**SpringBoot\_Interview\_Questions**

**Q1) How many types of annotation are available in springboot**

**A)** There are 4 types of annotation is available in springboot.

i.) Annotations for configuration

ii.) Annotations for data supply

iii.) Annotations for spring bean life cycle

iv.) misalliance annotation

**i.) Annotations for configuration:** @Service, @Controller, @Component, @Repository, @Configuration, @Bean, @RestController

**ii.) Annotations for data supply:** @Autowired, @Value, @Qualifier, @Primary

**iii.) Annotations for spring bean life cycle:** @PostContructor, @PreDestroy

**iv.) misalliance annotation:** @PropertySource, @ComponantScan, @ImportResource, @Import, @Scope, @Lazy

**Q2) What type of annotations you are used while creating Rest API.**

**A)** We can use below annotations in controller class

@Component

@Path

@GetMapping

@PostMapping

@PutMapping

@DeleteMapping

We can use the @Service annotation in service class to make java class as spring bean

We can use @Repository annotation in repository class to make java class as spring bean and DAO

**Q3) How you are connecting database**

**A)** We can connect the database by using application.properties file

We should update the respective database details in application.properties and add the respective dependences in pom.xml file then update the project.

**Q4) By using which annotation we can get the values from property file**

**A)** By using @Value annotation, we can read the values from any properties file under the project class path.

In class level we can use @PropertySource annotation

**Q5) Can we externalize the configuration in any of the springboot application**

**A)** Springboot allow us to externalize our configuration, So we can work with same application code in different environment.

We can use properties file, YAML file, environment variables and command line arguments to externalize our configuration. Property values can be injected directly into our application beans by using the @Value annotation, accessed via spring’s environment.

**Q6) What is the difference between Spring and SpringBoot**

**A)**

|  |  |
| --- | --- |
| **Spring** | **SpringBoot** |
| Spring framework helps to create loosely coupled applications | Springboot helps to create stand-alone applications |
| In the spring frame work we need to setup the servers explicitly | Springboot offers built-in or embedded server such as tomcat and jetty are available |
| To run spring application deployment descriptor is require | In springboot there is no need for the deployment descriptor |
| Spring framework does not support for the in-memory database | Springboot provide supports for the in-memory database such as H2 |
| Spring framework requires too many lines of code (boilerplate code) even for minimal tasks | In springboot we can avoid boilerplate code which reduce time and increase productivity |
| In the spring framework XML configuration required | No need for XML configuration in springboot |
| Spring framework does not provide any plugin for maven, gradle, etc like springboot | Springboot provide build tool plugins for maven and gradle. The plugin offers a variety of features |
| in spring framework, we have to build configuration manually | In springboot there are defaults configurations that allows faster bootstrapping |

**Q7) What is the use of springboot devtools**

**A)** The devtools stands for developer tools, the main aim of the module is to try and improve the development time while working with springboot application.

Springboot devtools pick up the changes and restart the application.

The main features of springboot devtools is

**i.) Property Defaults:** It will disable the caching and allows us to update pages without the need of restarting the application.

**ii.) Automatic Restart:** It means reloading of java classes and configure it at server side, after server-side changes it deployed dynamically.

**iii.) Live Reload:** The springboot devtools module include an embedded server called Live reload it allows the application to automatically trigger a browser refresh whenever we make changes in the resource. It is also known as Auto-refresh

Note: we can disable the live reload by setting the property spring.devtools.livereload.enable to false

**iv.) Remote Debug Tuning:** Springboot can tunnel JDWP (java debug wire protocal) over HTTP directly to the application. It can work application deployment cloud provides that only expose port 80 and 443

**v.) Remote Update and Restart:** it support the remote application update and restart

**Q8) What @SpringBootApplication annotation does**

**A)** @SpringBootApplication annotation is used to mark a configuration class that declares one or more @Bean methods and also triggers auto-configuration and component scanning

It is same as declaring a class with @Configuration, @EnableAutoConfiguration and @ComponentScna annotations.

**Q9) How to exclude some packages from base package**

**A)** We can use the exclude attribute with the @SpringBootApplication annotation.

@SpringBootApplication(exclude={Employee.class})

**Q10) What is the difference between @Controller and @RestController**

**A)**

|  |  |
| --- | --- |
| @Controller | @RestController |
| @Controller is used to mark class as spring mvc controller | @RestController annotation is a special controller used in RESTfull webservices, and it’s the combination of @Controller and @Response body annotation |
| It is specialized version of @Component annotation | It is specialized version of @Controller annotation |
| In @Controller we can return a view in spring web mvc | @RestController we cannot return a view |
| @Controller annotation indicates that the class is a controller like web controller | @RestController annotation is a controller where @Requestmethods assumes @Responsebody semantics by default |
| In @Controller, we need to use @ResponseBody on every handler method | In @Restcontroller we don’t need to use @ResponseBody on every handler method |

**Q11) What is the difference between Get() and Load()**

**A)**

|  |  |
| --- | --- |
| **Get()** | **Load()** |
| It is used to fetch the data from the database for the given identifier | It is also used to fetch the data from database for the given identifier |
| Get() method always hit the database | Load() method doesn’t hit the database every time |
| If object not found for the given identifier then it will return null value | It will throw object not found exception |
| It will return the fully initialized object so this method eager load object | It will always returns proxy object so this method is lazy load the object |
| It is slower than load() method, because it return fully initialized object which impact the performance of the application | It is slightly faster than Get() method |

**Q12) What is default view Resolver in springboot.**

**A)** InternalViewResourceResolver is the default view resolver.

**Q13) What is the default handler mapping in springboot**

**A)** RequestMappingHandlerMapping is the default handler mapping

**Q14) What is web services**

**A)** A web services are a set of open protocols and standards that allow data to be exchange between different applications or system

Web services can be used by software programs written in a verity of programming languages and running on a variety of platforms to exchange data via computers network such as the internet in a similar way to inter-process communication on a single computer.

Any software application or cloud technology that uses standardized web protocols (HTTP or HTTPS) to connect and exchange the data.

There are two API’s defined by java for developing web service applications

JAX-WS: For SOAP web service, There are two ways to write JAX-WS application code by RPC style and document style.

JAX-RS: For RESTful web service, There are mainly 2 implementations currently: Jersey and RESTeasy

**Q15) What is the different between RESTful web services and SOAP web services**

**A)**

|  |  |
| --- | --- |
| **SOAP web services** | **RESTful web services** |
| SOAP is a protocol | Rest is an architectural approach |
| In SOAP, the data exchange format is always XML | There is no strict data exchange format we can use JSON, XML, etc.. |
| XML is the most popular data exchange format in SOAP web services | JSON is the most popular data exchange format in RESTful web services |
| SOAP uses web service definition language (WSDL) | REST does not have any standard definition language |
| SOAP does not have any restriction on transport. We can use either HTTP or MQ | RESTful services use the most popular HTTP protocol |
| SOAP web services are typical to implement | RESTful services are easier to implement than SOAP |
| SOAP web services use the JAX-WS API | RESTful web services use the JAX-RS API |
| SOAP can not use RESTful services because it is a protocol | RESTful service can use SOAP web service because it is an architectural approach that can use any protocol like HTTP and SOAP |

**Q16) What is the use of application properties file**

**A)** Application properties file is used to write the application related property into that file. This file contains the different configurations which is required to run our application in a different environment, each environment will have a different defined by it.

Inside the application property file, we define every type of property like changing the server port, database connectivity, connection to the eureka server and many more.

**Q17) What is the application YML file**

**A)** The application properties file is not that readable, so most of the time developers choose the application.yml file over application.properties file.

YML is a superset of JSON and as such is a very convenient format for specifying hierarchical configuration data.

**Q18) What is Auto configuration**

**A)** We can enable the auto-configuration feature by using the annotation **@EnableAutoConfiguration**.

* @Conditional annotation acts as a base for the Spring Boot auto-configuration annotation extensions.
* It automatically registers the beans with @Component, @Configuration, @Bean, and meta-annotations for building custom stereotype annotations, etc.
* The annotation @EnableAutoConfiguration is used to enable the auto-configuration feature.
* The @EnableAutoConfiguration annotation enables the auto-configuration of Spring ApplicationContext by scanning the classpath components and registering the beans.
* This annotation is wrapped inside the @SpringBootApplication annotation along with @ComponentScan and @SpringBootConfiguration annotations.
* When running main() method, this annotation initiates auto-configuration.

**Q19) How to create prototype bean in spring boot**

**A)** Bean scope refers to the life cycle of the bean, that means when the object of bean will be initiated, how long does that object live and how many objects will be created for that bean throughout.

We use @Scope to define the scope of a @Component class or a [@Bean](http://www.javaguides.net/2018/09/spring-bean-annotation-with-example.html) definition. It can be either singleton, prototype, request, session, globalSession or some custom scope.

Prototype Scope: If the scope is declared prototype, then spring consider prototype scope. A new instance will be created for a single bean definition every time a request is made of that bean.

<bean

Id=””,

Class=”bean.Helloworld”

Scope=”prototype/>

**Q20) What is the component class**

**A)** The java class that contains state (non-static member variables) and behaviour (methods) and state must be used inside the business logics of behaviour (methods) is called bean class or component class.

Component or bean class totally different from java bean class because java bean class methods are just setter and getter methods not having any business logics where as the component methods are business methods having useful business logics and implementation logics of the application.

**Q21) What is Spring data JPA**

**A)** JPA is a Java specification (Jakarta Persistence API) and it manages relational data in Java applications. To access and persist data between Java object (Plain Old Java object)/ class and relational database, we can use JPA. Upon Object-Relation Mapping (ORM), it follows the mechanisms. It has the runtime Entity Manager API and it is responsible for processing queries and transactions on the Java objects against the database. The main highlight is it uses JPQL (Java Persistent Query Language) which is platform-independent. JPA mainly covers persistence in terms of

Spring data jpa consist of three repository interfaces

* CRUD Repository- it provides the CRUD operations
* Paging and Sorting Repository – Provide the methods to do pagination and sort records.
* JPA Repository – Provide JPA related methods such as flushing the persistence context and delete records in a batch.
* The Java Persistence API
* Object-Relational metadata
* Moreover, under the persistence package API is defined.
* We cannot say that JPA is a framework, but It defines a concept and it can be implemented by any framework.

**Q22) In springboot data jpa how we can create a custom repository**

**A)** Spring data jpa already provides many solution that allow us to query data easier such as query method or four repository interfaces ( JPA repository, paging and sorting repository, CRUD repository, Repository). Which help us a lot when building an application with spring data JPA. But sometimes we still need to customer query or method to meet expectations or requirements.

we can create a custom repository in Spring Data JPA. We just need to let EmployeeRepository implement the custom interface and define the implementation for the customized repository.

We just have to provide the implementation of the customizing repository, other parts let Spring Framework complete.

Then we can extends both custom repository along with any of one four repositories to get all the features of the built repository of spring data JPA.

**Q23) What is the springboot actuator**

**A)** Springboot actuator includes additional features that help us to monitor and manages the springboot application.

The actuator contains the HTTP and JMX end-points to manage and monitor the springboot application

There are three main features of springboot

**i.) Endpoint:** The endpoints allows us to monitor and interact with the application. Springboot provides a number of built-in endpoints but most of the applications choose HTTP endpoints.

**ii.) Metrics:** Springboot actuator provides dimensional metrics by integrating with the micrometer. The micrometer is interacted with springboot

**iii.) Audit:** Springboot provides a flexible audit framework that publish events to an audit event repository. It automatically publishes the authentication events if spring-security in execution.

**Q24) What is JNDI registry**

1. To provide global visibility and accessibility to any java class object we need to place that object reference in JNDI registry software. These references can be accessed from local JVM and remote JVM. JNDI (Java naming and directory interface)

Ex: RMI registry, COS registry, weblogic registry, glassfish registry

**Q25) What is dependency injection**

**A)** Dependency injection is a design pattern that allows us to remove the hard-coded dependencies and make our application loosely coupled, extendable and maintainable.

We can implement dependency injection in java to move the dependency resolution from compile time to run time.

The underlying server/container/jvm dynamically assigns/injects dependent class object to target class object.

Ex-1: The JVM assign/inject default values/initial values to object by executing constructor when the object is created.

Ex-2: The servlet container dynamically assigns/inject servlet config object to our class servlet object.

**Q26) How to implement polymorphism in spring framework**

**A)** We can achieve polymorphism by using the @Qualifier annotation. Inside the @Qualifier (“ ”) we should pass the name of the Bean to be inject.

**Q27) What is JPA specification executor**

**A)** To execute specifications, we need to extends the JPA specification executor interface in our spring data JPA repository.

Public Interface productorepository extends JpaRepository <product, String>, JpaSpecificationExecutor<Product>

{

}

The Jpa Specification Executor interface adds methods which will allow us to execute specifications. For example

List <T> findAll (Specification<T> spec);

Page<T> findAll (Specification<T> spec, Pageable pag);

List<T> findAll (Specification<T> spec, Sort sort );

Finally to execute our query we can call

List<Product> product=productRepository.findall(namelike(“reflectoring));

**Q28) What is the @EnableAutoConfiguration, @ComponentScan and @Configuration**

**A)**

**i.) @EnableAutoConfiguration:** *@EnableAutoConfiguration*annotation enables Spring Boot to auto-configure the application context. So, it automatically creates and registers beans based on both the included jar files in the classpath and the beans defined by us.

Example: when we define the [*spring-boot-starter-web*](https://www.baeldung.com/spring-boot-starters) dependency in our classpath, Spring boot auto-configures [Tomcat](https://www.baeldung.com/tomcat) and [Spring MVC](https://www.baeldung.com/spring-mvc-tutorial).

**ii.) @ComponentScan:** While developing an application, we need to tell the Spring framework to look for Spring-managed components. @ComponentScan enables Spring to scan for things like configurations, controllers, services, and other components we define.

@ComponentScan annotation is used with @Configuration annotation to specify the package for Spring to scan for components:

**iii.) @Configuration:** @Configuration annotation which indicates that the class has the one or more @Bean definition methods. So Spring container can process the class and generate Spring Beans to be used in the application. This annotation is part of the spring core framework.

**Q29) What is @Repository class**

**A)** @Repository is a Spring annotation that indicates that the decorated class is a repository. A repository is a mechanism for storage, retrieval, update, delete and search operation on objects. It is a specialization of the @Component annotation allowing for implementation classes to be autodetected through classpath scanning.

**Q30) What is the REST Resource**

**A)** REST architecture treats every content as a resource. These resources can be text files, HTML pages, images, videos and dynamic business data.

Rest server simply provides access to resource and Rest client access and modifies the resources.

Rest uses various representations to represent a resource where Text, JSON, XML. The most popular representations of resource are XML and JSON.