DataSouceTransactionManager

To configure a `DataSourceTransactionManager` in Spring Boot, you typically need to follow these steps:

1. Define a `DataSource` bean in your configuration class. This can be done using Spring Boot's auto-configuration if you have the necessary dependencies in your classpath, or you can manually configure the `DataSource` bean if needed.

2. Define a `DataSourceTransactionManager` bean and inject the `DataSource` bean into it.

3. Optionally, you can customize the transaction manager properties if needed.

Here's an example of how you can configure a `DataSourceTransactionManager` in Spring Boot:

```java
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.jdbc.datasource.DataSourceTransactionManager;
import org.springframework.transaction.PlatformTransactionManager;
import org.springframework.transaction.annotation.EnableTransactionManagement;
import org.springframework.transaction.annotation.TransactionManagementConfigurer;
import javax.sql.DataSource;

@Configuration
@EnableTransactionManagement
public class TransactionConfig implements TransactionManagementConfigurer {

    private final DataSource dataSource;

    public TransactionConfig(DataSource dataSource) {
        this.dataSource = dataSource;
    }

    @Bean
    public PlatformTransactionManager transactionManager() {
        return new DataSourceTransactionManager(dataSource);
    }
    @Override
    public PlatformTransactionManager annotationDrivenTransactionManager() {
        return transactionManager();
    }
}
```

In this configuration class:

- The `@Configuration` annotation marks the class as a configuration class.
- The `@EnableTransactionManagement` annotation enables Spring's annotation-driven transaction management.
- The `DataSource` bean is injected into the `TransactionConfig` class through constructor injection.
- The `transactionManager()` method defines a `DataSourceTransactionManager` bean and sets the `DataSource` as its data source.
- The `annotationDrivenTransactionManager()` method returns the `DataSourceTransactionManager` bean to provide transaction management support for annotated classes.

Make sure to adjust the configuration according to your specific `DataSource` setup and transaction management requirements. Additionally, ensure that your `DataSource` configuration is correctly set up in your `application.properties` or `application.yml` file if you're using Spring Boot's auto-configuration features.