Setting DB in JPA using Java Configuration Class

Yes, you can use HikariCP with JPA by configuring the data source using Java configuration classes. The HikariConfig class is typically used to configure the HikariCP connection pool, and you can use it to set up the data source as well.

Here's a basic example of how you can do this:

```java
import javax.sql.DataSource;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.orm.jpa.JpaTransactionManager;
import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;
import org.springframework.transaction.PlatformTransactionManager;
import org.springframework.transaction.annotation.EnableTransactionManagement;
import org.springframework.jdbc.datasource.DriverManagerDataSource;

@Configuration
@EnableTransactionManagement
public class JpaConfig {

    @Bean
    public DataSource dataSource() {
        HikariConfig config = new HikariConfig();
        config.setJdbcUrl("jdbc:mysql://localhost:3306/mydatabase");
        config.setUsername("username");
        config.setPassword("password");

        return new HikariDataSource(config);
    }

    @Bean
    public LocalContainerEntityManagerFactoryBean entityManagerFactory() {
        LocalContainerEntityManagerFactoryBean em = new
LocalContainerEntityManagerFactoryBean();
        em.setDataSource(dataSource());
        em.setPackagesToScan("com.example.entity");
        // Set JPA vendor adapter, properties, etc.

        return em;
    }

    @Bean
    public PlatformTransactionManager transactionManager() {
        JpaTransactionManager transactionManager = new
```

```
JpaTransactionManager();

transactionManager.setEntityManagerFactory(entityManagerFactory().getObjec
t());
    return transactionManager;
  }
}
```

In this example, we define a DataSource bean using HikariConfig to configure the connection pool settings. Then, we use this data source in the LocalContainerEntityManagerFactoryBean to set up the EntityManagerFactory for JPA. Finally, we configure the PlatformTransactionManager for managing transactions.

Make sure to adjust the configuration according to your database setup and requirements.