Пример конфигурации Hibernate 5 + HikariCP

*Опубликовано 13 мая 2017*

[HikariCP](https://brettwooldridge.github.io/HikariCP/) - это библиотека пула соединений с открытым исходным кодом JDBC, которая быстрее и меньше других библиотек пула соединений, таких как Apache DBCP, C3P0 и т. Д.

Hibernate обеспечивает поддержку приложения Java для использования HikariCP для пула соединений с дополнительными настройками конфигурации.

HikariCP - это пул соединений производственного качества с нулевыми издержками. В этом посте я собираюсь интегрировать hibernate с HikariCP.

**Пул подключений**

Пул соединений поддерживает кэш соединений с базой данных, он поддерживает некоторое количество объектов соединений с базой данных, которые можно использовать повторно, когда приложению требуется новый объект соединения с базой данных. Пул соединений поддерживается IBM DB2, Microsoft SQL Server, Oracle, MySQL, PostgreSQL и т. Д.,

**Зачем нужен пул подключений?**

Создание и открытие соединения с базой данных для каждого запроса занимает много времени. Используя пул соединений, после того, как соединение создано, мы можем поместить его в пул соединений и использовать его, когда нам потребуется.

В этой статье я покажу вам, как настроить библиотеку HikariCP с помощью Hibernate ORM.

Пример с прописанием свойств подключения в файле *hibernate.cfg.xml*

Следуя пошаговой процедуре, вы сможете настроить полное рабочее приложение в Eclipse.

**Шаг 1:** Создайте новый проект 'hibernate\_hiakricp' в Eclipse.

**Шаг 2:** Mavenize проекта. Щелкните правой кнопкой мыши по проекту -> Configure -> Convert To Maven Project.

**Шаг 3:** Обновите pom.xml для зависимостей maven.

**pom.xml**

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>4.3.11.Final</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-hikaricp</artifactId>

<version>4.3.11.Final</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.36</version>

</dependency>

**Step 4:**Создайте hibernate.cfg.xml и обновите его, как показано ниже. Убедитесь, что файл hibernate.cfg.xml находится в пути к классам.

**hibernate.cfg.xml**

<?xml version='1.0' encoding='utf-8'?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<!-- включить ведение журнала всех сгенерированных операторов SQL на консоли -->

<property name="show\_sql">true</property>

<!-- отформатировать сгенерированный оператор SQL, чтобы сделать его

более читабельным -->

<property name="format\_sql">false</property>

<!-- Hibernate поместит комментарии во все сгенерированные операторы SQL, чтобы

подсказать, что пытается сгенерированный SQL -->

<property name="use\_sql\_comments">true</property>

<!-- Это свойство заставляет Hibernate генерировать соответствующий SQL для

выбранной базы данных -->

<property name="dialect">org.hibernate.dialect.MySQLDialect</property>

<!-- Удалить и заново создать схему базы данных при запуске -->

<property name="hbm2ddl.auto">update</property>

<property name="connection.release\_mode">after\_transaction</property>

<!-- Специальные свойства Hikari и MySQL-->

<property name="connection.provider\_class">com.zaxxer.hikari.hibernate

.HikariConnectionProvider</property>

<property name="hikari.dataSource.cachePrepStmts">true</property>

<property name="hikari.dataSource.prepStmtCacheSize">250</property>

<property name="hikari.dataSource.prepStmtCacheSqlLimit">2048</property>

<property name="hikari.dataSource.useServerPrepStmts">true</property>

<property name="hikari.maximumPoolSize">30</property>

<property name="hikari.idleTimeout">30000</property>

<!-- Свойства подключения к базе данных -->

<property name="hibernate.hikari.dataSourceClassName">com.mysql.jdbc.jdbc2

.optional.MysqlDataSource</property>

<property name="hikari.dataSource.url">jdbc:mysql://127.0.0.1/sample</property>

<property name="hikari.dataSource.user">root</property>

<property name="hikari.dataSource.password">tiger</property>

<!-- mappings для аннотированных классов -->

<mapping class="com.sample.Employee" />

</session-factory>

</hibernate-configuration>

**Step 5:**Создайте пакет com.sample и определите класс Employee.

**Employee.java**

**package** com.sample;

**import** **javax.persistence.Entity**;

**import** **javax.persistence.GeneratedValue**;

**import** **javax.persistence.GenerationType**;

**import** **javax.persistence.Id**;

**import** **javax.persistence.TableGenerator**;

**@Entity**(name = "employee")

**public** **class** **Employee** {

**@Id**

**@TableGenerator**(name = "empId", table = "employeeIdTable", pkColumnName = "empKey",

pkColumnValue = "empValue", allocationSize = **1**)

**@GeneratedValue**(strategy = GenerationType.TABLE, generator = "empId")

**private** **int** id;

**private** String firstName;

**private** String lastName;

**public** **Employee**() {}

**public** **int** **getId**() {**return** id;}

**public** **void** **setId**(**int** id) {**this**.id = id;}

**public** String **getFirstName**() {**return** firstName;}

**public** **void** **setFirstName**(String firstName) {**this**.firstName = firstName;}

**public** String **getLastName**() {**return** lastName;}

**public** **void** **setLastName**(String lastName) {**this**.lastName = lastName;}

}

**Step 6:**Создайте, для создания сессии, класс HibernateUtil.java

**HibernateUtil.java**

**package** com.sample;

**import** **org.hibernate.Session**;

**import** **org.hibernate.SessionFactory**;

**import** **org.hibernate.boot.registry.StandardServiceRegistryBuilder**;

**import** **org.hibernate.cfg.Configuration**;

**public** **class** **HibernateUtil** {

**private** **static** SessionFactory sessionFactory = getSessionFactory();

**private** **static** SessionFactory **getSessionFactory**() {

**if** (sessionFactory == **null**) {

Configuration configuration = **new** Configuration().configure();

StandardServiceRegistryBuilder builder = **new** StandardServiceRegistryBuilder()

.applySettings(configuration.getProperties());

sessionFactory = configuration.buildSessionFactory(builder.build());

*//или вместо предыдущих 3-х строк*

sessionFactory = new Configuration().configure().buildSessionFactory();

}

**return** sessionFactory;

*//Кроме того все это вместо if лучше оборачивать в блок try/catch c ExceptionInInitializerError + использовать статический блок для заполнения переменной sessionFactory* *(в нем заполнение этой переменной выглядит гораздо более читаемым, чем так через метод)*:

private static final SessionFactory sessionFactory;

static {

try {

sessionFactory = new Configuration().configure().buildSessionFactory(); //метод “configure().” подгружает файл c конфигурацией “hibernate.cfg.xml”, а метод “buildSessionFactory()” настраивает сессионную фабрику

//Кроме того, я так пронимаю, что использование класса Configuration с методами .configure().buildSessionFactory() заменяет собой использование классов StandartServiceRegistryBuilder и MetadataSource.

} catch (Throwable e) {

throw new ExceptionInInitializerError(e);

}

}

**public static** Session **getSession**() throws HibernateException {

return sessionFactory.openSession();

}

}

**Step 7:**Создайтете класс Main.java.

**Main.java**

**package** com.sample;

**import** **org.hibernate.Session**;

**import** **org.hibernate.SessionFactory**;

**import** **org.hibernate.Transaction**;

**public** **class** **Main** {

**public** **static** **void** **insertObject**(Employee emp) {

Session session = **null**;

**try** (session = HibernateUtil.getSession()) {

session.beginTransaction();

session.save(emp);

Transaction trans = session.getTransaction();

trans.commit();

} **catch** (Exception e) {

System.out.println(e.getMessage());

e.printStackTrace();

}

}

**public** **static** **void** **main**(String[] args) {

Employee emp = **new** Employee();

emp.setFirstName("ptr");

emp.setLastName("Nayan");

insertObject(emp);

}

}

**Step 8:** Создайте таблицу в примере базы данных.

**create** **table** **employee** (id **integer** **not** **null**, firstName **varchar**(**255**), lastName **varchar**(**255**), **primary** **key** (id));

Запустите Main.java, вы получите следующий вид вывода в консоли.

INFO: HHH000399: Using default transaction strategy (direct JDBC transactions)

Hibernate: select sequence\_next\_hi\_value from hibernate\_sequences where sequence\_name = 'employee' for update

Hibernate: update hibernate\_sequences set sequence\_next\_hi\_value = ? where sequence\_next\_hi\_value = ? and sequence\_name = 'employee'

Hibernate: /\* insert com.sample.Employee \*/ insert into employee (firstName, lastName, id) values (?, ?, ?)

mysql> **select** \* **from** employee;

+--------+-----------+----------+

| id | firstName | lastName |

+--------+-----------+----------+

| **163840** | ptr | Nayan |

+--------+-----------+----------+

**1** row **in** **set** (**0**.**00** sec)

Пример с прописанием свойств подключения в коде в каком-то классе Environment

Jar Dependencies

В  pom.xml файле вашего Maven проекта, добавить   hibernate-core,  hibernate-hikaricp,  HikariCP и  mysql-connector-java зависимости следующим образом .

**pom.xml**

<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.boraji.tutorial.hibernate</groupId>

<artifactId>hibernate-hikari-example</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>hibernate-hikari-example</name>

<url>[http://maven.apache.org](http://maven.apache.org/)</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>6.0.5</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>5.2.10.Final</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-hikaricp</artifactId>

<version>5.2.10.Final</version>

</dependency>

<dependency>

<groupId>com.zaxxer</groupId>

<artifactId>HikariCP</artifactId>

<version>2.6.1</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.25</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-simple</artifactId>

<version>1.7.25</version>

</dependency>

</dependencies>

<build>

<sourceDirectory>src/main/java</sourceDirectory>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.5.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

Конфигурация Hibernate + HikariCP

Чтобы включить пул соединений HikariCP, вы можете установить параметры конфигурации HikariCP в Hibernate с префиксом hibernate.hikari.следующим образом.

*// HikariCP settings*

*// Maximum waiting time for a connection from the pool*

settings.put("hibernate.hikari.connectionTimeout", "20000");

*// Minimum number of ideal connections in the pool*

settings.put("hibernate.hikari.minimumIdle", "10");

*// Maximum number of actual connection in the pool*

settings.put("hibernate.hikari.maximumPoolSize", "20");

*// Maximum time that a connection is allowed to sit ideal in the pool*

settings.put("hibernate.hikari.idleTimeout", "300000");

Для всех доступных настроек в HikariCP, вы можете обратиться по этой ссылке -   [Конфигурация HikariCP](https://github.com/brettwooldridge/HikariCP#configuration-knobs-baby) .

Здесь полный  HibernateUtil класс для запуска приложения гибернации.

**HibernateUtil.java**

**package** com.boraji.tutorial.hibernate;

**import** java.util.HashMap;

**import** java.util.Map;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.boot.Metadata;

**import** org.hibernate.boot.MetadataSources;

**import** org.hibernate.boot.registry.StandardServiceRegistry;

**import** org.hibernate.boot.registry.StandardServiceRegistryBuilder;

**import** org.hibernate.cfg.Environment;

**import** com.boraji.tutorial.hibernate.entity.Person;

*/\*\**

*\* @author imssbora*

*\*/*

**public** **class** **HibernateUtil** {

**private** **static** StandardServiceRegistry registry;

**private** **static** SessionFactory sessionFactory;

**public** **static** SessionFactory **getSessionFactory**() {

**if** (sessionFactory == **null**) {

**try** {

StandardServiceRegistryBuilder registryBuilder =

**new** StandardServiceRegistryBuilder();

Map<String, Object> settings = **new** HashMap<>();

settings.put(Environment.DRIVER, "com.mysql.cj.jdbc.Driver");

settings.put(Environment.URL, "jdbc:mysql://localhost:3306/BORAJI?useSSL=false");

settings.put(Environment.USER, "root");

settings.put(Environment.PASS, "admin");

settings.put(Environment.HBM2DDL\_AUTO, "update");

settings.put(Environment.SHOW\_SQL, **true**);

*// HikariCP настройки*

*// Максимальное время ожидания соединения из пула l*

settings.put("hibernate.hikari.connectionTimeout", "20000");

*// Минимальное количество идеальных соединений в пуле*

settings.put("hibernate.hikari.minimumIdle", "10");

*// Максимальное количество фактических соединений в пуле*

settings.put("hibernate.hikari.maximumPoolSize", "20");

*//Максимальное время,в течение которого соединение может идеально находиться в пуле*

settings.put("hibernate.hikari.idleTimeout", "300000");

registryBuilder.applySettings(settings);

registry = registryBuilder.build();

MetadataSources sources = **new** MetadataSources(registry)

.addAnnotatedClass(Person.class);

Metadata metadata = sources.getMetadataBuilder().build();

sessionFactory = metadata.getSessionFactoryBuilder().build();

} **catch** (Exception e) {

**if** (registry != **null**) {

StandardServiceRegistryBuilder.destroy(registry);

}

e.printStackTrace();

}

}

**return** sessionFactory;

}

**public** **static** **void** **shutdown**() {

**if** (registry != **null**) {

StandardServiceRegistryBuilder.destroy(registry);

}

}

}

Класс сущности

Простой модельный класс с аннотациями JPA.

**Person.java**

**package** com.boraji.tutorial.hibernate.entity;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Entity

@Table(name = "PERSONS")

**public** **class** **Person** {

@Id

@GeneratedValue

@Column(name = "ID")

**private** Long id;

@Column(name = "NAME")

**private** String name;

*//Getter and Setter*

}

Запустить приложение

Создайте  MainApp класс для тестирования конфигурации HikariCP и Hibernate.

**MainApp.java**

**package** com.boraji.tutorial.hibernate;

**import** org.hibernate.Session;

**import** org.hibernate.Transaction;

**import** com.boraji.tutorial.hibernate.entity.Person;

*/\*\**

*\* @author imssbora*

*\*/*

**public** **class** **MainApp** {

**public** **static** **void** **main**(String[] args) {

Session session = **null**;

Transaction transaction = **null**;

**try** {

session = HibernateUtil.getSessionFactory().openSession();

transaction = session.beginTransaction();

transaction.begin();

Person person = **new** Person();

person.setName("Mike Lewis");

session.save(person);

transaction.commit();

} **catch** (Exception e) {

**if** (transaction != **null**) {

transaction.rollback();

}

} **finally** {

**if** (session != **null**) {

session.close();

}

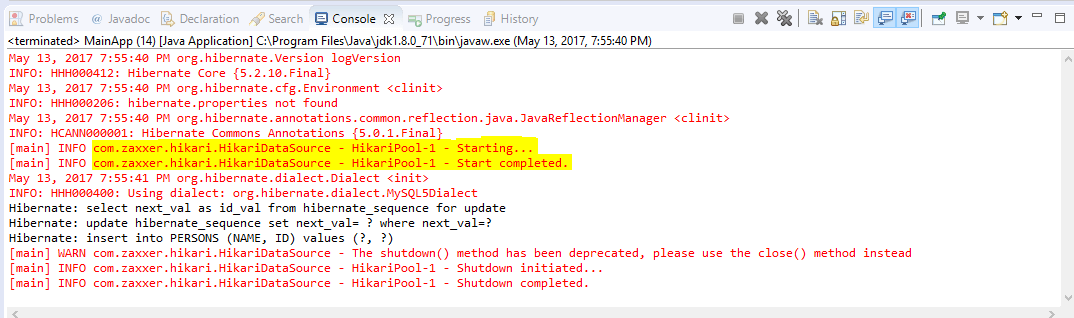
}

HibernateUtil.shutdown();

}

}

При выполнении  MainApp класса вывод консоли будет выглядеть следующим образом.



Т.е. о том, что нужно использовать HicariCP мы прописываем только в Maven, а в файле **HibernateUtil.java** мы просто заполняем setting данными для пула HicariCP (*без четкого написания, что эти данные для него*) и потом просто подхватываем этот setting в registryBuilder(*опять же без указания, что это для* HicariCP) – и все само заработает.