**SECTION2 :**

**package** com.in28minutes.learn\_spring\_framework;

**import** org.springframework.context.annotation.AnnotationConfigApplicationContext;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** com.in28minutes.learn\_spring\_framework.game.GameRunner;

**import** com.in28minutes.learn\_spring\_framework.game.GamingConsole;

**import** com.in28minutes.learn\_spring\_framework.game.MarioGame;

**import** com.in28minutes.learn\_spring\_framework.game.PacmanGame;

**import** com.in28minutes.learn\_spring\_framework.game.SuperContraGame;

@Configuration

**public** **class** App03GamingSpringBeans {

@Bean

**public** GamingConsole game()

{

**var** game = **new** PacmanGame();

**return** game;

}

@Bean

**public** GameRunner gameRunner()

{

**var** gameRunner = **new** GameRunner(game());

**return** gameRunner;

}

**public** **static** **void** main(String[] args) {

**var** context = **new** AnnotationConfigApplicationContext(App03GamingSpringBeans.**class**);

context.getBean(GamingConsole.**class**).up();

context.getBean(GameRunner.**class**).run();

}

}

In Spring Boot, @Component and @ComponentScan are essential annotations used for component scanning and managing the application's beans. Here's a detailed explanation:

**1. @Component**

* **Definition**: The @Component annotation is a generic stereotype for any Spring-managed component or bean. It is a marker annotation that indicates that a class is a Spring component.
* **Purpose**: When a class is annotated with @Component, Spring's component scanning mechanism automatically detects it and registers it as a bean in the application context.
* **Usage**: Typically, you use @Component to annotate classes that are neither a @Service, @Repository, nor @Controller. It is a generic annotation, while the other three are more specific stereotypes.

**2. @ComponentScan**

* **Definition**: The @ComponentScan annotation is used with configuration classes to specify the packages that Spring should scan for annotated components (like @Component, @Service, @Repository, @Controller, etc.).
* **Purpose**: It instructs Spring to scan the specified packages to find and register beans. This is necessary when your components are not in the default package (the same package as the Spring Boot application class).
* **Usage**: @ComponentScan can be used on configuration classes (classes annotated with @Configuration), or directly on the main application class in a Spring Boot application.

**3. Working Procedure in Spring Boot**

1. **Startup**:
   * When a Spring Boot application starts, it scans the classpath for components to manage as beans. This scanning is usually configured in the main application class (annotated with @SpringBootApplication), which implicitly includes @ComponentScan.
2. **Component Scanning**:
   * Spring Boot will automatically scan the package where the main application class is located and its sub-packages for components, unless specified otherwise using @ComponentScan.
3. **Bean Registration**:
   * As the components are found, Spring registers them as beans in the application context. Each bean is managed by Spring's Inversion of Control (IoC) container, which handles their lifecycle, dependencies, and injection into other components.
4. **Dependency Injection**:
   * If a component needs to use another bean, it can do so via dependency injection. Spring Boot will automatically inject the required beans based on the constructor, field, or method annotations (@Autowired, for example).
5. **Application Context**:
   * Once all components are scanned and registered, Spring Boot's application context is fully initialized, and the application is ready to process incoming requests or perform tasks.

This mechanism allows for modular and scalable application development in Spring Boot, where each component is independently managed and easily reusable.

**EXAMPLE:**

**package** com.in28minutes.learn\_spring\_framework;

**import** org.springframework.context.annotation.AnnotationConfigApplicationContext;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.ComponentScan;

**import** org.springframework.context.annotation.Configuration;

**import** com.in28minutes.learn\_spring\_framework.game.GameRunner;

**import** com.in28minutes.learn\_spring\_framework.game.GamingConsole;

**import** com.in28minutes.learn\_spring\_framework.game.MarioGame;

**import** com.in28minutes.learn\_spring\_framework.game.PacmanGame;

**import** com.in28minutes.learn\_spring\_framework.game.SuperContraGame;

@Configuration

@ComponentScan("com.in28minutes.learn\_spring\_framework.game")

**public** **class** App03GamingSpringBeans {

**public** **static** **void** main(String[] args) {

**var** context = **new** AnnotationConfigApplicationContext(App03GamingSpringBeans.**class**);

context.getBean(GamingConsole.**class**).up();

context.getBean(GameRunner.**class**).run();

}

}

--

**package** com.in28minutes.learn\_spring\_framework.game;

**import** org.springframework.stereotype.Component;

@Component

**public** **class** PacmanGame **implements** GamingConsole {

**public** **void** up()

{

System.***out***.println("PacmanGame JUMP");

}

**public** **void** down()

{

System.***out***.println("PacmanGame Go into hole");

}

**public** **void** left()

{

System.***out***.println(" PacmanGame Go back");

}

**public** **void** right()

{

System.***out***.println("PacmanGame Accerlate");

}

}

--

**package** com.in28minutes.learn\_spring\_framework.game;

**import** org.springframework.stereotype.Component;

@Component

**public** **class** GameRunner {

// MarioGame game ;

GamingConsole game;

**public** GameRunner(GamingConsole game) {

**this**.game = game ;

}

**public** **void** run()

{

System.***out***.println("RUNNING GAME IS " +game);

game.down();

game.up();

game.left();

game.right();

}

}

In Spring, @Primary and @Qualifier are annotations used to resolve bean conflicts when multiple beans of the same type are available in the application context. Here's a detailed explanation:

**1. @Primary**

* **Definition**: The @Primary annotation is used to indicate that a specific bean should be given preference when multiple beans of the same type are available.
* **Purpose**: If there are multiple beans of the same type and Spring needs to autowire one of them without further specification, the one marked with @Primary will be chosen. This avoids the ambiguity of which bean to inject.
* **Usage**: You annotate one of the beans with @Primary to designate it as the default bean to be injected when no other qualifier is specified.

**2. @Qualifier**

* **Definition**: The @Qualifier annotation is used to specify which bean should be injected when multiple beans of the same type are available in the application context.
* **Purpose**: @Qualifier provides more granular control over dependency injection by allowing you to explicitly name the bean that should be injected, even when @Primary is used on another bean.
* **Usage**: You annotate the injection point with @Qualifier followed by the name of the bean you want to inject.

--

**package** com.in28minutes.learn\_spring\_framework.game;

**import** org.springframework.beans.factory.annotation.Qualifier;

**import** org.springframework.stereotype.Component;

@Component

**public** **class** GameRunner {

// MarioGame game ;

GamingConsole game;

**public** GameRunner(@Qualifier("SuperContraGameQualifier")GamingConsole game) {

**this**.game = game ;

}

**public** **void** run()

{

System.***out***.println("RUNNING GAME IS " +game);

game.down();

game.up();

game.left();

game.right();

}

}

--

**package** com.in28minutes.learn\_spring\_framework.game;

**import** org.springframework.context.annotation.Primary;

**import** org.springframework.stereotype.Component;

@Component

@Primary

**public** **class** MarioGame **implements** GamingConsole{

**public** **void** up()

{

System.***out***.println("JUMP");

}

**public** **void** down()

{

System.***out***.println("Go into hole");

}

**public** **void** left()

{

System.***out***.println("Go back");

}

**public** **void** right()

{

System.***out***.println("Accerlate");

}

}

--

**package** com.in28minutes.learn\_spring\_framework.game;

**import** org.springframework.beans.factory.annotation.Qualifier;

**import** org.springframework.stereotype.Component;

@Component

@Qualifier("SuperContraGameQualifier")

**public** **class** SuperContraGame **implements** GamingConsole {

**public** **void** up()

{

System.***out***.println(" SUPER JUMP");

}

**public** **void** down()

{

System.***out***.println(" SUPER Go into hole");

}

**public** **void** left()

{

System.***out***.println(" SUPER Go back");

}

**public** **void** right()

{

System.***out***.println(" SUPER Accerlate");

}

}

**- \*\*@Primary\*\*:** Use `@Primary` to make one bean the default when there are multiple beans of the same type.

**- \*\*@Qualifier\*\*:** Use `@Qualifier` to choose a specific bean when there are multiple beans of the same type.