@SpringBootTest

https://stackoverflow.com/questions/67800209/cannot-load-configuration-class-in-component-test-marked-as-springboottest

Yes, by default, `@SpringBootTest` will scan all classes within the `src/main/java` and `src/main/test` directories of your Spring Boot application.

Here's a breakdown of how it works:

- **`src/main/java`**: This directory typically contains your application's main source code, including controllers, services, repositories, configuration classes, etc. `@SpringBootTest` will scan these classes to initialize the Spring ApplicationContext for integration testing.

- **`src/main/test`**: This directory contains your application's test code, including unit tests, integration tests, and test configuration classes. `@SpringBootTest` will also scan these classes to locate any additional configuration or test-specific beans required for testing.

When you annotate a test class with `@SpringBootTest`, Spring Boot automatically configures an ApplicationContext that includes all beans defined in the main application context, as well as any additional beans defined specifically for testing purposes. This allows you to write integration tests that interact with your application in a real-world environment.

Yes, you can customize component scanning behavior in your Spring Boot application, including for integration tests annotated with `@SpringBootTest`. You can configure component scanning explicitly by using the `@ComponentScan` annotation on your test configuration class or directly on your test classes.

Here's how you can do it:

1. **On Test Configuration Class**: If you have a separate test configuration class, you can use `@ComponentScan` to specify the base packages for component scanning:

```java
import org.springframework.boot.test.context.TestConfiguration;
import org.springframework.context.annotation.ComponentScan;

@TestConfiguration
```

```
   @ComponentScan(basePackages = {"com.example.myapp"})
   public class MyTestConfiguration {
       // Additional test configuration if needed
   }
```

2. **On Test Classes**: You can also specify component scanning directly on your test classes using the same `@ComponentScan` annotation:

```java
   import org.springframework.boot.test.context.SpringBootTest;
   import org.springframework.test.context.junit4.SpringRunner;
   import org.springframework.context.annotation.ComponentScan;

   @RunWith(SpringRunner.class)
   @SpringBootTest
   @ComponentScan(basePackages = {"com.example.myapp"})
   public class MyIntegrationTest {
       // Test methods
   }
```

By specifying the base packages for component scanning, you can control which packages Spring Boot will scan for components, configuration, and other Spring beans during test execution. This allows you to isolate and customize the ApplicationContext for your integration tests as needed.