Hibernate Hello World example using Maven build tool and SQLite database

By Abhijit Ghosh on Jul 18, 2012 1:05:40 PM 66,373 views

This tutorial will help you to write an annotation based hibernate java program which will save some records into a database and fetch them all using Hibernate API. We will use Maven tool to build the project, Eclipse IDE to code and SQLite database to save / retrieve records. SQLite is a self-contained, serverless, zero-configuration, transactional SQL database engine. To learn hibernate, following this tutorial you do not have to install any database or SQLite database separately. However to browse the database you can use 'SQLite Manager - Firefox addon' which provides a very nice GUI for SQLite database.

Tools and Technologies used in this article :

1. Hibernate Core 4.1
2. Maven
3. SQLite 3 database
4. SQLite Manager - Firefox addon
5. JDK 1.6
6. Eclipse 3.7

Download :

- Maven
- SQLite Manager - Firefox addon
- SQLite database

1. Create a Java Project using Maven Tool

In the command prompt execute the following command to generate Maven compatible Java project named as 'HibernateHelloWorld'.

```
mvn archetype:generate -DgroupId=com.srccodes.example.hibernate -DartifactId=HibernateHelloWorld
```

Generated Maven Java Project structure

2. Update pom.xml

Add dependency of Hibernate core and SQLite jdbc library. Also update 'maven-compiler-plugin' so that it uses compilation level 1.5 onwards. Otherwise annotation (introduced in JDK 5) will not work.

File : pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.srccodes.example.hibernate</groupId>
    <artifactId>HibernateHelloWorld</artifactId>
    <packaging>jar</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>HibernateHelloWorld</name>
    <url>http://maven.apache.org</url>
    <dependencies>
        <!-- hibernate -->
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>4.1.4.Final</version>
```


Similar Popular Recent

Full Text Hibernate Lucene Search Hello World Exa Using Maven and SQLite

Spring 3 Hello World Example / Tutorial using Mave Tool and Eclipse IDE

Maven Build Failure - Hadoop 2.2.0 - [ERROR] class for org.mortbay.component.AbstractLifeCycle not found

Spring 3 MVC Framework Based Hello World Web Application Example Using Maven, Eclipse IDE And Tomcat Server

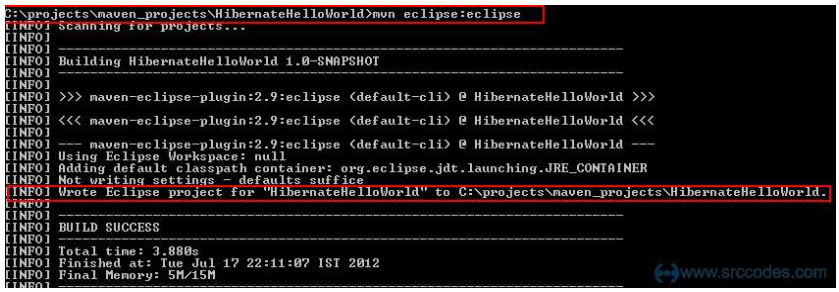
```
        </dependency>

        <!-- SQLite JDBC library -->
        <dependency>
            <groupId>org.xerial</groupId>
            <artifactId>sqlite-jdbc</artifactId>
            <version>3.7.2</version>
        </dependency>

        <!-- junit test -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>1.5</source>
                    <target>1.5</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```


3. Convert to Eclipse compatible Java project

Open the directory 'HibernateHelloWorld' in command prompt and run the following maven command.

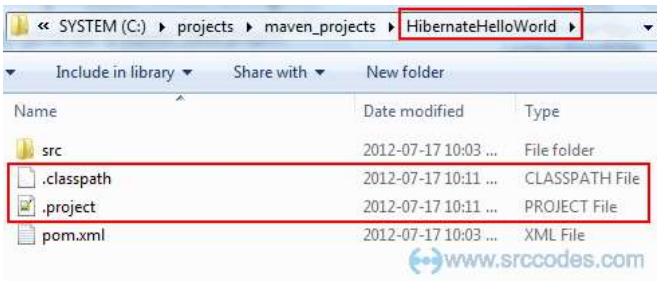
```
mvn eclipse:eclipse
```

Screenshot of command prompt



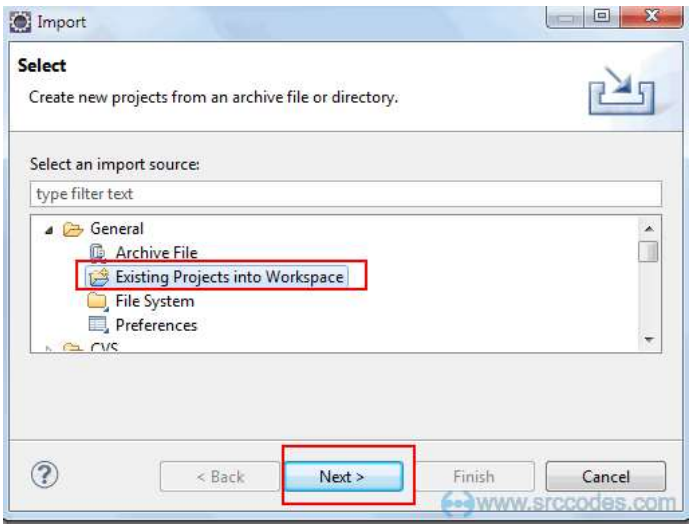
On completion of the above command, Maven Java project will be converted to a Eclipse compatible java project.

Eclipse compatible Java Project structure

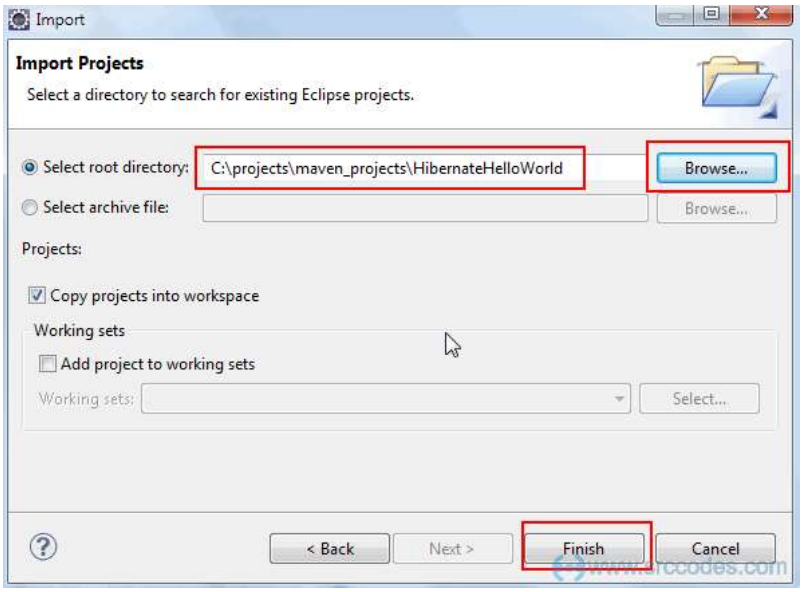


4. Import project in Eclipse

Open Eclipse IDE and select from the menu File --> Import --> General --> Existing Projects into Workspace



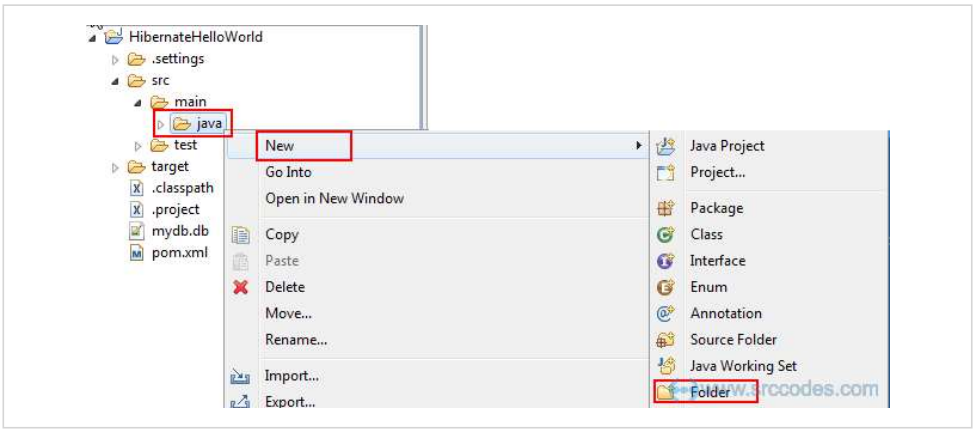
Browse to the directory of the newly converted Eclipse compatible Java Project and click 'Finish' button.



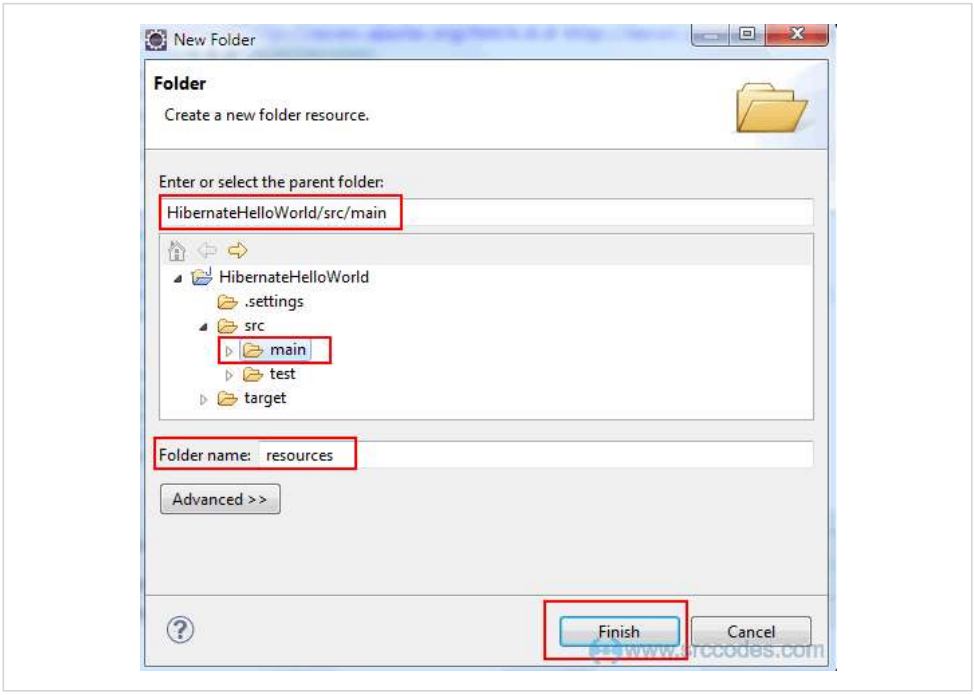
Screenshot of Eclipse project structure

5. Add Hibernate Configuration file

Right click on 'main' and select from context menu 'New' --> 'Folder'.



Enter 'resources' in the 'Folder name' field and click the 'Finish' button.



Copy the 'hibernate.cfg.xml' file in the 'resources' folder.

File: hibernate.cfg.xml

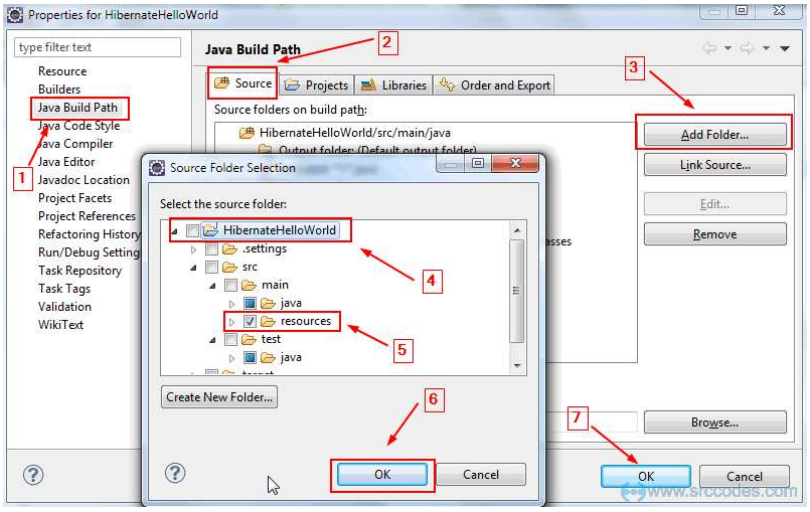
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
    <session-factory>
        <property name="show_sql">true</property>
        <property name="format_sql">true</property>
        <property name="dialect">org.hibernate.dialect.SQLiteDialect</property>
        <property name="connection.driver_class">org.sqlite.JDBC</property>
        <property name="connection.url">jdbc:sqlite:mydb.db</property>
        <property name="connection.username"></property>
        <property name="connection.password"></property>

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

        <mapping class="com.srccodes.example.hibernate.Contact"/>
    </session-factory>
</hibernate-configuration>
```

Note :
'mydb.db' is the SQLite database file included with the sourcecode attached in 'Download Source Code' section. You can create db file of your own using 'SQLite Manager - Firefox addon' UI. But copy that file inside 'HibernateHelloWorld' project directory directly.

I have set the property 'hibernate.hbm2ddl.auto' to 'update' so that when you will execute the code it will create the database tables of it's own based on the entity class 'com.srccodes.example.hibernate.Contact') we have written and referenced in this configuration file.

6. Configure Java Build Path

Right click on 'HibernateHelloWorld' project and select from context menu 'Properties' --> 'Java Build Path'.
Add 'resources' folder as shown in the screenshot below



7. Add SQLiteDialect

Copy from attached source code or download SQLiteDialect and add under package 'org.hibernate.dialect' in your project.

Note :
dialect is used to help hibernate framework to create underlying database specific SQL query.

8. Write Entity class

Create a class 'Contact' under the package 'com.srccodes.example.hibernate' and copy the following content.

```
1  package com.srccodes.example.hibernate;
2
3  import javax.persistence.Entity;
4  import javax.persistence.Id;
5  import javax.persistence.Table;
6
7  /**
8   * The persistent class for the contact database table.
9   *
10  */
11 @Entity
12 @Table(name = "contact")
13 public class Contact {
14     private Integer id;
15     private String name;
16     private String email;
17
18     public Contact() {
19
20     }
21
22     public Contact(Integer id, String name, String email) {
23         this.id = id;
24         this.name = name;
25         this.email = email;
26     }
27
28     @Id
29     public Integer getId() {
30         return this.id;
31     }
32
33     public void setId(Integer id) {
34         this.id = id;
35     }
36
37     public String getName() {
38         return this.name;
39     }
40
41     public void setName(String name) {
42         this.name = name;
```

```
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

As we have set 'hibernate.hbm2ddl.auto' property in 'hibernate.cfg.xml' file, hibernate will generate the schema for the first time and for every change of the schema based on the annotation we write in the 'Contact' class.

Annotation '@Table(name = "contact")' will create a table named as'contact' in the SQLite database. As 'id' field of 'Contact' class is annotated as '@Id' so 'id' will be the primary key for the generated table 'contact'.

9. Interact with database using Hibernate API

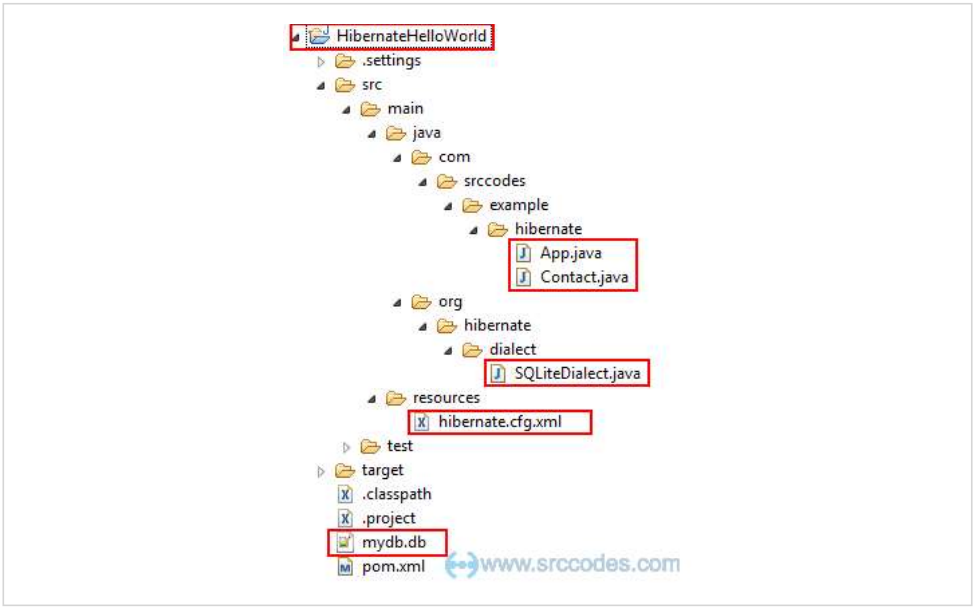
Copy the following code to 'App' class of package 'com.srccodes.example.hibernate'.

```
package com.srccodes.example.hibernate;

import java.util.List;
import java.util.Properties;

import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;
import org.hibernate.service.ServiceRegistry;
import org.hibernate.service.ServiceRegistryBuilder;

/**
 * Hello world!
 *
 */
public class App {
    private static SessionFactory sessionFactory = null;
    private static ServiceRegistry serviceRegistry = null;

    private static SessionFactory configureSessionFactory() throws HibernateException {
        Configuration configuration = new Configuration();
        configuration.configure();

        Properties properties = configuration.getProperties();

        serviceRegistry = new ServiceRegistryBuilder().applySettings(properties).buildServiceRe
        sessionFactory = configuration.buildSessionFactory(serviceRegistry);

        return sessionFactory;
    }

    public static void main(String[] args) {
        // Configure the session factory
        configureSessionFactory();

        Session session = null;
        Transaction tx=null;

        try {
            session = sessionFactory.openSession();
            tx = session.beginTransaction();

            // Creating Contact entity that will be save to the sqlite database
            Contact myContact = new Contact(3, "My Name", "my_email@email.com");
            Contact yourContact = new Contact(24, "Your Name", "your_email@email.com");

            // Saving to the database
            session.save(myContact);
            session.save(yourContact);

            // Committing the change in the database.
            session.flush();
            tx.commit();

            // Fetching saved data
            List<Contact> contactList = session.createQuery("from Contact").list();

            for (Contact contact : contactList) {
                System.out.println("Id: " + contact.getId() + " | Name:"  + contact.getName() +
            }

        } catch (Exception ex) {
            ex.printStackTrace();

            // Rolling back the changes to make the data consistent in case of any failure
            // in between multiple database write operations.
            tx.rollback();
        } finally{
            if(session != null) {
                session.close();
            }
        }
    }
```

```
76 }
```

10. Final project structure

After doing all the changes the overall project structure will look like this

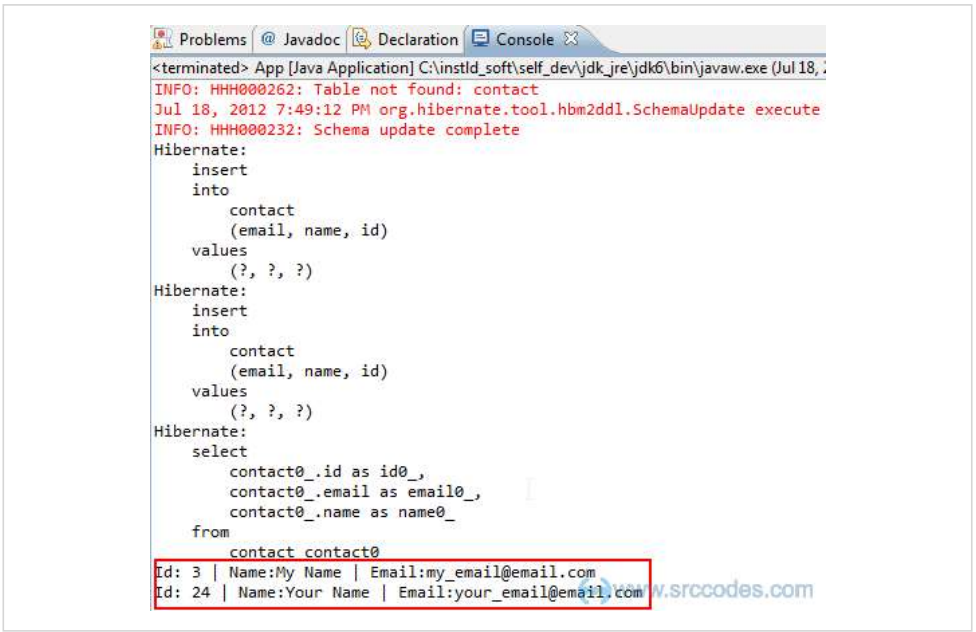


11. Run Your Code

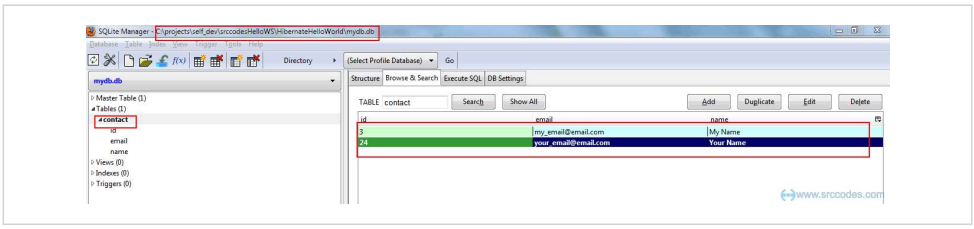
Right click on 'App.java' and select from context menu 'Run As' --> 'Java Application'.

12. Console Output

Now code will save two records in the database and fetch them to print in the console as well.



Screenshot of the table using 'SQLite Manager - Firefox addon' UI



Download Source Code

SrcCodes : HibernateHelloWorld.zip

References

- Hibernate JavaDoc
- Hibernate Developer Guide
- Maven Getting Started Guide
- SQLite Documents

Tags: maven sqlite-manager sqlite hibernate-core

Share This: 12

About the Author

Abhijit Ghosh

A java/j2ee developer. Love to share learning experiences on different technologies/frameworks.

Find me on

Note: To post source code in comment, use <pre><code>. For example: <pre><code>String message = "Hello World!";</code></pre>