**@SpringBootApplication**

* The ***@SpringBootApplication*** annotation is a convenience annotation in Spring Boot that combines several important annotations like ***@SpringBootConfiguration, @EnableAutoConfiguration, @ComponentScan*** to simplify the configuration of a Spring application.

**Equivalent Annotations**

The @SpringBootApplication annotation is a combination of three annotations:

* **@SpringBootConfiguration:** This is a specialisation of the @Configuration annotation, indicating that the class can be used by the Spring IoC container as a source of bean definitions.
* **@EnableAutoConfiguration:** This tells Spring Boot to start adding beans based on the classpath settings, other beans, and various property settings. It helps configure the application automatically.
* **@ComponentScan:** This enables component scanning, allowing Spring to find and register the components like @Service, @Controller, etc in the specified packages.

The @SpringBootApplication annotation is a powerful feature of Spring Boot, simplifying the setup and configuration of applications.

It reduces the amount of boilerplate code you need to write and provides a streamlined way to get your Spring application up and running quickly.

**@RestController**

* The ***@RestControlle***r annotation in Spring Framework is a specialised version of the @Controller annotation that is used to create RESTful web services.
* Using ***@RestController*** makes it easy to build RESTful APIs in Spring, streamlining the process of handling HTTP requests and responses.
* It provides a clean and concise way to create services that return data without the overhead of creating view templates.

**Example:**

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

@RequestMapping("/api")

public class MyRestController {

@GetMapping("/hello")

public String sayHello() {

return "Hello, World!";

}

@GetMapping("/users")

public List<User> getUsers() {

// Assume User is a predefined class and userService fetches user data

return userService.findAll();

}

}

**Explanation**

* **@RestController:** This annotation indicates that the class is a controller where every method returns a domain object instead of a view.
* **@RequestMapping("/api"):** This maps HTTP requests to “/api” to this controller.
* **@GetMapping("/hello"):** This handles GET requests for “/api/hello”, returning a simple string.
* **@GetMapping("/users"):** This handles GET requests for “/api/users”, returning a list of user objects.

**@GetMapping and @PostMapping:**

In Spring Boot, @GetMapping and @PostMapping are annotations used to handle HTTP GET and POST requests, respectively.

***@GetMapping***

* Used to handle HTTP GET requests, which are used to retrieve data from a server.
* Used for actions like fetching or displaying data.

***@PostMapping***

* Used to handle HTTP POST requests, which are used to submit data to be processed by the server.
* Used for actions like creating or updating resources.

**@PathVariable and @RequestParam:**

In Spring, both @PathVariable and @RequestParam are annotations used to extract values from the URL of an HTTP request.

***@PathVariable:***

* Used to extract values from the URI path.
* When you have a variable part in the URL path, you use @PathVariable to capture that value.
* We use @PathVariable for values embedded in the URL path.

***@RequestParam:***

* Used to extract query parameters from the URL.
* When you want to pass parameters in the query string of the URL, you use @RequestParam.
* We use @RequestParam for values passed as query parameters in the URL.

**@Service**

* The @Service annotation in Spring is used to mark a class as a service provider.
* It indicates that the class contains business logic and is part of the service layer in your application.
* The @Service annotation helps organise your business logic in a clear, manageable way, enabling better structure and maintainability in your Spring applications.

**@RestController vs @Service:**

***@Service***

* This annotation is used to mark a class as a service layer in your application.
* The service layer contains business logic.
* It typically doesn't handle HTTP requests directly.
* It performs the necessary operations like interacting with the database or other services and is often called by controllers.
* @Service is used for business logic.

***@RestController***

* This annotation is used to define a class as a RESTful controller that handles HTTP requests.
* It combines @Controller and @ResponseBody, meaning it not only defines endpoints but also ensures that data returned from these endpoints is directly written to the HTTP response as JSON or XML.
* @RestController is used for handling HTTP requests and responses.

**@Configuration:**

* @Configuration annotation in Spring is used to indicate that a class contains one or more Spring beans.
* It's a way to tell Spring that the class will be responsible for configuring and providing certain beans that other parts of your application might need.
  + It Marks a class as a source of bean definitions.
  + It is used to create and configure beans in a more flexible and reusable way.
  + When you define methods in the class, and each method returns an object (a bean). Spring will manage these beans for you.

**@ConfigurationProperties**

* The @ConfigurationProperties annotation in Spring Boot is used to bind values from external configuration files like application.properties or application.yml into Java objects.
* This allows you to easily access and manage configuration settings in your code.
* @ConfigurationProperties allows you to map configuration properties into Java objects.
* This approach helps organise and centralise your application configuration.