@ComponentScan

Before we rely completely on *@Component*, we must understand that it's only a plain annotation. The annotation serves the purpose of differentiating beans from other objects, such as domain objects.

Spring uses the *@ComponentScan* annotation to actually gather them all into its *ApplicationContext*.

If we're writing a Spring Boot application, it is helpful to know that *@SpringBootApplication* is a composed annotation that includes *@ComponentScan*

```
package com.stackroute.scannedscope;

@Component

public class ScannedScopeExample { }

-----------------------------------------------------------------

package com.baeldung.component.inscope;

@SpringBootApplication

@ComponentScan({"com.stackroute.component.inscope", "
com.stackroute.component.scannedscope"})

public class ComponentApplication

{

        //public static void main(String[] args) {...}

}
```

While developing an application, we need to tell the Spring framework to look for Spring-managed components. @ComponentScan enables Spring to scan for things like configurations, controllers, services, and other components we define.

the @ComponentScan annotation is used with @Configuration annotation to specify the package for Spring to scan for components:

```
@Configuration

@ComponentScan

public class EmployeeApplication {
```

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

ApplicationContext context = SpringApplication.run(EmployeeApplication.class, args); // ...

        }

}
```

Spring can also start scanning from the specified package, which we can define using basePackageClasses() or basePackages(). If no package is specified, then it considers the package of the class declaring the @ComponentScan annotation as the starting package

```
@Configuration
@ComponentScan(basePackages = "com.stackroute.*")
@EnableWebMvc
public class WebMVCConfig implements WebMvcConfigurer {

   @Bean
   public InternalResourceViewResolver resolver() {
      InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();
      viewResolver.setViewClass(JstlView.class);
      viewResolver.setSuffix(".jsp");
      viewResolver.setPrefix("/WEB-INF/view/");
      return viewResolver;
   }


}
```

, the Configuration classes can contain @Bean annotations, which register the methods as beans in the Spring application context. After that, the @ComponentScan annotation can auto-detect such beans:

```
@Configuration

public class Hospital {

@Bean

 public Doctor getDoctor() {

return new Doctor();

        }

}
```

@ComponentScan annotation can also scan, detect, and register beans for classes annotated with @Component, @Controller, @Service, and @Repository.