# Hibernate Tutorial For Beginners

APRIL 2, 2018 BY PANKAJ — 50 COMMENTS

Welcome to the Hibernate tutorial for Beginners. **Hibernate** is one of the most widely used **Java** Most of the applications use relational databases to store application information and at the low l use JDBC API for connecting to databases and perform CRUD operations.

**Table of Contents** [hide]



## Hibernate Tutorial for Beginners

If you look at the JDBC code, there is so much of boiler plate code and there are chances of resource leak and data inconsistency because all the work needs to be done by the developer. This is where an ORM tool comes handy.

**Object-relational mapping** or **ORM** is the programming technique to map application domain model objects to the relational database tables. Hibernate is java based ORM tool that provides framework for mapping application domain objects to the relational database tables and vice versa.

Some of the benefits of using Hibernate as ORM tool are:

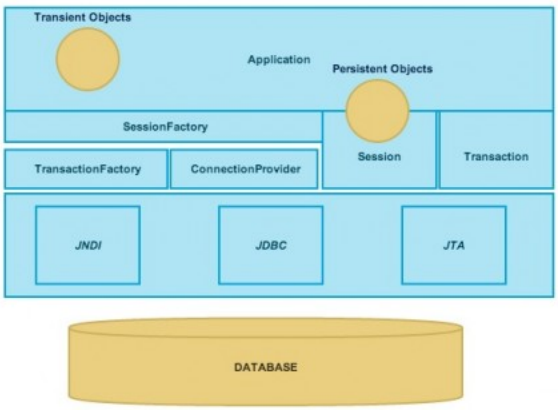
1. Hibernate supports mapping of java classes to database tables and vice versa. It provides features to perform CRUD operations across all the major relational databases.
2. Hibernate eliminates all the boiler-plate code that comes with JDBC and takes care of managing resources, so we can focus on business use cases rather than making sure that database operations are not causing resource leaks.

3. Hibernate supports transaction management and make sure there is no inconsistent data present in the system.
4. Since we use XML, property files or annotations for mapping java classes to database tables, it provides an abstraction layer between application and database.
5. Hibernate helps us in mapping joins, collections, inheritance objects and we can easily visualize how our model classes are representing database tables.
6. Hibernate provides a powerful query language (HQL) that is similar to SQL. However, HQL is fully object-oriented and understands concepts like inheritance, polymorphism and association.
7. Hibernate also offers integration with some external modules. For example Hibernate Validator is the reference implementation of Bean Validation (JSR 303).
8. Hibernate is an open source project from Red Hat Community and used worldwide. This makes it a better choice than others because learning curve is small and there are tons of online documentations and help is easily available in forums.
9. Hibernate is easy to integrate with other Java EE frameworks, it's so popular that Spring Framework provides built-in support for integrating hibernate with Spring applications.

I hope all the above benefits will convince you that Hibernate is the best choice for your application object-relational mapping requirements. Let's look at the Hibernate Framework architecture now and then we will jump into sample project where we will look into different ways to configure Hibernate in standalone java application and use it.

## Hibernate Architecture

Below image shows the Hibernate architecture and how it works as an abstraction layer between application classes and JDBC/JTA APIs for database operations. It's clear that Hibernate is built on top of JDBC and JTA APIs.




Let's look at the core components of hibernate architecture one by one.

- **SessionFactory (org.hibernate.SessionFactory)**: SessionFactory is an immutable thread-safe cache of compiled mappings for a single database. We can get instance of `org.hibernate.Session` using `SessionFactory`.

- **Session (org.hibernate.Session)**: Session is a single-threaded, short-lived object representing a conversation between the application and the persistent store. It wraps JDBC `java.sql.Connection` and works as a factory for `org.hibernate.Transaction`.
- **Persistent objects**: Persistent objects are short-lived, single threaded objects that contains persistent state and business function. These can be ordinary JavaBeans/POJOs. They are associated with exactly one `org.hibernate.Session`.
- **Transient objects**: Transient objects are persistent classes instances that are not currently associated with a `org.hibernate.Session`. They may have been instantiated by the application and not yet persisted, or they may have been instantiated by a closed `org.hibernate.Session`.
- **Transaction (org.hibernate.Transaction)**: Transaction is a single-threaded, short-lived object used by the application to specify atomic units of work. It abstracts the application from the underlying JDBC or JTA transaction. A `org.hibernate.Session` might span multiple `org.hibernate.Transaction` in some cases.
- **ConnectionProvider (org.hibernate.connection.ConnectionProvider)**: ConnectionProvider is a factory for JDBC connections. It provides abstraction between the application and underlying `javax.sql.DataSource` or `java.sql.DriverManager`. It is not exposed to application, but it can be extended by the developer.
- **TransactionFactory (org.hibernate.TransactionFactory)**: A factory for `org.hibernate.Transaction` instances.

### Hibernate and Java Persistence API (JPA)

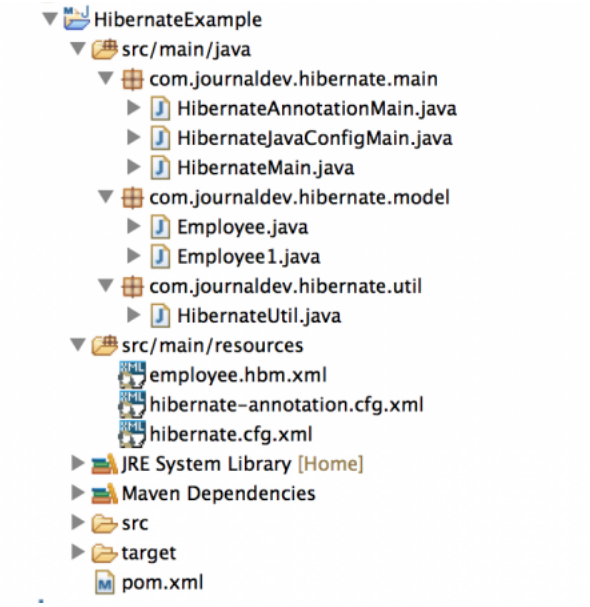
Hibernate provides implementation of **Java Persistence API**, so we can use JPA annotations with model beans and hibernate will take care of configuring it to be used in CRUD operations. We will look into this with annotations example.

## Hibernate Example

When developing hibernate applications, we need to provide two set of configuration. First set of configuration contains database specific properties that will be used to create Database connection and Session objects. Second set of configurations contains mapping between model classes and database tables.

We can use XML based or properties based configuration for database connection related configurations. We can use XML based or annotation based configurations for providing model classes and database tables mapping. We will use JPA annotations from `javax.persistence` for annotation based mappings.

Our final project will look like below image.



Create a Maven project in Eclipse or your favorite IDE, you can keep any name of your choice. Before we move on to the different components of the project, we will have to do the database setup.

## Database Table Setup

For my example, I am using MySQL database and below script is used to create necessary table.

```
CREATE TABLE `Employee` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(20) DEFAULT NULL,
  `role` varchar(20) DEFAULT NULL,
  `insert_time` datetime DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=19 DEFAULT CHARSET=utf8;
```

Notice that Employee table "id" column is automatically generated by MySQL, so we don't need to insert it.

## Hibernate Project Dependencies

Our final pom.xml file looks like below.

```
<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>com.journaldev.hibernate</groupId>
  <artifactId>HibernateExample</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>HibernateExample</name>

  <dependencies>
        <dependency>
                <groupId>org.hibernate</groupId>
                <artifactId>hibernate-core</artifactId>
                <version>4.3.5.Final</version>
        </dependency>
        <!-- Hibernate 4 uses Jboss logging, but older versions slf4j for logging -->
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-simple</artifactId>
            <version>1.7.5</version>
        </dependency>
```

**hibernate-core** artifact contains all the core hibernate classes, so we will get all the necessary features by including it in the project.

Note that I am using latest Hibernate version (4.3.5.Final) for my sample project and Hibernate is still evolving and I have seen that a lot of core classes change between every major release. So if you are using any other version, there is a small chance that you will have to modify the Hibernate configurations for it to work. However, I am sure that it will work fine for all the 4.x.x releases.

Hibernate 4 uses JBoss logging but older versions uses slf4j for logging purposes, so I have included **slf4j-simple** artifact in my project, although not needed because I am using Hibernate 4.

**mysql-connector-java** is the MySQL driver for connecting to MySQL databases, if you are using any other database then add corresponding driver artifact.

## Domain Model Classes

As you can see in above image that we have two model classes, `Employee` and `Employee1`.

Employee is a simple Java Bean class and we will use XML based configuration for providing it's mapping details.

Employee1 is a java bean where fields are annotated with JPA annotations, so that we don't need to provide mapping in separate XML file.

```
package com.journaldev.hibernate.model;

import java.util.Date;

public class Employee {

	private int id;
	private String name;
	private String role;
	private Date insertTime;

	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
```

Employee class is simple java bean, there is nothing specific to discuss here.

```
package com.journaldev.hibernate.model;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.UniqueConstraint;

@Entity
@Table(name="Employee",
		uniqueConstraints={@UniqueConstraint(columnNames={"ID"})})
public class Employee1 {
```

```
	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	@Column(name="ID", nullable=false, unique=true, length=11)
	private int id;
```

`javax.persistence.Entity` annotation is used to mark a class as Entity bean that can be persisted by hibernate, since hibernate provides JPA implementation.

`javax.persistence.Table` annotation is used to define the table mapping and unique constraints for the columns.

`javax.persistence.Id` annotation is used to define the primary key for the table.
`javax.persistence.GeneratedValue` is used to define that the field will be auto generated and **GenerationType.IDENTITY** strategy is used so that the generated "id" value is mapped to the bean and can be retrieved in the java program.

`javax.persistence.Column` is used to map the field with table column, we can also specify length, nullable and uniqueness for the bean properties.

## Hibernate Mapping XML Configuration

As stated above, we will use XML based configuration for Employee class mapping. We can choose any name, but it's good to choose with table or java bean name for clarity. Our hibernate mapping file for Employee bean looks like below.

**employee.hbm.xml**

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
	<class name="com.journaldev.hibernate.model.Employee" table="EMPLOYEE">
	<id name="id" type="int">
	    <column name="ID" />
	    <generator class="increment" />
	</id>
	<property name="name" type="java.lang.String">
	    <column name="NAME" />
	</property>
	<property name="role" type="java.lang.String">
	    <column name="ROLE" />
	</property>
	<property name="insertTime" type="timestamp">
		<column name="insert_time" />
	</property>
    </class>
</hibernate-mapping>
```

The xml configuration is simple and does the same thing as annotation based configuration.

## Hibernate Configuration Files

We will create two hibernate configuration xml files – one for xml based configuration and another for annotation based configuration.

**hibernate.cfg.xml**

```
		<!-- Connection Pool Size -->
		<property name="hibernate.connection.pool_size">1</property>

		<!-- org.hibernate.HibernateException: No CurrentSessionContext
configured! -->
		<property
name="hibernate.current_session_context_class">thread</property>

		<!-- Outputs the SQL queries, should be disabled in Production -->
		<property name="hibernate.show_sql">true</property>

		<!-- Dialect is required to let Hibernate know the Database Type,
MySQL, Oracle etc
			 Hibernate 4 automatically figure out Dialect from Database
Connection Metadata -->
		<property
name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>


		<!-- mapping file, we can use Bean annotations too -->
		<mapping resource="employee.hbm.xml" />
	</session-factory>
</hibernate-configuration>
```

Most of the properties are related to database configurations, other properties details are given in comment. Note the configuration for hibernate mapping file, we can define multiple hibernate mapping files and configure them here. Also note that mapping is specific to session factory.

**hibernate-annotation.cfg.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
		"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
		"http://hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
	<session-factory>
		<!-- Database connection properties - Driver, URL, user, password -->
		<property
name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
		<property
name="hibernate.connection.url">jdbc:mysql://localhost/TestDB</property>
		<property name="hibernate.connection.username">pankaj</property>
		<property name="hibernate.connection.password">pankaj123</property>

		<!-- org.hibernate.HibernateException: No CurrentSessionContext
configured! -->
		<property
name="hibernate.current_session_context_class">thread</property>

		<!-- Mapping with model class containing annotations -->
		<mapping class="com.journaldev.hibernate.model.Employee1"/>
	</session-factory>
```

Most of the configuration is same as XML based configuration, the only difference is the mapping configuration. We can provide mapping configuration for classes as well as packages.

## Hibernate SessionFactory

I have created a utility class where I am creating `SessionFactory` from XML based configuration as well as property based configuration. For property based configuration, we could have a property file and read it in the class, but for simplicity I am creating Properties instance in the class itself.

```
package com.journaldev.hibernate.util;

import java.util.Properties;

import org.hibernate.SessionFactory;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;
import org.hibernate.service.ServiceRegistry;

import com.journaldev.hibernate.model.Employee1;

public class HibernateUtil {

        //XML based configuration
        private static SessionFactory sessionFactory;

        //Annotation based configuration
        private static SessionFactory sessionAnnotationFactory;

        //Property based configuration
        private static SessionFactory sessionJavaConfigFactory;
```

Creating `SessionFactory` for XML based configuration is same whether mapping is XML based or annotation based. For properties based, we need to set the properties in `Configuration` object and add annotation classes before creating the `SessionFactory`.

Overall creating SessionFactory includes following steps:

1. Creating `Configuration` object and configure it
2. Creating `ServiceRegistry` object and apply configuration settings.
3. Use `configuration.buildSessionFactory()` by passing `ServiceRegistry` object as argument to get the `SessionFactory` object.

Our application is almost ready now, let's write some test programs and execute them.

## Hibernate XML Configuration Test

Our test program looks like below.

```
package com.journaldev.hibernate.main;

import java.util.Date;

import org.hibernate.Session;
```

```
import com.journaldev.hibernate.model.Employee;
import com.journaldev.hibernate.util.HibernateUtil;

public class HibernateMain {

        public static void main(String[] args) {
                Employee emp = new Employee();
                emp.setName("Pankaj");
                emp.setRole("CEO");
                emp.setInsertTime(new Date());

                //Get Session
                Session session =
HibernateUtil.getSessionFactory().getCurrentSession();
                //start transaction
                session.beginTransaction();
```

The program is self understood, when we execute the test program, we get following output.

```
May 06, 2014 12:40:06 AM
org.hibernate.annotations.common.reflection.java.JavaReflectionManager <clinit>
INFO: HCANN000001: Hibernate Commons Annotations {4.0.4.Final}
May 06, 2014 12:40:06 AM org.hibernate.Version logVersion
INFO: HHH000412: Hibernate Core {4.3.5.Final}
May 06, 2014 12:40:06 AM org.hibernate.cfg.Environment <clinit>
INFO: HHH000206: hibernate.properties not found
May 06, 2014 12:40:06 AM org.hibernate.cfg.Environment buildBytecodeProvider
INFO: HHH000021: Bytecode provider name : javassist
May 06, 2014 12:40:06 AM org.hibernate.cfg.Configuration configure
INFO: HHH000043: Configuring from resource: hibernate.cfg.xml
May 06, 2014 12:40:06 AM org.hibernate.cfg.Configuration getConfigurationInputStream
INFO: HHH000040: Configuration resource: hibernate.cfg.xml
May 06, 2014 12:40:07 AM org.hibernate.cfg.Configuration addResource
INFO: HHH000221: Reading mappings from resource: employee.hbm.xml
May 06, 2014 12:40:08 AM org.hibernate.cfg.Configuration doConfigure
INFO: HHH000041: Configured SessionFactory: null
Hibernate Configuration loaded
Hibernate serviceRegistry created
May 06, 2014 12:40:08 AM
org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl
configure
```

Notice that it's printing the generated employee id, you can check database table to confirm it.

## Hibernate Annotation Configuration Test

```
package com.journaldev.hibernate.main;

import java.util.Date;

import org.hibernate.Session;
import org.hibernate.SessionFactory;

import com.journaldev.hibernate.model.Employee1;
import com.journaldev.hibernate.util.HibernateUtil;
```

```
public class HibernateAnnotationMain {

	public static void main(String[] args) {
		Employee1 emp = new Employee1();
		emp.setName("David");
		emp.setRole("Developer");
		emp.setInsertTime(new Date());

		//Get Session
		SessionFactory sessionFactory =
HibernateUtil.getSessionAnnotationFactory();
```

When we execute above program, we get following output.

```
May 06, 2014 12:42:22 AM
org.hibernate.annotations.common.reflection.java.JavaReflectionManager <clinit>
INFO: HCANN000001: Hibernate Commons Annotations {4.0.4.Final}
May 06, 2014 12:42:22 AM org.hibernate.Version logVersion
INFO: HHH000412: Hibernate Core {4.3.5.Final}
May 06, 2014 12:42:22 AM org.hibernate.cfg.Environment <clinit>
INFO: HHH000206: hibernate.properties not found
May 06, 2014 12:42:22 AM org.hibernate.cfg.Environment buildBytecodeProvider
INFO: HHH000021: Bytecode provider name : javassist
May 06, 2014 12:42:22 AM org.hibernate.cfg.Configuration configure
INFO: HHH000043: Configuring from resource: hibernate-annotation.cfg.xml
May 06, 2014 12:42:22 AM org.hibernate.cfg.Configuration getConfigurationInputStream
INFO: HHH000040: Configuration resource: hibernate-annotation.cfg.xml
May 06, 2014 12:42:23 AM org.hibernate.cfg.Configuration doConfigure
INFO: HHH000041: Configured SessionFactory: null
Hibernate Annotation Configuration loaded
Hibernate Annotation serviceRegistry created
May 06, 2014 12:42:23 AM
org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl
configure
WARN: HHH000402: Using Hibernate built-in connection pool (not for production use!)
May 06, 2014 12:42:23 AM
```

Have a look at the output and compare it with the output from the XML based configuration, you will notice some differences. For example, we are not setting connection pool size for annotation based configuration, so it's setting to default value 20.

## Hibernate Java Configuration Test

```
package com.journaldev.hibernate.main;

import java.util.Date;

import org.hibernate.Session;
import org.hibernate.SessionFactory;

import com.journaldev.hibernate.model.Employee1;
import com.journaldev.hibernate.util.HibernateUtil;

public class HibernateJavaConfigMain {
```

```
        public static void main(String[] args) {
                Employee1 emp = new Employee1();
                emp.setName("Lisa");
                emp.setRole("Manager");
                emp.setInsertTime(new Date());

                //Get Session
                SessionFactory sessionFactory =
HibernateUtil.getSessionJavaConfigFactory();
                Session session = sessionFactory.getCurrentSession();
```

Output of the above test program is:

```
May 06, 2014 12:45:09 AM
org.hibernate.annotations.common.reflection.java.JavaReflectionManager <clinit>
INFO: HCANN000001: Hibernate Commons Annotations {4.0.4.Final}
May 06, 2014 12:45:09 AM org.hibernate.Version logVersion
INFO: HHH000412: Hibernate Core {4.3.5.Final}
May 06, 2014 12:45:09 AM org.hibernate.cfg.Environment <clinit>
INFO: HHH000206: hibernate.properties not found
May 06, 2014 12:45:09 AM org.hibernate.cfg.Environment buildBytecodeProvider
INFO: HHH000021: Bytecode provider name : javassist
Hibernate Java Config serviceRegistry created
May 06, 2014 12:45:09 AM
org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl
configure
WARN: HHH000402: Using Hibernate built-in connection pool (not for production use!)
May 06, 2014 12:45:09 AM
org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl
buildCreator
INFO: HHH000401: using driver [com.mysql.jdbc.Driver] at URL
[jdbc:mysql://localhost/TestDB]
May 06, 2014 12:45:09 AM
org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl
buildCreator
```

That's all for hibernate tutorial for beginners, I hope it will be enough to get you started. We will look into different features of Hibernate framework in future tutorials. Download the complete project from below link and play around with it to learn more.

Download Hibernate Beginners Project

NEXT »
[Solved] Hibernate Program Not Terminating

**About Pankaj**

If you have come this far, it means that you liked what you are reading. Why not reach little more and connect with me directly on **Google Plus**, **Facebook** or **Twitter**. I would love to hear your thoughts and opinions on my articles directly.

Recently I started creating video tutorials too, so do check out my videos on **Youtube**.

FILED UNDER: HIBERNATE

## Comments

**Sujan Maka says**
APRIL 6, 2018 AT 2:59 AM
Thank you. Simple and clear understanding
Reply

**Hemanth says**
OCTOBER 24, 2017 AT 12:39 AM
we can create sesion factory directly then what is the need of ServiceRegistry plz explian me
Thanks
Reply

**Akhand Singh says**
OCTOBER 11, 2017 AT 2:23 AM
finally it's work thank you very much,
Reply

**Rajinder singh says**
AUGUST 4, 2017 AT 1:29 AM
Hello sir
thanks for this tutorial you have define every steps very clearly
Reply

**Rajat Gupta says**
APRIL 13, 2017 AT 9:28 PM
Can you please do all your examples with annotations as well. I want to learn with annotations.
Thanks in advance
Reply

**Vijay Raj R says**
FEBRUARY 6, 2017 AT 6:52 PM
For Hibernate 5.x.x
private static SessionFactory buildSessionFactory() {
try {
StandardServiceRegistry standardRegistry = new StandardServiceRegistryBuilder()
.configure("hibernate.cfg.xml").build();
Metadata metadata = new MetadataSources(standardRegistry).getMetadataBuilder().build();
return metadata.getSessionFactoryBuilder().build();
} catch (HibernateException e) {
System.err.println("Initial SessionFactory creation failed." + e);
throw new ExceptionInInitializerError(e);
}
}
Reply

**Dheeraj says**
NOVEMBER 22, 2016 AT 4:13 PM
Getting compilation error:
Archive for required library: 'C:/Users/Dheeaj/.m2/repository/org/slf4j/slf4j-api/1.7.5/slf4j-api-1.7.5.jar' in project 'HibernateExample' cannot be read or is not a valid ZIP file
Reply

**Vishnu says**
SEPTEMBER 24, 2016 AT 12:48 PM
Thanks a lot for this tutorial □ . You explain with lot of clairty. Can you please share us the method you use to learn apis in such a clear manner .....
Reply

**BSK says**

SEPTEMBER 20, 2016 AT 7:57 AM
Very simple & clear. Thanks
Reply

**Costas says**
JULY 31, 2016 AT 10:40 AM
Amazing my friend very useful !! you are pro !
Reply

**valentino says**
JULY 21, 2016 AT 2:12 AM
Pankaj, what do you mean when you say "Transient objects are persistent classes instances", it is not clear to me, if they are transient how can they be persistent?
Reply

**Pankaj says**
JULY 21, 2016 AT 7:09 AM
For example, Employee instance created using "new" operator are in transient state. Once we call Hibernate Session save() method on it, then it's state changes to persistent.
Reply

**Wiro Sableng says**
JULY 4, 2016 AT 4:14 AM
This answer my question, great job!
Reply

**Anton says**
APRIL 18, 2016 AT 2:20 AM
Pankaj, thank you for this useful tutorial.
When I was making it, I met with one problem:
Exception in thread "main" org.hibernate.MappingException: Unknown entity: com.journaldev.hibernate.model.Employee
The reason is that I use Hibernate ver. 5.1.0 but in tutorial Hibernate ver. is 4.3.5, and the root of all evil is how we create SessionFactory object in class com.journaldev.hibernate.util.HibernateUtil, to be exact in methods buildSessionFactory() and buildSessionAnnotationFactory().
To solve this problem we need to change one line in that both methods, from :
SessionFactory sessionFactory = configuration.buildSessionFactory(serviceRegistry);
to
SessionFactory sessionFactory = configuration.buildSessionFactory();
and then everything will be work.
Also this link could be useful – http://stackoverflow.com/questions/19085816/how-to-fix-the-error-info-hhh000206-hibernate-properties-not-found
Reply

**Gokku says**
JULY 12, 2016 AT 2:26 AM
So is it true that we don't need serviceRegistry anymore?
Reply

**bharath says**
APRIL 12, 2016 AT 3:47 AM
Very useful material.
Reply

**yann says**
MARCH 24, 2016 AT 7:17 AM
Good Job …. Very usefull
Reply

**Savani Fu says**
FEBRUARY 11, 2016 AT 12:20 PM
Hey Pankaj – I'd like to create a annotation based model classes from the hibernate configuration using Jboss tools using reversed engineering process, but it does not created annotation instead it creates the .hbm files. Could you please guide how to create a annotation based files (i.e without .hbm files) ? Please see issue: http://stackoverflow.com/questions/35347623/unable-to-create-model-classes-from-db-schema-using-reversed-engineering-using-h
Reply

**mayank says**
FEBRUARY 9, 2016 AT 1:23 PM
Hi,
I am getting below error
Reply

**mayank says**
FEBRUARY 9, 2016 AT 1:24 PM
Exception in thread "main" org.hibernate.MappingException: Unknown entity: com.journaldev.hibernate.model.Employee
at org.hibernate.internal.SessionFactoryImpl.getEntityPersister(SessionFactoryImpl.java:781)
at org.hibernate.internal.SessionImpl.getEntityPersister(SessionImpl.java:1520)
at org.hibernate.event.internal.AbstractSaveEventListener.saveWithGeneratedId(AbstractSaveEventListener.java:100)
at org.hibernate.event.internal.DefaultSaveOrUpdateEventListener.saveWithGeneratedOrRequestedId(DefaultSaveOrUpdateEventListener.j
at org.hibernate.event.internal.DefaultSaveEventListener.saveWithGeneratedOrRequestedId(DefaultSaveEventListener.java:38)
at org.hibernate.event.internal.DefaultSaveOrUpdateEventListener.entityIsTransient(DefaultSaveOrUpdateEventListener.java:177)

at
org.hibernate.event.internal.DefaultSaveEventListener.performSaveOrUpdate(DefaultSaveEventListener.java:32)
at
org.hibernate.event.internal.DefaultSaveOrUpdateEventListener.onSaveOrUpdate(DefaultSaveOrUpdateEventListener.java:73)
at org.hibernate.internal.SessionImpl.fireSave(SessionImpl.java:679)
at org.hibernate.internal.SessionImpl.save(SessionImpl.java:671)
at org.hibernate.internal.SessionImpl.save(SessionImpl.java:666)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
at java.lang.reflect.Method.invoke(Unknown Source)
at
org.hibernate.context.internal.ThreadLocalSessionContext$TransactionProtectionWrapper.invoke(ThreadLocalSessionContext.java:338)
at com.sun.proxy.$Proxy19.save(Unknown Source)
at com.journaldev.hibernate.main.HibernateMain.main(HibernateMain.java:23)
Reply

**erayiz says**
OCTOBER 21, 2015 AT 1:23 AM
Hello,
I am getting this error
Exception in thread "main" org.hibernate.MappingException: Unknown entity
Reply

**erayiz says**
OCTOBER 21, 2015 AT 4:33 AM
I solve it , I missed to change "configuration.addAnnotatedClass" with my new class in buildSessionJavaConfigFactory/HibernateUtil
Reply

**JOHNYBASHA SHAIK says**
OCTOBER 31, 2015 AT 6:50 PM
If so, why author didn't use it in his code?
Reply

**JOHNYBASHA SHAIK says**
OCTOBER 31, 2015 AT 6:52 PM
got it… ignore my comment.
Reply

**Harish says**
OCTOBER 3, 2015 AT 1:50 AM
Thank you for this tutorial explained well and in very simple language. Worked for me with minor tweaking as "@GeneratedValue(strategy=GenerationType.IDENTITY)" doesn't work for Oracle DB. I used

GenerationType.TABLE instead and it solved my purpose.
Reply

**lawrence chids says**
SEPTEMBER 20, 2015 AT 12:09 PM
wow thats a great tutorial. Thank you for such materials. Keep up the good work.
Reply

**gundamaiah says**
AUGUST 25, 2015 AT 5:04 PM
I am getting the below exception please help
Initial SessionFactory creation failed.java.lang.NoClassDefFoundError:
javax/transaction/SystemException
Exception in thread "main" java.lang.ExceptionInInitializerError
at com.cognizant.hibernate.util.HibernateUtil.buildSessionFactory(HibernateUtil.java:41)
at com.cognizant.hibernate.util.HibernateUtil.getSessionFactory(HibernateUtil.java:101)
at com.cognizant.hibernate.main.HibernateMain.main(HibernateMain.java:19)
Caused by: java.lang.NoClassDefFoundError: javax/transaction/SystemException
at java.lang.Class.forName0(Native Method)
at java.lang.Class.forName(Class.java:270)
at org.jboss.logging.Logger$1.run(Logger.java:2252)
at java.security.AccessController.doPrivileged(Native Method)
at org.jboss.logging.Logger.getMessageLogger(Logger.java:2227)
at org.jboss.logging.Logger.getMessageLogger(Logger.java:2214)
at org.hibernate.cfg.Configuration.(Configuration.java:192)
at com.cognizant.hibernate.util.HibernateUtil.buildSessionFactory(HibernateUtil.java:27)
Reply

**J CHUI says**
APRIL 9, 2018 AT 11:41 PM
I have not the same problem. Have you solved now ? Thanks
10-Apr-2018
Reply

**J CHUI says**
APRIL 9, 2018 AT 11:43 PM
typo error. I have the same problem Have you solved now ? Thanks
Reply

**Aparna says**
AUGUST 13, 2015 AT 6:04 AM
I am trying out these examples in Eclipse. I created the pom.xml and right clicked and chose Run As -> Maven Install. It downloaded the jars in the local repository. Do I need to set the Classpath in eclipse to each of the jars in the repository? Is there a way to configure eclipse to point to the repo? Please reply ASAP

Reply

**Mark R says**
JUNE 24, 2015 AT 7:25 PM
How can you setup the hibernate.cfg.xml to use system environment variables to protect the passwords??
Reply

**prashanth says**
JUNE 9, 2015 AT 1:00 AM
Hi,
why we are getting the message " Using Hibernate built-in connection pool (not for production use!) ".
how to fix this?
Reply

**Pankaj says**
JUNE 9, 2015 AT 10:37 AM
Because we should not use it for production, we should always use Container (Tomcat) provided connection pooling for best results.
Reply

**dsgfgjhj says**
MAY 29, 2015 AT 7:59 AM
i know that onetoone relationship program but i want internally code explanation whats happeing please reply to as soon as possible
Reply

**osama says**
MARCH 30, 2015 AT 4:04 AM
I need to remove this info
[main] INFO org.hibernate.annotations.common.Version – Hibernate Commons Annotations 3.2.0.Final
271 [main] INFO org.hibernate.cfg.Environment – Hibernate 3.6.10.Final
275 [main] INFO org.hibernate.cfg.Environment – hibernate.properties not found
287 [main] INFO org.hibernate.cfg.Environment – Bytecode provider name : javassist
296 [main] INFO org.hibernate.cfg.Environment – using JDK 1.4 java.sql.Timestamp handling
Reply

**Afzaal says**
FEBRUARY 27, 2015 AT 10:54 PM
thanks good tutorial □
Reply

**trinsit.w says**

JANUARY 6, 2015 AT 2:53 AM

I have this problem also. Can you please help? Thank you very much.

Jan 06, 2015 5:51:10 PM org.hibernate.annotations.common.reflection.java.JavaReflectionManager
INFO: HCANN000001: Hibernate Commons Annotations {4.0.4.Final}
Jan 06, 2015 5:51:10 PM org.hibernate.Version logVersion
INFO: HHH000412: Hibernate Core {4.3.5.Final}
Jan 06, 2015 5:51:10 PM org.hibernate.cfg.Environment
INFO: HHH000206: hibernate.properties not found
Jan 06, 2015 5:51:10 PM org.hibernate.cfg.Environment buildBytecodeProvider
INFO: HHH000021: Bytecode provider name : javassist
Jan 06, 2015 5:51:10 PM org.hibernate.cfg.Configuration configure
INFO: HHH000043: Configuring from resource: hibernate.cfg.xml
Jan 06, 2015 5:51:10 PM org.hibernate.cfg.Configuration getConfigurationInputStream
INFO: HHH000040: Configuration resource: hibernate.cfg.xml
Initial SessionFactory creation failed.org.hibernate.HibernateException: Could not parse configuration: hibernate.cfg.xml
Exception in thread "main" java.lang.ExceptionInInitializerError
at com.journaldev.hibernate.util.HibernateUtil.buildSessionFactory(HibernateUtil.java:40)
at com.journaldev.hibernate.util.HibernateUtil.getSessionFactory(HibernateUtil.java:98)
at com.journaldev.hibernate.main.HibernateMain.main(HibernateMain.java:19)
Caused by: org.hibernate.HibernateException: Could not parse configuration: hibernate.cfg.xml
at org.hibernate.cfg.Configuration.doConfigure(Configuration.java:2163)
at org.hibernate.cfg.Configuration.configure(Configuration.java:2075)
at com.journaldev.hibernate.util.HibernateUtil.buildSessionFactory(HibernateUtil.java:27)
… 2 more
Caused by: org.dom4j.DocumentException: Connection timed out: connect Nested exception: Connection timed out: connect
at org.dom4j.io.SAXReader.read(SAXReader.java:484)
at org.hibernate.cfg.Configuration.doConfigure(Configuration.java:2155)
… 4 more

Reply

**niraj says**

DECEMBER 21, 2014 AT 12:47 AM

i got that error that is comes from hibernateUtil configuration.. plz help me..

Dec 21, 2014 2:16:12 PM org.hibernate.annotations.common.reflection.java.JavaReflectionManager
INFO: HCANN000001: Hibernate Commons Annotations {4.0.4.Final}
Dec 21, 2014 2:16:12 PM org.hibernate.Version logVersion
INFO: HHH000412: Hibernate Core {4.3.5.Final}
Dec 21, 2014 2:16:12 PM org.hibernate.cfg.Environment
INFO: HHH000205: Loaded properties from resource hibernate.properties: {hibernate.connection.driver_class=org.h2.Driver, hibernate.service.allow_crawling=false, hibernate.dialect=org.hibernate.dialect.H2Dialect, hibernate.max_fetch_depth=5, hibernate.format_sql=true, hibernate.generate_statistics=true, hibernate.connection.username=sa, hibernate.connection.url=jdbc:h2:mem:db1;DB_CLOSE_DELAY=-1;MVCC=TRUE, hibernate.bytecode.use_reflection_optimizer=false, hibernate.jdbc.batch_versioned_data=true, hibernate.connection.pool_size=5}
Dec 21, 2014 2:16:12 PM org.hibernate.cfg.Environment buildBytecodeProvider
INFO: HHH000021: Bytecode provider name : javassist
Dec 21, 2014 2:16:12 PM org.hibernate.cfg.Configuration configure
INFO: HHH000043: Configuring from resource: hibernate.cfg.xml
Dec 21, 2014 2:16:12 PM org.hibernate.cfg.Configuration getConfigurationInputStream

INFO: HHH000040: Configuration resource: hibernate.cfg.xml
Initial SessionFactory creation failed.java.lang.NullPointerException
Exception in thread "main" java.lang.ExceptionInInitializerError
at main.java.com.journaldev.hibernate.util.HibernateUtil.buildSessionFactory(HibernateUtil.java:40)
at main.java.com.journaldev.hibernate.util.HibernateUtil.getSessionFactory(HibernateUtil.java:99)
at main.java.com.journaldev.hibernate.main.HibernateMain.main(HibernateMain.java:19)
Caused by: java.lang.NullPointerException
at org.hibernate.internal.util.ConfigHelper.getResourceAsStream(ConfigHelper.java:170)
at org.hibernate.cfg.Configuration.getConfigurationInputStream(Configuration.java:2093)
at org.hibernate.cfg.Configuration.configure(Configuration.java:2074)
at main.java.com.journaldev.hibernate.util.HibernateUtil.buildSessionFactory(HibernateUtil.java:27)
... 2 more
Reply

**niraj says**
DECEMBER 20, 2014 AT 3:08 AM
how much jar we need to create a hibernate configuration or run a simple program..
Reply

**anonymous says**
NOVEMBER 10, 2014 AT 1:45 PM
thank you Pankaj this was really very helpful. after learning this I am trying to implement all the entities through persistence.xml is there any tutorial available on that?
Reply

**Rafa says**
NOVEMBER 5, 2014 AT 12:24 AM
By far the best and most explanatory beginners hibernate example I found.
Thanks a lot!
Reply

**STEVEN JOSEPH FONTALVO MORA says**
NOVEMBER 2, 2014 AT 6:46 AM
hello good, thanks for the tutorial Might upload some sample file .. thanks
Reply

**A B says**
OCTOBER 17, 2014 AT 4:20 AM
posted stuff is quiet helpful.
Reply

**Savan Patel says**
OCTOBER 17, 2014 AT 3:44 AM

Great Work..
Reply

**Danish says**
SEPTEMBER 8, 2014 AT 4:24 AM
Thanks for the tutorials....however it would be perfect if you could make videos of tutorials
Reply

**Tokala Sai Teja says**
JULY 28, 2014 AT 6:46 AM
I am getting this error if i am running this same project
Initial SessionFactory creation failed.org.hibernate.HibernateException: Could not parse configuration: hibernate.cfg.xml
Exception in thread "main" java.lang.ExceptionInInitializerError
at com.journaldev.hibernate.util.HibernateUtil.buildSessionFactory(HibernateUtil.java:40)
at com.journaldev.hibernate.util.HibernateUtil.getSessionFactory(HibernateUtil.java:98)
at com.journaldev.hibernate.main.HibernateMain.main(HibernateMain.java:19)
Caused by: org.hibernate.HibernateException: Could not parse configuration: hibernate.cfg.xml
at org.hibernate.cfg.Configuration.doConfigure(Configuration.java:2163)
at org.hibernate.cfg.Configuration.configure(Configuration.java:2075)
at com.journaldev.hibernate.util.HibernateUtil.buildSessionFactory(HibernateUtil.java:27)
... 2 more
Caused by: org.dom4j.DocumentException: Connection refused: connect Nested exception: Connection refused: connect
at org.dom4j.io.SAXReader.read(SAXReader.java:484)
at org.hibernate.cfg.Configuration.doConfigure(Configuration.java:2155)
... 4 more
Reply

**Pankaj says**
JULY 28, 2014 AT 7:11 AM
This exception comes when you are not online, so your program is not able to access the DTD files. Read https://www.journaldev.com/2959/how-to-configure-hibernate-cfg-xml-to-work-offline for the resolution.
Reply

**trinsit.w says**
JANUARY 6, 2015 AT 3:06 AM
It works now. Thank you.
Reply

**sanjeev kumar says**
JULY 23, 2014 AT 1:48 AM
Hello Sir,

Really appreciate your knowledge on the different subjects and your way of sharing these to us.
I was looking for hibernate tutorial and came here.
Request you to kindly link the hibernate tutorials one by one as you have done for JSP servlet tutorial.
That way it would be great for beginners to follow the different posts in an organised way.
Reply

**Pankaj says**
JULY 23, 2014 AT 10:13 AM
I will do that in coming weeks.
Reply

## Leave a Reply

Your email address will not be published. Required fields are marked *
Comment

Name *

Email *

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

Search for tutorials...

DOWNLOAD ANDROID APP

HIBERNATE FRAMEWORK

## Hibernate Tutorial

- Hibernate Example
- Hibernate SessionFactory
- Hibernate Session get load
- Hibernate Session save
- HQL Example
- Hibernate Criteria
- Hibernate SQL
- Hibernate Named Query
- Hibernate Log4J
- Hibernate Validator
- Hibernate Tomcat DataSource

### Hibernate Mapping

- Hibernate One to One Mapping
- Hibernate One to Many Mapping
- Hibernate Many to Many Join Tables

### Hibernate Caching

- Hibernate Cache
- Hibernate EHCache

### Hibernate Integrations

- Hibernate Spring
- Hibernate Spring MVC
- Hibernate Struts 2
- Hibernate Primefaces
- Hibernate Primefaces Spring
- Hibernate SpringRoo Primefaces
- Hibernate JSF Spring

### Miscellaneous

- Hibernate Tools Eclipse Plugin
- Hibernate Configuration Offline
- [Solved] No identifier specified
- Hibernate Program Not Terminating
- Access to DialectResolutionInfo
- get is not valid
- No CurrentSessionContext configured
- Hibernate Interview Questions

RECOMMENDED TUTORIALS

### Java Tutorials

- Java IO
- Java Regular Expressions
- Multithreading in Java
- Java Logging
- Java Annotations
- Java XML
- Collections in Java
- Java Generics
- Exception Handling in Java
- Java Reflection
- Java Design Patterns
- JDBC Tutorial

### Java EE Tutorials

- Servlet JSP Tutorial

- Struts2 Tutorial
- Spring Tutorial
- Hibernate Tutorial
- Primefaces Tutorial
- Apache Axis 2
- JAX-RS
- Memcached Tutorial