Pagination in Hibernate

# Maven Configuration (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>pagination-hibernate</groupId>

<artifactId>pagination-hibernate</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>many-to-one1</name>

<url>http://maven.apache.org</url>

<properties>

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

<spring.version>4.1.2.RELEASE</spring.version>

<spring.security.version>3.2.3.RELEASE</spring.security.version>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

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

<version>5.1.30</version>

</dependency>

<!-- C3P0 library -->

<dependency>

<groupId>com.mchange</groupId>

<artifactId>c3p0</artifactId>

<version>0.9.5</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>4.3.7.Final</version>

</dependency>

</dependencies>

</project>

# SQL Configuration(ddl.sql)

**drop** **table** if **exists** animal;

**create** **table** animal(

id **int**(11) **NOT** **NULL** AUTO\_INCREMENT,

name **varchar**(10),

**PRIMARY** **KEY** (id)

);

**select** \* **from** animal;

# Spring-Hibernate Configuration

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:jee=*"http://www.springframework.org/schema/jee"*

xmlns:tx=*"http://www.springframework.org/schema/tx"* xmlns:jdbc=*"http://www.springframework.org/schema/jdbc"*

xmlns:osgi=*"http://www.springframework.org/schema/osgi"* xmlns:security=*"http://www.springframework.org/schema/security"*

xsi:schemaLocation=*"http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd*

*http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd*

*http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-3.0.xsd*

*http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd*

*http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-3.0.xsd*

*http://www.springframework.org/schema/osgi http://www.springframework.org/schema/osgi/spring-osgi.xsd*

*http://www.springframework.org/schema/security http://www.springframework.org/schema/security/spring-security-3.0.3.xsd"*>

<tx:annotation-driven transaction-manager=*"discussionTransactionManager"* />

<bean id=*"dataSourceInternal"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"*

destroy-method=*"close"*>

<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"* />

<property name=*"jdbcUrl"* value=*"jdbc:mysql://localhost/test"* />

<property name=*"user"* value=*"deba"* />

<property name=*"password"* value=*"deba"* />

<!-- these are C3P0 properties -->

<property name=*"acquireIncrement"* value=*"5"* />

<property name=*"initialPoolSize"* value=*"5"* />

<property name=*"minPoolSize"* value=*"5"* />

<property name=*"maxPoolSize"* value=*"20"* />

</bean>

<!-- This is the lazy DataSource proxy that interacts with the target DataSource once a real statement is sent to the database. Users use this DataSource to set up their Hibernate session factory, which in turn forces the Hibernate second-level cache and also everything that interacts with that Hibernate session factory to use it. -->

<bean id=*"dataSource"* class=*"org.springframework.jdbc.datasource.LazyConnectionDataSourceProxy"*>

<property name=*"targetDataSource"*>

<ref bean=*"dataSourceInternal"* />

</property>

</bean>

<!-- <bean id="hibSessionFactory" class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean"> -->

<bean id=*"hibSessionFactory"* class=*"org.springframework.orm.hibernate4.LocalSessionFactoryBean"*>

<property name=*"dataSource"* ref=*"dataSource"* />

<property name=*"hibernateProperties"*>

<value>

hibernate.id.new\_generator\_mappings=true

<!-- hibernate.current\_session\_context\_class=thread -->

</value>

</property>

<property name=*"annotatedClasses"*>

<list>

<value>com.ddlab.rnd.orm.Animal</value>

</list>

</property>

</bean>

</beans>

# Java Source Code

## Animal.Java

**package** com.ddlab.rnd.orm;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Entity

@Table(name = "Animal")

**public** **class** Animal {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "id")

**private** **long** id;

@Column(name = "name")

**private** String name;

**public** **long** getId() {

**return** id;

}

**public** **void** setId(**long** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

}

## App.java (To insert into table)

**package** com.ddlab.rnd.orm;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** App {

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

ApplicationContext context = **new** ClassPathXmlApplicationContext(

"app-context.xml");

SessionFactory sessionFactory = (SessionFactory) context

.getBean("hibSessionFactory");

Session session = sessionFactory.openSession();

Transaction tx = **null**;

**try** {

tx = session.beginTransaction();

**for** (**int** i = 0; i < 100; i++) {

Animal animal = **new** Animal();

animal.setName("Animal-" + i);

session.save(animal);

}

tx.commit();

} **catch** (Exception e) {

System.***out***.println("Exception occured. " + e.getMessage());

e.printStackTrace();

} **finally** {

**if** (!sessionFactory.isClosed()) {

System.***out***.println("Closing SessionFactory");

sessionFactory.close();

}

}

}

}

## TestPagination.java

**package** com.ddlab.rnd.orm;

**import** java.util.List;

**import** org.hibernate.Criteria;

**import** org.hibernate.Query;

**import** org.hibernate.ScrollMode;

**import** org.hibernate.ScrollableResults;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** TestPagination {

**public** **static** **void** showAnimalsPageByPage(**int** minValue ) {

ApplicationContext context = **new** ClassPathXmlApplicationContext(

"app-context.xml");

SessionFactory sessionFactory = (SessionFactory) context

.getBean("hibSessionFactory");

Session session = sessionFactory.openSession();

System.***out***.println("Initial Value :::"+minValue);

Query query = session.createQuery("From Animal");

query.setFirstResult(minValue);

query.setMaxResults(10);

List<Animal> animalList = query.list();

**for**( Animal animal : animalList ) {

System.***out***.println("Animal Name :::"+animal.getName());

}

}

**public** **static** **void** showAnimals(Session session, **int** minValue ) {

System.***out***.println("Initial Value :::"+minValue);

Query query = session.createQuery("From Animal");

query.setFirstResult(minValue);

query.setMaxResults(10);

List<Animal> animalList = query.list();

**for**( Animal animal : animalList ) {

System.***out***.println("Animal Name :::"+animal.getName());

}

}

//The Total Count and the Last Page

**public** **static** **void** getAllAnimals() {

**int** pageSize = 10;

String countQ = "Select count(a.id) from Animal a";

Session session = *getSession*();

Query countQuery = session.createQuery(countQ);

Long countResults = (Long) countQuery.uniqueResult();

System.***out***.println("countResults---->"+countResults);//100

**int** lastPageNumber = (**int**) ((countResults / pageSize) + 1);

System.***out***.println("lastPageNumber :::"+lastPageNumber);//11

Query selectQuery = session.createQuery("From Animal");

**int** initialPage = (lastPageNumber - 1) \* pageSize;

System.***out***.println("Initial Page :::"+initialPage);//100

selectQuery.setFirstResult((lastPageNumber - 1) \* pageSize);

selectQuery.setMaxResults(pageSize);

List<Animal> lastPage = selectQuery.list();

}

/\*

\* Using ScrollableResults to implement pagination has the potential to reduce database calls.

\* This approach streams the result set as the program scrolls though it,

\* therefore eliminating the need to repeat the query to fill each page:

\*/

**public** **static** **void** getAllAnimalsByScrollableResults() {

Session session = *getSession*();

String hql = "FROM Animal";

Query query = session.createQuery(hql);

**int** pageSize = 10;

ScrollableResults resultScroll = query.scroll(ScrollMode.***FORWARD\_ONLY***);

resultScroll.first();

resultScroll.scroll(0);//You can increase like 10, 20 ,30

**int** i = 0;

**while** (pageSize > i++) {

Animal animal = (Animal) resultScroll.get(0);

System.***out***.println("Animal Name :::"+animal.getName());

**if** (!resultScroll.next())

**break**;

}

}

**public** **static** **void** getAllAnimalsByCriteria() {

Session session = *getSession*();

**int** pageSize = 10;

String hql = "FROM Animal";

Criteria criteria = session.createCriteria(Animal.**class**);

criteria.setFirstResult(0);//You can set like 10,20,30

criteria.setMaxResults(pageSize);

List<Animal> firstPage = criteria.list();

**for**(Animal animal : firstPage) {

System.***out***.println("Animal Name :::"+animal.getName());

}

}

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

ApplicationContext context = **new** ClassPathXmlApplicationContext(

"app-context.xml");

SessionFactory sessionFactory = (SessionFactory) context

.getBean("hibSessionFactory");

Session session = sessionFactory.openSession();

**return** session;

}

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

// ApplicationContext context = new ClassPathXmlApplicationContext(

// "app-context.xml");

// SessionFactory sessionFactory = (SessionFactory) context

// .getBean("hibSessionFactory");

// Session session = sessionFactory.openSession();

// for( int i = 0 ; i < 100 ; i = i+10) {

// int minValue = i;

//// showAnimals(session, minValue);

// showAnimalsPageByPage(minValue);

// System.out.println("============================================");

// }

// getAllAnimals();

// getAllAnimalsByScrollableResults();

*getAllAnimalsByCriteria*();

}

}