@Component

The above annotation is used to indicate a particular class as a spring component. It identifies the class as a bean or a component.

@Service

When we have a class that performs some business logic, calls external API’s or performs other calculations, then, we annotate the class with @Service. This is a form of @Component which is used for classes that provide services.

@Controller

In an application or a system, the classes created as controllers as annotated with the tag @Controller. It identifies component classes in the class path and also registers the bean definitions for those classes. Component scanning is to be configured for identifying components. It handles request mappings.

@EnableAutoConfiguration

The above annotation is used on the main class of the application. This annotation tells Spring Boot to start adding beans based on class path settings and various property settings and defines a base search package implicitly.

@RequestMapping

The above annotation is used to map requests onto specific handler classes or handler methods. Therefore, it can be used at class level or method level or both.

@RequestBody

The above annotation helps in annotating the arguments for request handler method. The annotation makes sure that a method parameter should be bound to the value of the request body. In order to convert the HTTP request message into object, the annotation uses HttpMessageConveter.

@ResponseBody

It is similar to @RequestBody annotation. It annotates request handler methods. It indicates that the result must be written into the response body in the requireed format. The returned object is converted into a response body by using the HttpMessageConveter.

@ExceptionHandler

The above annotation is used to handle exceptions at controller level. It defines the type of exception it will catch. It can define a list of exceptions also. The annotation is used in method level. When there is any exception that matches with any of the value in exception list then the method annotated with the @ExceptionHandler will get invoked.

@ControllerAdvice

By using @ExceptionHandler we can define a method that will be invoked when an exception occurs in a controller. But this cannot be applied for exceptions across controllers. So inorder to overcome the problem we use @ControllerAdvice which will again define @ExceptionHandler, @InitBinder, and @ModelAttribute methods that apply to all @RequestMapping methods. Therefore, if a @ExceptionHandler annotationis defined on a method in one @ControllerAdvice class, it will be applied to all the controllers.

@RestController

The above annotation is used to make a class as a controller. This is a class level annotation. The methods of the class annotated with @RestController will return an object as the http response which has the format that is understood by the consumers. Therefore a class annotated with @RestController doesn’t need an annotation @ResponseBody.

@RestControllerAdvice

The above annotation is a combination of @ControllerAdvice and @ResponseBody. Similar to @ControllerAdvice, the above annotation is used in combination with @ExceptionHandler.

@ResponseStatus

The above annotation is used to mark methods or exception classes with a corresponding status code and a string data that must be returned along with the status code. When the method gets invoked, the status code is set to the HTTP response which overrides the status information provided by any other method, etc.

@CrossOrigin

Inorder to handle requests from multiple origins i.e., to enable cross-origin we use the annotation @CrossOrigin.

@PathVariable

The URI contains various values which act as parameters. We can specify those paramters using the annotation @PathVariable and annotate the request handler arguments.

@RequestHeader

The above annotation is used to map the controller parameters with the values of the request header. By using this annotation we can get the header details within the controller itself.