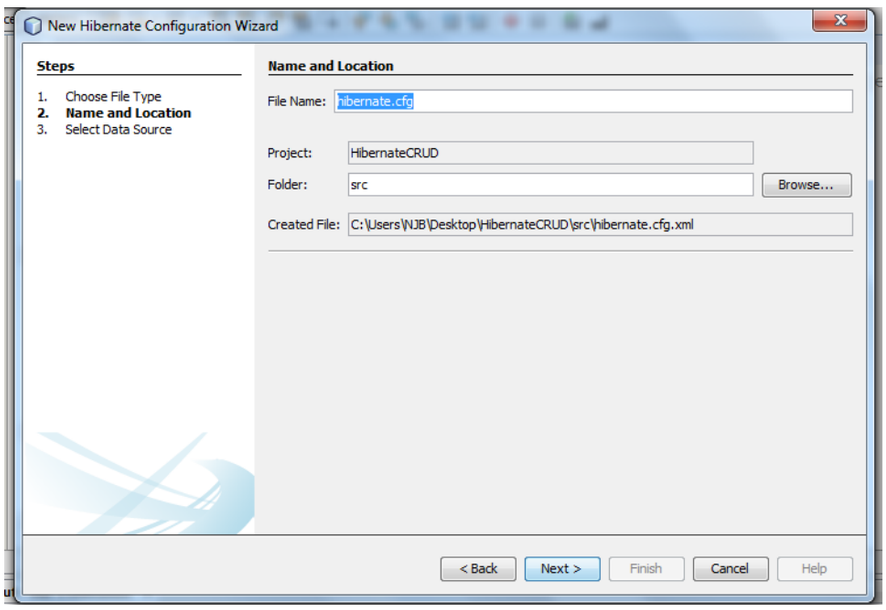
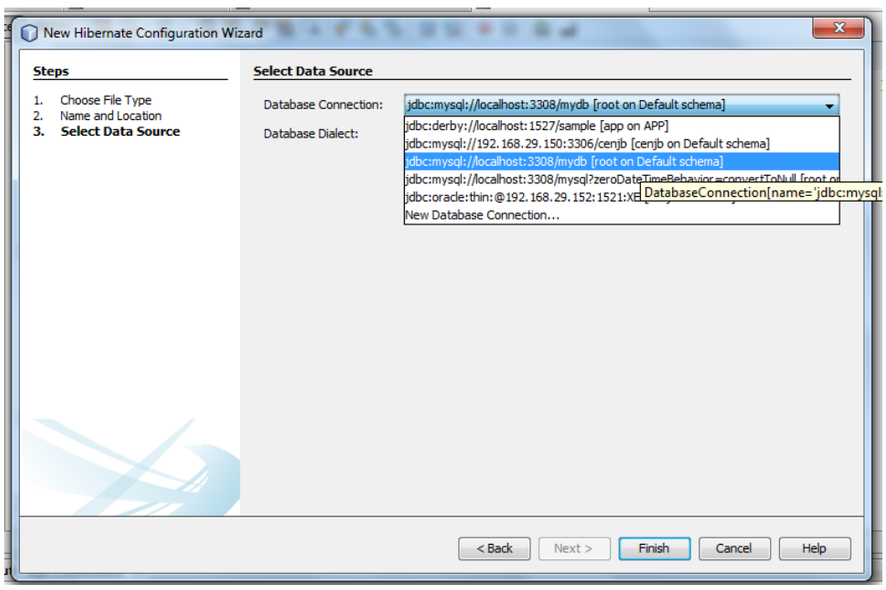
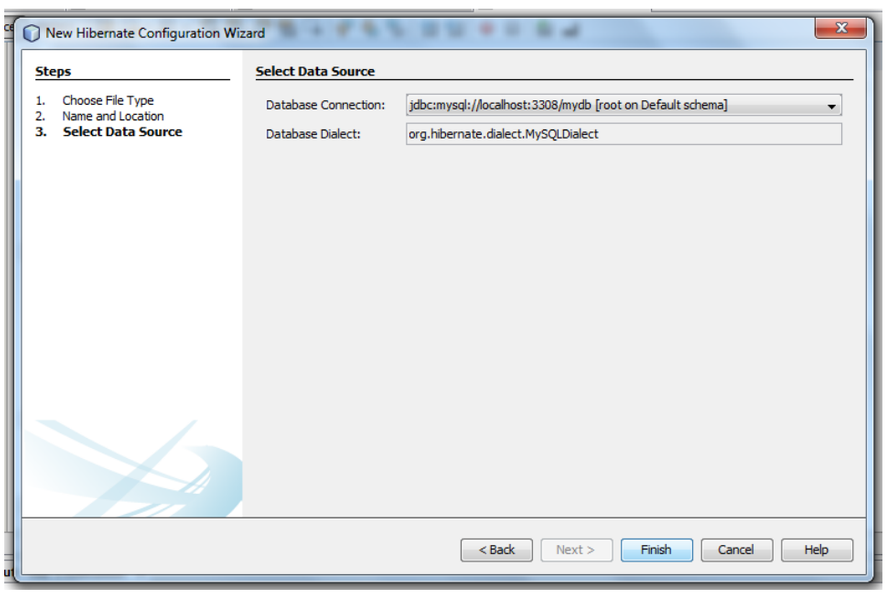
Steps for CRUD Hibernate application

1. Hibernate configuration wizard





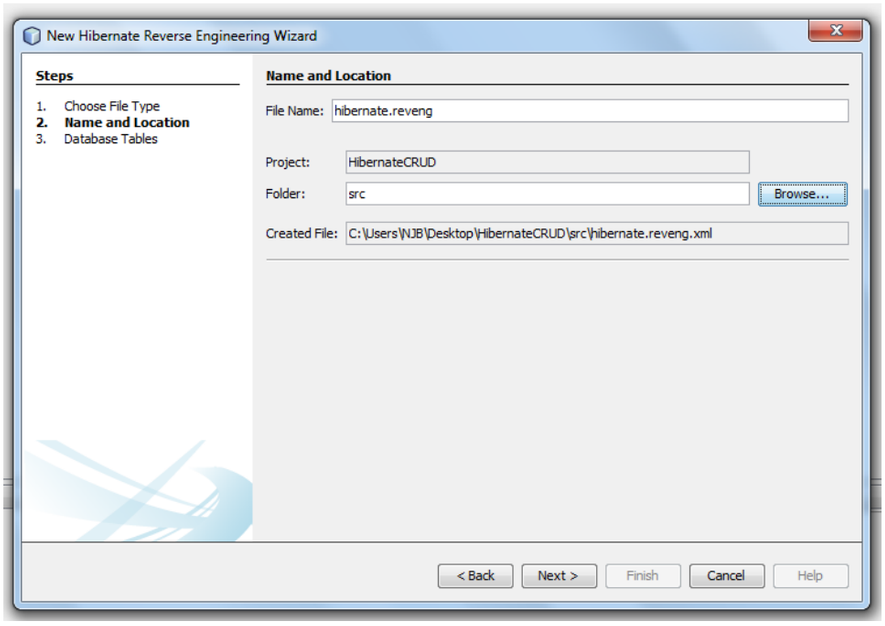


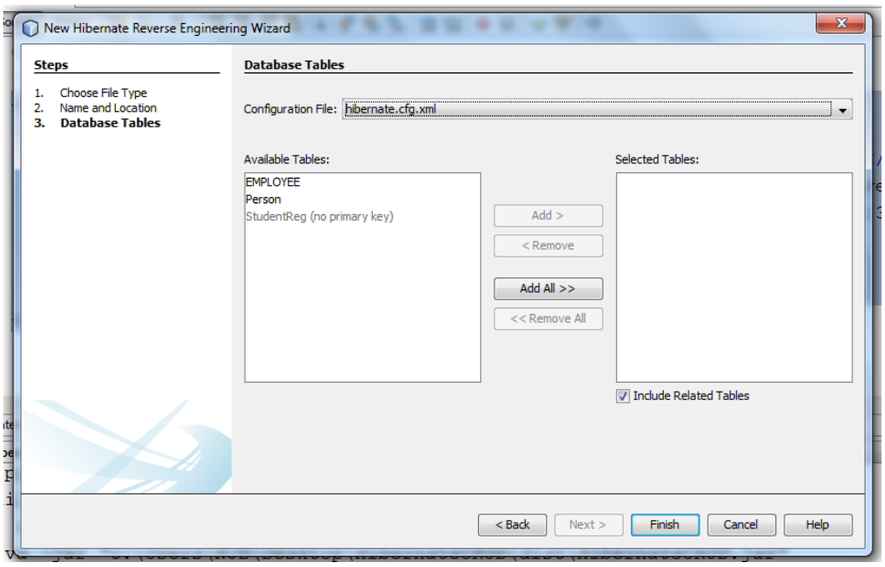
hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
<session-factory>
<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
<property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
<property
name="hibernate.connection.url">jdbc:mysql://localhost:3308/mydb</property>
<property name="hibernate.connection.username">root</property></session-factory>
</hibernate-configuration>
```

2. Add database driver for selected database to the project.

3. Hibernate Reverse Engineering Wizard

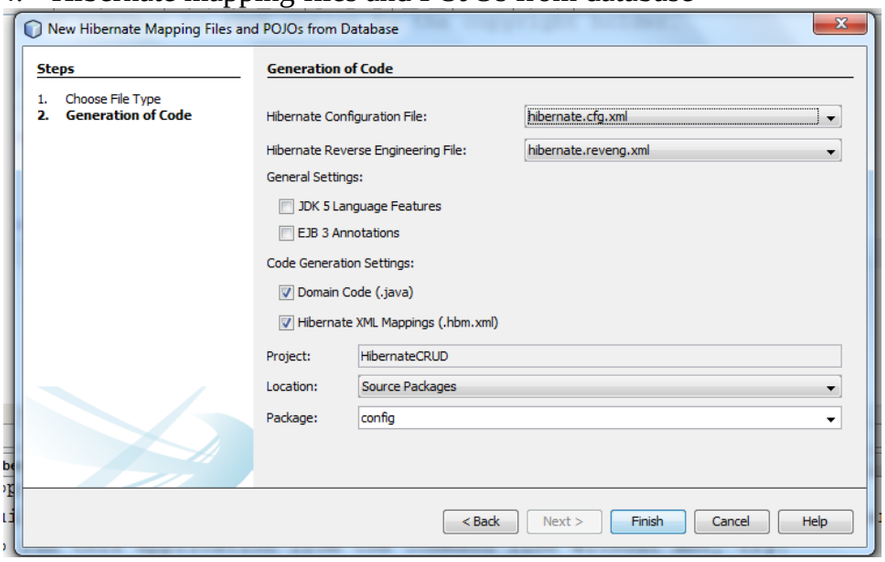




hibernate.reveng.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-reverse-engineering PUBLIC "-//Hibernate/Hibernate Reverse Engineering DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-reverse-engineering-3.0.dtd">
<hibernate-reverse-engineering>
        <schema-selection match-catalog="cenjb"/>
        <table-filter match-name="Person"/>
</hibernate-reverse-engineering>
```

4. Hibernate mapping files and POJOs from database



This file is generate by the wizard. Person.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<!-- Generated Feb 22, 2020 11:52:01 AM by Hibernate Tools 4.3.1 -->

<hibernate-mapping>
   <class name="config.Person" table="Person" catalog="cenjb" optimistic-lock="version">
       <id name="id" type="int">
          <column name="id" />
          <generator class="assigned" />
       </id>
       <property name="name" type="string">
          <column name="name" length="30" not-null="true" />
       </property>
       <property name="hobby" type="string">
          <column name="hobby" length="30" />
       </property>
   </class>
</hibernate-mapping>
```

Person.java

```
package config;
// Generated Feb 22, 2020 11:52:01 AM by Hibernate Tools 4.3.1
/**
 * Person generated by hbm2java
 */
public class Person  implements java.io.Serializable {
	private int id;
	private String name;
	private String hobby;

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

5. HibernateUtil.java

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

/**
 * Hibernate Utility class with a convenient method to get Session Factory
 * object.
 *
 * @author NJB
 */
public class HibernateUtil {
	private static final SessionFactory sessionFactory;
	private static ServiceRegistry serviceRegistry;
	static {
		try {
			// Create the SessionFactory from standard (hibernate.cfg.xml)
			// config file.
			Configuration configuration = new
				Configuration().configure("/config/hibernate.cfg.xml");

			serviceRegistry = new StandardServiceRegistryBuilder()
				.applySettings(configuration.getProperties()).build();

			// builds a session factory from the service registry
			sessionFactory =
				configuration.buildSessionFactory(serviceRegistry);

		} catch (Throwable ex) {
			// Log the exception.
			System.err.println("Initial SessionFactory creation failed." + ex);
			throw new ExceptionInInitializerError(ex);
		}
	}

	public static SessionFactory getSessionFactory() {
		return sessionFactory;
	}

	public static void closeSessionFactory() {
		if (serviceRegistry != null) {
			StandardServiceRegistryBuilder.destroy(serviceRegistry);
		}
	}
}
```

6. Main class for CRUD operations

```
package mypack;
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import utilities.HibernateUtil;

public class CrudOperations {
	public static void main(String[] args) {
		insertRecord(new Person(1, "abc", "reading"));
		insertRecord(new Person(2, "def", "surfing"));
		insertRecord(new Person(4, "pqr", "cooking"));
		insertRecord(new Person(3, "xyz", "travelling"));
		showRecord(1);
		int id = 1;
		String hobby = "sleeping";
		updateRecord(id, hobby);
		showRecord(id);
		int i = 4;
		showRecord(i);
		deleteRecord(i);
		showRecord(i);
		HibernateUtil.closeSessionFactory();
	}

	private static void insertRecord(Person p) {
		SessionFactory sessionFactory = HibernateUtil.getSessionFactory();
		Session session = sessionFactory.openSession();
		session.beginTransaction();
		session.save(p);
		session.getTransaction().commit();
	}

	private static void showRecord(int id) {
		Session session = HibernateUtil.getSessionFactory().openSession();
		Transaction transaction = null;
		try {
			transaction = session.beginTransaction();
			Object get = session.get(Person.class, id);
			Person temp = (Person) get;
			if (temp == null) {
				System.out.println("record not found");
			} else {
				System.out.println(temp);
			}
			transaction.commit();
		} catch (HibernateException e) {
			if (transaction != null) {
				transaction.rollback();
			}
			System.out.println("Error : " + e);
		} finally {
			session.close();
		}
	}
```

```
	private static void updateRecord(int id, String hobby) {
		Session session = HibernateUtil.getSessionFactory().openSession();
		Transaction transaction = null;
		try {
			transaction = session.beginTransaction();
			Person temp = (Person) session.get(Person.class, id);
			if (temp != null) {
				temp.setHobby(hobby);
				session.update(temp);
			} else {
				System.out.println("record not found");
			}
			transaction.commit();
		} catch (HibernateException e) {
			if (transaction != null) {
				transaction.rollback();
			}
			System.out.println("Error : " + e);
		} finally {
			session.close();
		}
	}

	private static void deleteRecord(int id) {
		Session session = HibernateUtil.getSessionFactory().openSession();
		Transaction transaction = null;
		try {
			transaction = session.beginTransaction();
			Person temp = (Person) session.get(Person.class, id);
			if (temp != null) {
				session.delete(temp);
			} else {
				System.out.println("record not found");
			}
			transaction.commit();
		} catch (HibernateException e) {
			if (transaction != null) {
				transaction.rollback();
			}
			System.out.println("Error : " + e);
		} finally {
			session.close();
		}
	}
}
```