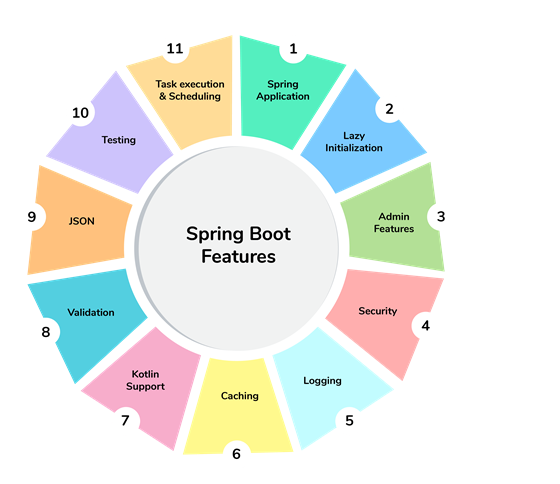
## Spring Boot FAQs (Collected From Different Sources)

What is Spring boot?

Sprint boot is a Java-based spring framework used for Rapid Application Development (to build stand-alone microservices). It has extra support of auto-configuration and embedded application server like tomcat, jetty, etc.

Features of Spring Boot that make it different?

* Creates stand-alone spring application with minimal configuration needed.
* It has embedded tomcat, jetty which makes it just code and run the application.
* Provide production-ready features such as metrics, health checks, and externalized configuration.
* Absolutely no requirement for XML configuration.



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:

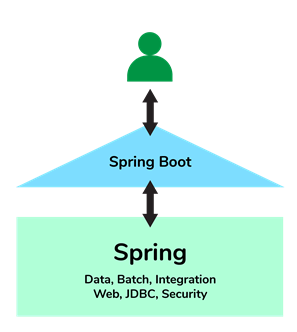
* Spring Boot auto-configuration.
* Spring Boot CLI.
* Spring Boot starter POMs.
* Spring Boot Actuators.

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:



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?

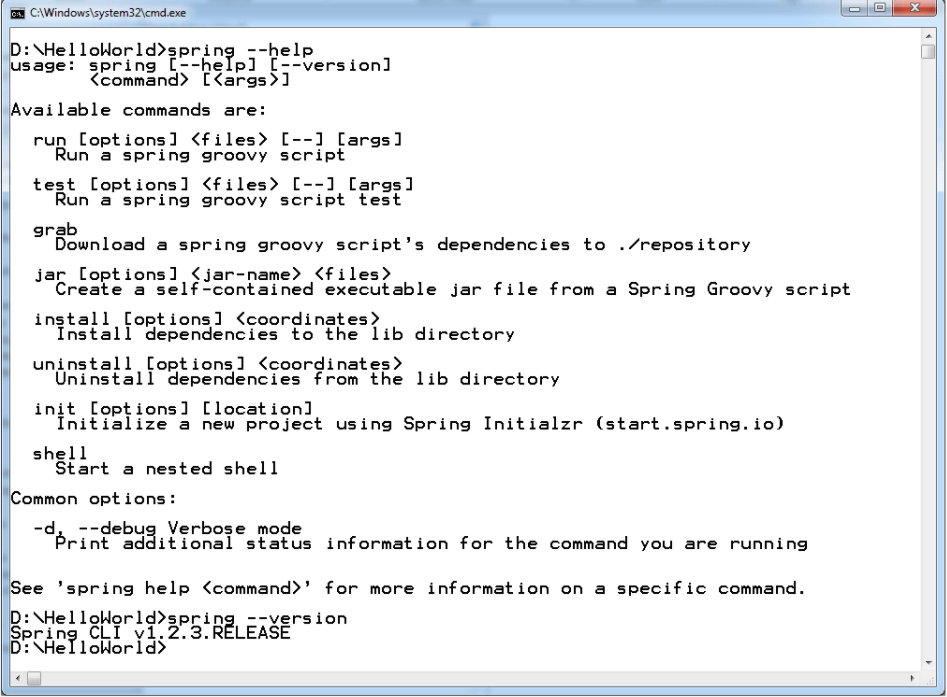
Spring Boot CLI is a command-line interface that allows you to create a spring-based java application using Groovy.

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, -grap, -jar, -war, -install, -uninstall, --init, -shell, -help.

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



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-id 8080. It can be changed by adding sever.port properties in the application.property 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 Sprint boot?

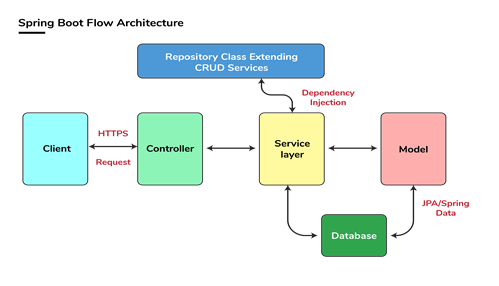
It is a combination of @Controller and @ResponseBody, used for creating a restful controller. It converts the response to JSON or XML. It ensures that data returned by each method will be written straight into the response body instead of returning a template.

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 Boot Flow Architecture

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 Data JPA

1. What is JPA?

Answer: JPA stands for Java Persistence API. It is a Java specification used to persist data between the relational database and Java objects. It acts as a bridge between object-oriented domain models and relational databases. Since interaction with database from Java application is very common, JPA was created to standardize this interaction.

There are many popular JPA implementations available in the Java world like Hibernate. You can further see these Spring Data JPA using Hibernate course to learn more about how to use Hibernate with Spring Data JPA in Java application.

Spring Data JPA Interview Questions with answers

2. What are some advantages of using JPA?

Answer: Here are some advantages of Java Persistence API or JPA:

JPA reduces the burden of interacting with databases.

Annotation in JPA reduces the cost of creating a definition file.

It is user-friendly.

JPA providers help merge applications.

3. What is the Spring data repository?

Answer: Spring data repository is a very important feature of JPA. It helps in reducing a lot of boilerplate code. Moreover, it decreases the chance of errors significantly. This is also the key abstraction that is provided using the Repository interface. It takes the domain class to manage as well as the id type of the domain class as Type Arguments.

4. What is the naming convention for finder methods in the Spring data repository interface?

Answer: This is another key feature of Spring Data JPA API which makes writing query method really easy. The finder method should use a special keyword, i.e. "find", followed by the name of the variable. For example, findByLastName().

5. Why is an interface not a class?

Answer: Interface is not a class because it does not contain concrete methods. It can contain only abstract methods.

6. Can we perform actual tasks like access, persist, and manage data with JPA?

Answer: No, we can't because JPA is only a Java specification.

7. How can we create a custom repository in Spring data JPA?

Answer: To create a custom repository, we have to extend it to any of the following interfaces:

a) Repository

b) PagingAndSortingRepository

c) CrudRepository

d) JpaRepository

e) QueryByExampleRepository

8. What is PagingAndSortingRepository?

Answer: The PagingAndSortingRepository provides methods that are used to retrieve entities using pagination and sorting. It extends the CrudRepository interface.

9. What is @Query used for? (example)

Answer: Spring Data API provides many ways to define SQL query which can be executed and Query annotations one of them. The @Query is an annotation that is used to execute both JPQL and native SQL queries.

10. Give an example of using @Query annotation with JPQL.

Answer: Here is an example of @Query annotation from Spring Data Application which returns all active orders from the database:

@Query("SELECT order FROM Orders o WHERE o.Disabled= 0")

Collection<User> findAllActiveOrders();

and, here is another example, which returns matching employees from the database

@Query("select e from Employee e where se.name = ?1")

List<Employee> getEmployees(String name);

11. Can you name the different types of entity mapping?

Answer: one-to-one mapping, one-to-many mapping, many-to-one mapping, and many-to-many mapping.

12. Define entity and name the different properties of an entity.

Answer: An entity is a group of states bundled (or associated) together in a single unit. It behaves like an object. It also becomes a major constituent of the object-oriented paradigm.

13. What is PlatformTransactionMangaer?

Answer: PlatformTransactionMangaer is an interface that extends TransactionManager. It is the central interface in Spring's transaction infrastructure.

14. How can we enable Spring Data JPA features?

Answer: To enable Spring data JPA features, first we have to define a configuration class and then, we can use @EnableJpaRepositoties annotation with it. This annotation will enable the features.

15. Differentiate between findById() and getOne().

Answer: The findById() is available in CrudRepository while getOne() is available in JpaRepository. The findById() returns null if record does not exist while the getOne() will throw an exception called EntityNotFoundException.

Spring ReST Interview Questions

1. What does REST stand for?

REST stands for REpresentational State Transfer, which uses HTTP protocol to send data from client to server e.g. a book in the server can be delivered to the client using JSON or XML.

2. What is a resource?

A resource is how data is represented in REST architecture. By exposing entities as the resource it allows a client to read, write, modify, and create resources using HTTP methods like GET, POST, PUT, DELETE, etc.

3. What are safe REST operations?

REST API uses HTTP methods to perform operations. Some of the HTTP operations which don't modify the resource at the server are known as safe operations e.g. GET and HEAD. On the other hand, PUT, POST, and DELETE are unsafe because they modify the resource on the server.

4. What are idempotent operations? Why is idempotency important?

There are some HTTP methods e.g. GET which produce same response no matter how many times you use them e.g. sending multiple GET request to the same URI will result in same response without any side-effect hence it is known as idempotent.

On the other hand, the POST is not idempotent because if you send multiple POST requests, it will result in multiple resource creation on the server, but again, PUT is idempotent if you are using it to update the resource.

Even, multiple PUT requests to update a resource on a server will give the same end result. You can further take the HTTP Fundamentals course by Pluralsight to learn more about idempotent methods of the HTTP protocol and HTTP in general.

5. Is REST scalable and/or interoperable?

Yes, REST is Scalable and interoperable. It doesn't mandate a specific choice of technology either at the client or server end. You can use Java, C++, Python or JavaScript to create RESTful Web Services and Consume them at the client end. I suggest you read a good book on REST API e.g. RESTful Web Services to learn more about REST.

6. What are the advantages of the RestTemplate?

The RestTemplate class is an implementation of the Template method pattern in the Spring framework. Similar to other popular template classes like JdbcTemplate or JmsTempalte, it also simplifies the interaction with RESTful Web Services on the client-side. You can use it to consume a RESTful Web Servicer very easily as shown in this example.

7. Which HTTP methods does REST use?

REST can use any HTTP methods but the most popular ones are GET for retrieving a resource, POST for creating a resource, PUt for updating the resource and DELETE for removing a resource from the server.

8. What is an HttpMessageConverter in Spring REST?

An HttpMessageConverter is a Strategy interface that specifies a converter that can convert from and to HTTP requests and responses. Spring REST uses this interface to convert HTTP response to various formats e.g. JSON or XML.

Each HttpMessageConverter implementation has one or several MIME Types associated with it. Spring uses the "Accept" header to determine the content type the client is expecting.

It will then try to find a registered HTTPMessageConverter that is capable of handling that specific content-type and use it to convert the response into that format before sending it to the client.

9. How to create a custom implementation of HttpMessageConverter to support a new type of request/response?

You just need to create an implementation of AbstractHttpMessageConverter and register it using the WebMvcConfigurerAdapter#extendMessageConverters() method with the classes which generate a new type of request/response.

10. Is REST normally stateless?

Yes, REST API should be stateless because it is based on HTTP which is also stateless. A Request in REST API should contain all the details required it to process i.e. it should not rely on previous or next request or some data maintained at the server end e.g. Sessions. REST specification puts a constraint to make it stateless and you should keep that in mind while designing your REST API.

11. What does @RequestMapping annotation do?

The @RequestMapping annotation is used to map web requests to Spring Controller methods. You can map requests based upon HTTP methods like the GET and POST and various other parameters. For examples, if you are developing RESTful Web Service using Spring then you can use produces and consumes property along with media type annotation to indicate that this method is only used to produce or consumers JSON as shown below:

@RequestMapping (method = RequestMethod.POST, consumes="application/json")

public Book save(@RequestBody Book aBook) {

return bookRepository.save(aBook);

}

You can similarly create other handler methods to produce JSON or XML. If you are not familiar with these annotations then I suggest .

12. Is @Controller a stereotype? Is @RestController a stereotype?

Yes, both @Controller and @RestController are stereotypes. The @Controller is actually a specialization of Spring's @Component stereotype annotation. This means that class annotated with @Controller will also be automatically be detected by Spring container as part of the container's component scanning process.

And, @RestController is a specialization of @Controller for RESTful web service. It not only combines @ResponseBody and @Controller annotation but also gives more meaning to your controller class to clearly indicate that it deals with RESTful requests.

Spring Framework may also use this annotation to provide some more useful features related to REST API development in the future.

13. What is the difference between @Controller and @RestController?

There are many differences between @Controller and @RestController as discussed in my earlier article (see the answer) but the most important one is that with @RestController you get the @ResponseBody annotation automatically, which means you don't need to separately annotate your handler methods with @ResponseBody annotation. This makes the development of RESTful web service easier using Spring. You can see here learn more about the difference between @RestController and @Controller annotations in Spring API.

14. When do you need @ResponseBody annotation in Spring MVC?

The @ResponseBody annotation can be put on a method to indicates that the return type should be written directly to the HTTP response body (and not placed in a Model, or interpreted as a view name).

For example:

@RequestMapping(path = "/hello", method = RequestMethod.PUT)

@ResponseBody

public String helloWorld() {

return "Hello World";

}

Alternatively, you can also use @RestController annotation instead of @Controller annotation. This will remove the need for using @ResponseBody because as discussed in the previous answer, it comes automatically with @RestController annotation.

15. What does @PathVariable do in Spring MVC? Why it's useful in REST with Spring?

It's one of the useful annotations from Spring MVC which allows you to read values from URI like query parameter. It's particularly useful in case of creating RESTful web service using Spring because in REST resource identifiers are part of URI. This question is normally asked experienced Spring MVC developers e.g. 4 to 6 years of experience.

For example, in the URL http://myapp.com/books/101 if you want to extract 101 the id, then you can use @PathVariable annotation of Spring MVC. If you are not familiar with Spring MVC annotations then Spring MVC For Beginners: Build Java Web App in 25 Steps is a good place to start with.

16. What is the HTTP status return code for a successful DELETE statement?

There is no strict rule with respect to what status code your REST API should return after a successful DELETE i.e it can return 200 Ok or 204 No Content. In general, if the DELETE operation is successful and the response body is empty return 204. If the DELETE request is successful and the response body is NOT empty, return 200

17. What does CRUD mean?

CRUD is a short form of Create, Read, Update and Delete. In REST API, the POST is used to create a resource, GET is used to read a resource, PUT is used to updated a resource and DELETE is used to remove a resource from the server. This one is another beginner level Spring MVC questions for 1 to 3 years experienced programmers

18. Where do you need @EnableWebMVC?

The @EnableWebMvc annotation is required to enable Spring MVC when Java configuration is used to configure Spring MVC instead of XML. It is equivalent to <mvc: annotation-driven> in XML configuration.

It enables support for @Controller-annotated classes that use @RequestMapping to map incoming requests to handler methods not already familiar with Spring's support for Java configuration,

19. When do you need @ResponseStatus annotation in Spring MVC?

A good questions for 3 to 5 years experienced spring developers. The @ResponseStatus annotation is required during error handling in Spring MVC and REST. Normally when an error or exception is thrown at the server-side, web server returns a blanket HTTP status code 500 - Internal server error.

This may work for a human user but not for REST clients. You need to send them a proper status code like 404 if the resource is not found. That's where you can use @ResponseStatus annotation, which allows you to send custom HTTP status code along with proper error message in case of Exception.

In order to use it, you can create custom exceptions and annotated them using @ResponseStatus annotation and proper HTTP status code and reason.

When such exceptions are thrown from controller's handler methods and not handled anywhere else, then appropriate HTTP response with the proper HTTP status code, which you have set is sent to the client.

For example, if you are writing a RESTful Web Service for a library which provides book information then you can use @ResponseStatus to create Exception which returns HTTP response code 404 when a book is not found instead of Internal Server Error (500), as shown below:

@ResponseStatus(value=HttpStatus.NOT\_FOUND, reason="No such Book") // 404

public class BookNotFoundException extends RuntimeException {

// ...

}

If this Exception is thrown from any handler method then HTTP error code 404 with reason "No such Book" will be returned to the client.

20. Is REST secure? What can you do to secure it?

This question is mostly asked with experienced Java programmers e.g. 2 to 5 years experience with both REST and Spring. Security is a broad term, it could mean security of message which is provided by encryption or access restriction which is provided using authentication and authorization. REST is normally not secure but you can secure it by using Spring security.

At the very least you can enable HTTP basic authentication by using HTTP in your Spring security configuration file. Similarly, you can expose your REST API using HTTPS if the underlying server supports HTTPS.

21. Does REST work with transport layer security (TLS)?

TLS or Transport Layer Security is used for secure communication between client and server. It is the successor of SSL (Secure Socket Layer). Since HTTPS can work with both SSL and TLS, REST can also work with TLS.

Actually, REST says anything about Security, it's up to the server which implements that. The same RESTful Web Service can be accessed using HTTP and HTTPS if the server supports SSL.

If you are using Tomcat, you can see here to learn more about how to enable SSL in Tomcat.

22. Do you need Spring MVC in your classpath for developing RESTful Web Service?

This question is often asked Java programmers with 1 to 2 years of experience in Spring. The short answer is Yes, you need Spring MVC in your Java application's classpath to develop RESTful web services using the Spring framework. It's actually Spring MVC which provides all useful annotations e.g. @RestController, @ResponseCode, @ResponseBody, @RequestBody, and @PathVariable, hence you must spring-mvc.jar or appropriate Maven entry in your pom.xml