Profile DB

If you're referring to a separate class specifically dedicated to configuring H2 database properties, you can certainly create one. While it's not strictly necessary, having a separate configuration class can help organize your code, especially if you have multiple data sources or if you want to keep your main configuration class cleaner.

Here's an example of how you can create a separate configuration class for H2 database properties:

```java
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Profile;
import org.springframework.jdbc.datasource.DataSourceBuilder;

import javax.sql.DataSource;

@Configuration
@Profile("dev")
public class H2DatabaseConfig {

    @Value("${spring.datasource.url}")
    private String dataSourceUrl;

    @Value("${spring.datasource.username}")
    private String dataSourceUsername;

    @Value("${spring.datasource.password}")
    private String dataSourcePassword;

    @Bean
    public DataSource dataSource() {
        return DataSourceBuilder
                .create()
                .url(dataSourceUrl)
                .username(dataSourceUsername)
                .password(dataSourcePassword)
                .driverClassName("org.h2.Driver")
                .build();
    }
}
```

In this class:

- We have a `H2DatabaseConfig` class annotated with `@Configuration` and `@Profile("dev")`, indicating that this configuration applies only when the "dev" profile is active.
- We inject datasource-related properties (`dataSourceUrl`, `dataSourceUsername`, `dataSourcePassword`) using `@Value` annotation.
- We define a `dataSource()` method annotated with `@Bean` to create and configure the DataSource bean using `DataSourceBuilder`.

This class encapsulates the configuration related to the H2 database and keeps it separate from other configurations in your application. You can include additional methods in this class if you need to configure other aspects of the database.

Different Way:

When the development profile is active in a Spring Boot application, using an in-memory database like H2 is a common choice. H2 is lightweight, fast, and ideal for development and testing purposes. Here's how you can configure H2 in a Spring Boot application when the dev profile is active:

1. **Add H2 Dependency**: Ensure that the H2 dependency is included in your `pom.xml` or `build.gradle` file. For Maven, it looks like this:

   ```xml
   <dependency>
       <groupId>com.h2database</groupId>
       <artifactId>h2</artifactId>
       <scope>runtime</scope>
   </dependency>
   ```

2. **Configure Database Properties**: In your `application-dev.properties` or `application-dev.yml` file, configure the database connection properties for H2. For example:

   ```properties
   spring.datasource.url=jdbc:h2:mem:testdb
   spring.datasource.driver-class-name=org.h2.Driver
   spring.datasource.username=sa
   spring.datasource.password=password
   spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
   spring.h2.console.enabled=true
   ```

```

This configuration sets up an in-memory H2 database named "testdb" with a username "sa" and password "password". It also enables the H2 console for accessing the database console in a web browser.

3. **Annotate Entities**: Annotate your JPA entities with `@Entity`, `@Table`, and other necessary annotations as per your requirements.

4. **Create Repositories**: Create Spring Data JPA repositories for your entities to perform database operations.

5. **Development Profile**: Ensure that the development profile is active when running your Spring Boot application. You can set this in the `application.properties` file or via command-line arguments.

   For example, in `application.properties`:

   ```properties
   spring.profiles.active=dev
   ```

6. **Accessing H2 Console**: Run your Spring Boot application, and you can access the H2 console by navigating to `http://localhost:8080/h2-console` (assuming your application is running on port 8080). Use the JDBC URL (`jdbc:h2:mem:testdb` in this case), username, and password configured earlier to connect to the H2 console.

Using H2 in the development profile allows you to quickly iterate and test your application without the need for setting up a separate database server. However, remember that H2 is an in-memory database, so any data stored in it will be lost when the application is stopped.
   ```