Component Stereotype Annotations

1. What are Stereotype Annotations?

Stereotype annotations in Spring are specialized annotations that define and manage Spring Beans in the application context. These annotations assign specific roles to classes within the application, enabling Spring to detect and register them automatically as beans.

2. Types of Stereotype Annotations

1. **@Component**:
   - General-purpose annotation to mark a class as a Spring-managed component.
   - Automatically registers the class as a bean when component scanning is enabled.

Example:
@Component

```
public class Alien {
    public void code() {
        System.out.println("Alien is coding!");
    }
}
```

2. **@Service**:
   - Specialization of @Component for the service layer.
   - Indicates a class containing business logic.

Example:

```
@Service
public class AlienService {
    public String assist() {
        return "Assistance provided by AlienService.";
    }
}
```

3. **@Repository**:
    - Specialization of @Component for the data access layer.
    - Indicates a class interacting with the database.

Example:

```
@Repository
public class AlienRepository {
    public List<String> fetchAliens() {
        return List.of("Alien1", "Alien2");
    }
}
```

4. **@Controller**:
    - Specialization of @Component for the presentation layer.
    - Handles HTTP requests in a Spring MVC application.

Example:

```
@Controller
public class AlienController {
    @GetMapping("/alien")
    public String greet() {
        return "Greetings from Alien!";
    }
}
```

3. The @Component Annotation

- **Purpose**:
    - Marks a class as a Spring component.
    - Simplifies bean creation and management.

- **Key Features**:
    - Generic stereotype annotation for any Spring-managed component.
    - Automatically detected during component scanning.

Example:

```
@Component
public class Alien {
    public void code() {
        System.out.println("Alien is coding!");
    }
}
```

4. @ComponentScan with @Configuration

1. **What is @ComponentScan?**
    - Specifies which packages Spring should scan for components, services, repositories, and controllers.
    - Detects and registers classes annotated with @Component, @Service, @Repository, and @Controller.
2. **Use of @ComponentScan with @Configuration**:
    - Often used alongside @Configuration to specify the base packages for scanning.

Example:

```
@Configuration
@ComponentScan("com.telusko")
public class AppConfig {
}
```

3. **Explanation**:
    - **@Configuration**:
        - Indicates the class provides Spring configuration.
    - **@ComponentScan("com.telusko")**:

        - Scans the com.telusko package and sub-packages for stereotype annotations.
    - Ensures all annotated classes in the specified package are registered as beans.

5. Advantages of Stereotype Annotations

- **Simplifies Configuration**:
    - No need for explicit bean definitions in XML or Java-based configuration.
- **Improves Readability**:
    - Clarifies the role of a class in the application.
- **Enables Component Scanning**:
    - Automatically detects and registers beans, reducing boilerplate code.

6. Practical Implementation

1. **Define Components**:
    - Create a class annotated with @Component.

```
@Component
public class Alien {
    public void code() {
        System.out.println("Alien is coding!");
    }
}
```

2. **Set Up Configuration**:
    - Use @Configuration and @ComponentScan to enable component scanning.

```
@Configuration
@ComponentScan("com.telusko")
public class AppConfig {
}
```

3. **Access Beans**:
    - Retrieve the bean from the Spring ApplicationContext.

```
public class MainApp {
    public static void main(String[] args) {
        AnnotationConfigApplicationContext context = new
AnnotationConfigApplicationContext(AppConfig.class);
        Alien alien = context.getBean(Alien.class);
        alien.code();
    }
}
```