https://www.google.com/search?q=difference+between+controller+and+restcontroller&rlz=1C1GGRV\_enIN1009IN1009&oq=difference+between+controller+&aqs=chrome.0.0i512j69i57j0i512l8.8861j0j7&sourceid=chrome&ie=UTF-8

**Spring**

* **What is the difference between Dependency Injection and IoC?**

<https://www.geeksforgeeks.org/spring-difference-between-inversion-of-control-and-dependency-injection/#:~:text=Dependency%20Injection%20is%20the%20method,subtypes%20of%20the%20IOC%20principle>.

* **What is difference between @service and @component?**

@Component is a generic stereotype for any Spring-managed component. **@Service annotates classes at the service layer.** **@Repository annotates classes at the persistence layer, which will act as a database repository**.

<https://www.baeldung.com/spring-component-repository-service>

* **Why do we use @autowired annotation?**

The @Autowired annotation **provides more fine-grained control over where and how autowiring should be accomplished**. The @Autowired annotation can be used to autowire bean on the setter method just like @Required annotation, constructor, a property or methods with arbitrary names and/or multiple arguments.

Starting with Spring 2.5, the framework introduced annotations-driven Dependency Injection. The main annotation of this feature is @Autowired. It **allows Spring to resolve and inject collaborating beans into our bean**.

<https://www.tutorialspoint.com/spring/spring_autowired_annotation.htm>

https://www.baeldung.com/spring-autowire

* **What is @RequestMapping in Spring?**

One of the most important annotations in spring is the @RequestMapping Annotation which is **used to map HTTP requests to handler methods of MVC and REST controllers**. In Spring MVC applications, the DispatcherServlet (Front Controller) is responsible for routing incoming HTTP requests to handler methods of controllers.

<https://www.geeksforgeeks.org/spring-requestmapping-annotation-with-example/>

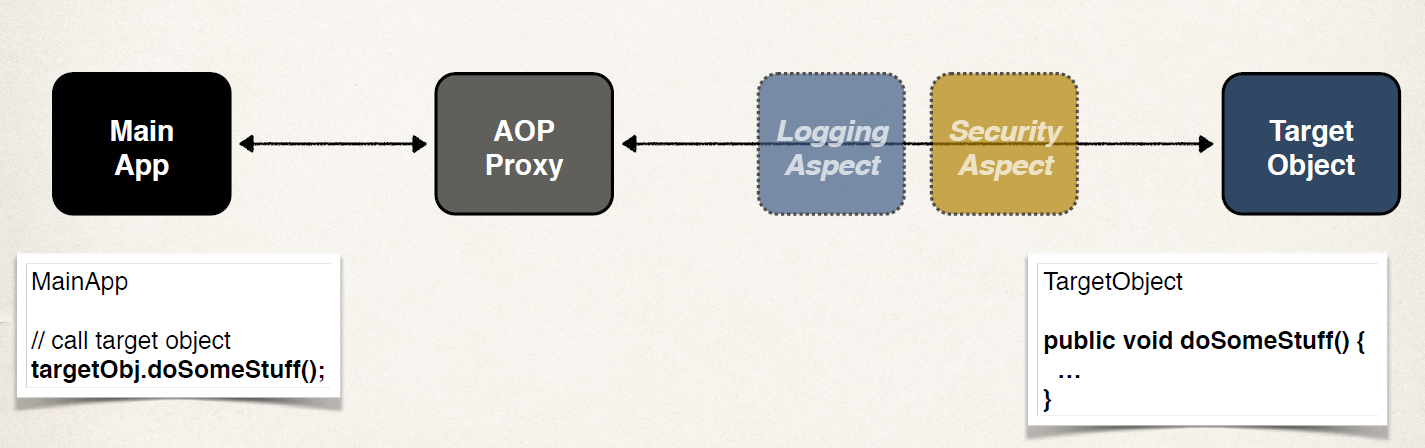
* **What is the use of @Bean annotation? Why do we need it?**
* **Why should you use constructor based injection and avoid @autowired on private fields?**
* **Lifecycle management of objects?**
* **bean scopes - singleton, prototype, resquest, session, global session**
* **How do you resolve ambiguity in injecting types?**
* **Is DI a compile time activity or run time activity?**
* **What is the use of @transactional?**
* **Difference between Dependency Injection and Factory Pattern.**

**Spring Batch**

**Spring AOP**

* **What is AOP?**
  + Programming technique based on concept of an Aspect
  + Aspect encapsulates cross-cutting logic
  + “Concern” means logic / functionality
* **Which pattern AOP uses to solve problems?**

It applies the Proxy design pattern.

****

* **What are the benefits of AOP?**

Code for Aspect is defined in a single class

* + Much better than being scattered everywhere
  + Promotes code reuse and easier to change

Business code in your application is cleaner

* + Only applies to business functionality: addAccount
  + Reduces code complexity

Configurable

* + Based on configuration, apply Aspects selectively to different parts of app
  + No need to make changes to main application code … very important!
* **What are the advantages and disadvantages of AOP?**

Advantages:

* 1. Reusable modules
  2. Resolve code tangling
  3. Resolve code scatter
  4. Applied selectively based on configuration

Disadvantages:

1. Too many aspects and app flow is hard to follow
2. Minor performance cost for aspect execution and run time coding.

* **What are the different terminologies in AOP?**

Aspect: module of code for a cross-cutting concern (logging, security, …)

Advice: What action is taken and when it should be applied

Join Point: When to apply code during program execution

Pointcut: A predicate expression for where advice should be applied

* **Give some scenarios where you will use AOP.**

Most common

* + logging, security, transactions

Audit logging

* + who, what, when, where

Exception handling

* + log exception and notify DevOps team via SMS/email

API Management

* + how many times has a method been called user
  + analytics: what are peak times? what is average load? who is top user?
* **What are the different types of advices?**

Before advice: run before the method

After finally advice: run after the method (finally)

After returning advice: run after the method (success execution)

After throwing advice: run after method (if exception thrown)

Around advice: run before and after method

* **What is weaving?**

Connecting aspects to target objects to create an advised object

Different types of weaving

* + Compile-time, load-time or run-time

Regarding performance: run-time weaving is the slowest

* **Can you give some examples where spring framework has used AOP?**

Security, transactions (@Transactional), caching etc

**Spring Security**

* **Security in REST API: Authentication & Authorization - simple/Basic, jwt, OAuth, OAuth2, 2 factor authentications,**
* **Refresh token**
* **How to configure security in spring boot application?**

**Spring REST**

* **What is the use of @RestController?**

Spring RestController annotation is used **to create RESTful web services using Spring MVC**. Spring RestController takes care of mapping request data to the defined request handler method. Once response body is generated from the handler method, it converts it to JSON or XML response.

* **What is the difference between the @Controller annotation and @RestController annotation?**

@Controller is used to mark classes as Spring MVC Controller.

@RestController annotation is a special controller used in RESTful Web services, and it's the combination of @Controller and @ResponseBody annotation. It is a specialized version of @Component annotation.

<https://www.geeksforgeeks.org/difference-between-controller-and-restcontroller-annotation-in-spring/#:~:text=%40RestController-,%40Controller%20is%20used%20to%20mark%20classes%20as%20Spring%20MVC%20Controller,specialized%20version%20of%20%40Component%20annotation>.

* **How and when to design REST API for async call?**
* **How does request get processed when we call REST API in spring?**
* **What is the difference between @RequestParam vs @PathVariable ?** <https://medium.com/1developer/spring-requestparam-vs-queryparam-vs-pathvariable-vs-pathparam-7c5655e541ad#:~:text=Even%20though%20%40PathVariable%20and%20%40RequestParam,data%20found%20in%20query%20parameters>.

<https://stackoverflow.com/questions/26709560/what-is-the-difference-b-w-requestparam-and-queryparam-anotation#:~:text=What%20is%20main%20difference%20between,QueryParam%20in%20Spring%20MVC%20controller%3F&text=They're%20functionally%20the%20same,want%20a%20more%20useful%20answer>.

* **How do you handle exceptions in REST API? Is it global or any custom exceptions?**
* **Design Product API? What are the steps of designing REST API to create and update product?**
* **Create a data model Student and mark detail of a school?**
* **What are filters in REST API?**
* **How do you secure your REST APIs?**
* **Why should you prefer using DTOs instead of returning entity objects?**
* **How a request is processed when a controller receives?**
* **How to handle exception globally?**

[**https://www.baeldung.com/exception-handling-for-rest-with-spring**](https://www.baeldung.com/exception-handling-for-rest-with-spring)

* **What is Swagger? How do you integrate Swagger in spring boot?**

<https://www.baeldung.com/swagger-2-documentation-for-spring-rest-api>

* **What are the return types in REST API?**

<https://stackoverflow.com/questions/29365833/what-are-valid-return-types-of-a-spring-mvc-controller-method>

* **What is the use of returning ResponseEntity over returning custom types from methods?**
* **How do you handle multiple parameters in a GET request?** <https://stackoverflow.com/questions/60421302/spring-boot-handling-multiple-parameters-in-a-get-request>
* **How do you convert data into xml and return as response from REST API?**
* **What is interceptor in REST API?**
* **How a REST API call is processed? Explain the workflow when a request is sent and how it is processed?**
* **How do you handle exception in REST API?**
* **What are DTOs and why they should be used?**
* **How do you handle exceptions - at global level or at REST method level?**
* **How would you return xml from REST API method?**
* **What is Richardson Maturity Model?**

<https://www.javatpoint.com/restful-web-services-richardson-maturity-model>

* **What are the RESTful Web Services best practices?**

<https://www.javatpoint.com/restful-web-services-best-practice>

**Spring Data**

**Spring Boot**

**What is the difference between @configuration and @EnableAutoConfiguration in Spring boot?**

**@EnableAutoConfiguration**: enable Spring Boot's auto-configuration mechanism. @ComponentScan: enable @Component scan on the package where the application is located (see the best practices)

@Configuration: allow to register extra beans in the context or import additional configuration classes.

<https://docs.spring.io/spring-boot/docs/2.0.x/reference/html/using-boot-using-springbootapplication-annotation.html>

**How do you create custom endpoints in spring boot actuator?**