SPRING BOOT ANNOTATIONS

Spring Boot works more on annotations to define and process bean definitions. Annotation is a tag that represents the metadata i.e. attached with class, interface, methods or fields to indicate some additional information which can be used by java compiler and JVM.
They are like hints given to the compiler about the program.
They do not affect the normal execution of the program.

1. Spring Core Framework annotations

[@Required, @Autowired, @Configuration, @Bean, @ComponentScan, @Qualifier, @Primary]

@Required: It applies to the bean setter method. It indicates that the annotated bean must be populated at configuration time with the required property, else it throws an exception BeanInitilizationException.
Ex: public class Machine
{
private Integer cost;
@Required
public void setCost(Integer cost)
{ this.cost = cost; }
public Integer getCost()
{ return cost; }
}

@Autowired: . It is used to connect the spring bean (with @Component annotation) on the instance variables, setter methods and constructor. When we use @Autowired annotation, the spring container auto-wires the bean by matching data-type.
Ex:
Class Alien {
// It automatically searches for the laptop object in the container by its type (Laptop.class)
@Autowired
Laptop laptop;
}

@Component
Class Laptop {
}

@Configuration: It is a class-level annotation. The class annotated with

@Configuration used by Spring Containers as a source of bean definitions.

@ComponentScan: It is used when we want to scan all the beans and the package declarations. It is used with the annotation @Configuration. We can also specify the base packages to scan for Spring Components.
Example
```
@ComponentScan(basePackages = "com.javatpoint")
@Configuration
public class ScanComponent
{
// ...
}
```

@Bean: It is a method-level annotation. It is an alternative of XML <bean> tag. It tells the method to produce a bean (object) to be managed by Spring Container.
Note: A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container.

@Qualifier: to autowire a specific bean among same type of beans. It is used at @Autowired annotation
———————————————————————————-
2. Spring Framework Stereotype Annotations
[@Component, @Controller, @Service, @Repository]

@Component: It is a class-level annotation. It is used to mark or register a Java class as a bean. A Java class annotated with @Component is found during the classpath. The Spring Framework pick it up and configure it in the application context as a Spring Bean.

Note:
@Controller, @Service, @Repository are the specialized forms of @Component

@Controller: The @Controller is a class-level annotation. It is a specialization of @Component. It marks a class as a web request handler. It is often used to serve web pages. By default, it returns a string that indicates which route to redirect. It is mostly used with @RequestMapping annotation.

Example

```
@Controller
@RequestMapping("books")
public class BooksController
{
```

```
@RequestMapping(value = "/{name}", method = RequestMethod.GET)
public Employee getBooksByName()
{  return booksTemplate;
}
}
```

@Service: It is also used at class level. It tells the Spring that class contains the business logic.

@Repository: It is a class-level annotation. The repository is a DAOs (Data Access Object) that access the database directly. The repository does all the operations related to the database.

———————————————————————————-

3. Spring Boot Annotations
[@EnableAutoConfiguration, @SpringBootApplication]

@EnableAutoConfiguration: It auto-configures the bean that is present in the classpath and configures it to run the methods. The use of this annotation is reduced in Spring Boot 1.2.0 release because developers provided an alternative of the annotation, i.e. @SpringBootApplication.
@SpringBootApplication: It is a combination of three annotations @EnableAutoConfiguration, @ComponentScan, and @Configuration.

———————————————————————————-

4. Spring MVC and REST Annotations
[@RequestMapping, @GetMapping, @PostMapping, @PutMapping, @DeleteMapping, @PatchMapping, @RequestBody, @ResponseBody, @PathVariable, @RequestParam, @RequestHeader, @RestController, @RequestAttribute]

@RequestMapping: It is used to map the web requests. It has many optional elements like consumes, header, method, name, params, path, produces, and value. We use it with the class as well as the method.
@GetMapping: It maps the HTTP GET requests on the specific handler method. It is used to create a web service endpoint that fetches It is used instead of using: @RequestMapping(method = RequestMethod.GET)
@PostMapping: It maps the HTTP POST requests on the specific handler method. It is used to create a web service endpoint that creates It is used instead of using: @RequestMapping(method = RequestMethod.POST)
@PutMapping: It maps the HTTP PUT requests on the specific handler method. It is used to create a web service endpoint that creates or updates It is used instead of using: @RequestMapping(method = RequestMethod.PUT)
@DeleteMapping: It maps the HTTP DELETE requests on the specific handler method. It is used to create a web service endpoint that deletes a resource. It is used instead of using: @RequestMapping(method = RequestMethod.DELETE)

@PatchMapping: It maps the HTTP PATCH requests on the specific handler method. It is used instead of using: @RequestMapping(method = RequestMethod.PATCH)
@RequestBody: It is used to bind HTTP request with an object in a method parameter. Internally it uses HTTP MessageConverters to convert the body of the request. When we annotate a method parameter with @RequestBody, the Spring framework binds the incoming HTTP request body to that parameter.
@ResponseBody: It binds the method return value to the response body. It tells the Spring Boot Framework to serialize a return an object into JSON and XML format.
@PathVariable: It is used to extract the values from the URI. It is most suitable for the RESTful web service, where the URL contains a path variable. We can define multiple @PathVariable in a method.
@RequestParam: It is used to extract the query parameters form the URL. It is also known as a query parameter. It is most suitable for web applications. It can specify default values if the query parameter is not present in the URL.
@RequestHeader: It is used to get the details about the HTTP request headers. We use this annotation as a method parameter. The optional elements of the annotation are name, required, value, defaultValue. For each detail in the header, we should specify separate annotations. We can use it multiple time in a method
@RestController: It can be considered as a combination
of @Controller and @ResponseBody annotations. The @RestController annotation is itself annotated with the @ResponseBody annotation. It eliminates the need for annotating each method with @ResponseBody.
@RequestAttribute: It binds a method parameter to request attribute. It provides convenient access to the request attributes from a controller method. With the help of @RequestAttribute annotation, we can access objects that are populated on the server-side.