

[Home](#) [Spring Boot](#) [Spring](#) [Hibernate](#) [C](#) [Java](#) [PHP](#) [HTML](#) [CSS](#) [JavaScript](#) [jQuery](#)



Mini Porcket Printer

MMN Portable Mini Thermal Printer Wirelessly
AilExpress

Spring Boot Annotations

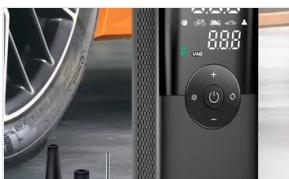
Spring Boot Annotations is a form of metadata that provides data about a program. In other words, annotations are used to provide **supplemental** information about a program. It is not a part of the application that we develop. It does not have a direct effect on the operation of the code they annotate. It does not change the action of the compiled program.

In this section, we are going to discuss some important **Spring Boot Annotation** that we will use later in this tutorial.

Core Spring Framework Annotations

@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 **BeanInitalizationException**.

Example



Fast Inflation Portable Air Pump

CARSUN Rechargeable Air Pump
AilExpress

```
public class Machine
{
    private Integer cost;
    @Required
    public void setCost(Integer cost)
    {
        this.cost = cost;
    }
    public Integer getCost()
```

```
{  
    return cost;  
}  
}
```

@Autowired: Spring provides annotation-based auto-wiring by providing @Autowired annotation. It is used to autowire spring bean on setter methods, instance variable, and constructor. When we use @Autowired annotation, the spring container auto-wires the bean by matching data-type.

Example

```
@Component  
public class Customer  
{  
    private Person person;  
    @Autowired  
    public Customer(Person person)  
    {  
        this.person=person;  
    }  
}
```

@Configuration: It is a class-level annotation. The class annotated with @Configuration used by Spring Containers as a source of bean definitions.

Example

```
@Configuration  
public class Vehicle  
{  
    @BeanVehicle engine()  
    {  
        return new Vehicle();  
    }  
}
```

```
}
```

@ComponentScan: It is used when we want to scan a package for beans. It is used with the annotation @Configuration. We can also specify the base packages to scan for Spring Components.

Example



Bluetooth Gamepads

BROODIO Compatible Nintendo Switch Controller

AilExpress

```
@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 to be managed by Spring Container.

Example

```
@Bean
public BeanExample beanExample()
{
    return new BeanExample ();
}
```

```
}
```

Spring Framework Stereotype Annotations

@Component: It is a class-level annotation. It is used to mark 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**.

Example

```
@Component
public class Student
{
    .....
}
```

@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**.



Retro mechanical bluetooth keyboard

7KEYS Retro Typewriter Keyboard

Amazon

Example

```
package com.javatpoint;
@Service
public class TestService
{
    public void service1()
    {
        //business code
    }
}
```

@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.

```
package com.javatpoint;
@Repository
public class TestRepository
{
    public void delete()
    {
        //persistence code
    }
}
```

```
}
```

Spring Boot Annotations

- **@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**.

Spring MVC and REST Annotations

- **@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.

Example



Lenovo XT80 Bluetooth

Bluetooth 5.3 and stereo sound quality, IPX5 waterproof, Ergonomic earhook design
AilExpress

@Controller

```
public class BooksController
```

```
{
```

```
@RequestMapping("/computer-science/books")
```

```
public String getAllBooks(Model model)
```

```
{
```

```
//application code  
return "bookList";  
}
```

- **@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 from 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.

Note: We have used all the above annotations in our [RESTful Web Services Tutorial](#) with real-world examples.

[← Prev](#)[Next →](#)

Mini Porcket Printer

MMN Portable Mini Thermal Printer Wirelessly
AilExpress



For Videos Join Our Youtube Channel: [Join Now](#)


Feedback


- Send your Feedback to feedback@jvatpoint.com

Help Others, Please Share





Learn Latest Tutorials


 Splunk tutorial
Splunk


 SPSS tutorial
SPSS

 Swagger
tutorial
Swagger


 T-SQL tutorial
Transact-SQL


 Tumblr tutorial
Tumblr


 React tutorial
ReactJS

 Regex tutorial
Regex

 Reinforcement
learning tutorial
Reinforcement
Learning


 R Programming
tutorial
R Programming


 RxJS tutorial
RxJS

 React Native
tutorial
React Native

 Python Design
Patterns
Python Design
Patterns


 Python Pillow
tutorial
Python Pillow


 Python Turtle
tutorial
Python Turtle

 Keras tutorial
Keras

Preparation

 Aptitude
Aptitude

 Logical
Reasoning
Reasoning

 Verbal Ability
Verbal Ability

 Interview
Questions
Interview Questions



Company
Interview
Questions

Company Questions

Trending Technologies



Artificial
Intelligence

Artificial
Intelligence



AWS Tutorial

AWS



Selenium
tutorial

Selenium



Cloud
Computing

Cloud Computing



Hadoop tutorial

Hadoop



ReactJS
Tutorial

ReactJS



Data Science
Tutorial

Data Science



Angular 7
Tutorial

Angular 7



Blockchain
Tutorial

Blockchain



Git Tutorial

Git



Machine
Learning Tutorial

Machine Learning



DevOps
Tutorial

DevOps

B.Tech / MCA



DBMS tutorial

DBMS



Data Structures
tutorial

Data Structures



DAA tutorial

DAA



Operating
System

Operating System



Computer
Network tutorial



Compiler
Design tutorial



Computer
Organization and
Architecture



Discrete
Mathematics
Tutorial

 Computer Network Ethical Hacking Ethical Hacking	 Compiler Design Computer Graphics Tutorial Computer Graphics	 Computer Organization Software Engineering Software Engineering	 Discrete Mathematics Web Technology
 Cyber Security tutorial Cyber Security	 Automata Tutorial Automata	 C Language tutorial C Programming	 C++ tutorial C++
 Java tutorial Java	 .Net Framework tutorial .Net	 Python tutorial Python	 List of Programs Programs
 Control Systems tutorial Control System	 Data Mining Tutorial Data Mining	 Data Warehouse Tutorial Data Warehouse	



Inflation Portable Air Pump

Rechargeable Air Pump

SS