Primary and Qualifier

In some cases, there are multiple beans of the same type, and Spring does not know which bean to inject. This is where @Qualifier comes in. It allows you to specify which bean should be injected.

Example with @Qualifier:

```
@Configuration
public class AppConfig {
      @Bean
      public Desktop desktop() {
      return new Desktop();
      @Bean
      public Laptop laptop() {
      return new Laptop();
@Component
class Alien {
      private Computer computer;
     // Use @Qualifier to specify which bean to inject
      @Autowired
      @Qualifier("laptop")
      public Alien(Computer computer) {
      this.computer = computer;
      public void code() {
```



```
System.out.println("Coding...");
    computer.compile(); // Calls compile on the Laptop object
@Component
class Laptop implements Computer {
      @Override
      public void compile() {
    System.out.println("Compiling using Laptop");
@Component
class Desktop implements Computer {
      @Override
      public void compile() {
    System.out.println("Compiling using Desktop");
// Main class to demonstrate @Qualifier
public class App {
      public static void main(String[] args) {
      ApplicationContext context = new
AnnotationConfigApplicationContext(AppConfig.class);
     // Get the Alien bean
      Alien alien = context.getBean(Alien.class);
    alien.code(); // Output: "Coding..." followed by "Compiling using Laptop"
```

Key Points:

- **Qualifier**: Used to resolve ambiguity when multiple beans of the same type exist.
- Example Output: "Compiling using Laptop", as we specified to inject the Laptop bean using @Qualifier("laptop").

Using @Primary for Default Bean Injection

You can also use the @Primary annotation to specify the default bean that should be autowired when multiple candidates are present.

Example with @Primary:

```
@Configuration
public class AppConfig {

    @Bean
    @Primary // This will be the default bean if no @Qualifier is specified
    public Laptop laptop() {
        return new Laptop();
        }

    @Bean
    public Desktop desktop() {
        return new Desktop();
        }
}
```

With this configuration, if no @Qualifier is used, Spring will inject the Laptop bean by default because of the @Primary annotation.



Conclusion:

- 1. **Field-Level Autowiring**: The simplest, but can be harder to test due to private fields.
- 2. **Setter-Based Autowiring**: More flexible and allows optional dependencies or pre-injection logic.
- 3. **Constructor-Based Autowiring**: The most robust and preferred method for mandatory dependencies, enforcing immutability.
- 4. **Autowiring with @Qualifier**: Used when multiple beans of the same type exist, to explicitly specify which one to inject.
- 5. **Autowiring with @Primary**: Used to mark one bean as the default if multiple beans of the same type are present.

