Как настроить ведение журнала в Hibernate - SLF4j Log4j

В Java у нас обычно делается лог ошибок, для чего пишется дополнительный код в программу. А тут делается лог всех подключений, действий и запросов, которые осуществляет hibernate. Причем логирование запускается само и не нужно писать никакого дополнительного кода в программе, чтобы логгирование начало выполняться. Все прописывается один раз в настроечных файлах pom.xml и log4j.properties.

Hibernate использует [Simple Logging Facade для Java (SLF4J)](http://www.slf4j.org/), чтобы перенаправить вывод журналирования в вашу удобную среду журналирования (log4j, JCL, JDK logging, lofback …​). В этом уроке мы покажем вам, как выполнить вход в систему **Hibernate с каркасом ведения журнала SLF4j Log4j**.

Технологии, использованные в этой статье:

Hibernate 3.6.3.Final, Maven 3.0.3, SLF4J-api-1.6.1, SLF4J-log4j12-1.6.1

1. Получить SLF4j Log4j

Для входа в Hibernate вам понадобится « **slf4j-api.jar**» и предпочитаемая привязка, например, log4j « **slf4j-log4j12.jar**«. Просто объявляйте зависимость в вашем pom.xml .

*File: pom.xml*

<project ...>

<repositories>

<repository>

<id>JBoss repository</id>

<url>http://repository.jboss.org/nexus/content/groups/public/</url>

</repository>

</repositories>

<dependencies>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>3.6.3.Final</version>

</dependency>

<!-- slf4j-log4j -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>1.6.1</version>

</dependency>

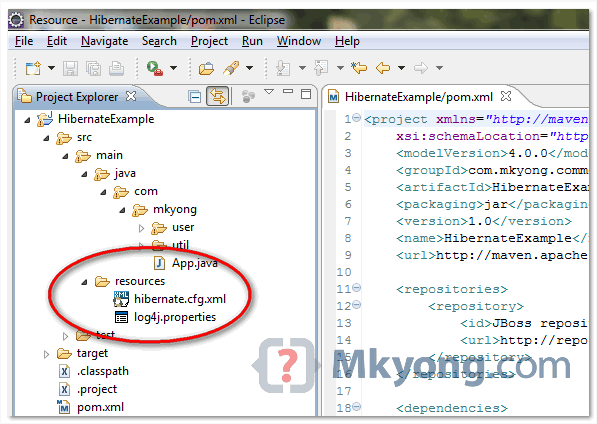
</dependencies>

</project>

* Где находится slf4j-api.jar?  Slf4j-api.jar **определяется как зависимость «**hibernate-core \*\* », поэтому вам не нужно объявлять его снова.

2. Файл свойств Log4j

Создайте файл « **log4j.properties**» и поместите его в путь к классам вашего проекта,



*File: log4j.properties*

# Direct log messages to a log file

log4j.appender.file=org.apache.log4j.RollingFileAppender

log4j.appender.file.File=C:\\mkyongapp.log

log4j.appender.file.MaxFileSize=1MB

log4j.appender.file.MaxBackupIndex=1

log4j.appender.file.layout=org.apache.log4j.PatternLayout

log4j.appender.file.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L - %m%n

# Direct log messages to stdout

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.Target=System.out

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L - %m%n

# Root logger option

log4j.rootLogger=INFO, file, stdout

# Log everything. Good for troubleshooting

log4j.logger.org.hibernate=INFO

# Log all JDBC parameters

log4j.logger.org.hibernate.type=ALL

С этой конфигурацией log4j он перенаправит весь вывод журнала на консоль, а также файл в «C: \\ mkyongapp.log`«.

* Примечание \*\* Hibernate предоставляет множество настроек, позволяющих разработчику решать, что регистрировать.

Всегда обращайтесь к этому [Hibernate Log Categories](http://docs.jboss.org/hibernate/core/3.6/reference/en-US/html/session-configuration.html" \l "configuration-logging), выберите его и внедрите в свой журнальный файл.

3. Вывод

Попробуйте запустить ваше веб-приложение Hibernate, все выходные данные будут записаны в файл «C: \\ mkyongapp.log`».

Как логировать Hibernate? Используем правильную конфигурацию

[05/10/2016](http://akorsa.ru/2016/10/kak-logirovat-hibernate-ispolzuem-pravilnuyu-konfiguratsiyu/)  [admin](http://akorsa.ru/author/admin/)  [Hibernate](http://akorsa.ru/category/hibernate/)

Введение

Для поиска источника проблем с производительностью нашего приложения мы подбираем нужную конфигурацию логирования. Выбор конфигурации также влияет на создаваемую лишнюю нагрузку, которая замедляет работу приложения. Вы должны решить, какую именно информацию нужно извлекать в лог.

Поэтому я предпочитаю использовать две разных конфигурации:

* Конфигурация для этапа разработки, в которой логируется достаточно внутренней информации для понимания взаимодействий БД, можно увидеть количество выполненных запросов и проверить SQL выражения.
* Конфигурация для этапа эксплуатации, в которой логируется меньше сообщений, насколько это возможно во избежание перегрузок.

Фреймворки логирования

До того, как мы обратим внимание на разные категории логирования и уровни, давайте посмотрим на фреймворки логирования, которые поддерживаются Hibernate. Начиная с версии 4.0, Hibernate использует JBoss Logging библиотеку для записи сообщения в лог файл. Эта библиотека это мост, который связывает разные лог фреймворки. Вы сами решаете, какие из следующих фреймворков Вы хотите использовать в своем приложении:

1. JBoss LogManager
2. Log4j 2
3. Log4j 1
4. Slf4j
5. JDK logging

Вам нужно всего лишь добавить предпочитаемый фреймворк в classpath и JBoss Logging подхватит его. Если есть несколько доступных фреймворков, то только один с высшим приоритетом будет выбран.

Я лично предпочитаю log4j 2 и буду использовать его в качестве примера. Концепция и категории логирования одинаковы во всех фреймворках, кроме того что названия уровней логирования могут отличаться, если Вы используете другой фреймоворк.

Категории логирования

Как все приложения и фреймворки, Hibernate записывает сообщения в разные категории и уровни логирования.

Категории группируют сообщения лога по специальным темам, например, выполненные SQL выражения или взаимодействия с кэшем. Следующая таблица показывает самые главные категории логов используемых Hibernate:

|  |  |
| --- | --- |
| **Категория** | **Описание** |
| org.hibernate | Эта категория содержит все сообщения, написанные Hibernate. Вы можете использовать их для анализа неспецифичных проблем.  Будьте осторожны, настроив эту категорию на глубокий уровень логирования, Вы рискуете создать большую нагрузку на вывод лога. |
| org.hibernate.SQL | Все SQL выражения, выполненные через JDBC записываются в эту категорию логов.  Вы можете использовать ее вместе с org.hibernate.type.descriptor.sql для получения подробной информации о JDBC параметрах и результатах. |
| org.hibernate.type.descriptor.sql | Hibernate записывает значения, привязанные к JDBC параметрам и выводит их из JDBC результата в эту категорию логов.  Эта категория должна использоваться вместе с org.hibernate.SQL для логирования самих SQL выражений. |
| org.hibernate.pretty | Hibernate логирует соcтояние сущностей во время flush (максимально -20 сущностей). |
| org.hibernate.cache | Информация об активностях в кэше второго уровня записывается в эту категорию логов. |
| org.hibernate.stat | Hibernate записывает некоторую статистику для каждого запроса в эту категорию. Статистику нужно активировать отдельно. |
| org.hibernate.hql.internal.ast.AST | Эта категория группирует АСД  HQL и SQL  (абстрактное синтаксическое дерево) в процессе парсинга запроса. |
| org.hibernate.tool.hbm2ddl | Hibernate записывает DDL SQL выражения, выполняемые в процессе миграции схемы в эту категорию |

Названия уровней логирования определены Вашим фреймворком логирования и определяют количество и глубину детализации сообщений. Вы можете назначать уровень логирования каждой категории. Если Вы не задаете уровень для отдельной категории, она будет наследовать уровень из родительской категории.

Не используйте show\_sql для логирования SQL запросов

Вопросы такого вида «Как можно логировать выполняемые SQL запросы» являются самыми часто задаваемыми вопросами и большинство популряных ответов в интернете выглядит так, «выставьте параметру show\_sql в persistence.xml значение true» *(как написано в оф.документации)*. **Пожалуйста, не делайте этого!**

Hibernate предоставляет два способа активации логирования выполненных SQL запросов и **установка параметра show\_sql в значение true самый худший**. Активация логирования таким способом несет два больших недостатка:

1. Hibernate записывает все выполняемые SQL выражения в стандартный вывод без использования фреймворка логирования (log4j, JDK Logging и т.д). Поэтому, трудно найти лог файл, который должен содержать эти специфичные сообщения и запись в стандартный вывод в большинстве случаев, намного медленнее, чем использование оптимизированных фреймворков логирования.
2. Persistence.xml это часть jar файла Вашего приложения и Вам необходимо открыть этот бинарник, если хотите активировать или деактивировать логирование SQL выражений. Конфигурация фреймворка логирования чаще всего это внешний текстовый файл, который легко изменяется в любой системе.

**Лучший способ активировать логирование выполняемых SQL выражений это установить уровень логирования категории org.hibernate.SQL в режим DEBUG.**

Hibernate будет записывать SQL выражения без всякой информации о значениях JDBC параметров. Если Вы так же хотите получить эту информацию, то нужно установить уровень логирования категории org.hibernate.type.descriptor.sql в режим DEBUG.

Рекомендуемые конфигурации логирования

Требования для среды продакшн и для среды разработки очень различаются. В процессе разработки, я хочу знать, что делает Hibernate в закулисье. Но логирование этой информации замедляет приложение и оно не требуется для продакшн. Поэтому я рекомендую использовать две разных стратегии:

**Разработка**

Hibernate проделывает много операций вместо Вас на заднем плане и поэтому легко забыть, что Вы еще работаете с БД. Но если Вы хотите быть уверенным, что приложение работает как ожидается, то необходимо проверить количество выполняемых запросов и их SQL выражения.

Для получения данной информации, Вам нужно установить уровень логирования в категории org.hibernate.SQL в значение DEBUG и я рекомендую также активировать Hibernate стастистику для получения сводки большинства важных метрик в конце сессии.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | log4j.appender.stdout=org.apache.log4j.ConsoleAppender  log4j.appender.stdout.Target=System.out  log4j.appender.stdout.layout=org.apache.log4j.PatternLayout  log4j.appender.stdout.layout.ConversionPattern=%d{HH:mm:ss,SSS} %-5p [%c] - %m%n    log4j.rootLogger=info, stdout  # basic log level for all messages  log4j.logger.org.hibernate=info    # SQL statements and parameters  log4j.logger.org.hibernate.SQL=debug  log4j.logger.org.hibernate.type.descriptor.sql=trace |

Следующий кусок кода демонстрирует сообщения Hibernate для сессии, в которой я сохранил две новых сущности «Author», потом обновил одну из них и выбрал всех авторов с данной фамилией.

|  |  |
| --- | --- |
| 1  2  3  4  5 | 17:34:50,353 DEBUG [org.hibernate.SQL] - select author0\_.id as id1\_0\_, author0\_.firstName as firstNam2\_0\_, author0\_.lastName as lastName3\_0\_, author0\_.version as version4\_0\_ from Author author0\_ where author0\_.id=1  17:34:50,362 TRACE [org.hibernate.type.descriptor.sql.BasicExtractor] - extracted value ([id1\_0\_] : [BIGINT]) - [1]  17:34:50,373 TRACE [org.hibernate.type.descriptor.sql.BasicExtractor] - extracted value ([firstNam2\_0\_] : [VARCHAR]) - [Joshua]  17:34:50,373 TRACE [org.hibernate.type.descriptor.sql.BasicExtractor] - extracted value ([lastName3\_0\_] : [VARCHAR]) - [Bloch]  17:34:50,374 TRACE [org.hibernate.type.descriptor.sql.BasicExtractor] - extracted value ([version4\_0\_] : [INTEGER]) - [0] |

Если нужно получить больше информации, Вы можете установить org.hibernate.type.descriptor на уровень TRACE, для того чтобы увидеть значения JDBC параметров и извлеченных из результата и org.hibernate.stat на DEBUG, для извлечения статистики о каждом запросе.

**Продакшн**

До тех пор пока Вам не нужно анализировать проблемы на продуктивной среде, Вы должны логировать настолько мало информации, насколько это возможно. Это можно сделать, установив уровень ERROR во всех связанных с Hibernate категориях. Вы должны так же убедиться, что компонент статистики Hibernate отключен.

Вывод

Hibernate поддерживает несколько провайдерова логирования и Вы можете выбрать одного, добавив в зависимости или просто положив в classpath проекта. Сообщения лога сгруппированы по разным категориям и Вы можете активировать или деактивировать их, выставляя нужный уровень логирования для каждой категории.

Запись сообщений лога и сбор требуемой информации занимает некоторое время. Поэтому вы должны выключить ненужные сообщения в конфигурации на продуктивной среде. Конфигурация на девелоперской среде должна включать логирование для некоторых важных категорий, как org.hibernate.SQL и я также советую включить статистику для нахождения потенциальных проблем с производительностью на раннем этапе.