

Hibernate is a framework that provides some abstraction layer meaning that the programmer does not have to worry about the implementations, Hibernate does the implementations for you internally like Establishing a connection with the database, writing queries to perform CRUD operations, etc. In this article, let us see a Hibernate Example using XML along with MySQL database in eclipse.

Requirements:

- Eclipse
- Maven
- Hibernate
- MySQL
- JDK 6 onwards

Example

As we are going to check about the maven kind of project, let us see the **pom.xml**

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/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.geeksforgeeks</groupId>
  <artifactId>HibernateSampleExample</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>HibernateSampleExample</name>
  <url>http://maven.apache.org</url>
  <dependencies>
```

```

    </artifactId>hibernate-core</artifactId>
    <version>4.3.5.Final</version>
  </dependency>
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.6</version>
  </dependency>

</dependencies>

<build>
  <pluginManagement>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.1</version>
        <configuration>
          <source>1.7</source>
          <target>1.7</target>
        </configuration>
      </plugin>
    </plugins>
  </pluginManagement>
</build>
</project>

```

For Hibernate, we should keep the details on the “**hibernate.cfg.xml**” file which specifies what kind of database and its credentials, its required drivers, etc., are placed. In this example, we are going to use MySQL and hence

The advantage of hibernate is it will create a mapping of a database table with a Java application class file. That is also specified in an XML file. Let us create a table in MySQL first

```
-- Here "geeksforgeeks" is the name of the database
-- "GeekUserDetails" is the name of the table
-- geekUserId is the Primary Key
CREATE TABLE geeksforgeeks.GeekUserDetails (
    geekUserId INT (5) NOT NULL,
    geekUsername VARCHAR (20) NOT NULL,
    numberOfPosts INT(5) NOT NULL,
    CREATED_BY VARCHAR (20) NOT NULL,
    CREATED_DATE DATE NOT NULL,
    PRIMARY KEY ( geekUserId )
)
```

Let us see the mapping file in Hibernate. i.e. each and every column has to be mapped between a table and class. First, let us create POJO(Model class) in Java for that

Java

```
import java.util.Date;
// We can check that column name in "GeekUserDetails" is
// matching with each and every field here. It is always good
// to have the same column name and field name here
public class GeekUserDetails {
    private int geekUserId;
    private String geekUsername;
    private int numberOfPosts;
    public int getNumberOfPosts() { return numberOfPosts; }

    public int getGeekUserId() { return geekUserId; }
```

```

    }
    this.geekUsername = geekUsername;
}

public void setNumberOfPosts(int numberOfPosts)
{
    this.numberOfPosts = numberOfPosts;
}

private String createdBy;
private Date createdAt;

public String getCreatedBy() { return createdBy; }

public void setCreatedBy(String createdBy)
{
    this.createdBy = createdBy;
}

public Date getCreatedAt() { return createdAt; }

public void setCreatedAt(Date createdAt)
{
    this.createdAt = createdAt;
}
}

```

Now, the mapping file related to the POJO file.

resources/geekuser.hbm.xml

XML

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <!-- com.geeksforgeeks.GeekUserDetails

```

Hibernate Example using XML in Eclipse - GeeksforGeeks
 </property>
 <property name="numberOfPosts" type="int">
 <column name="numberOfPosts"/>
 </property>
 <property name="createdBy">
 <column name="CREATED_BY"/>
 </property>
 <property name="createdDate" type="date">
 <column name="CREATED_DATE"/>
 </property>
 </class>
 </hibernate-mapping>

Now, let us see the main configuration file

resources/hibernate.cfg.xml

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">
<!-- Main configuration file -->
<hibernate-configuration>
  <session-factory>
    <!-- As we are connecting MySQL, com.mysql.jdbc.
         Driver is required(JDBC driver class) -->
    <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver<
    <!-- Here geeksforgeeks is the name of the database
         that we are connecting(JDBC URL) -->
    <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/geek
    <!-- Username to connect to MySQL -->
    <property name="hibernate.connection.username">root</property>
    <!-- Password to connect to MySQL Provide your correct password -->
  
```

[Java Arrays](#)
[Java Strings](#)
[Java OOPS](#)
[Java Collection](#)
[Java 8 Tutorial](#)
[Java Multithreading](#)
[Java Exception H](#)

```

    <!-- We need to provide the exact mapping file
         which we have created earlier -->
    <mapping resource="geekuser.hbm.xml" />
</session-factory>
</hibernate-configuration>

```

After creating the mapping between the MySQL table and Java class via “geekuser.hbm.xml” and “hibernate.cfg.xml”, let us try to do a simple insertion of a record in the table. Let us try to run that via a java application file. We need to look upon certain files like HibernateUtil.java. We need to create the SessionFactory from hibernate.cfg.xml. Hence first all the entries in this XML have to be satisfied before entering into the main code. Otherwise need to provide the required code to throw the exception.

Java

```

import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {
    private static final SessionFactory sessionFactory
        = buildSessionFactory();
    private static SessionFactory buildSessionFactory()
    {
        try {
            // We need to create the SessionFactory from
            // hibernate.cfg.xml
            return new Configuration()
                .configure()
                .buildSessionFactory();
        }
        catch (Throwable ex) {
            // Make sure you log the exception, as it might
            // be swallowed

```

```

    }
    return sessionFactory;
}

public static void shutdown()
{
    // Close caches and connection pools
    getSessionFactory().close();
}
}

```

GeekUserDetailsTest.java -> Session is getting created by using HibernateUtil.java. Hibernate SessionFactory has three methods namely `getCurrentSession()`, `openSession()` and `openStatelessSession()`. In our code, we are using `openSession()`. If it is not given, we will get into exception as Exception in thread "main" org.hibernate.HibernateException: No CurrentSessionContext configured! For `openSession()`, it will always open a new session and it has to be closed

Java

```

import java.util.Date;

import org.hibernate.Session;

public class GeekUserDetailsTest {
    public static void main(String[] args) {

        // open the session
        Session session = HibernateUtil.getSessionFactory().openSession();

        // For doing any CRUD operation,
        // let us start a transaction
        session.beginTransaction();

        // Create an object of GeekUserDetails
        GeekUserDetails geekUser = new GeekUserDetails();
    }
}

```

Just a save statement is enough which automatically creates an insert statement

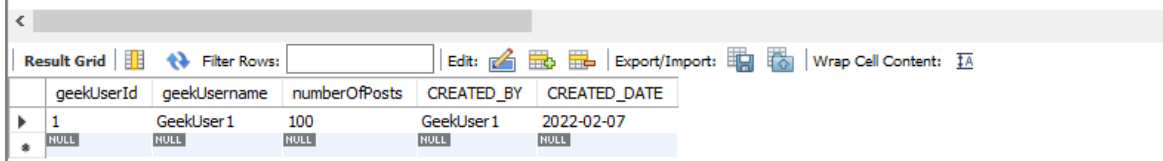
```
// Just a save statement is enough which
// automatically creates an insert statement
session.save(geekUser);

// commit will help to complete
// the changes in the table
session.getTransaction().commit();

// close the session
session.close();
}
```

Once this file is run as a “Java application”, we can able to see a record is inserted into the “GeekUserDetails” table

```
34 SELECT * FROM geeksforgeeks.GeekUserDetails;
35
```



The screenshot shows a database result grid with the following data:

	geekUserId	geekUsername	numberOfPosts	CREATED_BY	CREATED_DATE
1	1	GeekUser1	100	GeekUser 1	2022-02-07
*	NULL	NULL	NULL	NULL	NULL

Video explanation about the code:

Feeling lost in the vast world of Backend Development? It's time for a change!

Join our [Java Backend Development - Live Course](#) and embark on an exciting journey to master backend development efficiently and on schedule.

What We Offer:

- Comprehensive Course
- Expert Guidance for Efficient Learning
- Hands-on Experience with Real-world Projects
- Proven Track Record with 100,000+ Successful Geeks



priya...

3

Previous Article

Hibernate - Annotations

Next Article

Hibernate - Create Hibernate Configuration File with the Help of Plugin

Similar Reads

Hibernate - Table Per Subclass Example using XML File

In Table Per Subclass, subclass tables are mapped to the Parent class table by primary key and foreign key relationship. In a Table per Subclass strategy : For...

5 min read

Eclipse/STS IDE is generally used for developing application and what happens is whenever we are creating a simple Maven project and if the web.xml...

2 min read

Hibernate - Table Per Concrete Class using XML File

Hibernate is capable of storing the inherited properties of an object along with its new properties in its database when an object is saved in the database. In...

5 min read

Hibernate - Table Per Hierarchy using XML File

Hibernate is capable of storing the inherited properties of an object along with its new properties in its database when an object is saved in the database. In...

6 min read

[View More Articles](#)

Article Tags :

[Java-Hibernate](#)

[Java](#)

Practice Tags :

[Java](#)



Company

About Us
Legal
Careers
In Media
Contact Us
Advertise with us
GFG Corporate Solution
Placement Training Program

Languages

Python
Java
C++
PHP
GoLang
SQL
R Language
Android Tutorial

Data Science & ML

Data Science With Python
Data Science For Beginner
Machine Learning Tutorial
ML Maths
Data Visualisation Tutorial
Pandas Tutorial

Explore

Job-A-Thon Hiring Challenge
Hack-A-Thon
GfG Weekly Contest
Offline Classes (Delhi/NCR)
DSA in JAVA/C++
Master System Design
Master CP
GeeksforGeeks Videos
Geeks Community

DSA

Data Structures
Algorithms
DSA for Beginners
Basic DSA Problems
DSA Roadmap
DSA Interview Questions
Competitive Programming

Web Technologies

HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS

Python Tutorial
Web Scraping
OpenCV Tutorial
Python Interview Question

DevOps

Git
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

School Subjects

Mathematics
Physics
Chemistry
Biology
Social Science
English Grammar

Databases

SQL
MYSQL
PostgreSQL
PL/SQL
MongoDB

Competitive Exams

JEE Advanced
UGC NET
UPSC
SSC CGL
SBI PO
SBI Clerk
IBPS PO

Database Management System
Software Engineering
Digital Logic Design
Engineering Maths

System Design

High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns
OOAD
System Design Bootcamp
Interview Questions

Commerce

Accountancy
Business Studies
Economics
Management
HR Management
Finance
Income Tax

Preparation Corner

Company-Wise Recruitment Process
Resume Templates
Aptitude Preparation
Puzzles
Company-Wise Preparation
Companies
Colleges

More Tutorials

Software Development
Software Testing
Product Management
Project Management
Linux
Excel
All Cheat Sheets

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved