Spring MVC Annotation's

1. @Controller Annotation

This annotation serves as a specialization of @Component, allowing for implementation classes autodetected through classpath scanning.*@Controller annotation* tells the Spring IOC container to treat this class as Spring MVC controller.

For XML based configuration use the

```
<context:component-scan base-package="com.comcast">
```

```
@Controller
public class SpringMVCController {


 //HTTP Mappings
}
```

2. @RestController Annotation

A convenience annotation that is itself annotated with @Controller and @ResponseBody.

```
@RestController
public class FilterExampleController {
 @GetMapping
 public String greeting() {
  return "Hello World";
 }
```

```
 @GetMapping(value = "/greeting")

 public String customGreetings() {

  return "Hello From Custom Greetings";

 }

}
```

3. @RequestMapping

Annotation for mapping web requests methods in the *Spring MVC Controller*.
Both *Spring MVC and Spring WebFlux* support this annotation.*@RequestMapping* annotation provides several options to customize its behavior.

- Consumes – The consumable media types of the mapped request, narrowing the primary mapping. (e.g. `@RequestMapping(consumes = {"application/json", "application/xml"})`).
- method – The HTTP request methods to map (e.g. `method = {RequestMethod.GET,RequestMethod.POST}`).
- header – The headers of the mapped request.
- name – the name of the mapping.
- value – The primary mapping expressed by this annotation
- produces – The producible media types of the mapped request.

Here is an example for the *@RequestMapping*

```
@Controller

public class SpringMVCController {

 @RequestMapping(value = {

  "/greetings",

  "/hello-world"}, method = {RequestMethod.GET,RequestMethod.POST},

   consumes = {"application/json","application/xml"},
```

```
  produces = { "application/json"},headers = {"application/json"
 })
public String hellpWorld() {
 return "Hello";
}
}
```

4. @RequestParam

Annotation which shows that it binds a method parameter to a web request parameter. Request parameters passed by the browser/client as part of the HTTP request, the @RequestParam annotation help to map these parameters easily at the controller level.

```
@GetMapping("/request-mapping-example")
public String requestMappingExample(@RequestParam("code") String code) {
 //
}
```

5. @PathVariable

This annotation shows that a method parameter bound to a URI template variable. We specify the variable as part of the @RequestMapping and bind a method argument with @PathVariable. Let’s take an example where we want to pass productCode as part of the URI and not request parameter.

```
@GetMapping("/products/{id}")
public String getProduct(@PathVariable("id") String id) {
 //
}
```

6. @SessionAttribute

Annotation to bind a method parameter to a session attribute.@SessionAttribute used to pass value across different requests through the session. Rather than using HttpSession object directly, using this annotation can benefit auto type conversion and optional/required check.

```
@GetMapping("/user")
public String sessionexample(@SessionAttribute(name = "userLoginTime")
LocalDateTime startDateTime) {
 //
}
```

7) @Bean: Indicates that a method produces a bean to be managed by the Spring container. This is one of the most used and important spring annotation.

8) @ExceptionHandler

ExceptionHandler is a Spring annotation handle exceptions thrown by request handling. This annotation works at the @Controller level.

```
@GetMapping("/greeting")
String greeting() throws Exception {
 //
}

@ExceptionHandler({
 Exception.class
```

```
})
public handleException() {
 //
}
```

Others:

1. @ComponentScan: Configures component scanning directives for use with @Configuration classes. Here we can specify the base packages to scan for spring components.
2. @Component: Indicates that an annotated class is a “component”. Such classes are considered as candidates for auto-detection when using annotation-based configuration and classpath scanning.
3. @PropertySource: provides a simple declarative mechanism for adding a property source to Spring’s Environment. There is a similar annotation for adding an array of property source files i.e @PropertySources.
4. @Service: Indicates that an annotated class is a “Service”. This annotation serves as a specialization of @Component, allowing for implementation classes to be autodetected through classpath scanning.
5. @Repository: Indicates that an annotated class is a “Repository”. This annotation serves as a specialization of @Component and advisable to use with DAO classes.
6. @Autowired: Spring @Autowired annotation is used for automatic injection of beans. Spring @Qualifier annotation is used in conjunction with Autowired to avoid confusion when we have two of more bean configured for same type.