



Share on your Social Media

8681884318
www.softlogicsys.in
enquiry@softlogicsys.in

Hibernate Tutorial for beginners

Published On: September 17, 2024

Hibernate Tutorial for Java Aspirants

A Java framework called Hibernate makes it easier to create Java applications that communicate with databases. Learn what is required in the industries to begin a [career in Java development](#) in this Hibernate tutorial.

[Download Hibernate Tutorial PDF](#)

Introduction to Hibernate

Hibernate is a lightweight, open-source ORM tool (Object Relational Mapping) for Java development. For data persistence, Hibernate implements the JPA (Java Persistence API) features. In this **Hibernate tutorial**, you will learn the following:

- Overview of Hibernate
- Hibernate Architecture
- Web Application with Hibernate
- Generator Classes in Hibernate
- Caching in Hibernate
- Hibernate Configuration

Overview of Hibernate

For web apps, object-oriented domain models can be mapped to relational databases using the Hibernate framework, an open-source object-relational mapping (ORM) tool. Based on object containerization and the abstraction that offers that capability, object-relational mapping works.

ORM: Data creation, manipulation, and access are all made easier with the use of an ORM tool. It is a method of programming that associates an object with database data. Internally, the ORM tool communicates with the database via the JDBC API.

JPA: A Java specification called **Java Persistence API** (JPA) gives ORM tools some standardization and capabilities. The JPA classes and interfaces are included in the javax.persistence package.

[Hibernate Interview Questions and Answers](#)

Advantages of using the Hibernate framework:

Here are some of the advantages of the hibernate framework:

- **LOPL license:** Hibernate is open-source and lightweight.
- **Quick Performance:** The hibernate framework's quick performance is a result of its internal use of caching.



Featured Articles

Want to know more about becoming an expert in IT?

[Click Here to Get Started >>](#)

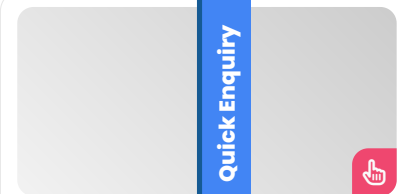
100% Placement
Assurance

AUTHORISED
CERTIFICATION
PARTNER **IBM**

Related Courses at SLA

- [Hibernate Online Training](#)
- [Hibernate Training in OMR](#)
- [Hibernate Train in Chennai](#)

Related Posts



Hibernate Project Topics

Published On: October 14, 2024

Introduction A Hibernate Professional specializes in the Hibernate ORM framework for Java, managing database interactions,...



Tableau Developer Salary in Chennai

Published On: October 12, 2024

Introduction A Tableau Developer designs, develops, and maintains dashboards and visualizations using Tableau software. Key...



VMware Tutorial for Cloud Computing Aspirants

Published On: October 12, 2024

VMware Tutorial for Cloud Computing Aspirants VMware software allows you to run a virtual machine...

Quick Enquiry



- **Database Independent Query:** The object-oriented variant of **SQL** is called HQL, or Hibernate Query Language.
- **Automatic Table Creation:** The Hibernate framework offers the ability to automatically construct database tables.
- **Simplifies Complex Join:** With the hibernate framework, retrieving data from several tables is simple.
- **Gives Statistics About Query and Database State:** Hibernate facilitates query caching and furnishes statistics regarding query and database state.

Hibernate Architecture

Numerous items, including persistent objects, session factories, transaction factories, connection factories, sessions, and transactions, are part of the Hibernate architecture.

There are four layers to the hibernate architecture.

- Java application layer
- Hibernate framework layer
- Backhand API layer
- Database layer

Understanding the components of the Hibernate architecture is necessary before developing the first Hibernate application. They are listed in the following order:

SessionFactory: The SessionFactory is a ConnectionProvider session and client factory. It stores an optional second-level data cache. The "org.hibernate.SessionFactory" interface will be implemented here.

Session: An interface between the application and database-stored data is provided by the session object. It encapsulates the JDBC connection and has a brief lifespan. Inserting, updating, and deleting the object is possible using the "org.hibernate.Session" interface.

Transaction: The atomic unit of work is specified by the transaction object. It's not required. The "org.hibernate.Transaction" interface offers methods for transaction management.

Connection Provider: It is a JDBC connection manufacturer. The application is abstracted from DriverManager or DataSource by it.

TransactionFactory: It functions as a transaction factory.

[Hibernate Syllabus PDF](#)

Web Application with Hibernate

Here, we'll use Hibernate to develop a web application. We are using the DAO class for database codes, the Bean class for data representation, and JSP for presentation logic to create the **web application**.

We are going to add the user's record to the database in this example. It is merely a form for registration.

Index.jsp

The register.jsp file receives user input from this page using the post function.

```
<form action="register.jsp" method="post">

Name:<input type="text" name="name"/><br><br/>

Password:<input type="password" name="password"/><br><br/>

Email ID:<input type="text" name="email"/><br><br/>

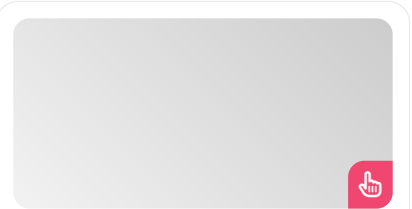
<input type="submit" value="register"/>

</form>
```

register.jsp

All request parameters are obtained by this file, which then stores the data in a User class object. Additionally, it passes the User class object to the UserDao class's register method.

```
<%@page import="com.sla.mypack.UserDao"%>
```



VBA Macros Tutorial for Beginners

Published On: October 10, 2024

VBA Macros Tutorial for Beginners VBA macros are programs that automate repetitive operations in Microsoft...

```

<jsp:useBean id="obj" class="com.sla.mypack.User">

</jsp:useBean>

<jsp:setProperty property="*" name="obj"/>

<% int i=UserDao.register(obj);

if(i>0)

out.print("You are successfully registered");

%>

```

User.java

The Persistent class in Hibernate is represented by this straightforward bean class.

```

package com.sla.mypack;

public class User {

private int id;

private String name,password,email;

}

```

user.hbm.xml

It maps the database table with the User class.

```

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

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 5.3//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-5.3.dtd">

<hibernate-mapping>

<class name="com.sla.mypack.User" table="u400">

<id name="id">

<generator class="increment"></generator>

</id>

<property name="name"></property>

<property name="password"></property>

<property name="email"></property>

</class>

</hibernate-mapping>

```

UserDao.java

A Dao class that has a method for keeping User class instances.

```

package com.sla.mypack;

```

```

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.boot.Metadata;

import org.hibernate.boot.MetadataSources;

import org.hibernate.boot.registry.StandardServiceRegistry;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

public class UserDao {

    public static int register(User u){

        int i=0;

        StandardServiceRegistry ssr = new
        StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();

        Metadata meta = new MetadataSources(ssr).getMetadataBuilder().build();

        SessionFactory factory = meta.getSessionFactoryBuilder().build();

        Session session = factory.openSession();

        Transaction t = session.beginTransaction();

        i=(Integer)session.save(u);

        t.commit();

        session.close();

        return i;

    }

}

```

hibernate.cfg.xml

It is a configuration file that holds details about the mapping file and database.

```

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

<!DOCTYPE hibernate-configuration PUBLIC

    "-//Hibernate/Hibernate Configuration DTD 5.3//EN"

    "http://hibernate.sourceforge.net/hibernate-configuration-5.3.dtd">

<hibernate-configuration>

<session-factory>

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

<property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>

```

```

<property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>

<property name="connection.username">system</property>

<property name="connection.password">jtp</property>

<property
name="connection.driver_class">oracle.jdbc.driver.OracleDriver</property>

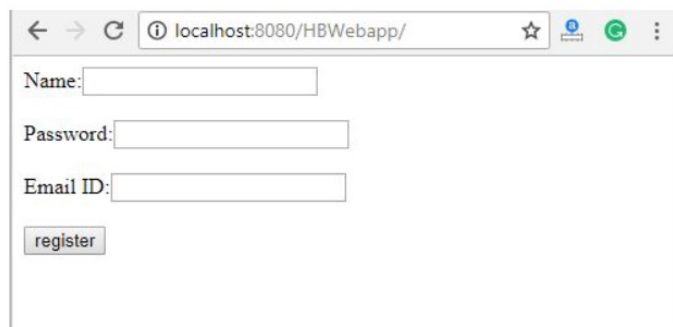
<mapping resource="user.hbm.xml"/>

</session-factory>

</hibernate-configuration>

```

Output



Hibernate Tutorial

[Hibernate Salary](#)

Generator Classes in Hibernate

One of the components of id is the <generator> class. It is employed in the creation of the persistent class objects' unique identifiers. The Hibernate Framework defines numerous generator classes.

- The "org.hibernate.id.IdentifierGenerator" interface is implemented by every generator class.
- An application programmer can implement the IdentifierGenerator interface to develop their generator classes.

The Hibernate framework contains various built-in generator classes:

- assigned
- increment
- sequence
- hilo
- native
- identity
- seqhilo
- uuid
- guid
- select
- foreign
- sequence-identity

Assigned

If there isn't a <generator> element, this is the generator strategy that runs by default. The program in this instance assigns the ID. Example:

....

```
<hibernate-mapping>
```

```
<class ...>
```

```
<id ...>
```

```
<generator class="assigned"></generator>
```

```
</id>
```

```
.....
```

```
</class>
```

```
</hibernate-mapping>
```

Increment

Only when no other process is attempting to insert data into this table does it generate the unique ID.

- It produces a type identifier that is short, int, or long.
- If a table has an identification, the program takes into account its maximum value; if not, it assumes that the first created identifier is 1.
- The hibernate increments the identification by one for every value of an attribute.

Syntax:

```
....
```

```
<hibernate-mapping>
```

```
<class ...>
```

```
<id ...>
```

```
<generator class="increment"></generator>
```

```
</id>
```

```
.....
```

```
</class>
```

```
</hibernate-mapping>
```

Sequence

It makes use of the database's sequence.

- If no sequence is defined, one is automatically created;
- If an Oracle database, the sequence HIBERNATE_SEQUENCE is created.
- A sequence is used in cases of Oracle, DB2, SAP DB, PostgreSQL, and McKoi; however, the generator is used in InterBase cases.

```
.....
```

```
<id ...>
```

```
<generator class="sequence"></generator>
```

```
</id>
```

```
.....
```

Make use of the generator's param subelement to define your sequence.

```
.....
```

```
<id ...>
```

```
<generator class="sequence">
```

```

        <param name="sequence">your_sequence_name</param>

    </generator>

</id>

.....

```

Hilo

It generates the id of types short, int, and long using the high and low methods.

Syntax:

```

.....

<id ...>

    <generator class="hilo"></generator>

</id>

.....

```

Native

Identical, sequential, or hilo is used, based on the database provider.

Syntax:

```

.....

<id ...>

    <generator class="native"></generator>

</id>

.....

```

Identity

It is utilized to support the id column in Sybase, **MySQL**, MS SQL Server, DB2, and HypersonicSQL. The type of returned ID is either long, int, or short. The database is in charge of creating unique identifiers.

Sequilo

It applies the high-low algorithm to the given sequence name. The type of returned ID is either long, int, or short.

UUID

A 128-bit UUID approach is utilized to generate the ID. Because IP is used, the returned ID is unique within a network and is of type String. The 32-digit UUID is represented in hexadecimal. GUID

It takes advantage of a GUID that a string-type database created. It functions with MySQL and MS SQL Server.

Select

It makes use of the main key that the database trigger returned.

Foreign

When using <one-to-one> association, it typically utilizes the id of another related object.

Sequence-identity

It uses a specific sequence generation approach. Only Oracle 10g drivers support it.

Caching in Hibernate

Hibernate caching increases the application's performance by pooling the cached object. When we need to retrieve the same data more than once, it is helpful.

Two primary categories of caching exist:

First-level cache: The session object contains the data from the first-level cache. Normally, it is activated. The first-level cache data won't be accessible to the application as a whole. Applications can make use of several session objects.

Second Level Cache: The second level cache data is stored in the SessionFactory object. The entire program will have access to the data kept in the second-level cache. However, we must explicitly activate it.

- EH (Easy Hibernate) Cache
- Swarm Cache
- OS Cache
- JBoss Cache

Hibernate Configuration

Hibernate needs a wide range of configuration options because it can function in various environments. The mapping data that gives Java classes various functionality is contained in these settings.

- Typically, we include mappings in the configuration file that are relevant to the databases. Hibernate makes it easier to offer configurations in properties files (like `hibernate.properties`) or XML files (like `hibernate.cfg.xml`).
- Properties and mappings to apps can be specified using an instance of the Configuration class. Additionally, this class creates an immutable "SessionFactory."
- By simply instantiating the Configuration class and providing mappings in the configuration file, we can obtain the Configuration class instance. Use the `addResource()` method if the mapping files are present in the classpath.

Syntax:

```
Configuration cfg = new Configuration()

.addResource("employee.hbm.xml")
```

Example: XML-Based Configuration

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

<!DOCTYPE hibernate-configuration PUBLIC

    "-//Hibernate/Hibernate Configuration DTD 5.3//EN"

    "http://www.hibernate.org/dtd/hibernate-configuration-5.3.dtd">

<hibernate-configuration>

    <session-factory>

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

        <property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>

        <property
name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>

        <property name="connection.username">system</property>

        <property name="connection.password">jtp</property>

        <property
name="connection.driver_class">oracle.jdbc.driver.OracleDriver</property>

    </session-factory>
```


</hibernate-configuration>

Properties File Configuration

```
hibernate.dialect= org.hibernate.dialect.Oracle9Dialect

hibernate.connection.driver_class= oracle.jdbc.driver.OracleDriver

hibernate.connection.url= jdbc:oracle:thin:@localhost:1521:xe

hibernate.connection.username= system

hibernate.connection.password=jtp

hibernate.show_sql=true

hibernate.hbm2ddl=update
```

JDBC Properties in Hibernate

- **hibernate.connection.driver_class:** It stands for the class of JDBC drivers.
- **hibernate.connection.url:** It is an image of the JDBC URL.
- **hibernate.connection.username:** It stands for the username in the database.
- **hibernate.connection.password:** It stands for the password for the database.
- **hibernate.connection.pool_size:** It stands for the greatest number of connections that the connection pool can accommodate.

Conclusion

This hibernate tutorial covers the basic concepts of mapping and configuration. Gain expertise with other important concepts in our [hibernate training in Chennai](#).

Share on your Social Media



Softlogic Academy

Softlogic Systems

KK Nagar [Corporate Office]

No.10, PT Rajan Salai, K.K. Nagar, Chennai – 600 078.

Landmark: Karnataka Bank Building

Phone: [+91 86818 84318](tel:+918681884318)

Email: enquiry@softlogicsys.in

Map: [Google Maps Link](#)

OMR

No. E1-A10, RTS Food Street
92, Rajiv Gandhi Salai (OMR),
Navalur, Chennai – 600 130.

Landmark: Adj. to AGS Cinemas

Phone: [+91 89256 88858](tel:+918925688858)

Email: info@softlogicsys.in

Map: [Google Maps Link](#)

Navigation

[About Us](#)
[Blog Posts](#)
[Careers](#)
[Contact](#)
[Placement Training](#)
[Corporate Training](#)
[Hire With Us](#)
[Job Seekers](#)
[SLA's Recently Placed Students](#)
[Reviews](#)
[Sitemap](#)

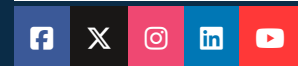
Important Links

[Disclaimer](#)
[Privacy Policy](#)
[Terms and Conditions](#)

Courses

[Python](#)
[Software Testing](#)
[Full Stack Developer](#)
[Java](#)
[Power BI](#)
[Clinical SAS](#)
[Data Science](#)
[Embedded](#)
[Cloud Computing](#)
[Hardware and Networking](#)
[VBA Macros](#)
[Mobile App Development](#)
[DevOps](#)

Social Media Links



Review Sources

[Google](#)
[Trustpilot](#)
[Glassdoor](#)
[Mouthshut](#)
[Sulekha](#)
[Justdial](#)
[Ambitionbox](#)
[Indeed](#)
[Software Suggest](#)
[Sitejabber](#)

