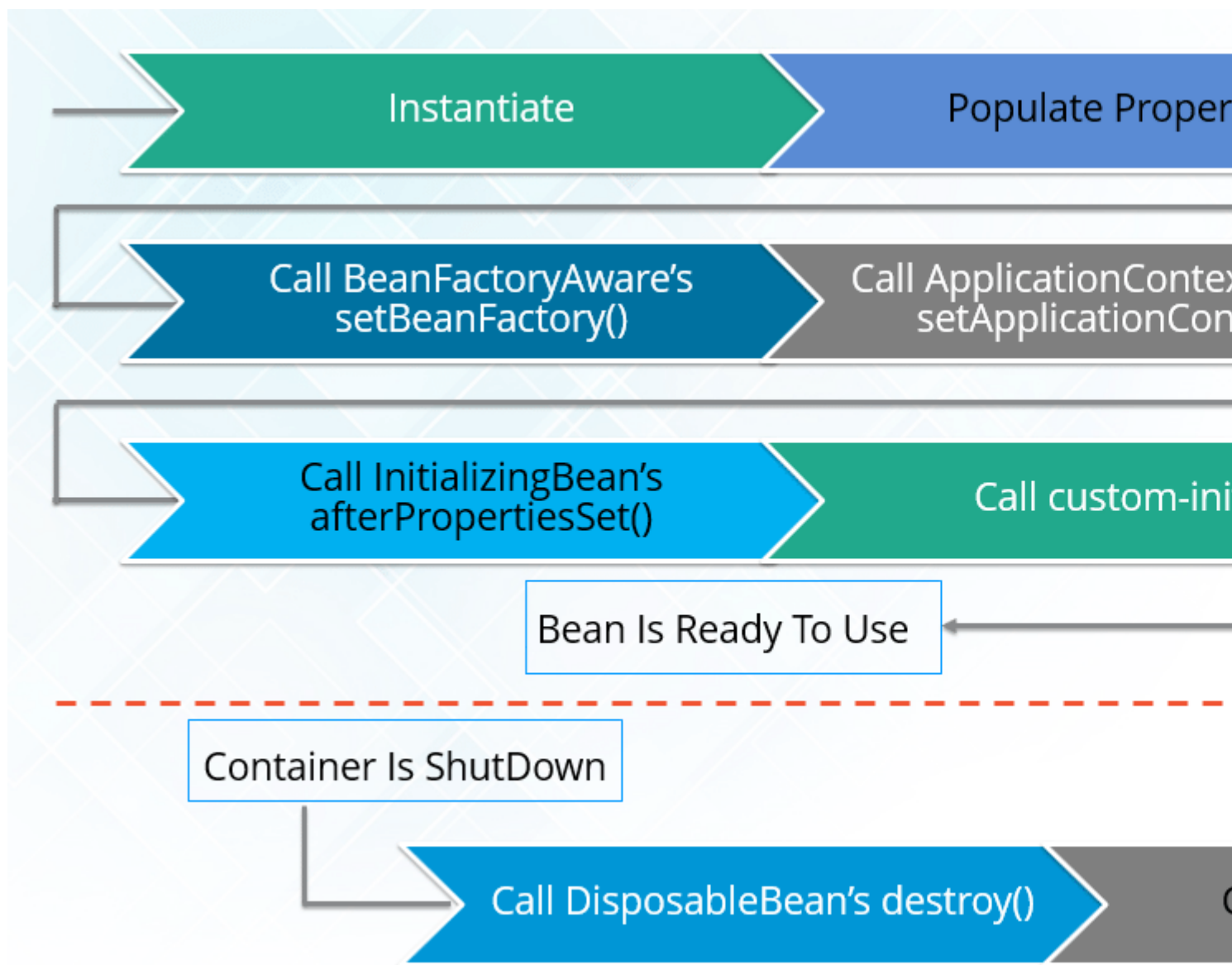
-
-
-
-

19. What is the Bean life cycle in Spring Bean Factory Container?

Bean life cycle in Spring Bean Factory Container is as follows:

1. The Spring container instantiates the bean from the bean's definition in the XML file.
2. Spring populates all of the properties using the dependency injection, as specified in the bean definition.
3. The factory calls `setBeanName()` by passing the bean's ID, if the bean implements the `BeanNameAware` interface.
4. The factory calls `setBeanFactory()` by passing an instance of itself, if the bean implements the `BeanFactoryAware` interface.
5. `preProcessBeforeInitialization()` methods are called if there are any `BeanPostProcessors` associated with the bean.
6. If an `init-method` is specified for the bean, then it will be called.
7. Finally, `postProcessAfterInitialization()` methods will be called if there are any `BeanPostProcessors` associated with the bean.

To understand it in better way check the below diagram:



20. Explain inner beans in Spring.

A bean can be declared as an inner bean only when it is used as a property of another bean. For defining a bean, the Spring's XML based configuration metadata provides the use of `<bean>` element inside the `<property>` or `<constructor-arg>`. Inner beans are always anonymous and they are always scoped as prototypes. For example, let's say we have one Student class having reference of Person class. Here we will be creating only one instance of Person class and use it inside Student.

Here's a Student class followed by bean configuration file:

Student.java

```
1 public class Student
2 {
3     private Person person;
4     //Setters and Getters
5 }
6 public class Person
7 {
8     private String name;
9     private String address;
10    //Setters and Getters
11 }
```

studentbean.xml

```
1 <bean id="StudentBean" class="com.edureka.Student">
2 <property name="person">
3 <!--This is inner bean -->
4 <bean class="com.edureka.Person">
5 <property name="name" value="Scott"></property>
6 <property name="address" value="Bangalore"></property>
7 </bean>
8 </property>
9 </bean>
```

21. Define Bean Wiring.

When beans are combined together within the Spring container, it's called wiring or bean wiring. The Spring container needs to know what beans are needed and how the container should use dependency injection to tie the beans together, while wiring beans.

22. What do you understand by auto wiring and name the different modes of it?

The Spring container is able to autowire relationships between the collaborating beans. That is, it is possible to let Spring resolve collaborators for your bean automatically by

inspecting the contents of the BeanFactory.

Different modes of bean auto-wiring are:

1. **no:** This is default setting which means no autowiring. Explicit bean reference should be used for wiring.
2. **byName:** It injects the object dependency according to name of the bean. It matches and wires its properties with the beans defined by the same names in the XML file.
3. **byType:** It injects the object dependency according to type. It matches and wires a property if its type matches with exactly one of the beans name in XML file.
4. **constructor:** It injects the dependency by calling the constructor of the class. It has a large number of parameters.
5. **autodetect:** First the container tries to wire using autowire by *constructor*, if it can't then it tries to autowire by *byType*.

23. What are the limitations with auto wiring?

Following are some of the limitations you might face with auto wiring:

- **Overriding possibility:** You can always specify dependencies using `<constructor-arg>` and `<property>` settings which will override autowiring.
- **Primitive data type:** Simple properties such as primitives, Strings and Classes can't be autowired.
- **Confusing nature:** Always prefer using explicit wiring because autowiring is less precise.

In the next section, we will discuss on Spring Annotations Interview Questions.

Spring Annotations – Spring Interview Questions

24. What do you mean by Annotation-based container configuration?

Instead of using XML to describe a bean wiring, the developer moves the configuration into the component class itself by using annotations on the relevant class, method, or field declaration. It acts as an alternative to XML setups. For example:

```
1|@Configuration
2|public class AnnotationConfig
```

```

3 {
4 @Bean
5 public MyDemo myDemo()
6 { return new MyDemoImpl(); }
7 }

```

25. How annotation wiring can be turned on in Spring?

By default, Annotation wiring is not turned on in the Spring container. Thus, to use annotation based wiring we must enable it in our Spring configuration file by configuring **<context:annotation-config/>** element. For example:

```

1 <beans xmlns="http://www.springframework.org/schema/beans"
2       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3       xmlns:context="http://www.springframework.org/schema/context"
4       xsi:schemaLocation="http://www.springframework.org/schema/beans
5                           http://www.springframework.org/schema/beans
6                           http://www.w3.org/2001/XMLSchema-instance
7                           http://www.w3.org/2001/XMLSchema-instance
8                           http://www.springframework.org/schema/context
9                           http://www.springframework.org/schema/context">
10
11     <context:annotation-config/>
12
13     <beans ..... />
14
15 </beans>

```

26. What's the difference between @Component, @Controller, @Repository & @Service annotations in Spring?

@Component: This marks a java class as a bean. It is a generic stereotype for any Spring-managed component. The component-scanning mechanism of spring now can pick it up and pull it into the application context.

@Controller: This marks a class as a Spring Web MVC controller. Beans marked with it are automatically imported into the Dependency Injection container.

@Service: This annotation is a specialization of the component annotation. It doesn't provide any additional behavior over the @Component annotation. You can use @Service over @Component in service-layer classes as it specifies intent in a better way.

@Repository: This annotation is a specialization of the @Component annotation with similar use and functionality. It provides additional benefits specifically for DAOs. It imports the DAOs into the DI container and makes the unchecked exceptions eligible for translation into Spring DataAccessException.

27. What do you understand by @Required annotation?

@Required is applied to bean property setter methods. This annotation simply indicates that the affected bean property must be populated at the configuration time with the help of an explicit property value in a bean definition or with autowiring. If the affected bean property has not been populated, the container will throw BeanInitializationException.

For example:

```
1 public class Employee
2 {
3     private String name;
4     @Required
5     public void setName(String name)
6     { this.name=name; }
7     public String getName()
8     { return name; }
9 }
```

28. What do you understand by @Autowired annotation?

The **@Autowired** annotation provides more accurate control over where and how autowiring should be done. This annotation is used to autowire bean on the setter methods, constructor, a property or methods with arbitrary names or multiple arguments. By default, it is a type driven injection.

For Example:

```
1 public class Employee
```

```

2{
3private String name;
4@Autowired
5public void setName(String name)
6{this.name=name; }
7public string getName()
8{ return name; }
9}

```

29. What do you understand by @Qualifier annotation?

When you create more than one bean of the same type and want to wire only one of them with a property you can use the **@Qualifier** annotation along with **@Autowired** to remove the ambiguity by specifying which exact bean should be wired.

For example, here we have two classes, Employee and EmpAccount respectively. In EmpAccount, using @Qualifier its specified that bean with id emp1 must be wired.

Employee.java

```

1public class Employee
2{
3private String name;
4@Autowired
5public void setName(String name)
6{ this.name=name; }
7public string getName()
8{ return name; }
9}

```

EmpAccount.java

```

1public class EmpAccount
2{
3private Employee emp;
4@Autowired
5@Qualifier(emp1)
6public void showName()
7{
8System.out.println("Employee name : "+emp.getName());
9}
10}

```

30. What do you understand by @RequestMapping annotation?

@RequestMapping annotation is used for mapping a particular HTTP request method to a specific class/ method in controller that will be handling the respective request. This annotation can be applied at both levels:

- **Class level**: Maps the URL of the request
- **Method level**: Maps the URL as well as HTTP request method

Next section of Spring Interview Questions is on Data Access.

Data Access – Spring Interview Questions

31. Describe Spring DAO support?

The Data Access Object (DAO) support in Spring makes it easy to work with data access technologies like JDBC, Hibernate or JDO in a consistent way. This allows one to switch between the persistence technologies easily. It also allows you to code without worrying about catching exceptions that are specific to each of these technology.

Question 3: What is Bean Factory, have you used XMLBeanFactory?

Ans: `BeanFactory` is a factory Pattern that is based on IOC [design principles](#). It is used to make a clear separation between application configuration and dependency from actual [code](#). The `XmlBeanFactory` is one of the implementations of Bean Factory which we have used in our project.

The `org.springframework.beans.factory.xml.XmlBeanFactory` is used to create bean instances defined in our XML file.

```
BeanFactory factory = new XmlBeanFactory(new FileInputStream("beans.xml"));
```

Or

```
ClassPathResource resorce = new ClassPathResource("beans.xml");  
XmlBeanFactory factory = new XmlBeanFactory(resorce);
```

Question 4: What are the difference between BeanFactory and ApplicationContext in Spring? ([answer](#))

Answer: This one is a very popular Spring interview question and often asks in an entry-level interview. `ApplicationContext` is the preferred way of using spring because of the [functionality](#) provided by it and the interviewer wanted to check whether you are familiar with it or not.

ApplicationContext.	BeanFactory
Here we can have more than one config files possible	In this only one config file or .xml file
Application contexts can publish events to beans that are registered as listeners	Don't support.
Support internationalization (I18N) messages	It's not
Support application life-cycle events, and validation.	Doesn't support.
Supports many enterprise services such as JNDI access, EJB integration, remoting	Doesn't support.

Question 5: What is AOP?

Answer: The core construct of AOP is the aspect, which encapsulates behaviors affecting multiple classes into reusable modules. AOP is a programming technique that allows a [developer](#) to modularize crosscutting concerns, that cut across the typical divisions of responsibility, such as **logging and transaction management**.

Spring AOP, aspects are implemented using regular classes or regular classes annotated with the `@Aspect` annotation. You can also check out these [30+ Spring MVC interview questions](#) for more focus on Java [web development](#) using the Spring MVC framework.

Question 6: Explain Advice?

Answer: It's an [implementation](#) of aspect; advice is inserted into an application at join points. Different types of advice include "around," "before" and "after" advice

Question 7: What are the joint Point and point cut?

Ans: This is not really a spring [interview](#) question I would say an AOP one. Similar to [Object-oriented programming](#), AOP is another popular programming concept that complements OOPS. A join point is an opportunity within the code for which we can apply an aspect. In [Spring](#) AOP, a join point always represents a method execution.

Pointcut: a predicate that matches join points. A pointcut is something that defines what join-points advice should be applied.

If you need more [Spring](#)

AOP questions then you can also check out my article about [17 Spring AOP Interview Questions with Answers](#) for Java developers.

Question 8: Difference between the setter and constructor injection in Spring? ([answer](#))

Setter injection is more flexible than constructor injection because you must remember the type and order of the constructor parameters. Also, constructor injection is generally used to inject the mandatory dependency, while setter can be used to inject the optional dependency.

Question 9: Difference between Factory Pattern and Dependency Injection in Java? ([answer](#))

Even though both allow you to reduce coupling in code, dependency injection is much more flexible and easier to test than Factory pattern.

Questions 10. Difference between @Autowired and @ Inject annotation in Spring? ([answer](#))

Question 11: What are the different modules in spring?

Answer: spring has seven core modules

1. The Core container module
2. Application context module
3. AOP [module](#) (Aspect Oriented Programming)
4. JDBC abstraction and DAO module
5. O/R [mapping](#) integration module (Object/Relational)
6. Web module
7. MVC framework module

2. Spring MVC Interview Questions Answers

So far we have seen the Spring core interview questions and answers and now let's see some interview questions from Spring MVC, one of the most important parts of Spring framework which allows you to develop web applications in Java using Model View Controller design pattern.

Questions 12: What is the difference between @Controller and @RestController in Spring MVC? ([answer](#))

Even though both are used to indicate that a Spring bean is a Controller in Spring MVC setup, @RestController is better when you are developing [RESTful web services](#) using the Spring MVC framework. It's a combination

of @Controller + @ResponseBody annotation which allows the controller to directly write the response and bypassing the view resolution process, which is not required for RESTful web service.

It also instructs DispatcherServlet to use different `HttpMessageConverters` to represent the response in the format client is expecting e.g. `HttpMessageJackson2Convert` to represent response in JSON format and JAXB based message converts to generate XML response.

You can further see the [REST with Spring](#) course by Baeldung to learn more about developing RESTful Web Services using Spring 4 and Spring 5.

```
22 import springfox.documentation.builders.PathSelectors;
23 import springfox.documentation.builders.RequestHandlerSelectors;
24 import springfox.documentation.spi.DocumentationType;
25 import springfox.documentation.spring.web.plugins.Docket;
26 import springfox.documentation.swagger2.annotations.EnableSwagger2;
27
28 @Configuration
29 @ComponentScan({ "org.baeldung.common.web", "org.baeldung.un.web" })
30 @EnableWebMvc
31 @EnableSwagger2
32 public class UnWebConfig extends WebMvcConfigurerAdapter {
33
34     public UnWebConfig() {
35         super();
36     }
37
38     // beans
39
40     @Bean
41     public Docket mainConfig() { // @formatter:off
42         return new Docket(DocumentationType.SWAGGER_2)
43             .select().apis(RequestHandlerSelectors.any())
44             .paths(PathSelectors.any())
45             .build()
46             .pathMapping("/api")
47             .directModelSubstitute(LocalDate.class, String.class)
48             .genericModelSubstitutes(ResponseEntity.class)
49             ;
50     } // @formatter:on
51 }
```

Question 13: What is the difference between a singleton and a prototype bean?

Ans: This is another popular *spring interview question* and an important [concept](#) to understand. Basically, a bean has scopes which define their existence on the application.

Singleton: means single bean definition to a single object instance per Spring IOC container.

Prototype: means a single bean definition to any number of object instances.

Whatever beans we defined in the spring framework are singleton beans.

There is an attribute in the bean tag named 'singleton' if specified true then the bean becomes singleton and if set to false then the bean becomes a prototype bean. By default, it is set to `true`. So, all the beans in the spring framework are by default singleton beans.

```
<bean id="createNewStock" class="springexample.stockMarket.CreateNewStockAccount"
    singleton="false">
    <property name="newBid"/>
</bean>
```

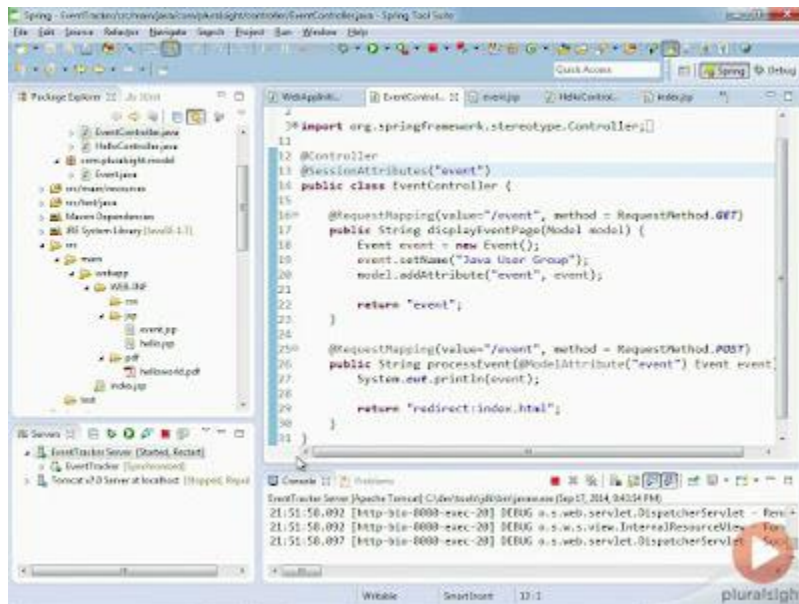
Question 14: What is the role of DispatcherServlet in Spring MVC? ([answer](#))

The `DispatcherServlet` is very important from the Spring MVC perspective, it acts as a FrontController i.e. all requests pass through it. It is responsible for routing the request to the controller and view the resolution before sending the response to the client.

When Controller returns a [Model](#) or View object, it consults all the view resolvers registered to find the correct type of `ViewResolver` which can render the response for clients.

In the case of RESTful [Web Services](#), the `DispatcherServlet` is also responsible for using `HttpMessageConverter` to represent the response in the JSON, XML, or TEXT format, depending on the content negotiation between Client and [Server](#) like if the client sends a request with HTTP accept header as "application/json" then `DispatcherServlet` will ask the `HttpMessageJackson2Converter` to convert the response into [JSON](#) format.

You can further see the free [Introduction to Spring MVC](#) course from Pluralsight to learn more about Spring MVC and `DispatcherServlet`.



Question 15: How to call the stored procedure from Java using Spring Framework? ([answer](#))

Question 16: How to Setup JDBC Database connection pool in Spring Web application? ([answer](#))

Questions 17: Difference between @RequestParam and @PathVariable in Spring MVC? ([answer](#))

Questions 18: Difference between @Component, @Service, @Controller, and @Repository annotation in Spring MVC? ([answer](#))

Question 19: What type of transaction Management Spring support?

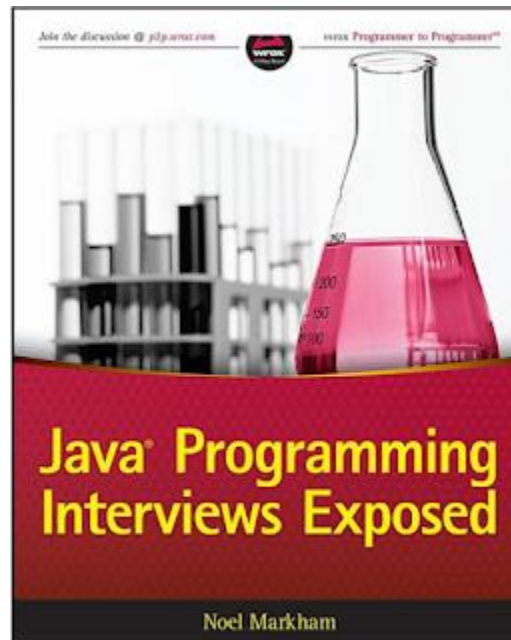
Ans: This spring interview question is a little difficult as compared to [previous](#) questions just because **transaction management** is a complex concept and not every developer familiar with it. Transaction management is critical in any application that will interact with the database.

The application has to ensure that the data is consistent and the integrity of the data is maintained. Following two types of transaction [management](#) is supported by spring:

1. Programmatic transaction management
2. Declarative transaction management.

If you need more questions, you can also check out the [Java Programming Interview exposed](#) book from Wrox publication, apart from various Java topics it also contains some really good Spring framework, Hibernate, and Spring MVC questions with detailed

explanation.



Also, Hibernate is mostly used in Spring, so don't forget to prepare some [Hibernate interview questions](#) along with Spring.

3. Spring Security Interview Questions Answer

Some of the [readers](#) requested to provide Spring Security interview questions and answer, So I thought to update this article with a few of the Spring security questions I came across.

Here are those:

20. How do you control the concurrent Active session using Spring Security? ([answer](#))

Another Spring interview question is based on Out of box feature provided by the Spring framework. You can easily control How many active sessions a user can have with a Java application by using Spring Security.

Apart from that Spring Security also provides the "remember me" feature which you can use to provide easier access for your users by remembering their login details on their [personal computer](#). You can further see [Learn Spring Security](#) course by Eugen Paraschiv to learn more about advanced details of Spring Security.

21. How do you set up LDAP Authentication using Spring Security? ([answer](#))

This is a very popular Spring Security interview question as Spring provides out-of-the-box support to connect Windows Active Directory for LDAP authentication and with few configurations in the Spring config file you can have this feature enabled.



22. How to implement Role-Based Access Control (RBAC) using Spring Security? ([answer](#))

Spring Security provides a couple of ways to implement Role-based access control like by using `GrantedAuthority`. See the article to learn more about it.

That's all about frequently asked Spring Interview Questions with Answers. These **Spring interview Questions and answers** are not very difficult and focused on spring fundamentals rather than focusing on an advanced feature of session management, spring security, authentication, etc. we will cover those questions in some other interview article. I would also suggest that share some spring questions asked to you guys during the interview and then I can put together those with their answers for quick reference of everybody

SPRING MVC

Q1. Why Should We Use Spring MVC?

Spring MVC implements a clear separation of concerns that allows us to develop and unit test our applications easily.

The concepts like:

Dispatcher Servlet

- Controllers
- View Resolvers
- Views, Models
- ModelAndView
- Model and Session Attributes

are completely independent of each other, and they are responsible for one thing only.

Therefore, **MVC gives us quite big flexibility**. It's based on interfaces (with provided implementation classes), and we can configure every part of the framework by using custom interfaces.

Another important thing is that we aren't tied to a specific view technology (for example, JSP), but we have the option to choose from the ones we like the most.

Also, we don't use Spring MVC only in web applications development but in the creation of RESTful web services as well.

Q2. What Is the Role of the @Autowired Annotation?

The @Autowired annotation can be used with fields or methods for injecting a **bean by type**. This annotation allows Spring to resolve and inject collaborating beans into your bean.

For more details, please refer to the tutorial about [@Autowired in Spring](#).

Q3. Explain a Model Attribute

The @ModelAttribute annotation is one of the most important annotations in Spring MVC. It **binds a method parameter or a method return value to a named model attribute and then exposes it to a web view**.

If we use it at the method level, it indicates the purpose of that method is to add one or more model attributes.

On the other hand, when used as a method argument, it indicates the argument should be retrieved from the model. When not present, we should first instantiate it and then add it to the model. Once present in the model, we should populate the arguments fields from all request parameters that have matching names.

More about this annotation can be found in our [article related to the @ModelAttribute annotation](#).

Q4. Explain the Difference Between @Controller and @RestController?

The main difference between the @Controller and @RestController annotations is that the @ResponseBody annotation is automatically included in the @RestController. This means that we don't need to annotate our handler methods with the @ResponseBody. We need to do this in a @Controller class if we want to write response type directly to the HTTP response body.

Q5. Describe a PathVariable

We can use the @PathVariable annotation as a handler method parameter in order to extract the value of a URI template variable.

For example, if we want to fetch a user by id from the `www.mysite.com/user/123`, we should map our method in the controller as `/user/{id}`:

```
@RequestMapping("/user/{id}")
public String handleRequest(@PathVariable("id") String userId, Model map) {}
```

The @PathVariable has only one element named value. It's optional and we use it to define the URI template variable name. If we omit the value element, then the URI template variable name must match the method parameter name.

It's also allowed to have multiple @PathVariable annotations, either by declaring them one after another:

```
@RequestMapping("/user/{userId}/name/{userName}")
public String handleRequest(@PathVariable String userId,
    @PathVariable String userName, Model map) {}
```

or putting them all in a `Map<String, String>` or `MultiValueMap<String, String>`:

```
@RequestMapping("/user/{userId}/name/{userName}")
```



```
🔗 public String handleRequest(@PathVariable Map<String, String> varsMap,  
Model map) {}
```

Q6. Validation Using Spring MVC

Spring MVC supports JSR-303 specifications by default. We need to add JSR-303 and its implementation dependencies to our Spring MVC application.

Hibernate Validator, for example, is one of the JSR-303 implementations at our disposal.

JSR-303 is a specification of the Java API for bean validation, part of Jakarta EE and JavaSE, which ensures that properties of a bean meet specific criteria, using annotations such as `@NotNull`, `@Min`, and `@Max`. More about validation is available in the [Java Bean Validation Basics](#) article.

Spring offers the `@Validator` annotation and the `BindingResult` class.

The Validator implementation will raise errors in the controller request handler method when we have invalid data. Then we may use the `BindingResult` class to get those errors.

Besides using the existing implementations, we can make our own. To do so, we create an annotation that conforms to the JSR-303 specifications first. Then, we implement the Validator class. Another way would be to implement Spring's `Validator` interface and set it as the validator via `@InitBinder` annotation in Controller class.

To check out how to implement and use your own validations, please see the tutorial regarding [Custom Validation in Spring MVC](#).

Q7. What Are the `@RequestBody` and the `@ResponseBody` Annotations?

The `@RequestBody` annotation, used as a handler method parameter, binds the HTTP Request body to a transfer or a domain object. Spring automatically deserializes incoming HTTP Request to the Java object using Http Message Converters.

When we use the `@ResponseBody` annotation on a handler method in the Spring MVC controller, it indicates that we'll write the return type of the method directly to the HTTP response body. We'll not put it in a Model, and Spring won't interpret as a view name.

Please check out the article on [@RequestBody and @ResponseBody](#) to see more details about these annotations.

Q8. Explain Model, ModelMap and ModelAndView?

The Model interface defines a holder for model attributes. The ModelMap has a similar purpose, with the ability to pass a collection of values. It then treats those values as if they were within a Map. We should note that in Model (ModelMap) we can only store data. We put data in and return a view name.

On the other hand, with the ModelAndView, we return the object itself. We set all the required information, like the data and the view name, in the object we're returning.

You can find more details in the article on [Model, ModelMap, and ModelAndView](#).

Q9. Explain SessionAttributes and SessionAttribute

The `@SessionAttributes` annotation is used for storing the model attribute in the user's session. We use it at the controller class level, as shown in our article about the [Session Attributes in Spring MVC](#):

`@Controller`

```

@RequestMapping("/sessionattributes")
@SessionAttributes("todos")
public class TodoControllerWithSessionAttributes {

    @GetMapping("/form")
    public String showForm(Model model,
        @ModelAttribute("todos") TodoList todos) {
        // method body
        return "sessionattributesform";
    }

    // other methods
}

```

In the previous example, the model attribute 'todos' will be added to the session if the @ModelAttribute and the @SessionAttributes have the same name attribute.

If we want to retrieve the existing attribute from a session that is managed globally, we'll use @SessionAttribute annotation as a method parameter:

```

@GetMapping
public String getTodos(@SessionAttribute("todos") TodoList todos) {
    // method body
    return "todoView";
}

```

Q10. What Is the Purpose of @EnableWebMvc?

The @EnableWebMvc annotation's purpose is to enable Spring MVC via Java configuration. It's equivalent to <mvc:annotation-driven> in an XML configuration. This annotation imports Spring MVC Configuration from WebMvcConfigurationSupport. It enables support for @Controller-annotated classes that use @RequestMapping to map incoming requests to a handler method.

🔗 You can learn more about this and similar annotations in our [Guide to the Spring @Enable Annotations](#).

Q11. What Is ViewResolver in Spring?

The ViewResolver enables an application to render models in the browser – without tying the implementation to a specific view technology – by mapping view names to actual views.

For more details about the ViewResolver, have a look at our [Guide to the ViewResolver in Spring MVC](#).

Q12. What Is the BindingResult?

BindingResult is an interface from org.springframework.validation package that represents binding results. We can use it to detect and report errors in the submitted form. It's easy to invoke — we just need to ensure that we put it as a parameter right after the form object we're validating. The optional Model parameter should come after the BindingResult, as it can be seen in the [custom validator tutorial](#):

```
@PostMapping("/user")
public String submitForm(@Valid NewUserForm newUserForm,
    BindingResult result, Model model) {
    if (result.hasErrors()) {
        return "userHome";
    }
    model.addAttribute("message", "Valid form");
    return "userHome";
}
```

🔗 When Spring sees the `@Valid` annotation, it'll first try to find the validator for the object being validated. Then it'll pick up the validation annotations and invoke the validator. Finally, it'll put found errors in the `BindingResult` and add the latter to the view model.

Q13. What Is a Form Backing Object?

The form backing object or a Command Object is just a POJO that collects data from the form we're submitting.

We should keep in mind that it doesn't contain any logic, only data.

To learn how to use form backing object with the forms in Spring MVC, please take a look at our article about [Forms in Spring MVC](#).

Q14. What Is the Role of the `@Qualifier` Annotation?

It is used simultaneously with the `@Autowired` annotation to avoid confusion when multiple instances of a bean type are present.

Let's see an example. We declared two similar beans in XML config:

```
<bean id="person1" class="com.baeldung.Person" >
  <property name="name" value="Joe" />
</bean>
<bean id="person2" class="com.baeldung.Person" >
  <property name="name" value="Doe" />
</bean>
```

When we try to wire the bean, we'll get an `org.springframework.beans.factory.NoSuchBeanDefinitionException`. To fix it, we need to use `@Qualifier` to tell Spring about which bean should be wired:

`@Autowired`

```
@Qualifier("person1")
private Person person;
```

Q15. What Is the Role of the @Required Annotation?

The @Required annotation is used on setter methods, and it indicates that the bean property that has this annotation must be populated at configuration time. Otherwise, the Spring container will throw a `BeanInitializationException` exception.

Also, @Required differs from @Autowired – as it is limited to a setter, whereas @Autowired is not. @Autowired can be used to wire with a constructor and a field as well, while @Required only checks if the property is set.

Let's see an example:

```
public class Person {
    private String name;

    @Required
    public void setName(String name) {
        this.name = name;
    }
}
```

Now, the name of the Person bean needs to be set in XML config like this:

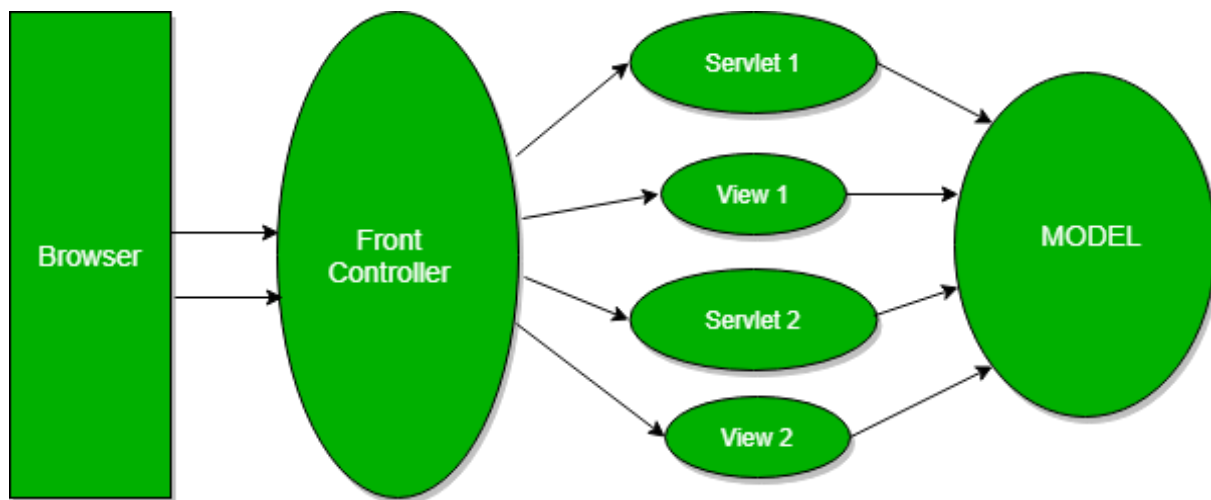
```
<bean id="person" class="com.baeldung.Person">
    <property name="name" value="Joe" />
</bean>
```

Please note that @Required doesn't work with Java based @Configuration classes by default. If you need to make sure that all your properties are set, you can do so when you create the bean in the @Bean annotated methods.

Q16. Describe the Front Controller Pattern

In the Front Controller pattern, all requests will first go to the front controller instead of the servlet. It'll make sure that the responses are ready and will send them back to the browser. This way we have one place where we control everything that comes from the outside world.

The front controller will identify the servlet that should handle the request first. Then, when it gets the data back from the servlet, it'll decide which view to render and, finally, it'll send the rendered view back as a response:

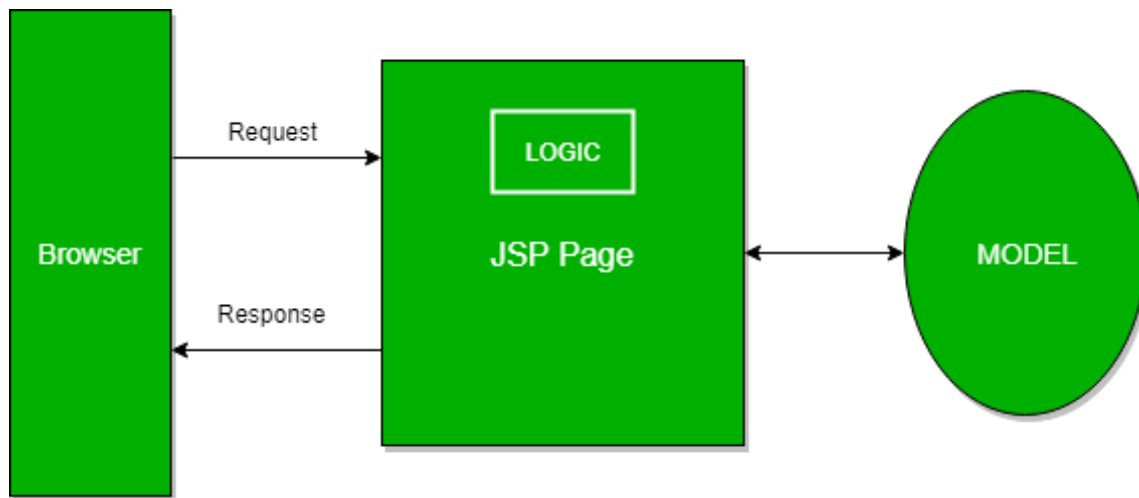


To see the implementation details, please check out our [Guide to the Front Controller Pattern in Java](#).

Q17. What Are Model 1 and Model 2 Architectures?

Model 1 and Model 2 represent two frequently used design models when it comes to designing Java Web Applications.

In Model 1, a request comes to a servlet or JSP where it gets handled. The servlet or the JSP processes the request, handles business logic, retrieves and validates data, and generates the response:



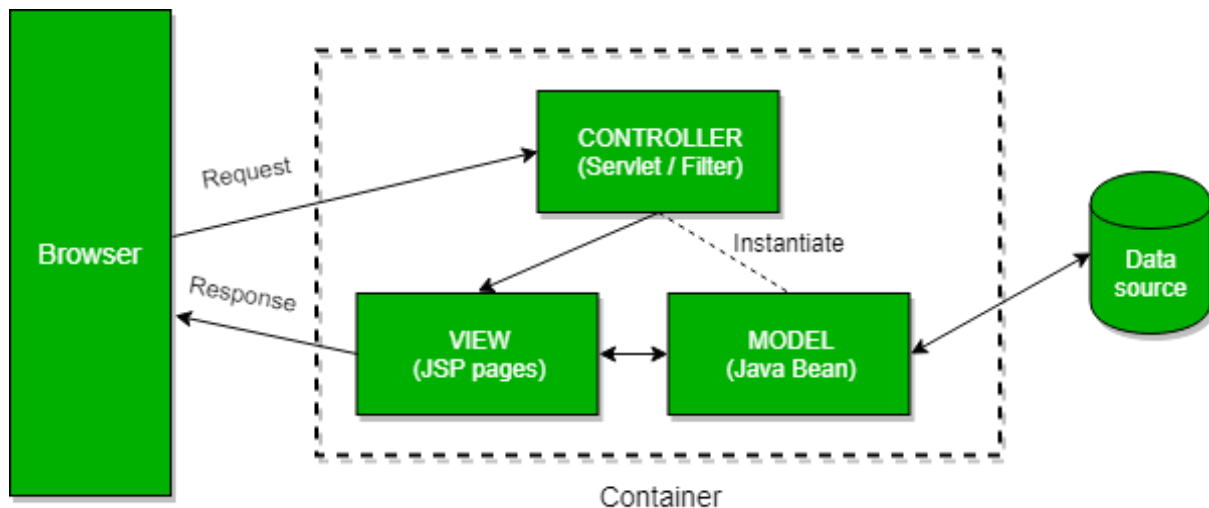
Since this architecture is easy to implement, we usually use it in small and simple applications.

On the other hand, it isn't convenient for large-scale web applications. The functionalities are often duplicated in JSPs where business and presentation logic are coupled.

The Model 2 is based on the Model View Controller design pattern and it separates the view from the logic that manipulates the content.

Furthermore, we can distinguish three modules in the MVC pattern: the model, the view, and the controller. The model is representing the dynamic data structure of an application. It's responsible for the data and business logic manipulation. The view is in charge of displaying the data, while the controller serves as an interface between the previous two.

In Model 2, a request is passed to the controller, which handles the required logic in order to get the right content that should be displayed. The controller then puts the content back into the request, typically as a JavaBean or a POJO. It also decides which view should render the content and finally passes the request to it. Then, the view renders the data:



3. Advanced Spring MVC Questions

Q18. What's the Difference Between @Controller, @Component, @Repository, and @Service Annotations in Spring?

According to the official Spring documentation, @Component is a generic stereotype for any Spring-managed component. @Repository, @Service, and @Controller are specializations of @Component for more specific use cases, for example, in the persistence, service, and presentation layers, respectively.

Let's take a look at specific use cases of last three:

- **@Controller** – indicates that the class serves the role of a controller, and detects @RequestMapping annotations within the class
- **@Service** – indicates that the class holds business logic and calls methods in the repository layer
- **@Repository** – indicates that the class defines a data repository; its job is to catch platform-specific exceptions and re-throw them as one of Spring's unified unchecked exceptions

Q19. What Are DispatcherServlet and ContextLoaderListener?

Simply put, in the Front Controller design pattern, a single controller is responsible for directing incoming `HttpRequests` to all of an application's other controllers and handlers.

Spring's `DispatcherServlet` implements this pattern and is, therefore, responsible for correctly coordinating the `HttpRequests` to the right handlers.

On the other hand, `ContextLoaderListener` starts up and shuts down Spring's root `WebApplicationContext`. It ties the lifecycle of `ApplicationContext` to the lifecycle of the `ServletContext`. We can use it to define shared beans working across different Spring contexts.

For more details on `DispatcherServlet`, please refer [to this tutorial](#).

Q20. What Is a MultipartResolver and When Should We Use It?

The `MultipartResolver` interface is used for uploading files. The Spring framework provides one `MultipartResolver` implementation for use with Commons FileUpload and another for use with Servlet 3.0 multipart request parsing.

Using these, we can support file uploads in our web applications.

Q21. What Is Spring MVC Interceptor and How to Use It?

Spring MVC Interceptors allow us to intercept a client request and process it at three places – before handling, after handling, or after completion (when the view is rendered) of a request.

The interceptor can be used for cross-cutting concerns and to avoid repetitive handler code like logging, changing globally used parameters in Spring model, etc.

For details and various implementations, take a look at [Introduction to Spring MVC HandlerInterceptor](#) article.

Q22. What Is an Init Binder?

A method annotated with `@InitBinder` is used to customize a request parameter, URI template, and backing/command objects. We define it in a controller and it helps in controlling the request. In this method, we register and configure our custom `PropertyEditors`, a formatter, and validators.

The annotation has the 'value' element. If we don't set it, the `@InitBinder` annotated methods will get called on each HTTP request. If we set the value, the methods will be applied only for particular command/form attributes and/or request parameters whose names correspond to the 'value' element.

It's important to remember that one of the arguments must be `WebDataBinder`. Other arguments can be of any type that handler methods support except for command/form objects and corresponding validation result objects.

Q23. Explain a Controller Advice

The `@ControllerAdvice` annotation allows us to write global code applicable to a wide range of controllers. We can tie the range of controllers to a chosen package or a specific annotation.

By default, `@ControllerAdvice` applies to the classes annotated with `@Controller` (or `@RestController`). We also have a few properties that we use if we want to be more specific.

If we want to restrict applicable classes to a package, we should add the name of the package to the annotation:

```
@ControllerAdvice("my.package")
@ControllerAdvice(value = "my.package")
@ControllerAdvice(basePackages = "my.package")
```

It's also possible to use multiple packages, but this time we need to use an array instead of the String.

Besides restricting to the package by its name, we can do it by using one of the classes or interfaces from that package:

```
@ControllerAdvice(basePackageClasses = MyClass.class)
```

The 'assignableTypes' element applies the @ControllerAdvice to the specific classes, while 'annotations' does it for particular annotations.

It's noteworthy to remember that we should use it along with @ExceptionHandler. This combination will enable us to configure a global and more specific error handling mechanism without the need to implement it every time for every controller class.

Q24. What Does the @ExceptionHandler Annotation Do?

The @ExceptionHandler annotation allows us to define a method that will handle the exceptions. We may use the annotation independently, but it's a far better option to use it together with the @ControllerAdvice. Thus, we can set up a global error handling mechanism. In this way, we don't need to write the code for the exception handling within every controller.

Let's take a look at the example from our article about [Error Handling for REST with Spring](#):

```
@ControllerAdvice
```

```

public class RestResponseEntityExceptionHandler
    extends ResponseEntityExceptionHandler {

    @ExceptionHandler(value = { IllegalArgumentException.class,
        IllegalStateException.class })
    protected ResponseEntity<Object> handleConflict(RuntimeException ex,
        WebRequest request) {
        String bodyOfResponse = "This should be application specific";
        return handleExceptionInternal(ex, bodyOfResponse, new HttpHeaders(),
            HttpStatus.CONFLICT, request);
    }
}

```

We should also note that this will provide `@ExceptionHandler` methods to all controllers that throw `IllegalArgumentException` or `IllegalStateException`. The exceptions declared with `@ExceptionHandler` should match the exception used as the argument of the method. Otherwise, the exception resolving mechanism will fail at runtime.

One thing to keep in mind here is that it's possible to define more than one `@ExceptionHandler` for the same exception. We can't do it in the same class though since Spring would complain by throwing an exception and failing on startup.

On the other hand, if we define those in two separate classes, the application will start, but it'll use the first handler it finds, possibly the wrong one.

Q25. Exception Handling in Web Applications

We have three options for exceptions handling in Spring MVC:

- per exception
- per controller
- globally

If an unhandled exception is thrown during web request processing, the server will return an HTTP 500 response. To prevent this, **we should annotate any of our custom exceptions with the `@ResponseStatus` annotation.** This kind of exceptions is resolved by `HandlerExceptionResolver`.

This will cause the server to return an appropriate HTTP response with the specified status code when a controller method throws our exception. We should keep in mind that we shouldn't handle our exception somewhere else for this approach to work.

Another way to handle the exceptions is by using the `@ExceptionHandler` annotation. We add `@ExceptionHandler` methods to any controller and use them to handle the exceptions thrown from inside that controller. These methods can handle exceptions without the `@ResponseStatus` annotation, redirect the user to a dedicated error view, or build a totally custom error response.

We can also pass in the servlet-related objects (`HttpServletRequest`, `HttpServletResponse`, `HttpSession`, and `Principal`) as the parameters of the handler methods. But, we should remember that we can't put the `Model` object as the parameter directly.

The third option for handling errors is by `@ControllerAdvice` classes. It'll allow us to apply the same techniques, only this time at the application level and not only to the particular controller. To enable this, we need to use the `@ControllerAdvice` and the `@ExceptionHandler` together. This way exception handlers will handle exceptions thrown by any controller.

For more detailed information on this topic, go through the [Error Handling for REST with Spring](#) article.

4. Conclusion

In this article, we've explored some of the Spring MVC related questions that could come up at the technical interview for Spring developers. You should take these questions into account as a starting point for further research since this is by no means an exhaustive list.

We wish you good luck in any upcoming interviews!

1. What are the advantages of using Spring Boot?

The advantages of Spring Boot are listed below:

- Easy to understand and develop spring applications.
- Spring Boot is nothing but an existing framework with the addition of an embedded HTTP server and annotation configuration which makes it easier to understand and faster the process of development.
- Increases productivity and reduces development time.
- Minimum configuration.
- We don't need to write any XML configuration, only a few annotations are required to do the configuration.

2. What are the Spring Boot key components?

Below are the four key components of spring-boot:

- Spring Boot auto-configuration.
- Spring Boot CLI.
- Spring Boot starter POMs.
- Spring Boot Actuators.

3. Why Spring Boot over Spring?

Below are some key points which spring boot offers but spring doesn't:

- Starter POM.
- Version Management.
- Auto Configuration.
- Component Scanning.
- Embedded server.
- InMemory DB.

- Actuators

Spring Boot simplifies the spring feature for the user:

Spring vs Spring Boot

You can download a PDF version of Spring Boot Interview Questions.



4. What is the starter dependency of the Spring boot module?

Spring boot provides numbers of starter dependency, here are the most commonly used -

- Data JPA starter.
- Test Starter.
- Security starter.
- Web starter.
- Mail starter.
- Thymeleaf starter.

5. How does Spring Boot works?

Spring Boot automatically configures your application based on the dependencies you have added to the project by using annotation. The entry point of the spring boot application is the class that contains `@SpringBootApplication` annotation and the main method.

Spring Boot automatically scans all the components included in the project by using `@ComponentScan` annotation.

6. What does the `@SpringBootApplication` annotation do internally?

The `@SpringBootApplication` annotation is equivalent to using `@Configuration`, `@EnableAutoConfiguration`, and `@ComponentScan` with their default attributes. Spring Boot enables the developer to use a single annotation instead of using multiple. But, as we know, Spring provided loosely coupled features that we can use for each annotation as per our project needs.

7. What is the purpose of using `@ComponentScan` in the class files?

Spring Boot application scans all the beans and package declarations when the application initializes. You need to add the `@ComponentScan` annotation for your class file to scan your components added to your project.

8. How does a spring boot application get started?

Just like any other Java program, a Spring Boot application must have a main method. This method serves as an entry point, which invokes the `SpringApplication#run` method to bootstrap the application.

```
@SpringBootApplication
public class MyApplication {

    public static void main(String[] args) {

        SpringApplication.run(MyApplication.class);
        // other statements
    }
}
```

9. What are starter dependencies?

Spring boot starter is a maven template that contains a collection of all the relevant transitive dependencies that are needed to start a particular functionality.

Like we need to import `spring-boot-starter-web` dependency for creating a web application.

```
<dependency>
<groupId> org.springframework.boot</groupId>
<artifactId> spring-boot-starter-web </artifactId>
</dependency>
```

10. What is Spring Initializer?

Spring Initializer is a web application that helps you to create an initial spring boot project structure and provides a maven or gradle file to build your code. It solves the problem of setting up a framework when you are starting a project from scratch.

11. What is Spring Boot CLI and what are its benefits?

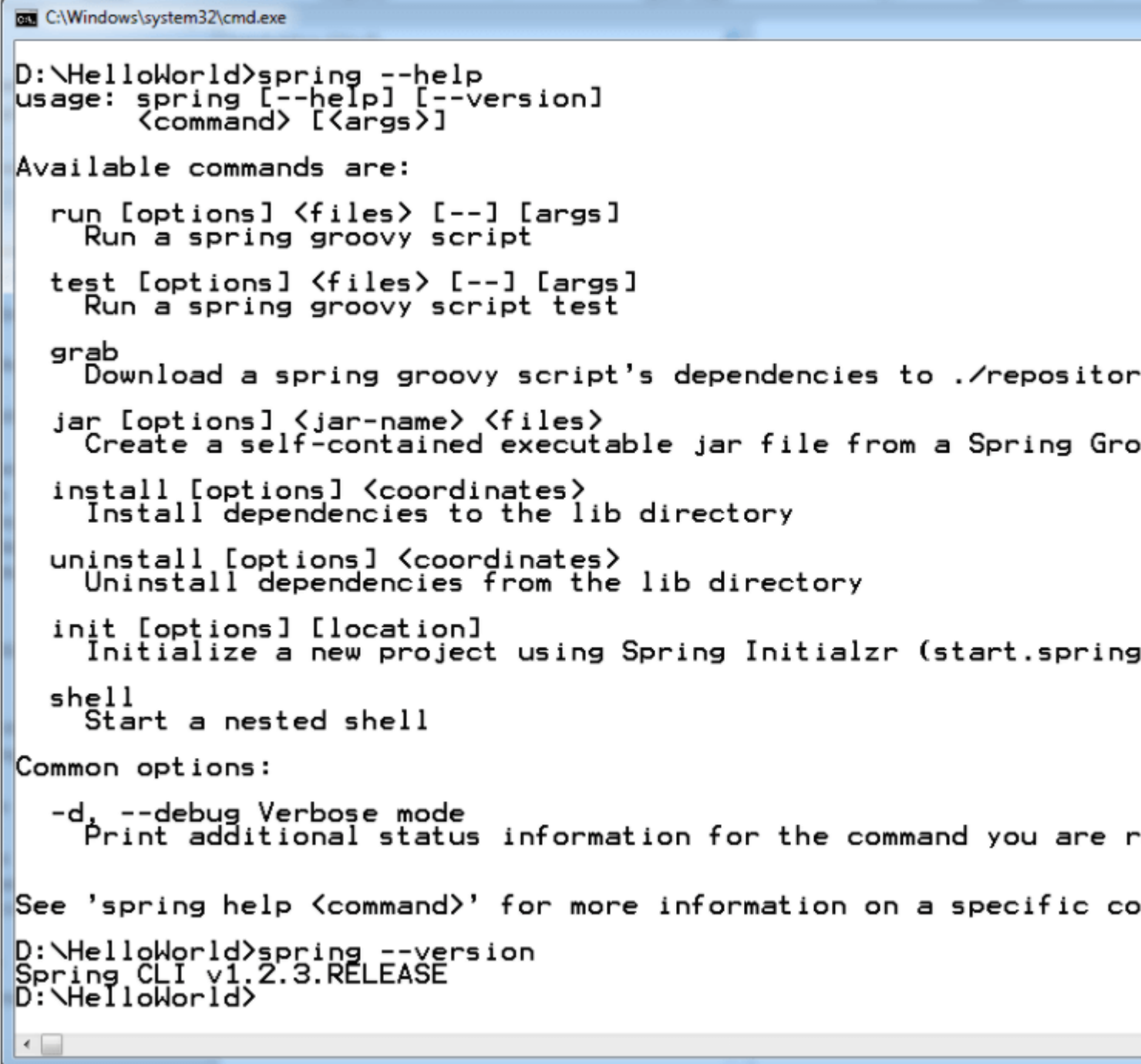
Spring Boot CLI is a command-line interface that allows you to create a spring-based java application using Groovy.

Example: You don't need to create getter and setter method or access modifier, return statement. If you use the JDBC template, it automatically loads for you.

12. What are the most common Spring Boot CLI commands?

-run, -test, -grab, -jar, -war, -install, -uninstall, --init, -shell, -help.

To check the description, run `spring --help` from the terminal.



```
C:\Windows\system32\cmd.exe

D:\HelloWorld>spring --help
usage: spring [--help] [--version]
       <command> [<args>]

Available commands are:

  run [options] <files> [--] [args]
      Run a spring groovy script

  test [options] <files> [--] [args]
      Run a spring groovy script test

  grab
      Download a spring groovy script's dependencies to ./repositor

  jar [options] <jar-name> <files>
      Create a self-contained executable jar file from a Spring Gro

  install [options] <coordinates>
      Install dependencies to the lib directory

  uninstall [options] <coordinates>
      Uninstall dependencies from the lib directory

  init [options] [location]
      Initialize a new project using Spring Initializr (start.spring

  shell
      Start a nested shell

Common options:

  -d, --debug Verbose mode
      Print additional status information for the command you are r

See 'spring help <command>' for more information on a specific co

D:\HelloWorld>spring --version
Spring CLI v1.2.3.RELEASE
D:\HelloWorld>
```

Spring Boot CLI Commands

Advanced Spring Boot Questions

13. What Are the Basic Annotations that Spring Boot Offers?

The primary annotations that Spring Boot offers reside in its org.springframework.boot.autoconfigure and its sub-packages. Here are a couple of basic ones:

`@EnableAutoConfiguration` – to make Spring Boot look for auto-configuration beans on its classpath and automatically apply them.

`@SpringBootApplication` – used to denote the main class of a Boot Application. This annotation combines `@Configuration`, `@EnableAutoConfiguration`, and `@ComponentScan` annotations with their default attributes.

14. What is Spring Boot dependency management?

Spring Boot dependency management is used to manage dependencies and configuration automatically without you specifying the version for any of that dependencies.

15. Can we create a non-web application in Spring Boot?

Yes, we can create a non-web application by removing the web dependencies from the classpath along with changing the way Spring Boot creates the application context.

16. Is it possible to change the port of the embedded Tomcat server in Spring Boot?

Yes, it is possible. By using the `server.port` in the `application.properties`.

17. What is the default port of tomcat in spring boot?

The default port of the tomcat server is 8080. It can be changed by adding `server.port` properties in the `application.properties` file.

18. Can we override or replace the Embedded tomcat server in Spring Boot?

Yes, we can replace the Embedded Tomcat server with any server by using the Starter dependency in the `pom.xml` file. Like you can use `spring-boot-starter-jetty` as a dependency for using a jetty server in your project.

19. Can we disable the default web server in the Spring boot application?

Yes, we can use `application.properties` to configure the web application type i.e `spring.main.web-application-type=none`.

20. How to disable a specific auto-configuration class?

You can use `exclude` attribute of `@EnableAutoConfiguration` if you want auto-configuration not to

apply to any specific class.

```
//use of exclude  
@EnableAutoConfiguration(exclude={className})
```

21. Explain @RestController annotation in Spring boot?

It is a combination of @Controller and @ResponseBody, used for creating a restful controller. It converts the response to JSON or XML. It ensures that data returned by each method will be written straight into the response body instead of returning a template.

22. What is the difference between @RestController and @Controller in Spring Boot?

@Controller Map of the model object to view or template and make it human readable but @RestController simply returns the object and object data is directly written in HTTP response as JSON or XML.

23. Describe the flow of HTTPS requests through the Spring Boot application?

On a high-level spring boot application follow the MVC pattern which is depicted in the below flow diagram.

Spring Book



24. What is the difference between RequestMapping and GetMapping?

RequestMapping can be used with GET, POST, PUT, and many other request methods using the method attribute on the annotation. Whereas getMapping is only an extension of RequestMapping which helps you to improve on clarity on request.

25. What is the use of Profiles in spring boot?

While developing the application we deal with multiple environments such as dev, QA, Prod, and each environment requires a different configuration. For eg., we might be using an embedded H2 database for dev but for prod, we might have proprietary Oracle or DB2. Even if DBMS is the same across the environment, the URLs will be different.

To make this easy and clean, Spring has the provision of Profiles to keep the separate configuration of environments.

26. What is Spring Actuator? What are its advantages?

An actuator is an additional feature of Spring that helps you to monitor and manage your application when you push it to production. These actuators include auditing, health, CPU usage, HTTP hits, and metric gathering, and many more that are automatically applied to your application.

27. How to enable Actuator in Spring boot application?

To enable the spring actuator feature, we need to add the dependency of “spring-boot-starter-actuator” in pom.xml.

```
<dependency>
<groupId> org.springframework.boot</groupId>
<artifactId> spring-boot-starter-actuator </artifactId>
</dependency>
```

28. What are the actuator-provided endpoints used for monitoring the Spring boot application?

Actuators provide below pre-defined endpoints to monitor our application -

- Health
- Info
- Beans

- Mappings
- Configprops
- Httptrace
- Heapdump
- Threaddump
- Shutdown

29. How to get the list of all the beans in your Spring boot application?

Spring Boot actuator “/Beans” is used to get the list of all the spring beans in your application.

30. How to check the environment properties in your Spring boot application?

Spring Boot actuator “/env” returns the list of all the environment properties of running the spring boot application.

31. How to enable debugging log in the spring boot application?

Debugging logs can be enabled in three ways -

- We can start the application with --debug switch.
- We can set the logging.level.root=debug property in application.property file.
- We can set the logging level of the root logger to debug in the supplied logging configuration file.

32. Where do we define properties in the Spring Boot application?

You can define both application and Spring boot-related properties into a file called application.properties. You can create this file manually or use Spring Initializer to create this file. You don't need to do any special configuration to instruct Spring Boot to load this file, If it exists in classpath then spring boot automatically loads it and configure itself and the application code accordingly.

33. What is dependency Injection?

The process of injecting dependent bean objects into target bean objects is called dependency injection.

- Setter Injection: The IOC container will inject the dependent bean object into the target bean object by calling the setter method.
- Constructor Injection: The IOC container will inject the dependent bean object into the target bean object by calling the target bean constructor.
- Field Injection: The IOC container will inject the dependent bean object into the target bean object

by Reflection API.

34. What is an IOC container?

IoC Container is a framework for implementing automatic dependency injection. It manages object creation and its life-time and also injects dependencies into the class.

Spring Boot MCQs

1.

Spring is used for?

Java Framework

Web Development Framework

MVC Framework

All of the Above

35 MySQL Interview Questions and Answers Every Developer Should Know

By [Simplilearn](#) Last updated on Mar 24, 20222096



Table of Contents

[Basic MySQL Interview Questions](#)

[Intermediate MySQL Interview Questions](#)

[Advanced MySQL Interview Questions](#)

[MySQL](#) is the most popular open-source relational database management system (RDBMS), typically used with [PHP](#). It is fast, reliable, and easy to run on the web and the server. MySQL is the world's most popular open-source [database software](#) and a preferred choice for critical business applications by giants like Yahoo, Suzuki, and NASA. Naturally, there are a lot of career opportunities for MySQL experts. This article is a Q/A guide on how to answer MySQL interview questions.

MySQL uses standard [SQL programming](#) for the creation, modification, and extraction of data

from a relational database. The data is stored in tables consisting of rows and columns. Users can interact directly with MySQL or use it to implement applications that need relational database capability. MySQL jobs range from MySQL Developer, [MySQL Database Administrator](#), MySQL Database Engineer, and more.

Here are some of the most frequently asked MySQL interview questions and how to answer them.

Full Stack Java Developer Course

In Partnership with HIRIST and HackerEarth [EXPLORE COURSE](#)



Basic MySQL Interview Questions

1. What is MySQL?

MySQL is a relational database management system based on SQL (Structured Query Language). It is an open source software owned by Oracle and can run on various platforms. Most websites or web applications are developed using MySQL.

2. In which language has MySQL been written?

MySQL is written in [C](#) and [C++](#). Its SQL parser is written in yacc.

3. What are the advantages of using MySQL?

MySQL is a fast, stable, and reliable solution that provides advantages like:

- Data Security – most secure and reliable [database management](#) system
- Flexibility – runs on all operating systems; features 24X7 support and enterprise indemnification
- High Performance – powerful, designed to meet highly demanding applications while maintaining optimum speed and high performance
- On-demand Scalability – offers on-demand scalability and complete customization
- Enterprise-level SQL Features – the enterprise edition includes advanced features and management tools, and technical support for enterprise
- Full-text Indexing and Searching – has support for full-text indexing and searching
- Query Caching – unique memory caches help enhance the speed of MySQL greatly
- Replication – one MySQL server can be duplicated on another, resulting in numerous benefits

4. What is a database?

A database is a structured repository of data stored electronically in a computer system and organized in a way that data can be quickly searched and information rapidly retrieved. A database is generally controlled by a database management system.

5. What does 'MySQL' stand for?

'My' in MySQL represents the first name of its co-founder, Michael Widenius' daughter, My Widenius. SQL is an abbreviation for the term "Structured Query Language". SQL is also used in databases like Oracle and Microsoft SQL Server.

6. How to check MySQL version?

The command 'MySQL-v' can be used to check MySQL version on Linux

7. What does a MySQL database contain?

A MySQL database contains one or many tables, with each table containing several records or rows. Within these rows, data is contained in various columns or fields.

8. List the ways to interact with MySQL.

There are 3 main ways users can interact with MySQL:

- Using a command line
- Through a web interface
- Using a [programming language](#)

9. What are the different tables in MySQL?

They are:

- MyISAM
- HeapMerge
- INNO DB
- ISAM

10. What are MySQL Database Queries?

A query is a request for data or information from a database. Users can query a database for specific information, and the resultant record/records are returned by MySQL.

FREE Java Certification Training

Learn A-Z of Java like never before **ENROLL NOW**



11. What are some common MySQL commands?

Some common MySQL commands are:

- CREATE – To create Table
- INSERT – To insert data
- JOIN – To join tables
- DELETE – To delete a row from a table
- ALTER – To alter database or table
- BACKUP – to back up a table
- DROP – To delete a database or table
- CREATE INDEX – to add indexing to a column in a table
- GRANT – To change user privileges
- TRUNCATE – Empty a table
- EXIT – to exit

12. How to create a database in MySQL?

The CREATE DATABASE command can be used to create a new database.

13. How to create table using MySQL?

The following query can be used to create a table:

```
CREATE TABLE 'history' (
```

```
'author' VARCHAR(128),
```

```
'title' VARCHAR(128),
```

```
'type' VARCHAR(16),
```

```
'year' CHAR(4))
```

```
ENGINE = InnoDB;
```

A table "history" gets created in the selected database.

14. How to insert data in MySQL?

The INSERT INTO statement is used to insert new records in a table in MySQL.

The two main syntaxes are:

```
INSERT INTO table_name (column 1, column 2, column 3,...columnN)
```

```
VALUES (value 1, value 2, value 3,...valueN)
```


15. How do you remove a column form a database?

The DROP command is used to remove a column from a database.

Alter table 'history' drop column title;

16. How to create an index?

There are different types of indexes in MySQL, like a regular INDEX, a [PRIMARY KEY](#), or a FULLTEXT index. Indexes are created on a column basis. Indexing helps to quickly search for results, either by ordering the data on disk or by telling the SQL engine which location to find your data in.

Syntax:

```
ALTER TABLE history ADD INDEX(author(10));
```

17. How do you delete data from MySQL table?

We use the DELETE statement to remove records from a table.

The syntax is as follows:

```
DELETE FROM table_name WHERE column_name
```

18. How can you view a database in MySQL?

The SHOW DATABASES command allows the user to view all databases on the MySQL server host.

```
mysql> SHOW DATABASES;
```

19. How to import database in MySQL?

There are two ways to import database or move data from one place to another:

- Command Line Tool
- MySQL Workbench

20. What are numeric data types in MySQL?

There are numeric data types for integer, fixed-point, floating-point, and bit values in MySQL. Except for BIT, the other numeric data types can be signed or unsigned.

Examples:

INT - Standard Integer

TINYINT - Very Small Integer

SMALLINT - Small Integer

MEDIUMINT - Medium-sized Integer

BIGINT - Large Integer

DECIMAL - Fixed-point number

FLOAT - Single-precision floating-point number

DOUBLE - Double-precision floating-point number

BIT - Bit-field

21. What are string data types in MySQL?

The string [data types in MySQL](#) are:

- CHAR
- VARCHAR
- BINARY
- VARBINARY
- TINYBLOB
- BLOB
- MEDIUMBLOB
- LONGBLOB
- TINYTEXT
- TEXT
- MEDIUMTEXT
- LONGTEXT
- ENUM
- SET
- NULL

Free Course: Introduction to SQL

Learn MySQL, PostgreSQL and SQL Server **ENROLL NOW**



22. What are temporal data types in MySQL?

MySQL provides temporal data types for date and time, as well as a combination of date and time. These are:

DATE - A date value in CCYY-MM-DD Format

TIME - A Time value in hh : mm :ss format

DATETIME - Date and time value in CCYY-MM-DD hh : mm :ss format

TIMESTAMP - A timestamp value in CCYY-MM-DD hh : mm :ss format

YEAR - A year value in CCYY or YY format

23. What is BLOB?

BLOB is an acronym for a binary large object. It is a string data type used to hold a variable amount of data.

24. How do you add users in MySQL?

The CREATE command, along with necessary credentials, can be used to add users.

```
CREATE USER 'testuser' IDENTIFIED BY 'sample password';
```

Intermediate MySQL Interview Questions

25. What are views in MySQL?

A view is a set of rows returned when a particular query is executed in MySQL. It is also known as a virtual table, which does not store any data of its own but displays data stored in other tables.

26. How to create and execute views?

The CREATE VIEW command is used to create a view in MySQL.

The syntax is:

```
CREATE VIEW [databasename.] view_name [(column_list)] AS select-statement;
```

27. What are MySQL triggers?

A task that is executed automatically in response to a predefined database event is known as a trigger. Each trigger is associated with a table and is activated by commands like INSERT, DELETE, or UPDATE.

28. How many triggers are possible in MySQL?

There are 6 different types of triggers in MySQL:

- Before Insert
- After Insert
- Before Update
- After Update
- Before Delete

- After Delete

29. What is MySQL server?

The server 'mysqld' is the MySQL server, which performs all manipulation of databases and tables.

30. What are the clients and utilities in MySQL?

There are several MySQL programs available to help users communicate with the server. Some important ones for administrative tasks are:

.mysql – this interactive program helps to send [SQL statements](#) to the server and view the results. One can even use MySQL to use batch scripts.

.mysqladmin – this administrative program helps perform tasks like shutting down the server, checking configuration, monitoring status if it is not functioning properly.

.mysqldump – for backing up databases or copying them to another server

.mysqlcheck and myisamchk – these programs help perform table checking, analysis, and optimization, plus repairs for damaged tables.

31. What types of relationships are used in MySQL?

Three types of relationships are used in MySQL:

- One-to-one – items with one-to-one relation can be included as columns in the same table
- One-to-many – or many-to-one relationships are seen when one row in a table is related to multiple rows in another table
- Many-to-many – many rows in a table are linked to many rows in another table

Advanced MySQL Interview Questions

32. Explain the logical architecture of MySQL

The top layer comprises the services required by most network-based client/server tools like connection handling, security, authentication, etc.

The 2nd layer comprises code for query parsing, optimization, analysis, caching, and all built-in functions.

The 3rd layer comprises storage engines where storage and retrieval of data stored in MySQL is performed.

33. What is Scaling?

Scaling capacity in MySQL is the ability to handle the load in terms of:

- Data quantity
- Number of users
- User activity
- Size of related datasets

34. What is Sharding?

The process of breaking up large tables into smaller chunks or shards spread across many servers is called sharding. It makes querying, maintenance, and other tasks faster.

35. What are Transaction Storage Engines?

The InnoDB storage engine enables users to use the transaction facility of MySQL.

Get a firm foundation in Java, the most commonly used programming language in software development with the [Java Certification Training Course](#).

We hope this list of MySQL interview questions will be helpful for you. Register with Simplilearn today to get access to top-rated courses on [database training](#) and [full stack web development](#).

1. Give an example of a project you have worked on and the technologies involved. How did you make these choices?



Project management

Ans: This helps in knowing the methodology of the full stack web developer and also gives an idea of his sharpness and precision in choosing the right toolset. You need to be as specific as possible and go in depth while speaking about the reason of choosing a particular toolset. Show a balance between your ability to develop both on the front-end and the back-end of the web

application. It is okay to show that you have more experience in one side of the development game than the other but to demonstrate that you have the ability to work on both the ends of the application.

2. Can you relate of an experience when you found your colleagues code to be inefficient? How did you deal with it?

Ans: This helps to determine a candidate's quality standards and also gives an impression of his team playing skills. You must demonstrate that you have high-quality assurance standards, and show comfort in pointing out flaws or for that matter bugs to others. However, you must pay focus on portraying that you are good at giving positive feedback, getting the work done without creating resentment.

3. Are you aware of design patterns?

Ans: While answering this question you should show your ability to understand common errors that might creep in while creating web applications. If your knowledge is formal and deep, you must push the employer to gain confidence in the experience you possess in understanding a clean and readable code.

4. What is the best implementation or debugging you have done in the past?

Ans: This is a tricky question. This will actually give the hiring manager an idea of both the complexity and the style of projects you have done in the past. You should explicitly mention the issues you faced and the measures you took to overcome that roadblock. You can additionally speak about the learnings you earned from the issue.

5. Do you enjoy management or execution more?

Ans: Every senior professional will be asked this **full stack web developer interview question**. Recruiters want to know if you want to stay in a technical role or would want to switch to a managerial position. They also at times might push a bit and ask if you prefer to work alone or in paired groups. It's

important, to be honest here. The company's requirements might be different from your interests, so it is ideal that the expectations are set straight from the get-go.

6. What is Continuous Integration?

Ans: Continuous Integration is the process of using codes that are specially designed & automated for testing. This process helps the developers to easily deploy codes during the time of production. Web developers use this process to integrate codes several times a day. These codes are checked automatically Continuous Integration helps in detecting errors quickly and locating them more easily.

7. What is Multi-threading?

Ans: Multi-threading is a process of improving the performance of CPU. Generally, it is the ability of a program to get managed by more than one user at a time or manage multiple requests by the same user. Multi-threading is done by executing multiple processes that can be supported by the operating system.

Common Full Stack Web Developer Interview Questions and Answers

There are always some **full stack web developer interview questions and answers** that would be very common to ask. And to make this post exhaustive in order to give the maximum benefit to the user, we have researched them as well. Here are the top commonly asked **full stack web developer interview questions**.

1. What's the most puzzling programming challenge you have come across recently?



Programming challenge

Ans: Speak about the most recent bug or a discrepancy you came across and explain how you went ahead to overcome it. Tell the interviewer that you are a person who believes in collaborative work. Depict how you could solve the issue with the help of a colleague, online community or your mentor. It is always good to ask when you don't know.

2. What are you coding currently?

Ans: A person who loves technology is always working with it. Whether it be for your company or for your own recreational reasons. Good programmers will always have something to share. Candidates who code willingly for personal learnings will always stand out here.

3. What is the biggest mistake you did in any of your projects? How did you rectify it?

Ans: As they say, 'No man is an island'. Similarly, you can't be working on technology and be right all the time. That is not imaginable. You need to be honest here and talk about a mistake that you think was serious. To add to it,

speak about your learnings from this mistake and explain the procedure you adopted to minimize the damage done.

4. Explain Inversion of Control.

Ans: This question is very commonly asked to check the candidate's understanding of design patterns. It is a broad term but is more specifically used by software developers for describing a pattern which is used to decouple layers and components in a system.

5. What is the main difference between GraphQL and REST?

Ans: This is a moderately difficult question but a good developer would be able to get through with this in ease. An important difference between GraphQL and REST is that GraphQL doesn't deal with dedicated resources. Everything referred to as a graph is connected and can be queried to app needs.

6. List some common ways to reduce the load time of a web application?

Ans: There are quite a lot of ways to reduce the loading time of a web application like enabling browser caching, optimizing images, minifying resources, minimizing HTTP requests and reducing redirects.

7. What is Long Polling?

Ans: Long Polling is a web development pattern that is used to surpass pushing data from the server to the client. By using Long Polling pattern, the Client requests information from the server. If the server does not have any information about the client, the server holds the request and waits for some information to be available rather than sending an empty resource.

Final Notes

Whether you are a fresher or an experienced candidate, no employer would expect you to know everything. But it is important to be prepared for the

interview and show interest in the job. At the same time, being honest and transparent will always pay off.

Taking [Python Programming course](#) can help you as a guiding light to build a successful career in Python Programming.

Wish you all the best for your full stack web developer interview!