HIBERNATE

java persistence framework

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

Hibernate is a high-performance Object/Relational persistence and query service, which is licensed under the open source GNU Lesser General Public License (LGPL) and is free to download. Hibernate not only takes care of the mapping from Java classes to database tables (and from Java data types to SQL data types), but also provides data query and retrieval facilities.

This tutorial will teach you how to use Hibernate to develop your database based web applications in simple and easy steps.

Audience

This tutorial is designed for all those Java programmers who would like to understand the Hibernate framework and its API.

Prerequisites

We assume you have a good understanding of the Java programming language. A basic understanding of relational databases, JDBC, and SQL will be very helpful in understanding this tutorial.

Copyright & Disclaimer

© Copyright 2015 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of the contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness, or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com

tutorialspoint
SIMPLYEASYLEARNING

Table of Contents

About the Tutorial i

Audience i

Prerequisites i

Copyright & Disclaimer i

Table of Contents ii

1. HIBERNATE – ORM 1

What is JDBC? 1

Pros and Cons of JDBC 1

Why Object Relational Mapping (ORM)? 1

What is ORM? 3

Java ORM Frameworks 4

2. HIBERNATE – OVERVIEW 5

Hibernate Advantages 5

Supported Databases 6

3. HIBERNATE – ARCHITECTURE 7

Configuration Object 8

SessionFactory Object 8

Session Object 8

Transaction Object 9

Query Object 9

Criteria Object 9

4. HIBERNATE – ENVIRONMENT SETUP 10

Installing IDE 10

Downloading Hibernate 10

tutorialspoint
SIMPLYEASYLEARNING

Installing Hibernate 11

Verification 16

Adding Required jars to Hibernate 17

5. HIBERNATE – CONFIGURATION 22

Hibernate Properties 22

Hibernate with MySQL Database 23

6. HIBERNATE – SESSIONS 26

Session 26

Sample Code 26

Session Interface Methods 28

7. HIBERNATE – PERSISTENT CLASS 30

Simple POJO Example 30

8. HIBERNATE – MAPPING FILES 32

9. HIBERNATE – MAPPING TYPES 35

Primitive Types 35

Date and Time Types 35

Binary and Large Object Types 36

JDK-related Types 36

10. HIBERNATE – EXAMPLES 37

Example 37

Create POJO Classes 37

Create Mapping File (Employee.hbm.xml) 39

Create Configuration File 40

Persistence Operations 41

11. HIBERNATE – O/R MAPPINGS 48

tutorialspoint
SIMPLYEASYLEARNING

Collection Mappings 48

Hibernate – Set Mappings 49

Hibernate – SortedSet Mappings 59

Hibernate – List Mappings 71

Hibernate – Bag Mappings 81

Hibernate – Map Mappings 91

Hibernate – SortedMap Mappings 101

Association Mappings 113

Hibernate – Many-to-One Mappings 113

Hibernate – One-to-One Mappings 125

Hibernate – One-to-Many Mappings 136

Hibernate – Many-to-Many Mappings 146

Inheritance Mappings 157

Hibernate – Table Per Class Hierarchy 158

Hibernate Table Per Sub-class 164

Hibernate Table Per Concrete Class 171

Component Mappings 177

Hibernate – Component Mappings 177

12. HIBERNATE – ANNOTATIONS 189

Environment Setup for Hibernate Annotation 189

Annotated Class Example 189

@Entity Annotation 191

@Table Annotation 191

@Id and @GeneratedValue Annotations 191

@Column Annotation 192

Create Application Class 192

Database Configuration 195

tutorialspoint
SIMPLYEASYLEARNING

Compilation and Execution 196

13. HIBERNATE – QUERY LANGUAGE 198

FROM Clause 198

AS Clause 198

SELECT Clause 199

WHERE Clause 199

ORDER BY Clause 199

GROUP by Clause 200

Using Named Parameters 200

UPDATE Clause 200

DELETE Clause 201

INSERT Clause 201

Aggregate Methods 201

Pagination using Query 202

14. HIBERNATE – CRITERIA QUERIES 203

Restrictions with Criteria 203

Pagination Using Criteria 205

Sorting the Results 205

Projections & Aggregations 205

Criteria Queries Example 206

Compilation and Execution 212

15. HIBERNATE – NATIVE SQL 214

Scalar Queries 214

Entity Queries 214

Named SQL Queries 214

Native SQL Example 215

tutorialspoint
SIMPLYEASYLEARNING

Compilation and Execution 220

16. HIBERNATE – CACHING 222

First-level Cache 222

Second-level Cache 223

Query-level Cache 223

The Second Level Cache 223

Concurrency Strategies 223

Cache Provider 224

The Query-level Cache 227

17. HIBERNATE – BATCH PROCESSING 228

Batch Processing Example 229

Compilation and Execution 234

18. HIBERNATE – INTERCEPTORS 235

How to Use Interceptors? 236

Create POJO Classes 238

Create Database Tables 239

Create Mapping Configuration File 239

Create Application Class 240

Compilation and Execution 243

tutorialspoint
SIMPLYEASYLEARNING

1. HIBERNATE – ORM

Any enterprise application performs database operations by storing and retrieving vast amounts of data. Despite all the available technologies for storage management, application developers normally struggle to perform database operations efficiently.

Generally, Java developers use lots of code or proprietary framework to interact with the database. Therefore, it is advisable to use Object Relational Mapping (ORM) to reduct the burden of interacting with the database. ORM forms a bridge between object models (Java program) and relational models (database program) like JDBC.

What is JDBC?

JDBC stands for **Java Database Connectivity**. It provides a set of Java API for accessing the relational databases from Java program. These Java APIs enables Java programs to execute SQL statements and interact with any SQL compliant database.

JDBC provides a flexible architecture to write a database independent application that can run on different platforms and interact with different DBMS without any modification.

Pros and Cons of JDBC

Pros of JDBC	Cons of JDBC
Clean and simple SQL processing.	Complex if it is used in large projects.
Good performance with large data.	Large programming overhead.
Very good for small applications.	No encapsulation.
Simple syntax so easy to learn.	Hard to implement MVC concept. Query is DBMS specific.

Why Object Relational Mapping (ORM)?

When we work with an object-oriented system, there is a mismatch between the object model and the relational database table. RDBMSs represent data in a tabular format whereas object-oriented languages, such as Java or C# represent it as an interconnected graph of objects.

Example

One may encounter a few problems when direct interaction takes place between object models and relation database tables. The following example highlights the problems associated with traditional database application structure.

tutorialspoint
SIMPLYEASYLEARNING

Here we have a Java Class called **Employee** with proper constructors and associated public functions. We have a simple java entity class for an employee having fields (variables) such as id, first_name, last_name, and salary.

```
public class Employee {
   private int id;
   private String first_name;
   private String last_name;
   private int salary;

   public Employee() {}
   public Employee(String fname, String lname, int salary) {
      this.first_name = fname;
      this.last_name = lname;
      this.salary = salary;
   }
   public int getId() {
      return id;
   }
   public String getFirstName() {
      return first_name;
   }
   public String getLastName() {
      return last_name;
   }
   public int getSalary() {
      return salary;
   }
}
```

Consider the above object needs to be stored and retrieved using the following RDBMS table. : Following is the Create table statement for Employee class. Its field structure should be same as the given object model (Employee class).

```
create table EMPLOYEE (
   id INT NOT NULL auto_increment,
```


tutorialspoint
SIMPLYEASYLEARNING

```
   first_name VARCHAR(20) default NULL,
   last_name  VARCHAR(20) default NULL,
   salary     INT  default NULL,
   PRIMARY KEY (id)
);
```

Thee following problems may arise when direction interaction takes place between object models and relation database tables:

- First, what if we need to modify the design of our database after having developed a few pages or our application?

- Second, loading and storing objects in a relational database exposes us to the following five mismatch problems:

Mismatch	Description
Granularity	Sometimes you will have an object model, which has more classes than the number of corresponding tables in the database.
Inheritance	RDBMSs do not define anything similar to Inheritance, which is a natural paradigm in object-oriented programming languages.
Identity	An RDBMS defines exactly one notion of 'sameness': the primary key. Java, however, defines both object identity (a==b) and object equality (a.equals(b)).
Associations	Object-oriented languages represent associations using object references whereas an RDBMS represents an association as a foreign key column.
Navigation	The ways you access objects in Java and in RDBMS are fundamentally different.

The **O**bject-**R**elational **M**apping (ORM) is the solution to handle all the above impedance mismatches.

What is ORM?

ORM stands for **O**bject-**R**elational **M**apping (ORM) is a programming technique for converting data between relational databases and object oriented programming languages such as Java, C#, etc.An ORM system has the following advantages over plain JDBC:

S.N.	Advantages
1	Let’s business code access objects rather than DB tables.
2	Hides details of SQL queries from OO logic.

tutorialspoint
SIMPLYEASYLEARNING

3	Based on JDBC 'under the hood.'
4	No need to deal with the database implementation.
5	Entities based on business concepts rather than database structure.
6	Transaction management and automatic key generation.
7	Fast development of application.

An ORM solution consists of the following four entities:

S.N.	Solutions
1	An API to perform basic CRUD operations on objects of persistent classes.
2	A language or API to specify queries that refer to classes and properties of classes.
3	A configurable facility for specifying mapping metadata.
4	A technique to interact with transactional objects to perform dirty checking, lazy association fetching, and other optimization functions.

Java ORM Frameworks

There are several persistent frameworks and ORM options in Java. A persistent framework is an ORM service that stores and retrieves objects into a relational database.

- Enterprise JavaBeans Entity Beans
- Java Data Objects
- Castor
- TopLink
- Spring DAO
- Hibernate, and many more

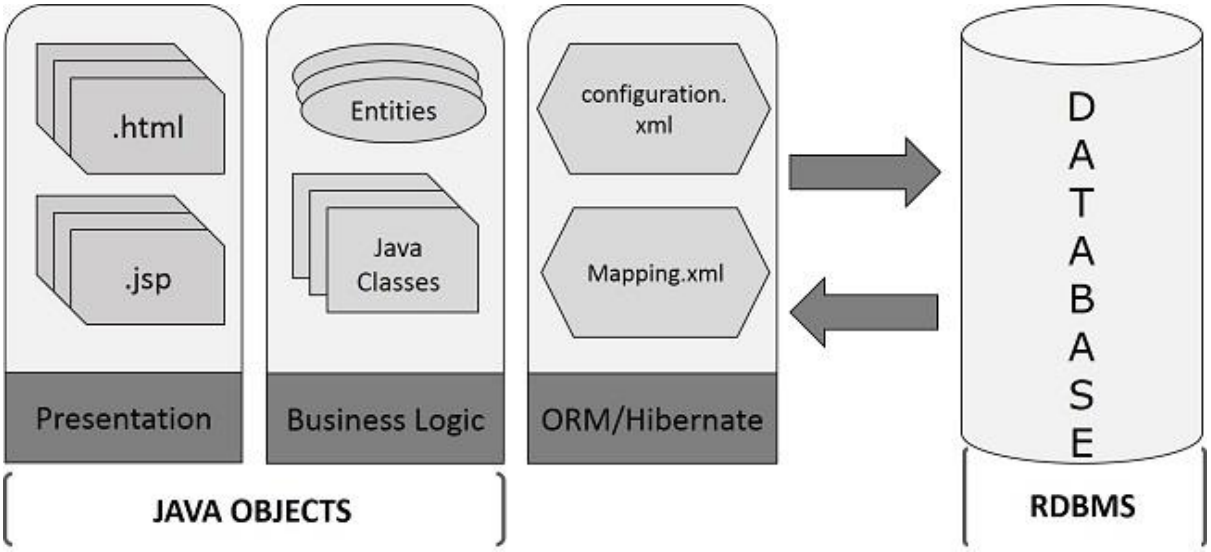
tutorialspoint
SIMPLYEASYLEARNING

2. HIBERNATE – OVERVIEW

Hibernate is an **O**bject-**R**elational **M**apping(ORM) solution for JAVA. It is an open source persistent framework created by Gavin King in 2001. It is a powerful, high performance Object-Relational Persistence and Query service for any Java Application.

Hibernate maps Java classes to database tables and from Java data types to SQL data types and relieves the developer from 95% of common data persistence related programming tasks.

Hibernate sits between traditional Java objects and database server to handle all the works in persisting those objects based on the appropriate O/R mechanisms and patterns.



Hibernate Advantages

Here we have listed down the advantages of using Hibernate:

- Hibernate takes care of mapping Java classes to database tables using XML files and without writing any line of code. If there is a change in the database or in any table, then all that you need to change are the XML file properties.
- Provides simple APIs (classes and methods) for storing and retrieving Java objects directly to and from the database.
- Hibernate supports **Inheritance**, **Association relations**, and **Collections**.
- Abstracts away the unfamiliar SQL types and provides a way to work around familiar Java Objects.
- Hibernate does not require an application server to operate.

tutorialspoint
SIMPLYEASYLEARNING

- Hibernate Supports only unchecked exceptions, so no need to write try, catch, or throws blocks. Generally we have a Hibernate translator which converts Checked exceptions to Unchecked.
- Minimizes database access with smart fetching strategies.
- Hibernate has its own query language. That is **Hiberate Query Language** (HQL) which contains database independent controlers.
- Manipulates Complex associations of objects of your database.
- Hibernate supports caching mechanism: It reduces the number of round trips (transactions) between an application and the database. It increases the application performance.
- Hibernate supports annotations, apart from XML..

Supported Databases

Hibernate supports almost all the major RDBMS database servers. Following is a list of few of the database engines that Hibernate supports:

- HSQL Database Engine
- DB2/NT
- MySQL
- PostgreSQL
- FrontBase
- Oracle
- Microsoft SQL Server Database
- Sybase SQL Server
- Informix Dynamic Server

Hibernate supports a variety of other technologies as well including:

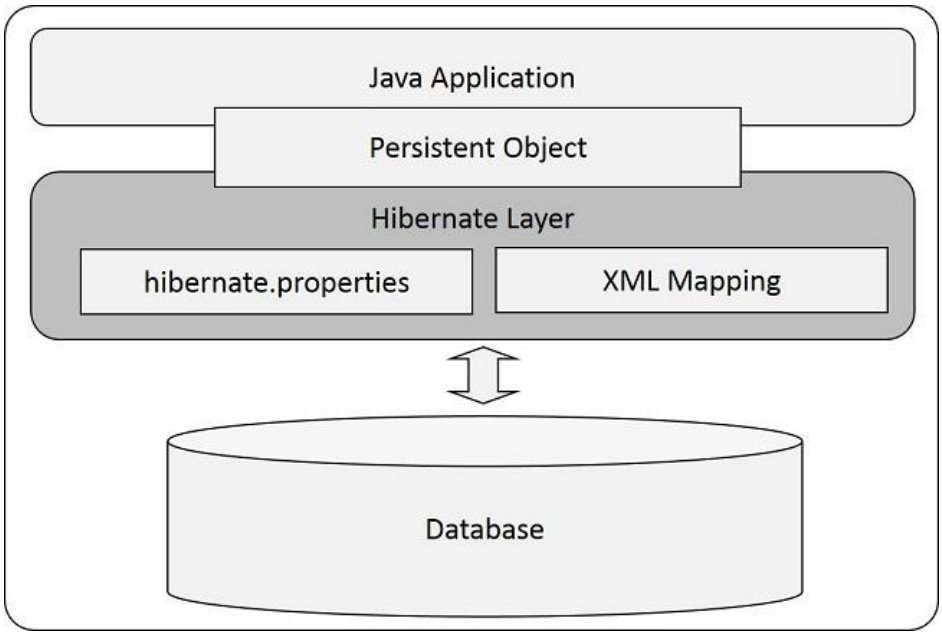
- XDoclet Spring
- J2EE
- Eclipse plug-ins
- Maven

tutorialspoint
SIMPLYEASYLEARNING

3. HIBERNATE – ARCHITECTURE

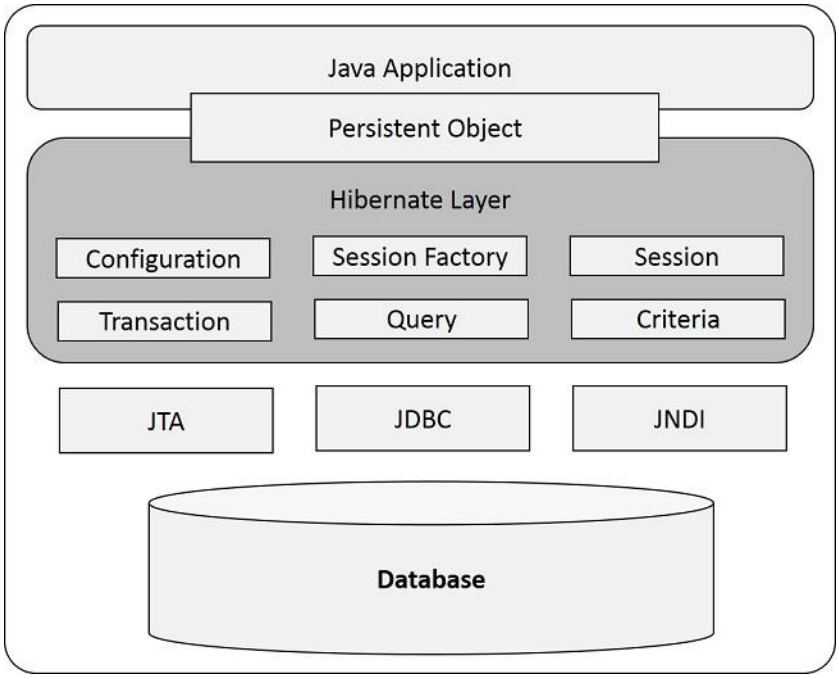
Hibernate has a layered architecture which helps the user to operate without having to know the underlying APIs. Hibernate makes use of the database and configuration data to provide persistence services (and persistent objects) to the application.

Following is a very high level view of the Hibernate Application Architecture.



Following is a detailed view of the Hibernate Application Architecture with a few important core classes in the Hibernate layer.

tutorialspoint
SIMPLYEASYLEARNING



Hibernate uses various existing Java APIs, like JDBC, Java Transaction API (JTA), and Java Naming and Directory Interface (JNDI). JDBC provides a rudimentary level of abstraction of functionality common to relational databases, allowing almost any database with a JDBC driver to be supported by Hibernate. JNDI and JTA support Hibernate to be integrated with J2EE application servers.

The following section gives a brief description about the class objects which are involved in the Hibernate Layer of the given architecture diagram. This sectiong gives you a theritical idea of how the hibernate class objects are used to build an application.

Configuration Object

Configuration is a serializable class. It is the first Hibernate object that you need to create in any Hibernate application. It is usually created only once during application initialization. It allows the application to specify properties and mapping documents to be used. The Configuration object provides two keys components:

- **Database Connection:** A database connection is most important for Enterprise and Database applications. It is handled through one or more configuration files supported by Hibernate. Those are **hibernate.properties** file and **hibernate.cfg.xml** file.
- **Mapping Setup:** This component creates the connection between the Java classes and database tables. It creates mapping between each entity java class and each table in the database.

tutorialspoint
SIMPLYEASYLEARNING

SessionFactory Object

SessionFactory is a Factory Interface used to create **Session** instances. After adding the properties and Mapping files to the Configuration object, it is used to create a SessionFactory object which in turn configures Hibernate (Front-end javaclasses and Back-end tables) for the application. SessionFactory is a thread-safe object and used by all the threads of an application. It is a heavyweight object, usually created during application start-up and kept for later use. You would need one SessionFactory object per database using a separate configuration file. So, if you are using multiple databases, then you would have to create multiple SessionFactory objects.

Session Object

Session is an Interface that wraps the JDBC connection. That means, it creates a physical connection between the application and a database. The Session object is lightweight and designed to be instantiated each time an interaction is needed with the database. Persistent objects are saved and retrieved through a Session object.

The Lifecycle of a Session is bounded by the beginning and end of a logical trasaction. It contains three states:

- **transient:** never persistent, currently not associated with any Session.
- **persistent:** currently associated with unique Session.
- **detached:** previously persistent, currently not associated with any Session.

The session objects should not be kept open for a long time because they are not usually thread-safe. They should be created and destroyed them as needed.

Transaction Object

Transaction is an Interface and it represents a unit of work with the database and most of the RDBMS supports transaction functionality. Transactions in Hibernate are handled by an underlying transaction manager.

This is an optional object and Hibernate applications may choose not to use this interface, instead managing transactions in their own application code.

Query Object

Query is an interface and it is used in **SQL** or Hibernate Query Language (**HQL**) string to retrieve data from the database and create objects. A Query instance is used to bind query parameters, limit the number of results returned by the query, and finally to execute the query.

tutorialspoint
SIMPLYEASYLEARNING

Criteria Object

Criteria is an interface and it is used for retrieving entity data by composing Criterion (Interface) objects. Criterion Objects work like a condition **(WHERE and IF)** in the SQL query, all the criterion objects (conditions) are added to the Criteria Object and that object will be executed and used for retriving entity data in objects.

tutorialspoint
SIMPLYEASYLEARNING

4. HIBERNATE – ENVIRONMENT SETUP

This chapter explains how to install Hibernate and other associated packages to prepare a develop environment for Hibernate applications. We will work with MySQL database for Hibernate examples, so make sure you already have setup for MySQL database. For more detail on MySQL, you can check our MySQL Tutorial.

We recommend that you use an IDE for Hibernate programming. We are using Eclipse IDE for further explanation in this tutorial.

Installing IDE

Follow the steps given below to download and install Eclipse IDE.

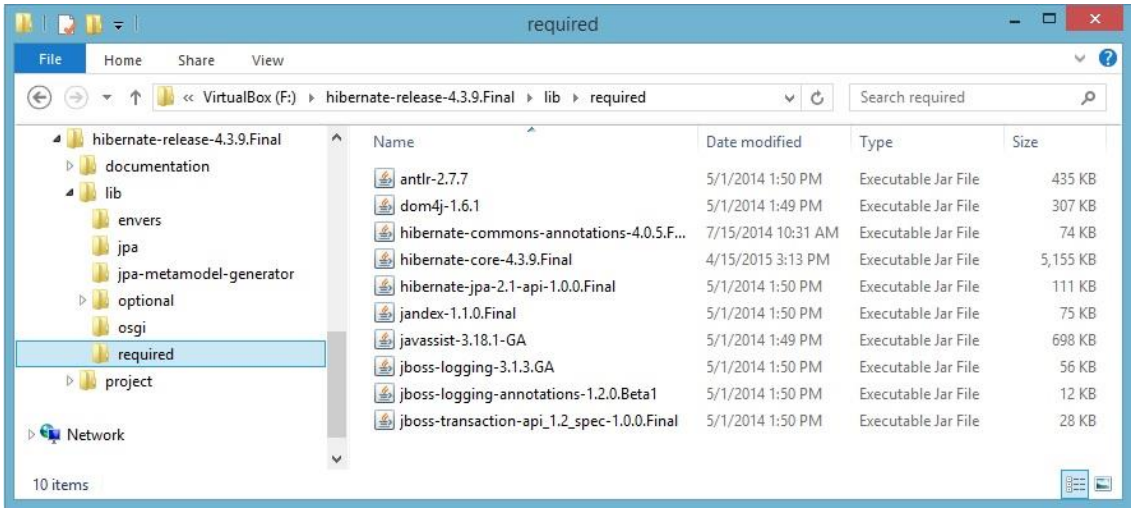
- Download the latest version of eclipse **here**.
- We are following **eclipse-jee-luna-R-win32-x86_64.zip** for this tutorial.

Downloading Hibernate

Make sure you already have the latest version of Java installed on your machine. Following are the simple steps to download Hibernate on your machine.

- Make a choice whether you want to install Hibernate on Windows or Unix. Then proceed to the next step to download either the .zip file for windows or the .tz file for Unix.
- Download the latest version of Hibernate from **here**.
- At the time of writing this tutorial, I downloaded **hibernate-release-4.3.9.Final**. When you unzip the downloaded file, it will produce the following directory structure.

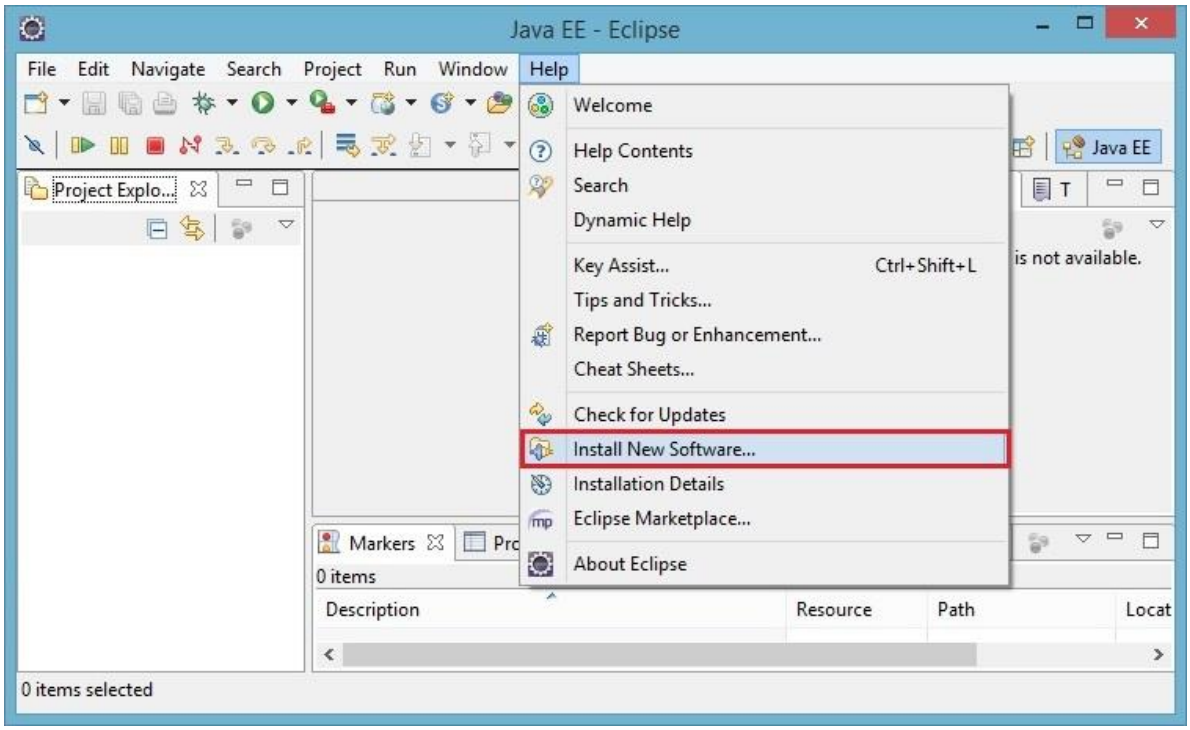
tutorialspoint
SIMPLYEASYLEARNING



Installing Hibernate

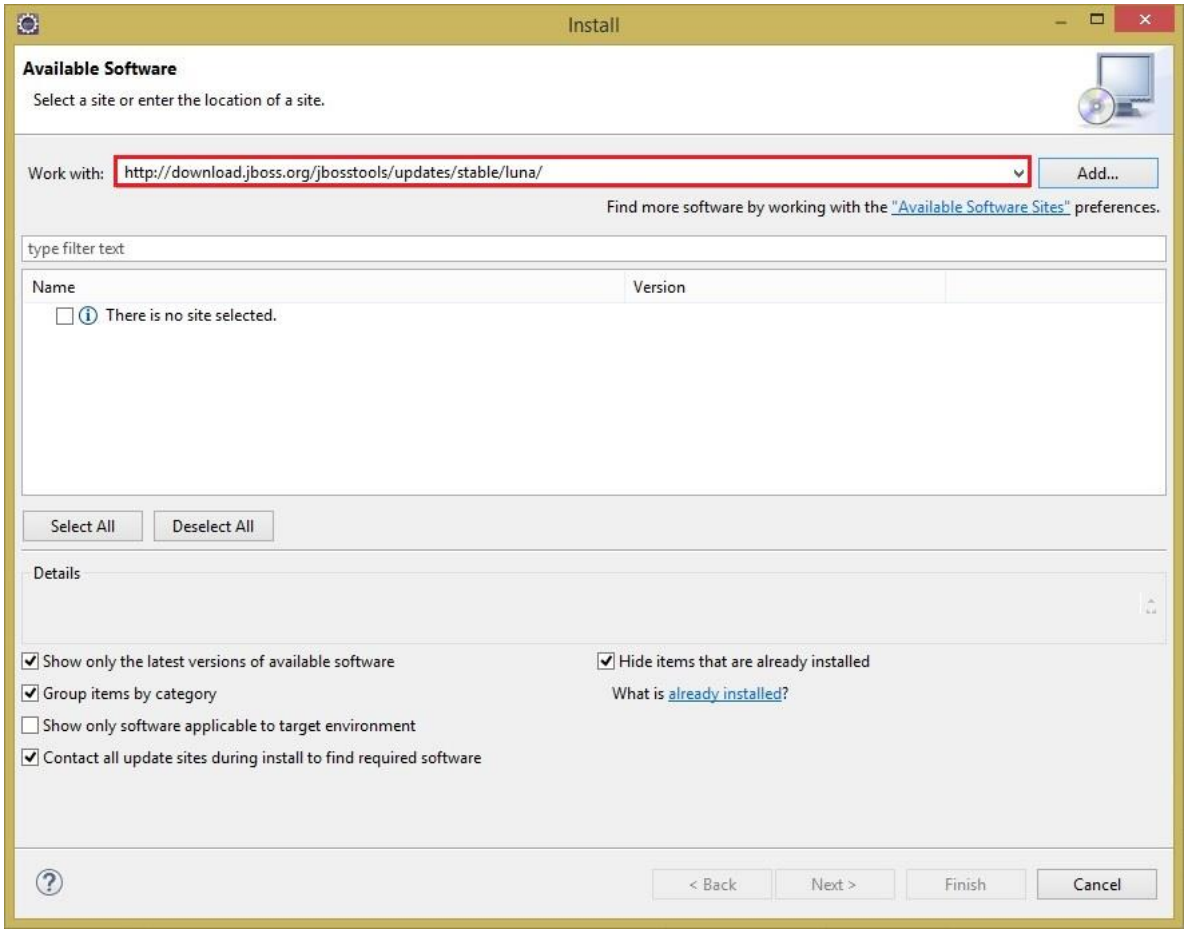
Follow the instructions given below for installing Hibernate on your system. We are using Eclipse IDE for this tutorial, therefore it is required to add Hibernate capabilities (Hibernate Tools) to Eclipse IDE.

Open Eclipse IDE and select **Help >> Install new Software**.

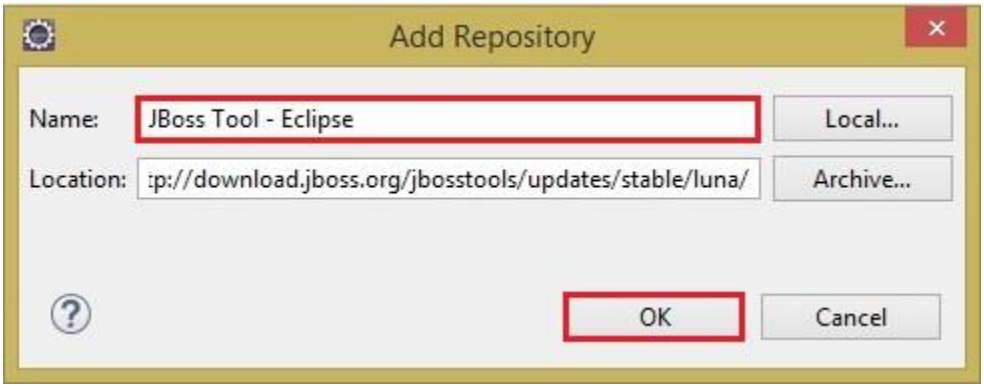


Once you click the new installation software option, you will find the following screenshot. Here you will provide the URL for downloading Hibernate tools. The URL is "http://download.jboss.org/jbosstools/updates/stable/luna/" and click the add button.

tutorialspoint
SIMPLYEASYLEARNING



After clicking the Add button, you will find the following dialog box. Here, in the place of **Name** field: Enter **JBoss Tool - Eclipse**. Location Field is automatically filled with the given JBoss URL (http://download.jboss.org/jbosstools/updates/stable/luna/), otherwise fill it. Click OK.



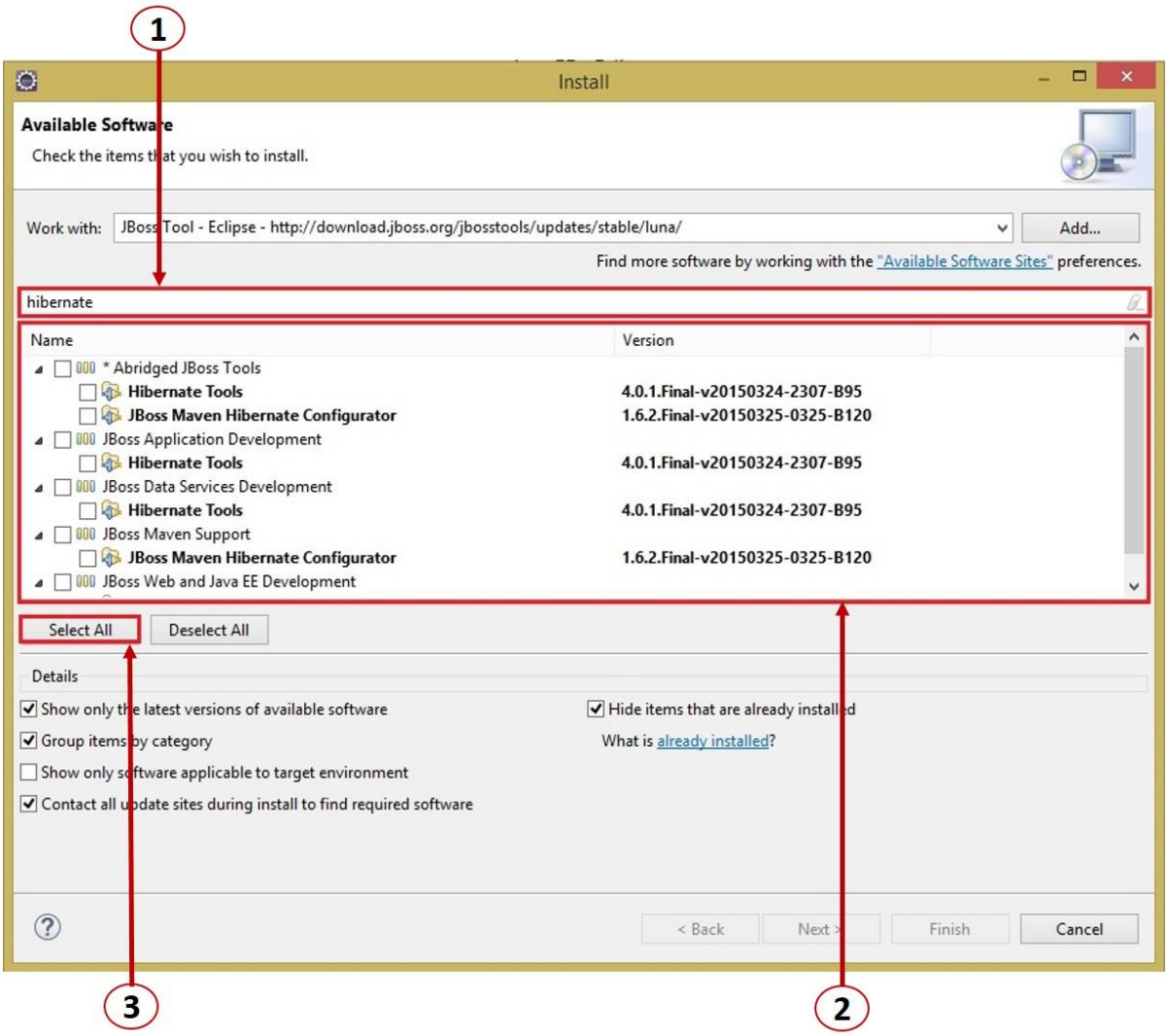
After clicking OK, you will find the following screenshot. Fill the following sections in the given screenshot.

- **SEC-1:** This is filter section. Enter the filter text as **hibernate**.

tutorialspoint
SIMPLYEASYLEARNING

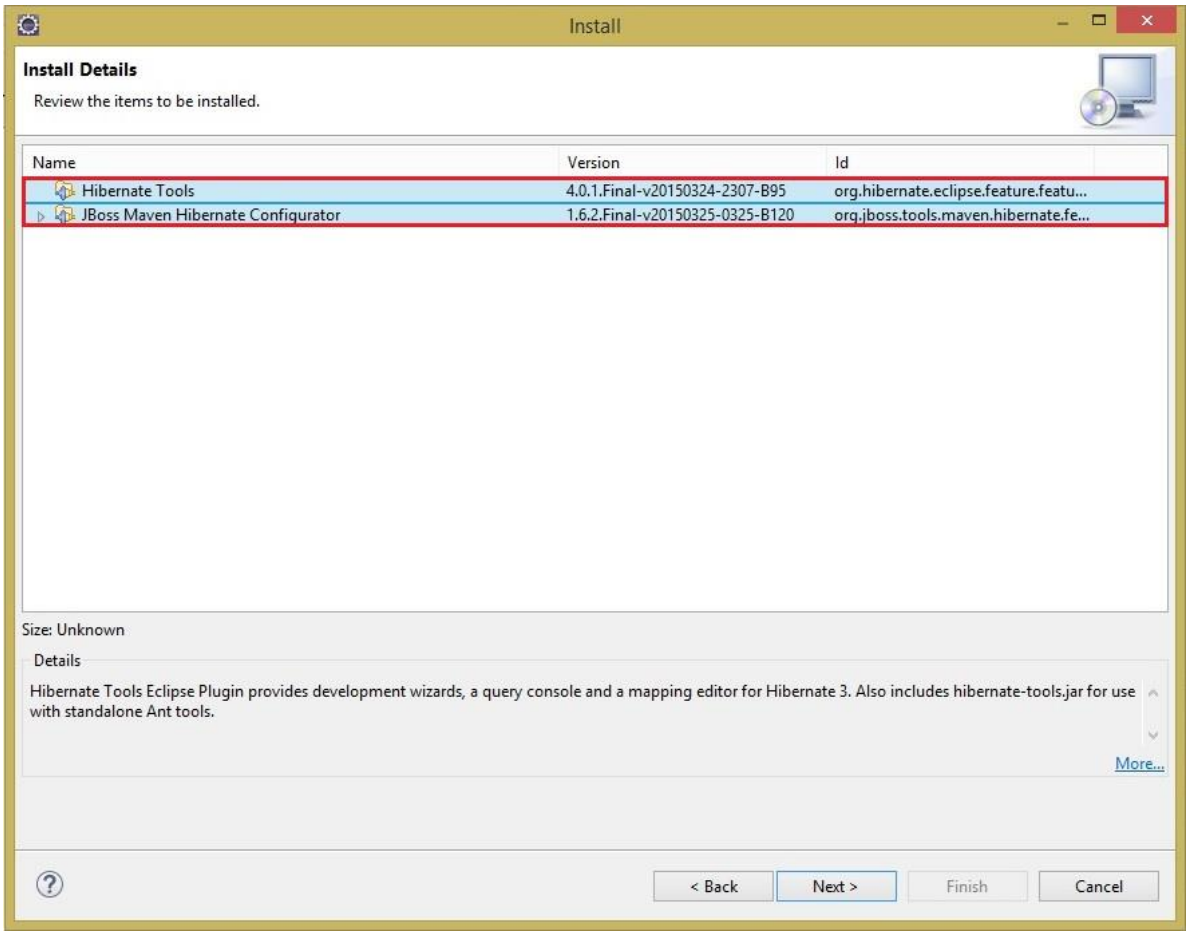
- **SEC-2:** Here, you will find a list of hibernate tools.
- **SEC-3:** Click **Select All** for selecting all Hibernate tools provided by JBoss organization.

Click Next to process. The process will take some time, please wait.



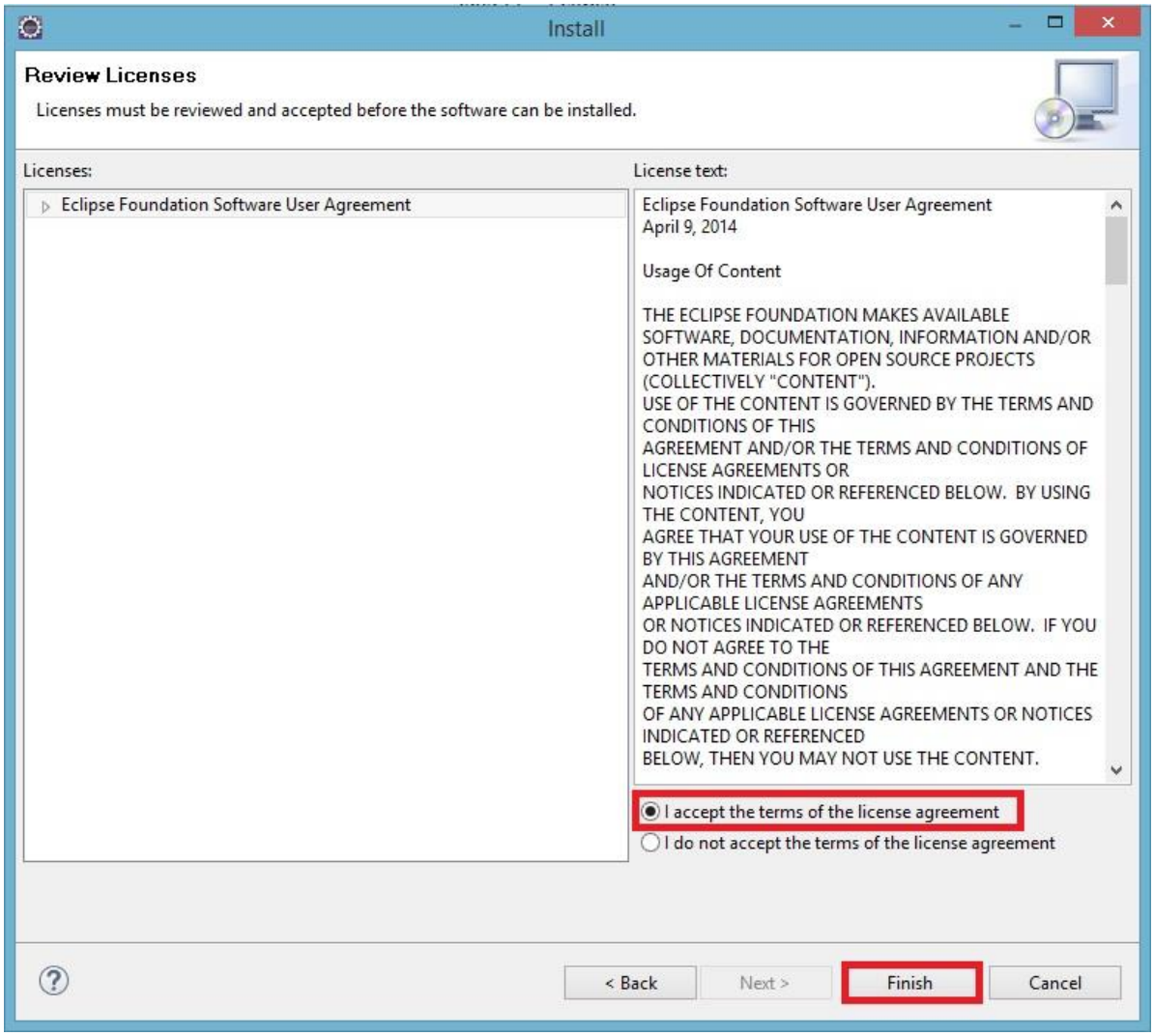
After completing the process, you will find the following screenshot. Here, you have to select all the options using the Control key and click Next.

tutorialspoint
SIMPLYEASYLEARNING

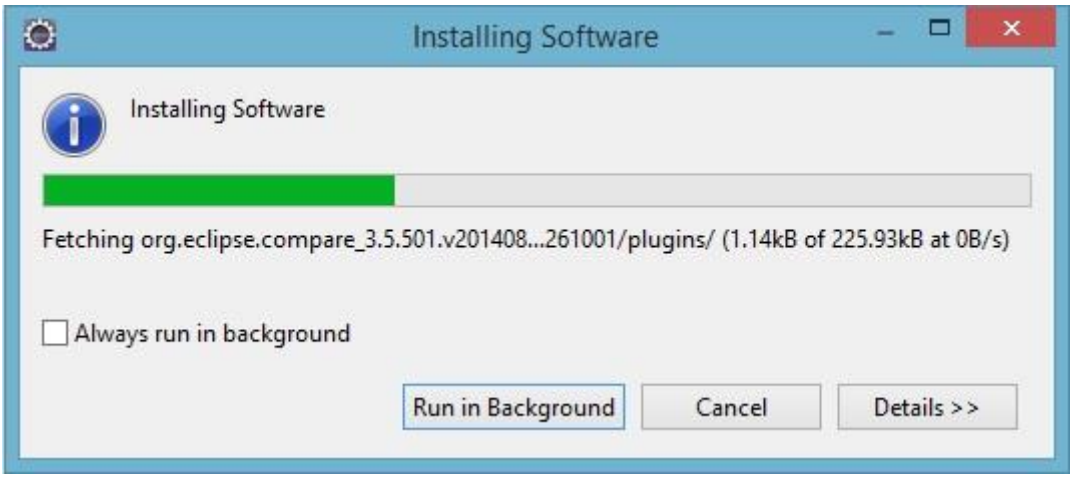


Observe the following screenshot. Here, accept the License Agreement by clicking the radiobutton and click Finish.

tutorialspoint
SIMPLYEASYLEARNING



After clicking the Finish button, it will display the following dialog box. It will take some time to install the software, please wait.



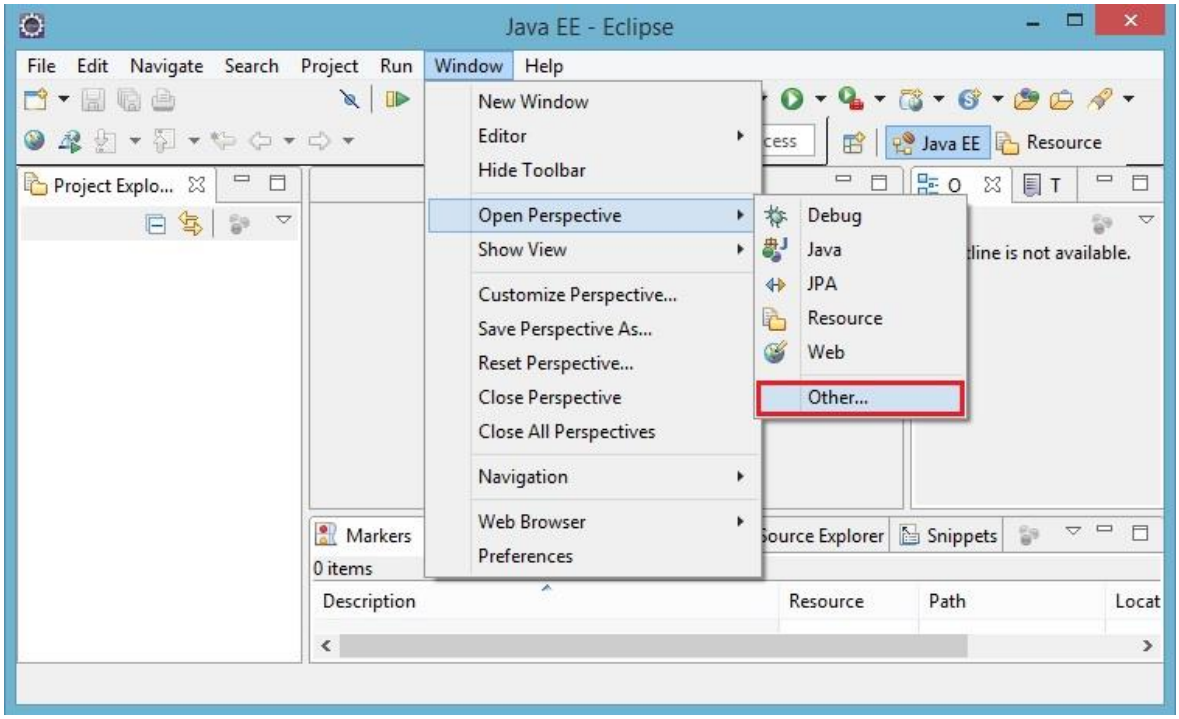
tutorialspoint
SIMPLYEASYLEARNING

After installation, you have to restart your system to save the configurations.

Verification

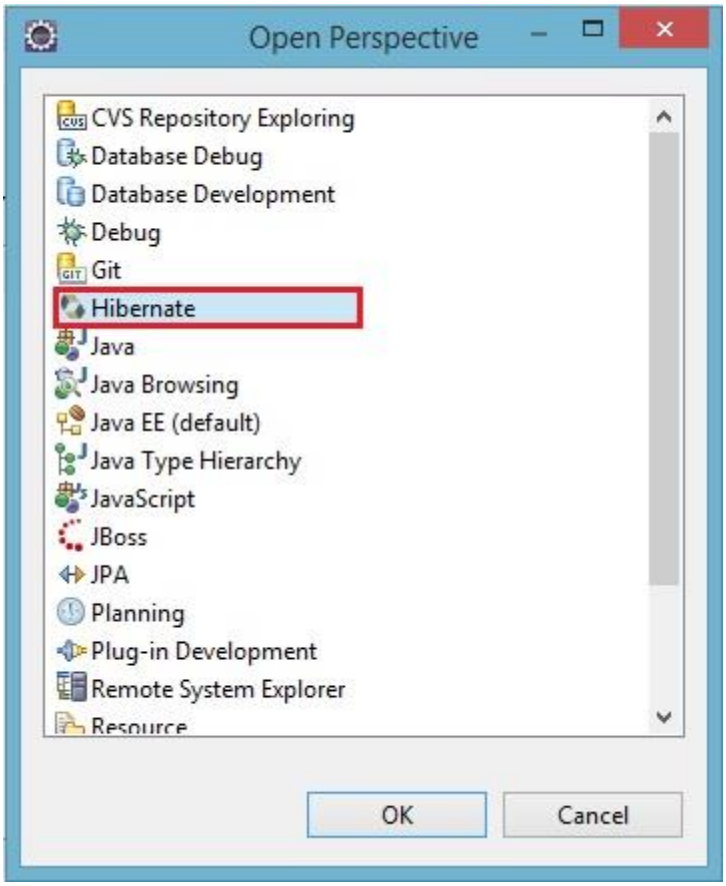
Follow the instructions given below to verify if the Hibernate Installation was successful or not.

Open Eclipse and go to **Windows >> Open Prespective >> Others**.



You will find the Hibernate Perspective in a dialog box as mentioned in the following screenshot.

tutorialspoint
SIMPLYEASYLEARNING



If you click Ok, then you will find the Eclipse IDE with Hibernate capabilities.

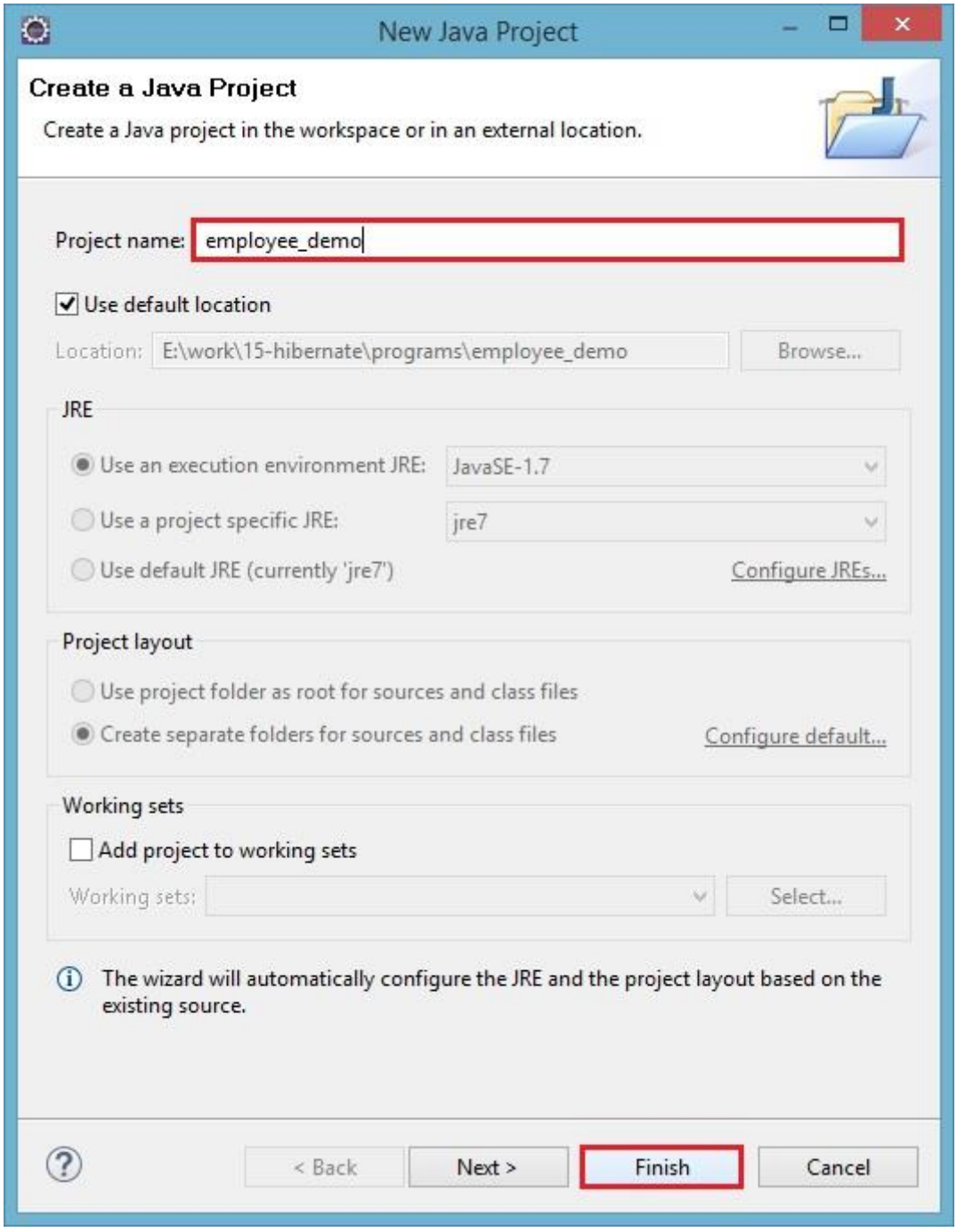
Adding Required jars to Hibernate

The following steps show how to create a Hibernate project and add the required jars to your project. This tutorial explains the concept of Hibernate using an employee database example. Therefore it is better to provide Employee database related names for project variables and components.

Step 1: Create a Java Project

Open Eclipse and create a sample Java project by following the given instructions. Go to **File -> new -> Java Project**. Then you will find the following screenshot, here you need to give the project name as **employee_demo** and click the Finish button to create the Java project.

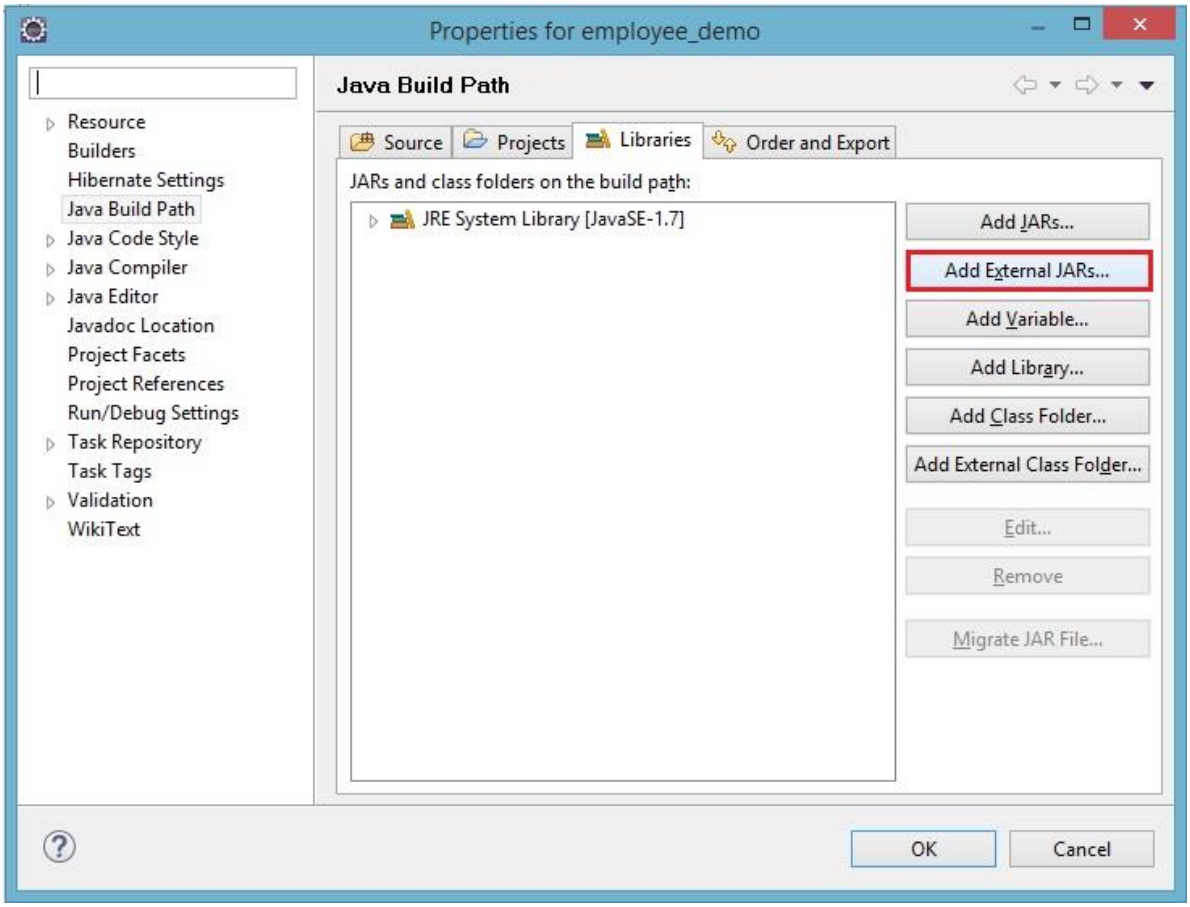
tutorialspoint
SIMPLYEASYLEARNING



Step 2: Add jars and configure the build path of your project

Right-clik on the project folder (Example_demo). Go to **Build Path** -> Configure Build Path. Then you will find the following screenshot. Here click on **Add External JARs** for adding the reuired jars to your project.

tutorialspoint
SIMPLYEASYLEARNING

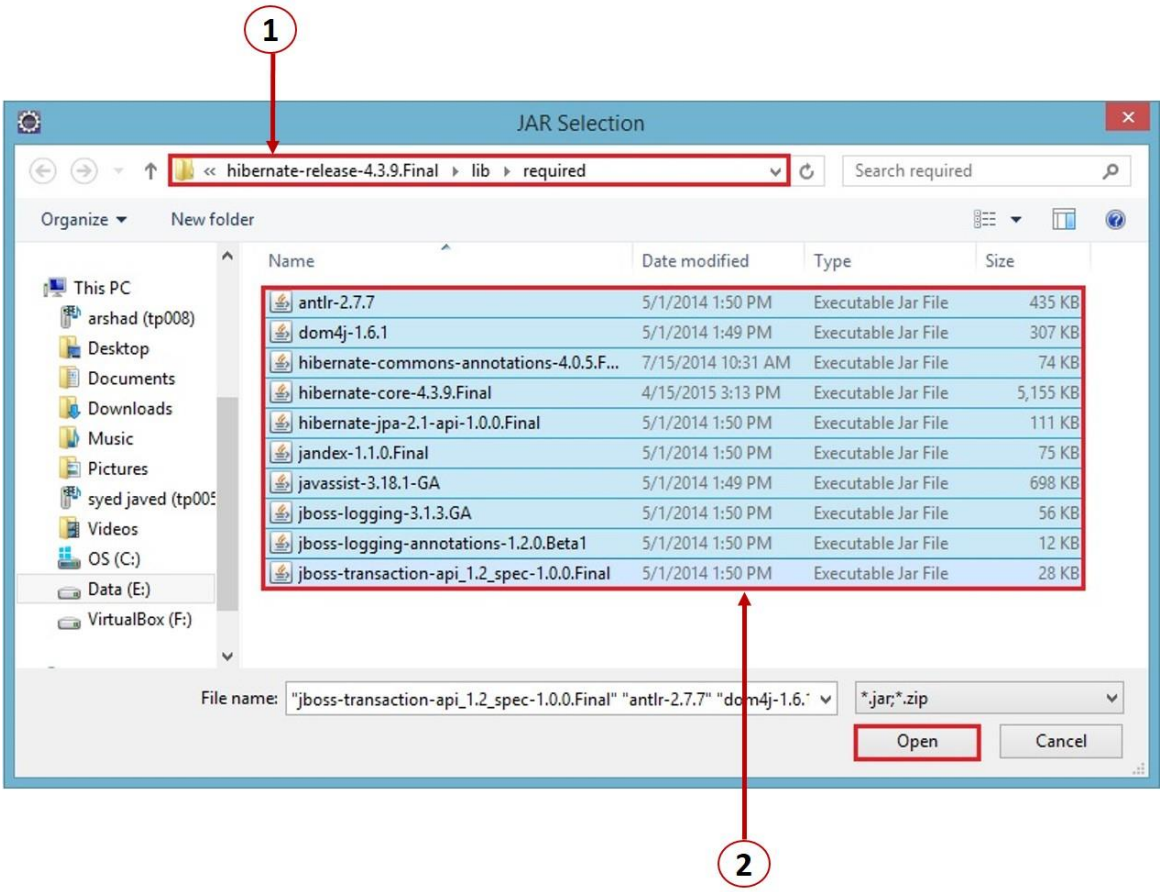


When you click on **Add External JARs**, you will find the file explorer as shown below. This screenshot shows two important sections:

1. The **address** where you downloaded the Hibernate jars that we have shown in the **Downloading Hibernate** section of this chapter.

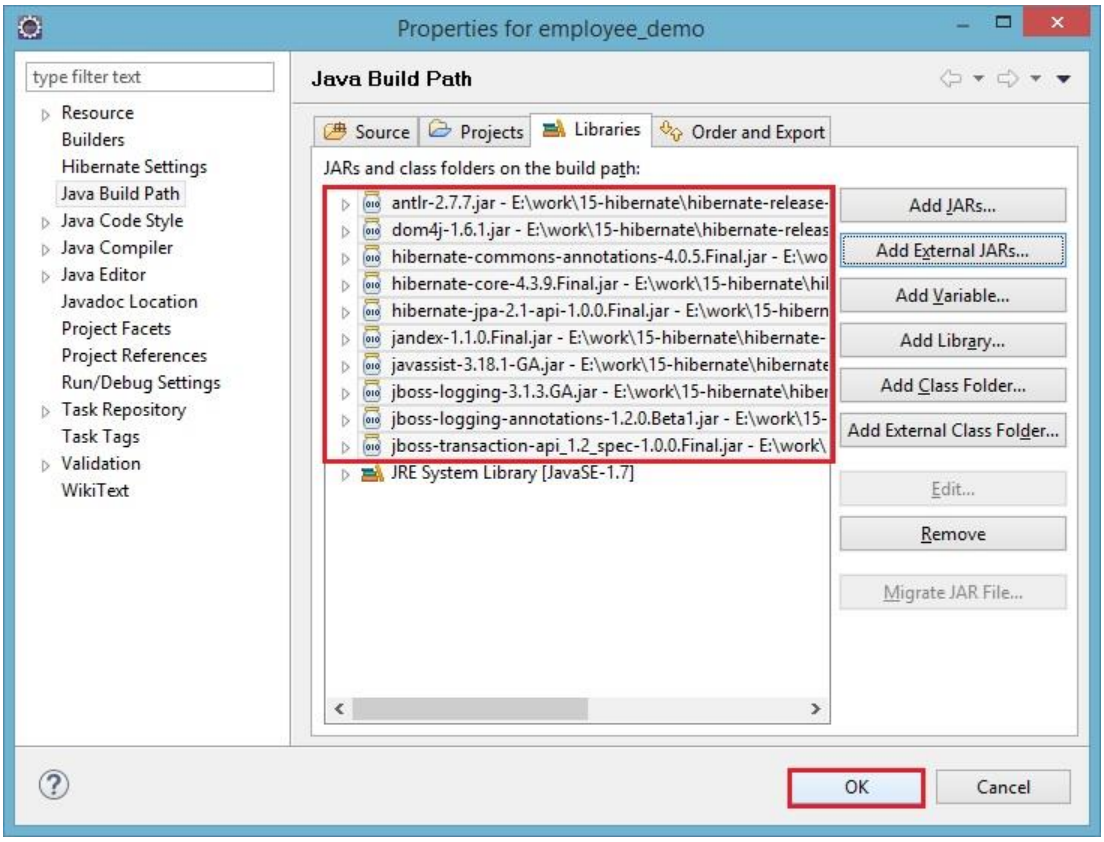
2. Select all jars in the required folder. These jars are used in Hibernate programming. Click **Open** to add all these jars to your project.

tutorialspoint
SIMPLYEASYLEARNING



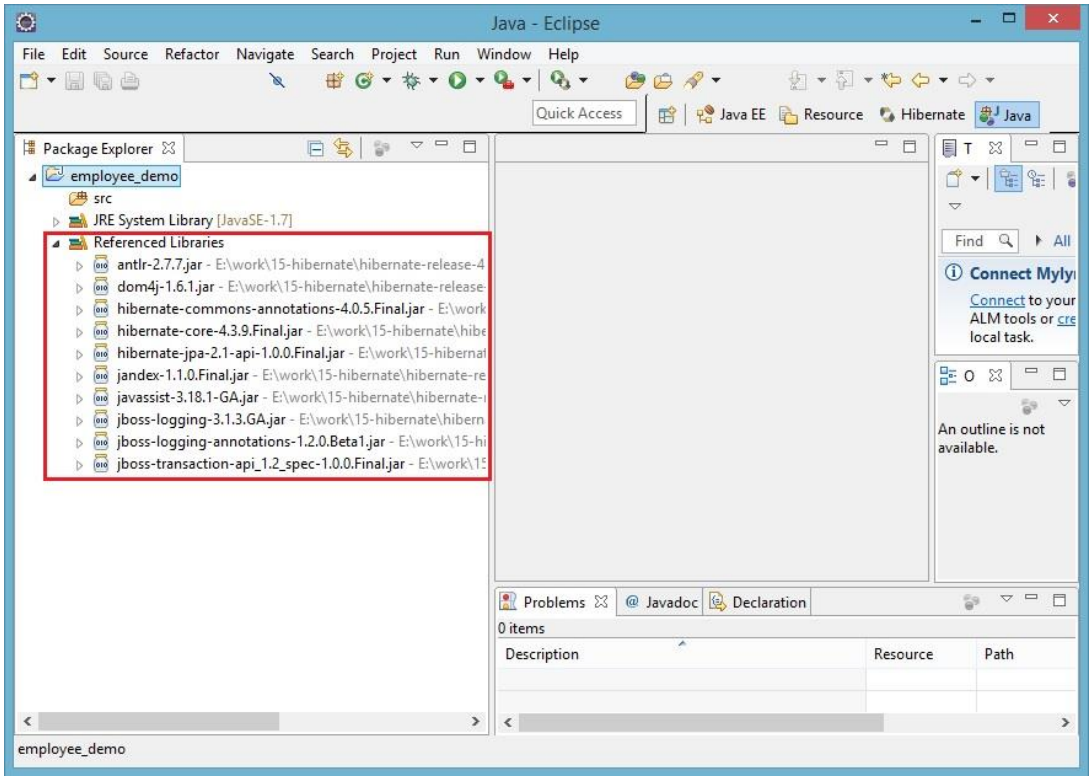
Now, you will find all the added jars in the Libraries section. Click OK to configure the build path for your project.

tutorialspoint
SIMPLYEASYLEARNING



Finally, after adding all the jars to your project, the directory structure will be as shown in the following screenshot.

tutorialspoint
SIMPLYEASYLEARNING



We have used this project hierarchy for the entire tutorial.

tutorialspoint
SIMPLYEASYLEARNING

End of ebook preview
If you liked what you saw…
Buy it from our store @ **https://store.tutorialspoint.com**

tutorialspoint
SIMPLYEASYLEARNING