Below is a **clear, concise, and beginner-friendly** overview of **Lombok basic annotations** and **basic Spring Boot annotations**, with **simple definitions** and **common use cases**.

Lombok Basic Annotations

Lombok helps **reduce boilerplate code** (getters, setters, constructors, etc.).

@Getter

- Automatically generates **getter methods** for fields.

```
@Getter
private String name;
```

@Setter

- Automatically generates **setter methods** for fields.

```
@Setter
private int age;
```

@ToString

- Generates a toString() method.

```
@ToString
public class User {}
```

@EqualsAndHashCode

- Generates equals() and hashCode() methods.

```
@EqualsAndHashCode
public class User {}
```

@NoArgsConstructor

- Generates a **no-argument constructor**.

```
@NoArgsConstructor
public class User {}
```

@AllArgsConstructor

- Generates a constructor with **all fields**.

```
@AllArgsConstructor
public class User {}
```

@RequiredArgsConstructor

- Generates a constructor for **final and @NonNull fields**.

```
@RequiredArgsConstructor
public class User {
   private final String name;
}
```

@Data ⭐ (Most Common)

- Combines:
  - @Getter
  - @Setter
  - @ToString
  - @EqualsAndHashCode
  - @RequiredArgsConstructor

```
@Data
public class User {}
```

@Builder

- Implements the **Builder design pattern**.

```
@Builder
public class User {
   private String name;
   private int age;
}
```

@Slf4j

- Creates a logger instance.

```
@Slf4j
public class UserService {
   log.info("Hello Lombok");
}
```

Basic Spring Boot Annotations

Spring Boot annotations define **application behavior**, **dependency injection**, and **web APIs**.

🔷 Core Spring Boot Annotations

@SpringBootApplication ⭐

- Main annotation to start a Spring Boot app.
- Combines:
  - @Configuration
  - @EnableAutoConfiguration
  - @ComponentScan

```
@SpringBootApplication
public class MyApp {}
```

🔷 Stereotype Annotations (Beans)

@Component

- Marks a class as a **Spring-managed bean**.

```
@Component
public class MyService {}
```

@Service

- Used for **business logic layer**.

```
@Service
public class UserService {}
```

@Repository

- Used for **database access layer**.

```
@Repository
public class UserRepository {}
```

@Controller

- Used for **MVC controllers** (returns views).

```
@Controller
public class HomeController {}
```

@RestController ⭐

- Used for **REST APIs** (returns JSON).
- Combines @Controller + @ResponseBody

```
@RestController
public class UserController {}
```

🔷 Dependency Injection

@Autowired

- Injects a bean automatically.

```
@Autowired
private UserService userService;
```

@Qualifier

- Used when **multiple beans** of same type exist.

```
@Qualifier("userService")
```

🔷 REST API Annotations

@RequestMapping

- Maps HTTP requests.

```
@RequestMapping("/users")
```

@GetMapping

```
@GetMapping("/users")
```

@PostMapping

```
@PostMapping("/users")
```

@PutMapping

@PutMapping("/users/{id}")

@DeleteMapping

@DeleteMapping("/users/{id}")

🔷 Request Handling

@PathVariable

```
@GetMapping("/users/{id}")
public User get(@PathVariable int id) {}
```

@RequestParam

```
@GetMapping("/users")
public User get(@RequestParam String name) {}
```

@RequestBody

```
@PostMapping("/users")
public User save(@RequestBody User user) {}
```

🔷 Configuration & Properties

@Configuration

- Defines configuration classes.

```
@Configuration
public class AppConfig {}
```

@Value

- Reads values from application.properties.

```
@Value("${app.name}")
private String appName;
```