Hibernate Logging

Overview

In this slide, we will explore:

- How to configure logging when interacting with a database.
- The role of Simple Logging Facade for Java
- The Log4j logging framework.
- Setting up and configuring logging in Hibernate.

Understanding Hibernate Logging

Hibernate generates extensive log output when interacting with the database. However, excessive logging may clutter the console and make debugging harder.

We can tailor the logging output to display only relevant information.

Simple Logging Facade can help us.

Simple Logging Facade for Java (SLF4J)

SLF4J is a **logging abstraction** that allows developers to use different logging frameworks (e.g., Log4j, Logback, java.util.logging).

It provides flexibility by allowing the logging framework to be chosen at runtime.

In this setup, we use SLF4J as the abstraction and Log4j as the underlying implementation.

log4j

log4j is a reliable, fast and flexible logging framework (APIs) written in Java, which is distributed under the Apache Software License.

Unlike Log4j, SLF4J is not a logging framework itself; instead, it provides an API to work with multiple logging frameworks, including Log4j.

Setting Up Logging in Hibernate

To enable logging, we need to include the necessary dependencies and configuration files.

slf4j.jar

log4j-1.2.17.jar

slf4j-log4j12-1.7.0.jar

and the following file in the resources:

log4j.properties

Required Dependencies (POM.xml)

Log 4j:

```
<!-- https://mvnrepository.com/artifact/log4j/log4j -->
<dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.17</version>
</dependency>
```

Required Dependencies (POM.xml)

slf4j-log4j12:

```
<!-- https://mvnrepository.com/artifact/org.slf4j/slf4j-log4j12 -->
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-log4j12</artifactId>
    <version>1.7.0</version>
    <scope>test</scope>
</dependency>
```

Configuring Log4j (log4j.properties)

Create a log4j.properties file inside the resources directory with the following configuration:

Set Hibernate logging level to WARN (hides DEBUG and INFO logs)

log4j.logger.org.hibernate=warn

 # Log SQL queries at INFO level

log4j.logger.org.hibernate.type=info

Configuring Log4j (log4j.properties)

We can choose *info* or *debug* instead of *warn*. However, 'warn' provides a cleaner log output compared to 'info' and 'debug'."

SQL Queries (DEBUG): Logs all SQL queries executed by Hibernate.

SQL Parameters (TRACE): Logs actual parameter values in prepared statements (instead of ?,?,?,?).

Additional Hibernate Logging Configuration

Modify the hibernate.cfg.xml file to fine-tune logging settings:

<!-- Enable SQL query logging -->

<property name="show_sql">true</property>

<!-- Format SQL output for better readability -->

<property name="format_sql">true</property>

<!-- Include SQL comments for better understanding of queries -->

<property name="use_sql_comments">true</property>

Additional Hibernate Logging Configuration

Explanation:

show_sql=true logs all SQL queries.

format_sql=true improves readability by splitting SQL into multiple lines.

use_sql_comments=true adds comments to SQL queries to help debugging.

Summary

SLF4J is a logging abstraction that allows using different logging frameworks.

Log4j is one of the most commonly used frameworks for logging.

Hibernate generates a lot of logs; we can control the verbosity using log4j.properties.

hibernate.cfg.xml provides additional SQL logging configurations.

By tailoring logging settings, we can reduce unnecessary log clutter and focus on essential debugging information.