Hibernate Log4j Logging

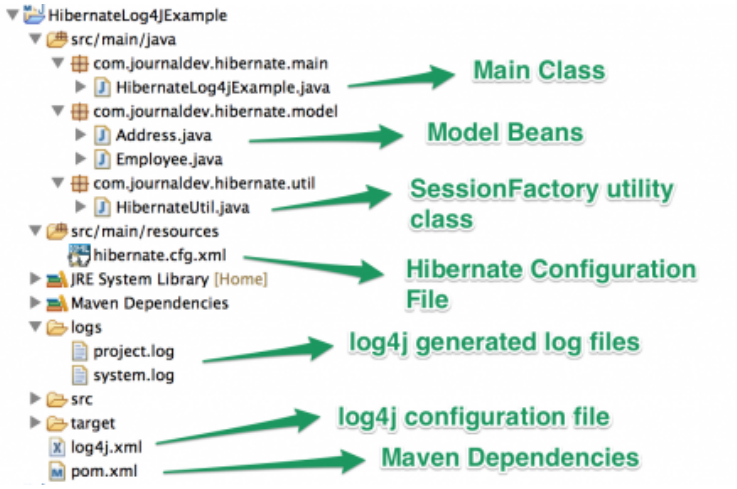
APRIL 2, 2018 BY PANKAJ — 3 COMMENTS

Welcome to Hibernate Log4j Logging example. Hibernate 4 uses JBoss logging rather than slf4j in older hibernate versions. Today we will look into how to configure log4j in a hibernate application.

Table of Contents [hide]

Hibernate Log4j Logging

Create a maven project in the Eclipse or your favorite IDE, final project structure will look like below image.



Let's look into each of the components one by one.

Hibernate Log4j Maven Dependencies

We need hibernate-core, mysql driver and log4j dependencies in our project, our final pom.xml file looks like below.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.journaldev.hibernate</groupId>
  <artifactId>HibernateLog4JExample</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>
        <dependency>
                <groupId>org.hibernate</groupId>
                <artifactId>hibernate-core</artifactId>
                <version>4.3.5.Final</version>
        </dependency>
        <dependency>
                <groupId>log4j</groupId>
                <artifactId>log4j</artifactId>
                <version>1.2.16</version>
        </dependency>
        <dependency>
                <groupId>mysql</groupId>
                <artifactId>mysql-connector-java</artifactId>
```

Log4j Configuration File

I am using XML based log4j configuration, we can also use property file based configuration.

log4j.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">

<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/"
        debug="false">

        <!-- A time/date based rolling appender -->
        <appender name="FILE" class="org.apache.log4j.RollingFileAppender">
                <param name="File" value="logs/system.log" />
                <param name="Append" value="true" />
                <param name="ImmediateFlush" value="true" />
                <param name="MaxFileSize" value="200MB" />
                <param name="MaxBackupIndex" value="100" />

                <layout class="org.apache.log4j.PatternLayout">
                        <param name="ConversionPattern" value="%d %d{Z} [%t] %-5p
(%F:%L) - %m%n" />
                </layout>
        </appender>

        <appender name="journaldev-hibernate"
class="org.apache.log4j.RollingFileAppender">
```

This file needs to be placed at the root folder, so that our main class can access it. Notice the location of log4j log files, our application generated logs will go into project.log whereas hibernate logs will go into system.log file.

Hibernate Configuration File

Our hibernate configuration xml looks like below.

hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration SYSTEM "classpath://org/hibernate/hibernate-
configuration-3.0.dtd">
<hibernate-configuration>
        <session-factory>
                <property
```

```
name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
                <property name="hibernate.connection.password">pankaj123</property>
                <property
name="hibernate.connection.url">jdbc:mysql://localhost/TestDB</property>
                <property name="hibernate.connection.username">pankaj</property>
                <property
name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

                <property
name="hibernate.current_session_context_class">thread</property>
                <property name="hibernate.show_sql">true</property>

                <mapping class="com.journaldev.hibernate.model.Employee" />
                <mapping class="com.journaldev.hibernate.model.Address" />
        </session-factory>
</hibernate-configuration>
```

Hibernate SessionFactory utility class

```
package com.journaldev.hibernate.util;

import org.apache.log4j.Logger;
import org.hibernate.SessionFactory;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;
import org.hibernate.service.ServiceRegistry;

public class HibernateUtil {

        private static SessionFactory sessionFactory;

        private static Logger logger = Logger.getLogger(HibernateUtil.class);

        private static SessionFactory buildSessionFactory() {
        try {
            // Create the SessionFactory from hibernate.cfg.xml
                Configuration configuration = new Configuration();
                configuration.configure("hibernate.cfg.xml");
                logger.info("Hibernate Configuration loaded");

                ServiceRegistry serviceRegistry = new
```

Notice that I am using log4j Logger for logging purpose, since this class is part of our application, the logs should go into project.log file.

Model Classes

We have two model classes – Employee and Address. I am using the same database setup as in HQL Example, so you can check that tutorial to look into database setup and model classes. Both of these classes are simple java beans with hibernate annotation based mapping.

Hibernate Log4j Logging Test Class

Our main class looks like below.

```
package com.journaldev.hibernate.main;

import java.util.List;

import org.apache.log4j.Logger;
import org.apache.log4j.xml.DOMConfigurator;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;

import com.journaldev.hibernate.model.Employee;
import com.journaldev.hibernate.util.HibernateUtil;

public class HibernateLog4jExample {

	static{
		System.out.println("Before log4j configuration");
		DOMConfigurator.configure("log4j.xml");
		System.out.println("After log4j configuration");
	}
```

Notice that we don't need to do anything to plugin log4j logging. Obviously we would need to configure it first before using it, that's why I am using static block to make sure log4j is configured before we start using it. I am using `org.apache.log4j.xml.DOMConfigurator` because our log4j configuration is XML based, if you have property file based configuration, you should use `org.apache.log4j.PropertyConfigurator` class.

Now when we execute above program, we get following data in log4j files.

project.log

```
2014-06-05 20:00:11,868 -0700 [main] INFO  (HibernateUtil.java:20) - Hibernate Configuration loaded
2014-06-05 20:00:11,936 -0700 [main] INFO  (HibernateUtil.java:23) - Hibernate serviceRegistry created
2014-06-05 20:00:12,698 -0700 [main] INFO  (HibernateLog4jExample.java:37) - List of Employees::1,San Jose
2014-06-05 20:00:12,698 -0700 [main] INFO  (HibernateLog4jExample.java:37) - List of Employees::2,Santa Clara
2014-06-05 20:00:12,698 -0700 [main] INFO  (HibernateLog4jExample.java:37) - List of Employees::3,Bangalore
2014-06-05 20:00:12,698 -0700 [main] INFO  (HibernateLog4jExample.java:37) - List of Employees::4,New Delhi
2014-06-05 20:00:12,712 -0700 [main] INFO  (HibernateLog4jExample.java:42) - DONE
```

Notice that above log file contains only entries generated by our program, this way we can separate our application logs from hibernate logs.

system.log

```
2014-06-05 22:49:58,415 -0700 [main] INFO  (JavaReflectionManager.java:66) - HCANN000001: Hibernate Commons Annotations {4.0.4.Final}
2014-06-05 22:49:58,423 -0700 [main] INFO  (Version.java:54) - HHH000412: Hibernate Core {4.3.5.Final}
2014-06-05 22:49:58,426 -0700 [main] INFO  (Environment.java:239) - HHH000206: hibernate.properties not found
2014-06-05 22:49:58,427 -0700 [main] INFO  (Environment.java:346) - HHH000021: Bytecode provider name : javassist
2014-06-05 22:49:58,446 -0700 [main] INFO  (Configuration.java:2073) - HHH000043: Configuring from resource: hibernate.cfg.xml
2014-06-05 22:49:58,447 -0700 [main] INFO  (Configuration.java:2092) - HHH000040: Configuration resource: hibernate.cfg.xml
2014-06-05 22:49:58,491 -0700 [main] INFO  (Configuration.java:2214) - HHH000041: Configured SessionFactory: null
2014-06-05 22:49:58,557 -0700 [main] WARN  (DriverManagerConnectionProviderImpl.java:93) - HHH000402: Using Hibernate built-in connection pool (not for production use!)
2014-06-05 22:49:58,560 -0700 [main] INFO  (DriverManagerConnectionProviderImpl.java:166) - HHH000401: using driver [com.mysql.jdbc.Driver] at URL [jdbc:mysql://localhost/TestDB]
2014-06-05 22:49:58,561 -0700 [main] INFO
```

```
(DriverManagerConnectionProviderImpl.java:175) - HHH000046: Connection properties:
```

Since our hibernate logs level is set to INFO mode, there are not many logs. If you will change it to debug, even this simple program generates more than 1400 lines of log. Analyzing these logs are sometime crucible in understanding how hibernate works internally and to debug issues related to hibernate.

Also, we got following logs generated in console.

```
Before log4j configuration
After log4j configuration
Hibernate: select employee0_.emp_id as emp_id1_1_, employee0_.emp_name as
emp_name2_1_, employee0_.emp_salary as emp_sala3_1_ from EMPLOYEE employee0_
Hibernate: select address0_.emp_id as emp_id1_0_0_, address0_.address_line1 as
address_2_0_0_, address0_.city as city3_0_0_, address0_.zipcode as zipcode4_0_0_,
employee1_.emp_id as emp_id1_1_1_, employee1_.emp_name as emp_name2_1_1_,
employee1_.emp_salary as emp_sala3_1_1_ from ADDRESS address0_ left outer join
EMPLOYEE employee1_ on address0_.emp_id=employee1_.emp_id where address0_.emp_id=?
Hibernate: select address0_.emp_id as emp_id1_0_0_, address0_.address_line1 as
address_2_0_0_, address0_.city as city3_0_0_, address0_.zipcode as zipcode4_0_0_,
employee1_.emp_id as emp_id1_1_1_, employee1_.emp_name as emp_name2_1_1_,
employee1_.emp_salary as emp_sala3_1_1_ from ADDRESS address0_ left outer join
EMPLOYEE employee1_ on address0_.emp_id=employee1_.emp_id where address0_.emp_id=?
Hibernate: select address0_.emp_id as emp_id1_0_0_, address0_.address_line1 as
address_2_0_0_, address0_.city as city3_0_0_, address0_.zipcode as zipcode4_0_0_,
employee1_.emp_id as emp_id1_1_1_, employee1_.emp_name as emp_name2_1_1_,
employee1_.emp_salary as emp_sala3_1_1_ from ADDRESS address0_ left outer join
EMPLOYEE employee1_ on address0_.emp_id=employee1_.emp_id where address0_.emp_id=?
Hibernate: select address0_.emp_id as emp_id1_0_0_, address0_.address_line1 as
address_2_0_0_, address0_.city as city3_0_0_, address0_.zipcode as zipcode4_0_0_,
employee1_.emp_id as emp_id1_1_1_, employee1_.emp_name as emp_name2_1_1_,
```

Notice that Hibernate is printing SQL queries to the console log, this is happening from `org.hibernate.engine.jdbc.spi.SqlStatementLogger` class `logStatement` method, code snippet is:

```
if (this.logToStdout)
        System.out.println("Hibernate: " + statement);
```

This is controlled by `hibernate.show_sql` property in hibernate.cfg.xml file. Actually if you are using log4j, then you should turn it off because they will also be part of log files if logging level is set to DEBUG mode. Having it in log files make more sense that getting it printed on console.

That's all for hibernate log4j logging example, as you can see it's very easy to plugin and all we need is to configure log4j properly. You can download the project from below link.

Download Hibernate Log4j Project

« PREVIOUS
Hibernate EHCache – Hibernate Second Level Cache

NEXT »
Hibernate Native SQL Query Example

About Pankaj

If you have come this far, it means that you liked what you are reading. Why not reach little more and connect with me directly on **Google Plus**, **Facebook** or **Twitter**. I would love to hear your thoughts and opinions on my articles directly.

Recently I started creating video tutorials too, so do check out my videos on **Youtube**.

FILED UNDER: HIBERNATE

Comments

Vaibahv Mankar says

SEPTEMBER 23, 2016 AT 3:52 AM

Hi,

We have hibernate 4.1.xx and log4j2. when we used hibernate.show_sql=true and in log4j2 used fileappender. Its not logging in file. Query display only on console.

Reply

Raghavendra M says

APRIL 21, 2015 AT 12:54 AM

Hello , this is very helpfull me. I have a doubt , here in log4j.xml , you created a loger as . Is this ok if the same name in my project or has to be changed?. The problem I am facing is , I am able to log only logger.error(“”); values in system.log file . I am not getting any values in project.log file. How to solve this please help me.

Reply

Prasad says

DECEMBER 17, 2014 AT 10:04 PM

I followed all ur steps but for me all logs are displaying in system.log file there is no seperation. Project.log is showing empty file.

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

Search for tutorials...

DOWNLOAD ANDROID APP

HIBERNATE FRAMEWORK

Hibernate Tutorial

- Hibernate Example
- Hibernate SessionFactory
- Hibernate Session get load
- Hibernate Session save
- HQL Example
- Hibernate Criteria
- Hibernate SQL
- Hibernate Named Query
- Hibernate Log4J
- Hibernate Validator
- Hibernate Tomcat DataSource

Hibernate Mapping

- Hibernate One to One Mapping
- Hibernate One to Many Mapping
- Hibernate Many to Many Join Tables

Hibernate Caching

- Hibernate Cache
- Hibernate EHCache

Hibernate Integrations

- Hibernate Spring
- Hibernate Spring MVC
- Hibernate Struts 2
- Hibernate Primefaces
- Hibernate Primefaces Spring
- Hibernate SpringRoo Primefaces
- Hibernate JSF Spring

Miscellaneous

- Hibernate Tools Eclipse Plugin
- Hibernate Configuration Offline
- [Solved] No identifier specified
- Hibernate Program Not Terminating
- Access to DialectResolutionInfo
- get is not valid
- No CurrentSessionContext configured
- Hibernate Interview Questions

RECOMMENDED TUTORIALS

Java Tutorials

- Java IO
- Java Regular Expressions
- Multithreading in Java
- Java Logging
- Java Annotations
- Java XML
- Collections in Java
- Java Generics
- Exception Handling in Java
- Java Reflection
- Java Design Patterns
- JDBC Tutorial

Java EE Tutorials

- Servlet JSP Tutorial
- Struts2 Tutorial
- Spring Tutorial
- Hibernate Tutorial

- Primefaces Tutorial
- Apache Axis 2
- JAX-RS
- Memcached Tutorial