**Java Libraries - use date in native query**

Copy JAR files which are listed below:

antlr-2.7.7.jar

classmate-1.3.4.jar

commons-lang3-3.6.jar

commons-logging-1.1.3.jar

dom4j-1.6.1.jar

ehcache-core-2.6.11.jar

geolatte-geom-1.1.0.jar

hibernate-commons-annotations-5.0.1.Final.jar

hibernate-core-5.2.11.Final.jar

hibernate-ehcache-5.2.12.Final.jar

hibernate-ejb3-persistence.jar

hibernate-enhance-maven-plugin-4.3.7.Final.jar

hibernate-entitymanager.jar

hibernate-java8-5.2.11.Final.jar

hibernate-jpa-2.1-api-1.0.0.Final.jar

hibernate-spatial-5.2.11.Final.jar

hibernate-validator-6.0.2.Final.jar

javassist-3.16.1-GA.jar

jboss-logging-3.3.1.Final.jar

jboss-transaction-api\_1.1\_spec-1.0.1.Final.jar

jts-1.11.jar

mysql-connector-java-5.1.36.jar

slf4j-api-1.7.21.jar

**Create Database**

Create a database with the name is **hibernate5**. This database have a table: **Invoice table**.

--

-- Table structure for table `invoice`

--

CREATE TABLE `invoice` (

`id` int(11) NOT NULL PRIMARY KEY AUTO\_INCREMENT,

`name` varchar(250) NOT NULL,

`dateCreated` date NOT NULL,

`payment` varchar(250) NOT NULL

) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--

-- Dumping data for table `invoice`

--

INSERT INTO `invoice` (`name`, `dateCreated`, `payment`) VALUES

('Invoice 1', '2017-12-08', 'cash');

INSERT INTO `invoice` (`name`, `dateCreated`, `payment`) VALUES

('Invoice 2', '2017-12-08', 'cash');

INSERT INTO `invoice` (`name`, `dateCreated`, `payment`) VALUES

('Invoice 3', '2017-12-05', 'cash');

INSERT INTO `invoice` (`name`, `dateCreated`, `payment`) VALUES

('Invoice 4', '2017-11-01', 'cash');

INSERT INTO `invoice` (`name`, `dateCreated`, `payment`) VALUES

('Invoice 5', '2017-12-01', 'cash');

INSERT INTO `invoice` (`name`, `dateCreated`, `payment`) VALUES

('Invoice 6', '2017-08-09', 'cash');

***Structures of Invoice Table***

***Invoice Table***

**Entities Class**

Create a entity class – **Invoice.java** to represent the above table

***Invoice.java***

package entities;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import static javax.persistence.GenerationType.IDENTITY;

import java.util.Date;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name = "invoice")

public class Invoice implements java.io.Serializable {

@Id

@GeneratedValue(strategy = IDENTITY)

@Column

private Integer id;

@Column

private String name;

@Column

private Date dateCreated;

@Column

private String payment;

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Date getDateCreated() {

return dateCreated;

}

public void setDateCreated(Date dateCreated) {

this.dateCreated = dateCreated;

}

public String getPayment() {

return payment;

}

public void setPayment(String payment) {

this.payment = payment;

}

}

**Hibernate Configuration File**

Puts Invoice.java in your Hibernate configuration file, and also MySQL connection details.

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "<http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd>">

<hibernate-configuration>

<session-factory>

<property name="hibernate.enable\_lazy\_load\_no\_trans">true</property>

<property name="hibernate.connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="hibernate.connection.password">123456</property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/hibernate5</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.dialect">org.hibernate.spatial.dialect.mysql.MySQLSpatialDialect</property>

<property name="hibernate.current\_session\_context\_class">thread</property>

<mapping class="entities.Invoice" />

</session-factory>

</hibernate-configuration>

**Create HibernateUtil class**

The HibernateUtil class helps in creating the SessionFactory from the Hibernate configuration file. The SessionFactory is threadsafe, so it is not necessary to obtain one for each thread.

package native\_language;

import org.hibernate.\*;

import org.hibernate.boot.\*;

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

public class HibernateUtil {

private static final SessionFactory sessionFactory;

static {

try {

StandardServiceRegistry standardRegistry = new

StandardServiceRegistryBuilder()

.configure("hibernate.cfg.xml")

.build();

Metadata metaData = new MetadataSources(

standardRegistry)

.getMetadataBuilder()

.build();

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

} catch (Throwable th) {

throw new ExceptionInInitializerError(th);

}

}

public static SessionFactory getSessionFactory() {

return sessionFactory;

}

}

**Create InvoiceModel class**

The InvoiceModel class contains methods to interact with the database.

package native\_query;

import java.util.List;

import org.hibernate.\*;

import entities.\*;

public class InvoiceModel {

private SessionFactory sessionFactory = HibernateUtil.getSessionFactory();

public List<Invoice> findLast7Days() {

List<Invoice> invoices = null;

Session session = null;

Transaction transaction = null;

try {

session = sessionFactory.openSession();

transaction = session.beginTransaction();

org.hibernate.query.Query query = session.createNativeQuery(

"select \* from invoice where dateCreated >= DATE\_SUB(CURDATE(), INTERVAL 7 DAY)", Invoice.class);

invoices = query.getResultList();

transaction.commit();

} catch (Exception e) {

invoices = null;

if (transaction != null) {

transaction.rollback();

}

} finally {

session.close();

}

return invoices;

}

public List<Invoice> findLastWeek() {

List<Invoice> invoices = null;

Session session = null;

Transaction transaction = null;

try {

session = sessionFactory.openSession();

transaction = session.beginTransaction();

org.hibernate.query.Query query = session.createNativeQuery(

"SELECT \* FROM invoice WHERE dateCreated >= curdate() - INTERVAL DAYOFWEEK(curdate()) + 6 DAY AND dateCreated < curdate() - INTERVAL DAYOFWEEK(curdate()) - 1 DAY",

Invoice.class);

invoices = query.getResultList();

transaction.commit();

} catch (Exception e) {

invoices = null;

if (transaction != null) {

transaction.rollback();

}

} finally {

session.close();

}

return invoices;

}

public List<Invoice> findThisWeek() {

List<Invoice> invoices = null;

Session session = null;

Transaction transaction = null;

try {

session = sessionFactory.openSession();

transaction = session.beginTransaction();

org.hibernate.query.Query query = session.createNativeQuery(

"SELECT \* FROM invoice WHERE YEARWEEK(dateCreated) = YEARWEEK(NOW())", Invoice.class);

invoices = query.getResultList();

transaction.commit();

} catch (Exception e) {

invoices = null;

if (transaction != null) {

transaction.rollback();

}

} finally {

session.close();

}

return invoices;

}

public List<Invoice> findThisMonth() {

List<Invoice> invoices = null;

Session session = null;

Transaction transaction = null;

try {

session = sessionFactory.openSession();

transaction = session.beginTransaction();

org.hibernate.query.Query query = session.createNativeQuery(

"SELECT \* FROM invoice WHERE MONTH(dateCreated) = MONTH(CURRENT\_DATE()) AND YEAR(dateCreated) = YEAR(CURRENT\_DATE())",

Invoice.class);

invoices = query.getResultList();

transaction.commit();

} catch (Exception e) {

invoices = null;

if (transaction != null) {

transaction.rollback();

}

} finally {

session.close();

}

return invoices;

}

public List<Invoice> findLastMonth() {

List<Invoice> invoices = null;

Session session = null;

Transaction transaction = null;

try {

session = sessionFactory.openSession();

transaction = session.beginTransaction();

org.hibernate.query.Query query = session.createNativeQuery(

"SELECT \* FROM invoice WHERE dateCreated >= CONCAT(LEFT(NOW() - INTERVAL 1 MONTH,7),'-01') AND dateCreated <= NOW() - INTERVAL 1 MONTH",

Invoice.class);

invoices = query.getResultList();

transaction.commit();

} catch (Exception e) {

invoices = null;

if (transaction != null) {

transaction.rollback();

}

} finally {

session.close();

}

return invoices;

}

public List<Invoice> findLast30Days() {

List<Invoice> invoices = null;

Session session = null;

Transaction transaction = null;

try {

session = sessionFactory.openSession();

transaction = session.beginTransaction();

org.hibernate.query.Query query = session.createNativeQuery(

"select \* from invoice where dateCreated >= DATE\_SUB(CURDATE(), INTERVAL 30 DAY)", Invoice.class);

invoices = query.getResultList();

transaction.commit();

} catch (Exception e) {

invoices = null;

if (transaction != null) {

transaction.rollback();

}

} finally {

session.close();

}

return invoices;

}

public List<Invoice> findThisQuarter() {

List<Invoice> invoices = null;

Session session = null;

Transaction transaction = null;

try {

session = sessionFactory.openSession();

transaction = session.beginTransaction();

org.hibernate.query.Query query = session.createNativeQuery(

"select \* from invoice where quarter(dateCreated) = quarter(curdate())", Invoice.class);

invoices = query.getResultList();

transaction.commit();

} catch (Exception e) {

invoices = null;

if (transaction != null) {

transaction.rollback();

}

} finally {

session.close();

}

return invoices;

}

public List<Invoice> findLastQuarter() {

List<Invoice> invoices = null;

Session session = null;

Transaction transaction = null;

try {

session = sessionFactory.openSession();

transaction = session.beginTransaction();

org.hibernate.query.Query query = session.createNativeQuery(

"SELECT \* from invoice where dateCreated >= quarter(curdate() - INTERVAL 1 QUARTER)",

Invoice.class);

invoices = query.getResultList();

transaction.commit();

} catch (Exception e) {

invoices = null;

if (transaction != null) {

transaction.rollback();

}

} finally {

session.close();

}

return invoices;

}

public List<Invoice> findLast90Days() {

List<Invoice> invoices = null;

Session session = null;

Transaction transaction = null;

try {

session = sessionFactory.openSession();

transaction = session.beginTransaction();

org.hibernate.query.Query query = session.createNativeQuery(

"select \* from invoice where dateCreated >= DATE\_SUB(CURDATE(), INTERVAL 90 DAY)", Invoice.class);

invoices = query.getResultList();

transaction.commit();

} catch (Exception e) {

invoices = null;

if (transaction != null) {

transaction.rollback();

}

} finally {

session.close();

}

return invoices;

}

public List<Invoice> findThisYear() {

List<Invoice> invoices = null;

Session session = null;

Transaction transaction = null;

try {

session = sessionFactory.openSession();

transaction = session.beginTransaction();

org.hibernate.query.Query query = session.createNativeQuery(

"select \* from invoice where year(dateCreated) = year(curdate())", Invoice.class);

invoices = query.getResultList();

transaction.commit();

} catch (Exception e) {

invoices = null;

if (transaction != null) {

transaction.rollback();

}

} finally {

session.close();

}

return invoices;

}

public List<Invoice> findLastYear() {

List<Invoice> invoices = null;

Session session = null;

Transaction transaction = null;

try {

session = sessionFactory.openSession();

transaction = session.beginTransaction();

org.hibernate.query.Query query = session.createNativeQuery(

"SELECT \* from invoice where dateCreated >= quarter(curdate() - INTERVAL 1 YEAR)", Invoice.class);

invoices = query.getResultList();

transaction.commit();

} catch (Exception e) {

invoices = null;

if (transaction != null) {

transaction.rollback();

}

} finally {

session.close();

}

return invoices;

}

public List<Invoice> findLast365Days() {

List<Invoice> invoices = null;

Session session = null;

Transaction transaction = null;

try {

session = sessionFactory.openSession();

transaction = session.beginTransaction();

org.hibernate.query.Query query = session.createNativeQuery(

"select \* from invoice where dateCreated >= DATE\_SUB(CURDATE(), INTERVAL 365 DAY)", Invoice.class);

invoices = query.getResultList();

transaction.commit();

} catch (Exception e) {

invoices = null;

if (transaction != null) {

transaction.rollback();

}

} finally {

session.close();

}

return invoices;

}

}

**Run It**

package native\_query;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.List;

import entities.Invoice;

public class Main {

public static void main(String[] args) {

SimpleDateFormat simpleDateFormat = new SimpleDateFormat("MM/dd/yyyy");

InvoiceModel invoiceModel = new InvoiceModel();

System.out.println("Today: " + simpleDateFormat.format(new Date()));

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Find By Week \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Find Invoices by This Week");

List<Invoice> invoices = invoiceModel.findThisWeek();

for (Invoice invoice : invoices) {

System.out.println("Id: " + invoice.getId());

System.out.println("Name: " + invoice.getName());

System.out.println("Date Created: " + simpleDateFormat.format(invoice.getDateCreated()));

System.out.println("Payment: " + invoice.getPayment());

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

}

System.out.println("\nFind Invoices by Last Week");

invoices = invoiceModel.findLastWeek();

for (Invoice invoice : invoices) {

System.out.println("Id: " + invoice.getId());

System.out.println("Name: " + invoice.getName());

System.out.println("Date Created: " + simpleDateFormat.format(invoice.getDateCreated()));

System.out.println("Payment: " + invoice.getPayment());

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

}

System.out.println("\nFind Invoices by Last 7 Days");

invoices = invoiceModel.findLast7Days();

for (Invoice invoice : invoices) {

System.out.println("Id: " + invoice.getId());

System.out.println("Name: " + invoice.getName());

System.out.println("Date Created: " + simpleDateFormat.format(invoice.getDateCreated()));

System.out.println("Payment: " + invoice.getPayment());

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

}

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Find By Month \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Find Invoices by This Month");

invoices = invoiceModel.findThisMonth();

for (Invoice invoice : invoices) {

System.out.println("Id: " + invoice.getId());

System.out.println("Name: " + invoice.getName());

System.out.println("Date Created: " + simpleDateFormat.format(invoice.getDateCreated()));

System.out.println("Payment: " + invoice.getPayment());

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

}

System.out.println("\nFind Invoices by Last Month");

invoices = invoiceModel.findLastMonth();

for (Invoice invoice : invoices) {

System.out.println("Id: " + invoice.getId());

System.out.println("Name: " + invoice.getName());

System.out.println("Date Created: " + simpleDateFormat.format(invoice.getDateCreated()));

System.out.println("Payment: " + invoice.getPayment());

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

}

System.out.println("\nFind Invoices by Last 30 Days");

invoices = invoiceModel.findLast30Days();

for (Invoice invoice : invoices) {

System.out.println("Id: " + invoice.getId());

System.out.println("Name: " + invoice.getName());

System.out.println("Date Created: " + simpleDateFormat.format(invoice.getDateCreated()));

System.out.println("Payment: " + invoice.getPayment());

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

}

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Find By Quarter \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Find Invoices by This Quarter");

invoices = invoiceModel.findThisQuarter();

for (Invoice invoice : invoices) {

System.out.println("Id: " + invoice.getId());

System.out.println("Name: " + invoice.getName());

System.out.println("Date Created: " + simpleDateFormat.format(invoice.getDateCreated()));

System.out.println("Payment: " + invoice.getPayment());

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

}

System.out.println("\nFind Invoices by Last Quarter");

invoices = invoiceModel.findLastQuarter();

for (Invoice invoice : invoices) {

System.out.println("Id: " + invoice.getId());

System.out.println("Name: " + invoice.getName());

System.out.println("Date Created: " + simpleDateFormat.format(invoice.getDateCreated()));

System.out.println("Payment: " + invoice.getPayment());

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

}

System.out.println("\nFind Invoices by Last 90 Days");

invoices = invoiceModel.findLast90Days();

for (Invoice invoice : invoices) {

System.out.println("Id: " + invoice.getId());

System.out.println("Name: " + invoice.getName());

System.out.println("Date Created: " + simpleDateFormat.format(invoice.getDateCreated()));

System.out.println("Payment: " + invoice.getPayment());

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

}

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Find By Year \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Find Invoices by This Year");

invoices = invoiceModel.findThisYear();

for (Invoice invoice : invoices) {

System.out.println("Id: " + invoice.getId());

System.out.println("Name: " + invoice.getName());

System.out.println("Date Created: " + simpleDateFormat.format(invoice.getDateCreated()));

System.out.println("Payment: " + invoice.getPayment());

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

}

System.out.println("\nFind Invoices by Last Year");

invoices = invoiceModel.findLastYear();

for (Invoice invoice : invoices) {

System.out.println("Id: " + invoice.getId());

System.out.println("Name: " + invoice.getName());

System.out.println("Date Created: " + simpleDateFormat.format(invoice.getDateCreated()));

System.out.println("Payment: " + invoice.getPayment());

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

}

System.out.println("\nFind Invoices by Last 365 Days");

invoices = invoiceModel.findLast365Days();

for (Invoice invoice : invoices) {

System.out.println("Id: " + invoice.getId());

System.out.println("Name: " + invoice.getName());

System.out.println("Date Created: " + simpleDateFormat.format(invoice.getDateCreated()));

System.out.println("Payment: " + invoice.getPayment());

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

}

}

}

**Output**

Today: 12/08/2017

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Find By Week \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Find Invoices by This Week

Id: 1

Name: Invoice 1

Date Created: 12/08/2017

Payment: cash

============================

Id: 2

Name: Invoice 2

Date Created: 12/08/2017

Payment: cash

============================

Id: 3

Name: Invoice 3

Date Created: 12/05/2017

Payment: cash

============================

Find Invoices by Last Week

Id: 5

Name: Invoice 5

Date Created: 12/01/2017

Payment: cash

============================

Find Invoices by Last 7 Days

Id: 1

Name: Invoice 1

Date Created: 12/08/2017

Payment: cash

============================

Id: 2

Name: Invoice 2

Date Created: 12/08/2017

Payment: cash

============================

Id: 3

Name: Invoice 3

Date Created: 12/05/2017

Payment: cash

============================

Id: 5

Name: Invoice 5

Date Created: 12/01/2017

Payment: cash

============================

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Find By Month \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Find Invoices by This Month

Id: 1

Name: Invoice 1

Date Created: 12/08/2017

Payment: cash

============================

Id: 2

Name: Invoice 2

Date Created: 12/08/2017

Payment: cash

============================

Id: 3

Name: Invoice 3

Date Created: 12/05/2017

Payment: cash

============================

Id: 5

Name: Invoice 5

Date Created: 12/01/2017

Payment: cash

============================

Find Invoices by Last Month

Id: 4

Name: Invoice 4

Date Created: 11/01/2017

Payment: cash

============================

Find Invoices by Last 30 Days

Id: 1

Name: Invoice 1

Date Created: 12/08/2017

Payment: cash

============================

Id: 2

Name: Invoice 2

Date Created: 12/08/2017

Payment: cash

============================

Id: 3

Name: Invoice 3

Date Created: 12/05/2017

Payment: cash

============================

Id: 5

Name: Invoice 5

Date Created: 12/01/2017

Payment: cash

============================

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Find By Quarter \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Find Invoices by This Quarter

Id: 1

Name: Invoice 1

Date Created: 12/08/2017

Payment: cash

============================

Id: 2

Name: Invoice 2

Date Created: 12/08/2017

Payment: cash

============================

Id: 3

Name: Invoice 3

Date Created: 12/05/2017

Payment: cash

============================

Id: 4

Name: Invoice 4

Date Created: 11/01/2017

Payment: cash

============================

Id: 5

Name: Invoice 5

Date Created: 12/01/2017

Payment: cash

============================

Find Invoices by Last Quarter

Id: 1

Name: Invoice 1

Date Created: 12/08/2017

Payment: cash

============================

Id: 2

Name: Invoice 2

Date Created: 12/08/2017

Payment: cash

============================

Id: 3

Name: Invoice 3

Date Created: 12/05/2017

Payment: cash

============================

Id: 4

Name: Invoice 4

Date Created: 11/01/2017

Payment: cash

============================

Id: 5

Name: Invoice 5

Date Created: 12/01/2017

Payment: cash

============================

Id: 6

Name: Invoice 6

Date Created: 08/09/2017

Payment: cash

============================

Find Invoices by Last 90 Days

Id: 1

Name: Invoice 1

Date Created: 12/08/2017

Payment: cash

============================

Id: 2

Name: Invoice 2

Date Created: 12/08/2017

Payment: cash

============================

Id: 3

Name: Invoice 3

Date Created: 12/05/2017

Payment: cash

============================

Id: 4

Name: Invoice 4

Date Created: 11/01/2017

Payment: cash

============================

Id: 5

Name: Invoice 5

Date Created: 12/01/2017

Payment: cash

============================

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Find By Year \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Find Invoices by This Year

Id: 1

Name: Invoice 1

Date Created: 12/08/2017

Payment: cash

============================

Id: 2

Name: Invoice 2

Date Created: 12/08/2017

Payment: cash

============================

Id: 3

Name: Invoice 3

Date Created: 12/05/2017

Payment: cash

============================

Id: 4

Name: Invoice 4

Date Created: 11/01/2017

Payment: cash

============================

Id: 5

Name: Invoice 5

Date Created: 12/01/2017

Payment: cash

============================

Id: 6

Name: Invoice 6

Date Created: 08/09/2017

Payment: cash

============================

Find Invoices by Last Year

Id: 1

Name: Invoice 1

Date Created: 12/08/2017

Payment: cash

============================

Id: 2

Name: Invoice 2

Date Created: 12/08/2017

Payment: cash

============================

Id: 3

Name: Invoice 3

Date Created: 12/05/2017

Payment: cash

============================

Id: 4

Name: Invoice 4

Date Created: 11/01/2017

Payment: cash

============================

Id: 5

Name: Invoice 5

Date Created: 12/01/2017

Payment: cash

============================

Id: 6

Name: Invoice 6

Date Created: 08/09/2017

Payment: cash

============================

Find Invoices by Last 365 Days

Id: 1

Name: Invoice 1

Date Created: 12/08/2017

Payment: cash

============================

Id: 2

Name: Invoice 2

Date Created: 12/08/2017

Payment: cash

============================

Id: 3

Name: Invoice 3

Date Created: 12/05/2017

Payment: cash

============================

Id: 4

Name: Invoice 4

Date Created: 11/01/2017

Payment: cash

============================

Id: 5

Name: Invoice 5

Date Created: 12/01/2017

Payment: cash

============================

Id: 6

Name: Invoice 6

Date Created: 08/09/2017

Payment: cash

========================