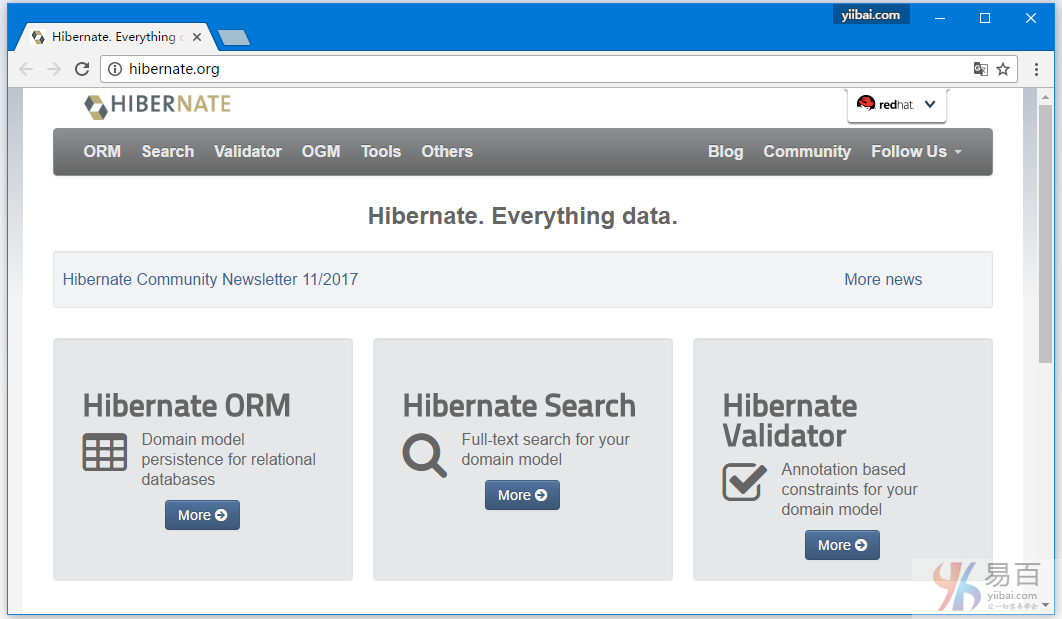
**Hibernate实例**

这个Hibernate教程通过简化的例子提供了Hibernate框架的深入概念。Hibernate是Gavin King于2001年开始，作为EJB2风格实体bean的替代品开发的对象关系映射工具。 Hibernate的稳定版本到目前(2017年3月)是[Hibernate 5.2.9](http://hibernate.org/orm/downloads/)。 这个Hibernate教程致力于提供Hibernate技术的简单和深入学习。没有什么东西在这个世界是完美的，也没有什么是永远最好的。 但我们可以尝试去作得更好，希望本教程会在您学习Hibernate技术路上帮助到你很多。

注：如果你想要节省时间，快速学习Hibernate,请参考快速入门文章：<http://www.yiibai.com/hibernate/hibernate-quick-guide.html>

Hibernate官方网站是：<http://hibernate.org/> ，有关 **Hibernate** 的资讯，文档，软件库等都可以这个网站上找到。



## Hibernate框架

Hibernate框架简化了java应用程序与数据库交互的开发。 Hibernate是一个开源，轻量级的ORM(对象关系映射)工具。

ORM工具简化了数据创建，数据处理和数据访问。它是将对象映射到数据库中存储的数据(表)的编程技术。



ORM工具内部使用JDBC API与数据库进行交互。

## Hibernate框架的优点

Hibernate框架有很多优点。它们分别如下：

* **开源和轻量级：** Hibernate框架是根据LGPL许可证和轻量级的开源工具。
* **快速性能：** Hibernate框架的性能很快，因为缓存在Hibernate框架内部使用。 hibernate框架中有两种类型的缓存：一级缓存和二级缓存。一级缓存默认是启用的。
* **数据库独立查询：** HQL(Hibernate查询语言)是面向对象的SQL版本。 它生成数据库独立查询。 所以你不需要编写数据库特定的查询语句。 在Hibernate之前，如果项目更改了数据库，我们需要更改SQL查询，从而导致维护变得非常复杂。
* **自动创建表：** Hibernate框架提供了自动创建数据库表的功能。 因此，无需手动在数据库中创建表。
* **简化复杂连接：** 在hibernate框架中可轻松获取多个表中的数据。
* **提供查询统计和数据库状态：** Hibernate支持查询缓存，并提供有关查询和数据库状态的统计信息。

# Hibernate体系结构

Hibernate架构包括许多对象持久对象，会话工厂，事务工厂，连接工厂，会话，事务等。

hibernate架构中有4层：Java应用层，hibernate框架层，反射api层和数据库层。请参见hibernate架构图：



这是Hibernate的高级架构，具有映射文件和配置文件。



Hibernate框架使用许多对象会话工厂，会话，事务等以及现有的Java API，如JDBC(Java数据库连接)，JTA(Java事务API)和JNDI(Java命名目录接口)。

### Hibernate体系结构的要素

要创建第一个hibernate应用程序，我们必须知道Hibernate架构的元素。 它们如下：

**会话工厂(SessionFactory)**

SessionFactory是ConnectionProvider的会话和客户端工厂。 它拥有数据的二级缓存(可选)。 org.hibernate.SessionFactory接口提供了工厂方法来获取Session的对象。

**会话(Session)**  
Session对象提供应用程序和存储在数据库中的数据之间的接口。 它是一个短生命周期的对象并包装JDBC连接。 它是事务，查询和标准的工厂。 它拥有一级缓存(强制性)数据。 org.hibernate.Session接口提供插入，更新和删除对象的方法。 它还提供了事务，查询和标准的工厂方法。

**事务(Transaction)**  
事务对象指定工作的原子单位,它是一个可选项。 org.hibernate.Transaction接口提供事务管理的方法。

**连接提供者(ConnectionProvider)**

它是一个JDBC连接工厂。 它从DriverManager或DataSource抽象出来的应用程序。 它是一个可选项。

**事务工厂(TransactionFactory)**

它是一个事务工厂，是一个可选项。

# Hibernate入门程序

在第一个Hibernate入门示例程序中，我们将使用eclipse IDE创建一个简单的hibernate应用程序示例。要在Eclipse IDE中创建第一个hibernate应用程序，需要遵循以下步骤：

重要提示：  
按照惯例，易百教程的每个入门教程程序都会详细介绍对应环境的安装，配置和运行步骤，以帮助读者/学习者快速入门。往后的系列教程中没有特别说明，都会以入门教程程序配置为参考。所以，在您阅读每一篇文章遇到配置或运行环境不清楚，请参考入门教程程序配置。

1. 创建java项目
2. 为hibernate添加jar文件
3. 创建持久类
4. 创建持久类的映射文件
5. 创建配置文件
6. 创建检索或存储持久对象的类
7. 运行应用程序

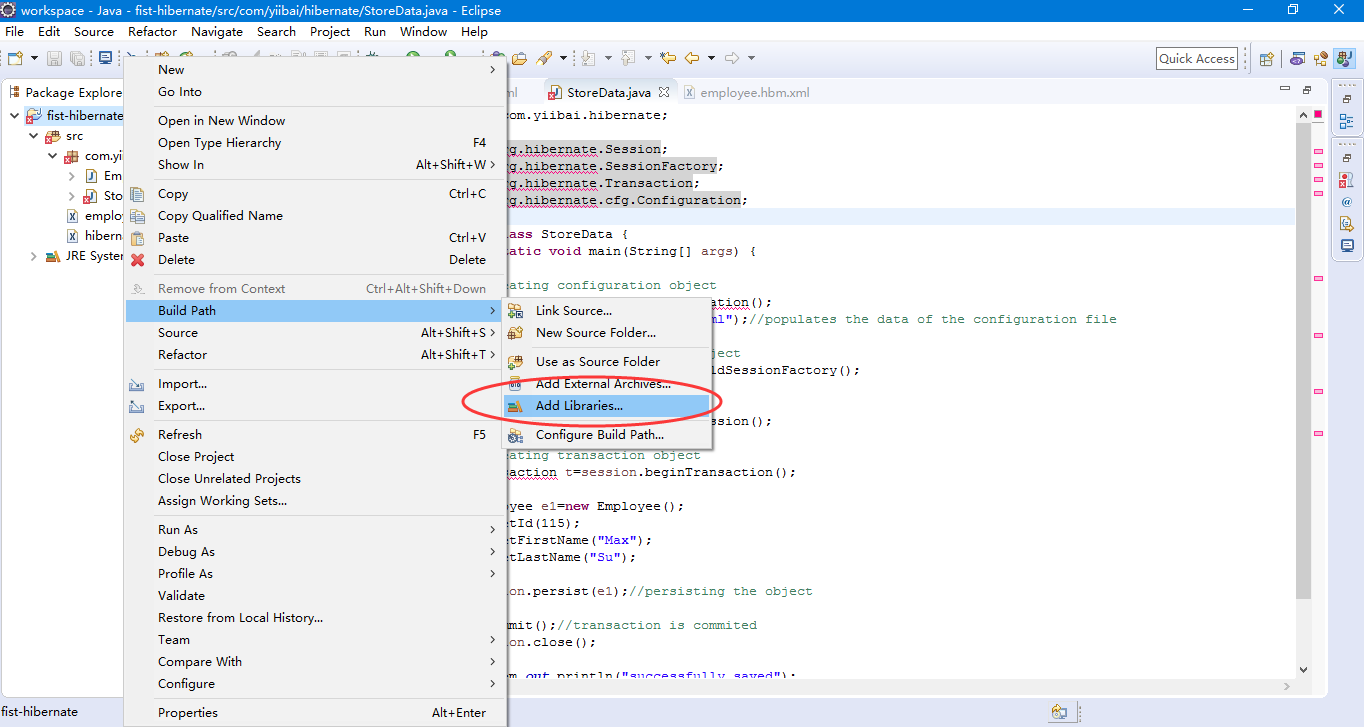
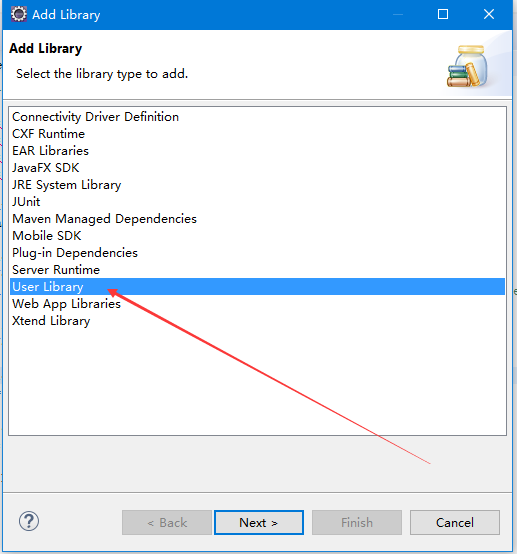
前提工作:

1. 安装好 MySQL 并启动。参考： <http://www.yiibai.com/mysql/>
2. Eclipse Neon.2 Release (4.6.2)。 参考：<http://www.yiibai.com/eclipse/>
3. 下载 Hibernate 相关类库(Jar文件)
4. 创建一个数据库表：tb\_employee

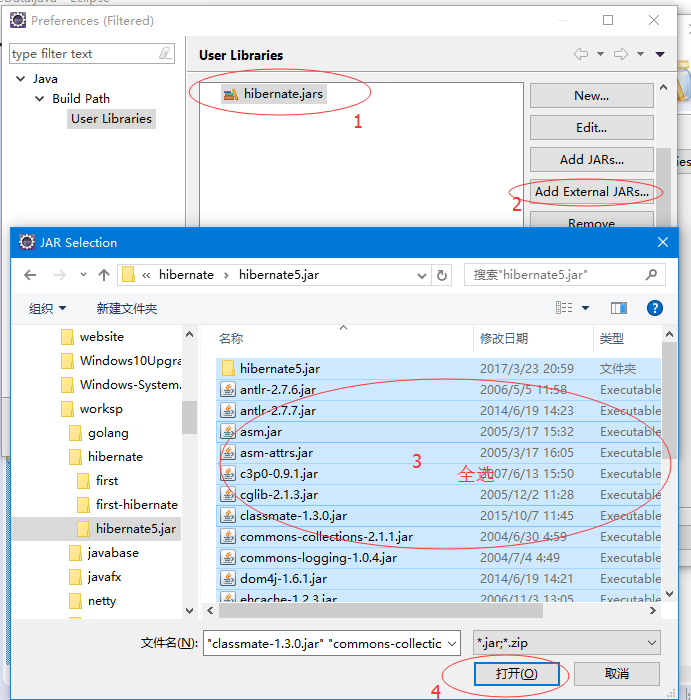
## 1. 创建java项目

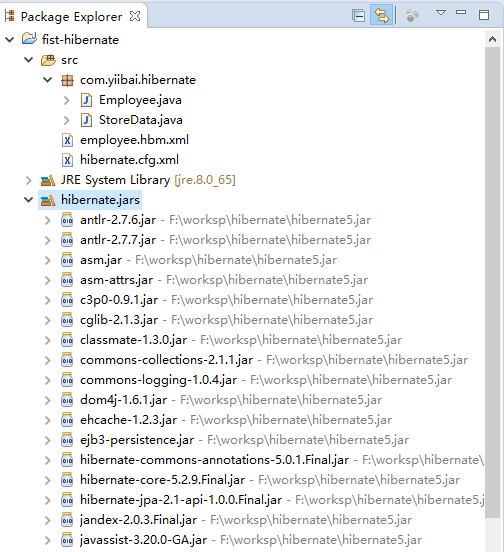
打开Eclipse，通过 **File** -> **New** -> **project** -> **java project** 创建java项目。 现在指定项目名称: first-hibernate， 然后 **next->** 完成。

## 2. 为hibernate添加jar文件

添加jar文件右键单击您的项目(first-hibernate) -> Build Path -> 添加外部存档。  
  
现在选择所有的jar文件，如下图所示 -  


创建一个用户自定义库，为这个自定义库起个名字：hibernate-jars，并选择 hibernate 相关jar包。



完成后加入效果如下所示 -  


下载所需的 [Hibernate-jars](http://www.yiibai.com/src/hb/hibernatejar.zip)文件：

在这个例子中，我们将应用程序与[MySQL](http://www.yiibai.com/mysql/)数据库连接。 所以你必须添加mysql-connect.jar文件。

## 3. 创建持久化类

在这里，要创建持久化类，右键单击src -> New -> Class - 使用包名指定类(例如：com.yiibai.mypackage) -> finish。

Employee.java

package com.yiibai.mypackage;

public class Employee {

private int id;

private String firstName,lastName;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

}

Java

## 4. 创建持久化类的映射文件

在这里，我们正在创建与上一主题中创建的相同的映射文件。 要创建映射文件，右键单击src -> new -> file -> 指定文件名(例如employee.hbm.xml) , 它必须在包外部。

**employee.hbm.xml**

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

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name="com.yiibai.mypackage.Employee" table="emp1000">

<id name="id">

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

</id>

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

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

</class>

</hibernate-mapping>

XML

## 5. 创建配置文件

配置文件包含数据库的所有信息，如：connection\_url，driver\_class，username，password等。hbm2ddl.auto属性用于自动在数据库中创建表。 我们将在下一个主题中深入学习Dialect类。 要创建配置文件，请右键单击src -> new -> file。 现在指定配置文件名，例如： hibernate.cfg.xml。

**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">

<!-- Generated by MyEclipse Hibernate Tools. -->

<hibernate-configuration>

<session-factory>

<property name="connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="connection.url">jdbc:mysql://localhost:3306/test</property>

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

<property name="connection.password">123456</property>

<property name="dialect">org.hibernate.dialect.MySQL5InnoDBDialect</property>

<property name="show\_sql">true</property>

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

</session-factory>

</hibernate-configuration>

XML

## 6. 创建检索或存储持久对象的类

在这个类中，我们只是将employee对象存储到数据库中。

package com.yiibai.mypackage;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

public class StoreData {

public static void main(String[] args) {

//creating configuration object

Configuration cfg=new Configuration();

cfg.configure("hibernate.cfg.xml");//populates the data of the configuration file

//creating seession factory object

SessionFactory factory=cfg.buildSessionFactory();

//creating session object

Session session=factory.openSession();

//creating transaction object

Transaction t=session.beginTransaction();

Employee e1=new Employee();

e1.setId(100);

e1.setFirstName("Max");

e1.setLastName("Su");

session.persist(e1);//persisting the object

t.commit();//transaction is committed

session.close();

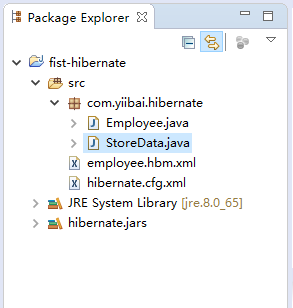
System.out.println("successfully saved");

}

}

Java

## 7. 运行应用程序

在运行应用程序之前，确定目录结构如下图所示 -  


在运行这个示例之前，还差关键的一个步骤，那就是创建对象对应的表：tb\_employee,其结构及创建语句语句如下-

CREATE TABLE `tb\_employee` (

`id` int(10) unsigned NOT NULL AUTO\_INCREMENT,

`firstName` varchar(32) NOT NULL DEFAULT '',

`lastName` varchar(32) NOT NULL DEFAULT '',

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=101 DEFAULT CHARSET=utf8;

SQL

接下来，就是运行hibernate应用程序，请右键单击StoreData类 -Run As - Java应用程序。

输入结果如下 -

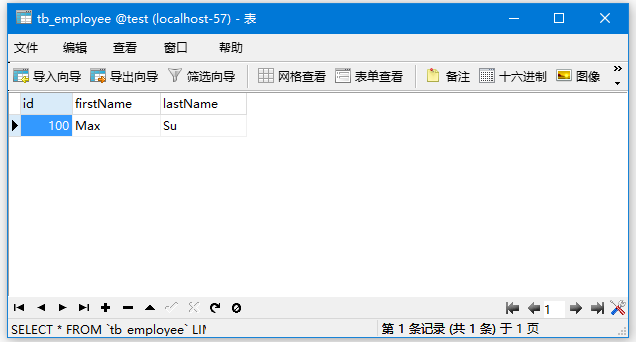
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect

Hibernate: insert into tb\_employee (firstName, lastName, id) values (?, ?, ?)

successfully saved

Shell

查看 tb\_employee 表中的数据，如下 -

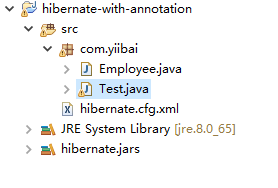


# Hibernate使用注释(注解)

可以使用注释创建hibernate应用程序。 有许多注释可用于创建hibernate应用程序，如[@Entity](https://github.com/Entity)，[@Id](https://github.com/Id)，[@Table](https://github.com/Table)等。

Hibernate注释基于JPA 2规范（me：jpa规范了需要的数据库操作的注解），并支持所有功能。  
所有JPA注释都在javax.persistence.\*包中定义。 Hibernate EntityManager实现由JPA规范定义的接口和生命周期。

使用hibernate注释的核心优点是我们不需要创建映射(\*.hbm.xml)文件直接创建对象关联。 在这里，hibernate注释用于提供元数据。

完整的项目结构图如下 -  


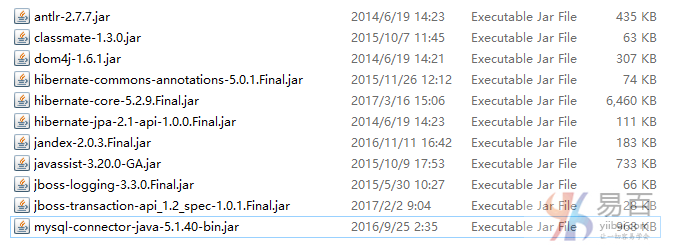
## 使用注释创建hibernate应用程序的示例

创建具有注释的hibernate应用程序有4个步骤。分别如下 -

* 为MySQL添加连接驱动程序的jar文件(如果数据库是MySQL)和注释
* 创建持久化类
* 在配置文件中添加持久化类的映射
* 创建检索或存储持久对象的类

## 1. 为MySQL和注释添加jar文件

对于MySQL，您需要添加mysql-connector-java-5.1.40-bin.jar文件，以及 hibernate5的核心类库，如下所示 -



## 2. 创建持久化类

在这里，我们创建一个持久化类: Employee 。 并使用注释完成与数据库表：tb\_employee 的映射关联。

[@Entity](https://github.com/Entity)注释将此类标记为实体。  
[@Table](https://github.com/Table)注释指定要保留此实体的数据的表名。 如果不使用[@Table](https://github.com/Table)注释，hibernate将使用类名作为表名称by default。  
[@Id](https://github.com/Id)注释标记此实体的标识符。  
[@Column](https://github.com/Column)注释指定此属性或字段的列的详细信息。如果未指定[@Column](https://github.com/Column)注释，则属性名称将用作列名称by default。

Employee.java 文件的代码如下 -

package com.yiibai;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name = "tb\_employee")

public class Employee {

@Id

private int id;

private String firstName, lastName;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

}

Java

## 3. 在配置文件中添加持久化类的映射

打开hibernate.cfg.xml文件，并添加如下的映射资源条目：

<mapping class="com.yiibai.Employee"/>

XML

现在完整的配置文件: hibernate.cfg.xml，将如下所示：

<?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">

<!-- Generated by MyEclipse Hibernate Tools. -->

<hibernate-configuration>

<session-factory>

<property name="connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="connection.url">jdbc:mysql://localhost:3306/test</property>

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

<property name="connection.password">123456</property>

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

<property name="show\_sql">true</property>

<mapping class="com.yiibai.Employee"/>

</session-factory>

</hibernate-configuration>

XML

## 4. 创建检索或存储持久对象的类

在这个类中，我们只是将Employee类的对象存储到数据库中。 在这里，我们使用StandardServiceRegistryBuilder类和MetadataSources类从持久化类获取映射的信息。

Test.java 测试类代码如下 -

package com.yiibai;

import org.hibernate.\*;

import org.hibernate.boot.MetadataSources;

import org.hibernate.boot.registry.StandardServiceRegistry;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import org.hibernate.cfg.\*;

public class Test {

public static void main(String[] args) {

//Session session = new AnnotationConfiguration().configure().buildSessionFactory().openSession();

final StandardServiceRegistry registry = new StandardServiceRegistryBuilder().configure("hibernate.cfg.xml")

.build();

// 2. 根据服务注册类创建一个元数据资源集，同时构建元数据并生成应用一般唯一的session工厂

SessionFactory sessionFactory = new MetadataSources(registry).buildMetadata().buildSessionFactory();

/\*\*\*\* 上面是配置准备，下面开始我们的数据库操作 \*\*\*\*\*\*/

Session session = sessionFactory.openSession();// 从会话工厂获取一个session

Transaction t = session.beginTransaction();

Employee e1 = new Employee();

e1.setId(1001);

e1.setFirstName("Yii");

e1.setLastName("Bai");

Employee e2 = new Employee();

e2.setId(1002);

e2.setFirstName("Min");

e2.setLastName("Su");

session.persist(e1);

session.persist(e2);

t.commit();

session.close();

System.out.println("successfully saved");

}

}

Java

运行上面(Test.java)示例，得到以下结果 -

... ...

INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect

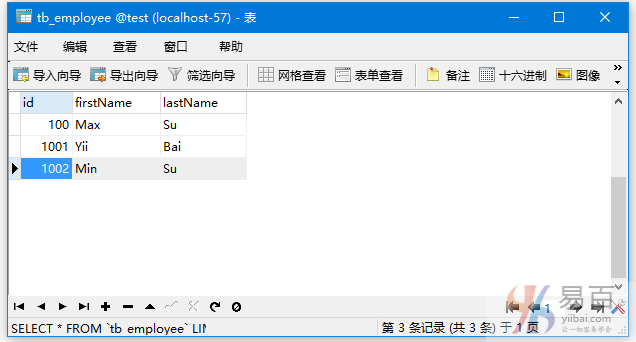
Hibernate: insert into tb\_employee (firstName, lastName, id) values (?, ?, ?)

Hibernate: insert into tb\_employee (firstName, lastName, id) values (?, ?, ?)

successfully saved

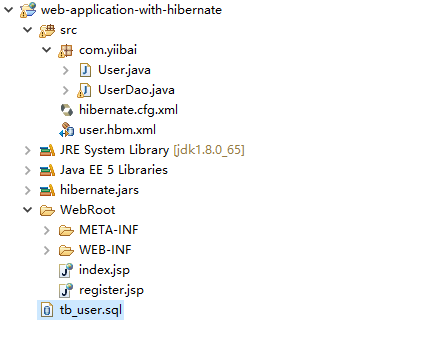
Shell

查看数据库表 tb\_employee 中的数据，应该会看到以下结果 -



# Web应用程序使用Hibernate

在本文中，我们将学习使用hibernate创建一个Web应用程序。 对于创建Web应用程序，我们使用JSP表示表现层，使用Bean类表示数据，以及使用DAO类操作数据库。  
在hibernate中创建简单的应用程序时，不需要在hibernate中执行额外的操作来创建Web应用程序。 在这个示例中，我们使用JSP文件获取用户的值。

首先打开 MyEclipse ，创建一个动态Web项目：web-application-with-hibernate，其结果如下 -  


## 使用hibernate创建Web应用程序的示例

在这个例子中，我们将在数据库中插入用户的记录。它只是实现一个简单的注册表单的处理，即：从表单获取用户的输入信息，并将数据插入到对应的数据表中。

**index.jsp**

该页面用于从用户获取输入，并使用post方法将其发送到register.jsp文件。

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>注册表单</title>

</head>

<body>

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

<h2>用户注册表单</h2>

<hr/>

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

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

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

<br /> <input type="submit" value="注册" />

</form>

</body>

</html>

HTML

**register.jsp**

这个文件获取所有请求参数，并将此信息存储到User类的对象中。 此外，它调用UserDao类的 register() 方法传递User类对象。

<%@page import="com.yiibai.UserDao"%>

<jsp:useBean id="obj" class="com.yiibai.User"></jsp:useBean>

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

<%

int i = UserDao.save(obj);

if (i > 0) {

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

}

%>

Jsp

**User.java**

它是在hibernate中表示持久类的简单bean类。

package com.yiibai;

public class User {

private int id;

private String name, password, email;

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;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

}

Java

**user.hbm.xml**

它用于与User类与数据库的表映射。

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

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name="com.yiibai.User" table="tb\_user">

<id name="id">

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

</id>

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

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

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

</class>

</hibernate-mapping>

XML

**UserDao.java**

一个Dao类，包含存储User类的实例的方法。

package com.yiibai;

import org.hibernate.\*;

import org.hibernate.boot.MetadataSources;

import org.hibernate.boot.registry.StandardServiceRegistry;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import org.hibernate.cfg.\*;

public class UserDao {

public static int save(User u) {

int i = 0;

final StandardServiceRegistry registry = new StandardServiceRegistryBuilder()

.configure("hibernate.cfg.xml").build();

// 2. 根据服务注册类创建一个元数据资源集，同时构建元数据并生成应用一般唯一的session工厂

SessionFactory sessionFactory = new MetadataSources(registry)

.buildMetadata().buildSessionFactory();

Session session = sessionFactory.openSession();// 从会话工厂获取一个session

Transaction t = session.beginTransaction();

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

t.commit();

session.close();

System.out.println(u.getName());

return i;

}

}

Java

**创建数据库表**

创建一个与持久化User类对象关联的表：tb\_user ， 其结构和创建语句如下 -

CREATE TABLE `tb\_user` (

`id` int(10) unsigned NOT NULL AUTO\_INCREMENT,

`name` varchar(32) NOT NULL DEFAULT '',

`password` varchar(32) NOT NULL DEFAULT '',

`email` varchar(32) NOT NULL DEFAULT '',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

SQL

**hibernate.cfg.xml**

它是一个配置文件，其中包含有关数据库和映射文件的信息。

<?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">

<!-- Generated by MyEclipse Hibernate Tools. -->

<hibernate-configuration>

<session-factory>

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

<property name="connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="connection.url">jdbc:mysql://localhost:3306/test</property>

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

<property name="connection.password">123456</property>

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

<property name="show\_sql">true</property>

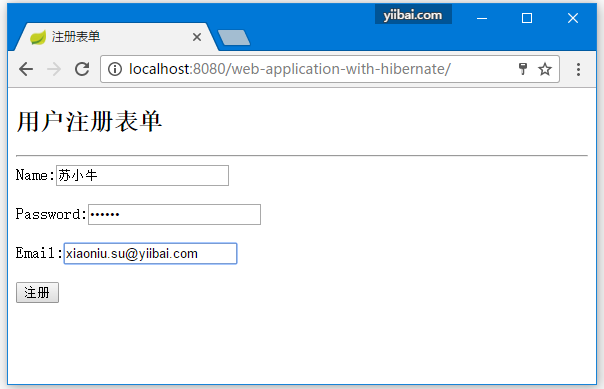
<mapping resource="user.hbm.xml"/>

</session-factory>

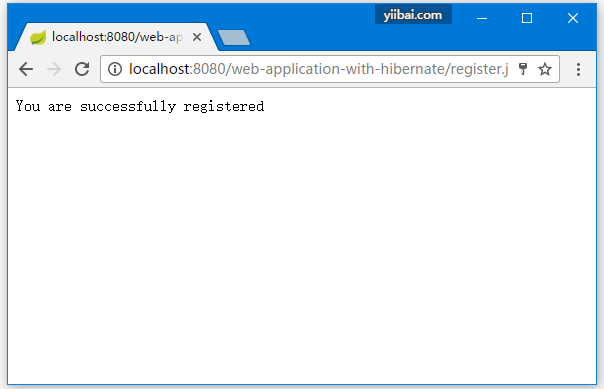
</hibernate-configuration>

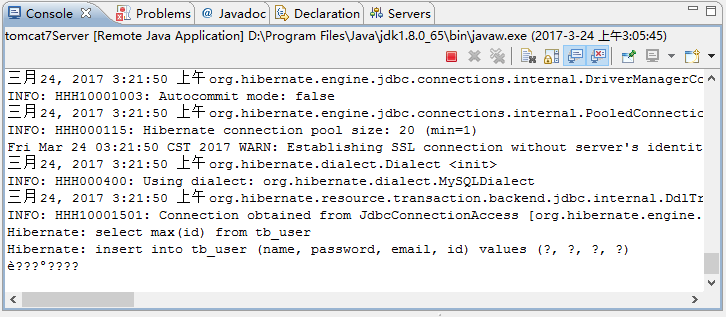
XML

发布上面的工程到 Tomcat Sever中，打开浏览器访问URL：<http://localhost:8080/web-application-with-hibernate/> , 没有错误应该会看到下面界面并写入相关内容 -



提交后，看到以下结果 -



MyEclipse终端输出结果如下 -  


打开数据库，查看 tb\_user 表应该会看到刚才插入的数据。

# Hibernate生成器类

在Hibernate中，id元素的<generator>子元素用于生成持久化类的对象的唯一标识符。 Hibernate框架中定义了许多生成器类。

所有的生成器类都实现了org.hibernate.id.IdentifierGenerator接口。 应用程序员可以通过实现IdentifierGenerator接口来创建自己的生成器类。 Hibernate框架提供了许多内置的生成器类：

1. assigned
2. increment
3. sequence
4. hilo
5. native
6. identity
7. seqhilo
8. uuid
9. guid
10. select
11. foreign
12. sequence-identity

## 1. assigned

如果没有使用<generator>元素，assigned是默认的生成器策略。在这种情况下，应用程序为对象分配ID。 例如：

....

<hibernate-mapping>

<class ...>

<id ...>

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

</id>

.....

</class>

</hibernate-mapping>

XML

## 2. increment

当没有其他进程将数据插入此表时，才会生成唯一的ID。 它生成short，int或long型标识符。 第一个生成的标识符通常为1，然后每次递增为1。语法：

....

<hibernate-mapping>

<class ...>

<id ...>

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

</id>

.....

</class>

</hibernate-mapping>

XML

## 3. sequence

它使用数据库的顺序序列。如果没有定义序列，它会自动创建一个序列。 在Oracle数据库的情况下，它将创建一个名为HIBERNATE\_SEQUENCE的序列。 在Oracle，DB2，SAP DB，Postgre SQL或McKoi的情况下，它使用序列(sequence)，但在interbase中使用生成器。  
**语法：**

.....

<id ...>

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

</id>

.....

Java

要定义自己的序列，请使用generator的 param 子元素。

.....

<id ...>

<generator class="sequence">

<param name="sequence">your\_sequence\_name</param>

</generator>

</id>

.....

XML

## 4. hilo

它使用高低算法来生成short，int和long类型的id。 **语法**：

.....

<id ...>

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

</id>

.....

Java

## 5. native

它使用标识，序列或希洛取决于数据库供应商。 **语法**：

.....

<id ...>

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

</id>

.....

Java

## 6. identity

它用于Sybase，Mysql，MS SQL Server，DB2和Hypersonic SQL以支持id列。 返回的ID类型为short，int或long。

## 7. seqhilo

它在指定的序列名称上使用高低算法。 返回的ID类型为short，int或long。

## 8. uuid

它使用128位UUID算法生成id。 返回的ID是String类型，在网络中是唯一的(因为使用了IP)。 UUID以十六进制数字表示，长度为32。

## 9. guid

它使用由字符串类型的数据库生成的GUID。 它适用于MS SQL Server和MySQL。

## 10. select

它使用数据库触发器返回主键。

## 11. foreign

它使用另一个关联对象的id，主要用于**<一对一>**关联。

## 12. sequence-identity

它使用特殊的序列生成策略。 仅在Oracle 10g驱动程序中支持。

# Hibernate使用Log4j日志记录(使用xml文件)

日志记录使程序员能够将日志详细信息永久写入文件。 [Log4j](http://www.yiibai.com/log4j/)和Logback框架可以在hibernate框架中使用来支持日志记录。

使用log4j执行日志记录有两种方法：

* 通过log4j.xml文件(或)
* 通过log4j.properties文件

## 使用xml文件执行Log4j执行Hibernate日志记录的步骤

使用xml文件使用log4j执行日志记录有两种方法：

1. 使用hibernate加载log4j的jar文件
2. 在src文件夹内创建log4j.xml文件(与hibernate.cfg.xml文件同个目录)

## 使用xml文件的Log4j进行Hibernate日志记录的示例

您可以通过在hibernate示例中执行两个步骤来在hibernate中启用日志记录。 这是使用log4j进行日志记录支持的hibernate应用程序的第一个例子。

## 加载所需的jar文件

您需要使用hibernate的jar文件和加载slf4j.jar和log4j.jar文件([http://logging.apache.org/log4j/1.2/)。](http://logging.apache.org/log4j/1.2/)%E3%80%82)

## 创建log4j.xml文件

现在您需要创建log4j.xml文件。 在此示例中，所有日志详细信息将被写入当前项目的log4j\_hn.log文件。

log4j.xml文件的内容如下

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

<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">

<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/"

debug="false">

<appender name="CONSOLE" class="org.apache.log4j.ConsoleAppender">

<layout class="org.apache.log4j.PatternLayout">

<param name="ConversionPattern" value="[%d{dd/MM/yy hh:mm:ss:sss z}] %5p %c{2}: %m%n" />

</layout>

</appender>

<appender name="ASYNC" class="org.apache.log4j.AsyncAppender">

<appender-ref ref="CONSOLE" />

<appender-ref ref="FILE" />

</appender>

<appender name="FILE" class="org.apache.log4j.RollingFileAppender">

<param name="File" value="log4j\_hn.log" />

<param name="MaxBackupIndex" value="100" />

<layout class="org.apache.log4j.PatternLayout">

<param name="ConversionPattern" value="[%d{dd/MM/yy hh:mm:ss:sss z}] %5p %c{2}: %m%n" />

</layout>

</appender>

<category name="org.hibernate">

<priority value="DEBUG" />

</category>

<category name="java.sql">

<priority value="debug" />

</category>

<root>

<priority value="INFO" />

<appender-ref ref="FILE" />

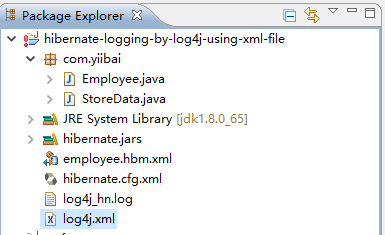
</root>

</log4j:configuration>

XML

此示例的项目名称：hibernate-logging-by-log4j-using-xml-file，请到下面的下载页面中下载代码。

完整的项目目录结构如下 -



# Hibernate使用Log4j日志记录(使用properties文件)

我们知道，Log4j和Logback框架可用于支持日志记录hibernate，使用log4j有两种执行日志记录的方法：

1. 通过log4j.xml文件(或)
2. 通过log4j.properties文件

在这里，我们将使用log4j通过属性(properties)文件启用日志记录。

## 使用属性文件执行Log4j执行Hibernate日志记录的步骤

使用属性(properties)文件使用log4j执行日志记录有两种方法：

1. 使用hibernate加载log4j jar文件
2. 在src文件夹内创建log4j.properties文件(与hibernate.cfg.xml文件同个目录)

## 使用属性文件通过Log4j进行Hibernate日志记录的示例

您可以通过在任何hibernate示例中执行两个步骤来在hibernate中启用日志记录。 这是使用log4j进行日志记录支持的hibernate应用程序的第二个例子。

## 加载所需的jar文件

需要使用hibernate的jar文件和加载slf4j.jar和log4j.jar文件。这些Jar文件请参考下载页面下载。

## 创建log4j.properties文件

现在创建log4j.properties文件。 在此示例中，所有日志详细信息将被写入yiibai-log4j.log文件中。

log4j.properties文件的内容如下 -

# Direct log messages to a log file

log4j.appender.file=org.apache.log4j.RollingFileAppender

log4j.appender.file.File=yiibai-log4j.log

log4j.appender.file.MaxFileSize=1MB

log4j.appender.file.MaxBackupIndex=1

log4j.appender.file.layout=org.apache.log4j.PatternLayout

log4j.appender.file.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L - %m%n

# Direct log messages to stdout

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.Target=System.out

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L - %m%n

# Root logger option

log4j.rootLogger=INFO, file, stdout

# Log everything. Good for troubleshooting

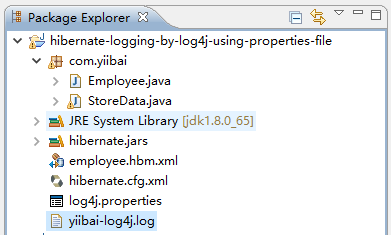
log4j.logger.org.hibernate=INFO

# Log all JDBC parameters

log4j.logger.org.hibernate.type=ALL

Shell

完整的目录结构如下所示 -



提示： 如果未加入

log4j.jar ， 就无法生成 yiibai-log4j.log 文件。

# Hibernate继承映射

我们可以将继承层次结构类与数据库的表映射。 在hibernate中定义了三个继承映射策略：

* 每个层次类一张表
* 每个具体类一张表
* 每个子类一张表

## 1. 表每个层次类

在每个层次映射一张表中，需要单个表来映射整个层次结构，添加一个额外的列(称为标识符列)来标识该类。 但可空(null)值存储在表中。

* [使用xml文件的每个类层次一张表](http://www.yiibai.com/hibernate/table-per-hierarchy-example-using-xml-file.html)
* [使用注释的表每个类层次一张表](http://www.yiibai.com/hibernate/table-per-hierarchy-using-annotation-tutorial-example.html)

## 2. 每个具体类一张表

在每个具体类一张表的情况下，按照类创建表。 但是在子类表中添加了重复列。

* [使用XML文件的每个具体类一张表](http://www.yiibai.com/hibernate/table-per-concrete-class.html)
* [使用注释的每个具体类一张表](http://www.yiibai.com/hibernate/table-per-concrete-class-using-annotation-tutorial-example.html)

## 3. 每个子类一张表

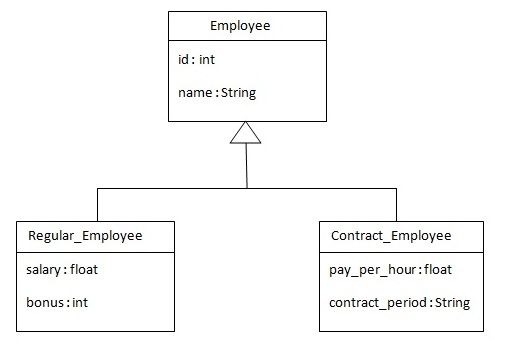
在这个策略中，表是根据类创建的，但是由外键相关。 所以没有重复的列。

* [使用xml文件的每个子类一张表](http://www.yiibai.com/hibernate/table-per-subclass)
* [使用注释的每个子类一张表](http://www.yiibai.com/hibernate/table-per-subclass-using-annotation-tutorial-example.html)

# Hibernate使用xml文件的每个类层次一张表

通过这种继承策略，我们可以通过单表映射整个层次结构。 这里，在表中创建一个额外的列(也称为discriminator列)来标识该类。

让我们先了解问题。下面给出的整个层次类映射到数据库的一个表中图解说明。



这个层次结构中有三个类。Employee是Regular\_Employee和Contract\_Employee类的超类(父类)。 让我们来看看这个层次结构类的映射文件。

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

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name="com.yiibai.Employee" table="emp121"

discriminator-value="emp">

<id name="id">

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

</id>

<discriminator column="type" type="string"></discriminator>

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

<subclass name="com.yiibai.Regular\_Employee"

discriminator-value="reg\_emp">

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

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

</subclass>

<subclass name="com.yiibai.Contract\_Employee"

discriminator-value="con\_emp">

<property name="pay\_per\_hour"></property>

<property name="contract\_duration"></property>

</subclass>

</class>

</hibernate-mapping>

XML

在每个类层次结构一个表的情况下，hibernate框架添加了一个标识符列，该框架指定了记录的类型。 它主要用于区分记录。要指定它，必须指定类的 discriminator 子元素。  
类的子类subelement指定子类。 在这种情况下，Regular\_Employee和Contract\_Employee是Employee类的子类。

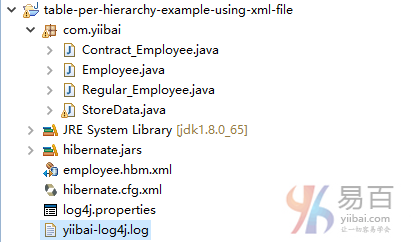
此层次结构的表结构如下所示：

| **列名** | **数据类型** | **是否为空** | **默认值** | **是否主键** |
| --- | --- | --- | --- | --- |
| id | int(10) | 否 | - | 是 |
| type | varchar(254) | 否 | - | - |
| name | varchar(254) | 是 | - | - |
| salary | float | 是 | - | - |
| bonus | int(10) | 是 | - | - |
| pay\_per\_hour | float | 是 | - | - |
| contract\_duration |  | 是 | - | - |

## 每个类层次结构一个表示例

在这个例子中，我们创建了三个类，并在employee.hbm.xml文件中提供了这些类的映射。

完整的工程目录如下所示 -



下载代码的项目名称：table-per-hierarchy-example-using-xml-file

### 1)创建持久类

您需要创建表示继承的持久化类。让我们为上面的层次结构创建三个类：

文件: Employee.java

package com.yiibai;

public class Employee {

private int id;

private String name;

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;

}

}

Java

文件: Regular\_Employee.java

package com.yiibai;

public class Regular\_Employee extends Employee {

private float salary;

private int bonus;

public float getSalary() {

return salary;

}

public void setSalary(float salary) {

this.salary = salary;

}

public int getBonus() {

return bonus;

}

public void setBonus(int bonus) {

this.bonus = bonus;

}

}

Java

文件: Contract\_Employee.java

package com.yiibai;

public class Contract\_Employee extends Employee {

private float pay\_per\_hour;

private String contract\_duration;

public float getPay\_per\_hour() {

return pay\_per\_hour;

}

public void setPay\_per\_hour(float payPerHour) {

pay\_per\_hour = payPerHour;

}

public String getContract\_duration() {

return contract\_duration;

}

public void setContract\_duration(String contractDuration) {

contract\_duration = contractDuration;

}

}

Java

### 2)创建持久化类的映射文件

上面我们已经了解了层次结构，下面来看看如何配置映射文件。

文件: employee.hbm.xml

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

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name="com.yiibai.Employee" table="emp121"

discriminator-value="emp">

<id name="id">

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

</id>

<discriminator column="type" type="string"></discriminator>

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

<subclass name="com.yiibai.Regular\_Employee"

discriminator-value="reg\_emp">

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

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

</subclass>

<subclass name="com.yiibai.Contract\_Employee"

discriminator-value="con\_emp">

<property name="pay\_per\_hour"></property>

<property name="contract\_duration"></property>

</subclass>

</class>

</hibernate-mapping>

XML

### 3)在配置文件中添加hbm文件的映射

打开hibernate.cfg.xml文件，并添加如下映射资源的条目：

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

XML

现在配置文件将如下所示：

文件：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">

<!-- Generated by MyEclipse Hibernate Tools. -->

<hibernate-configuration>

<session-factory>

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

<property name="connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="connection.url">jdbc:mysql://localhost:3306/test</property>

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

<property name="connection.password">123456</property>

<property name="dialect">org.hibernate.dialect.MySQL5InnoDBDialect</property>

<property name="show\_sql">true</property>

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

</session-factory>

</hibernate-configuration>

XML

hbm2ddl.auto属性定义是用于在数据库中创建自动表。

## 4)创建存储持久对象的类

在这个类中，我们只是将Employee对象存储到数据库表中。

文件：StoreData.java

package com.yiibai;

import org.hibernate.\*;

import org.hibernate.boot.MetadataSources;

import org.hibernate.boot.registry.StandardServiceRegistry;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import org.hibernate.cfg.\*;

public class StoreData {

public static void main(String[] args) {

// Session session = new

// AnnotationConfiguration().configure().buildSessionFactory().openSession();

final StandardServiceRegistry registry = new StandardServiceRegistryBuilder()

.configure("hibernate.cfg.xml").build();

// 2. 根据服务注册类创建一个元数据资源集，同时构建元数据并生成应用一般唯一的的session工厂

SessionFactory sessionFactory = new MetadataSources(registry)

.buildMetadata().buildSessionFactory();

/\*\*\*\* 上面是配置准备，下面开始我们的数据库操作 \*\*\*\*\*\*/

Session session = sessionFactory.openSession();// 从会话工厂获取一个session

Transaction t = session.beginTransaction();

Employee e1 = new Employee();

e1.setName("Yiibai");

Regular\_Employee e2 = new Regular\_Employee();

e2.setName("Max su");

e2.setSalary(50000);

e2.setBonus(5);

Contract\_Employee e3 = new Contract\_Employee();

e3.setName("Hippo su");

e3.setPay\_per\_hour(1000);

e3.setContract\_duration("15 hours");

session.persist(e1);

session.persist(e2);

session.persist(e3);

t.commit();

session.close();

System.out.println("success");

}

}

Java

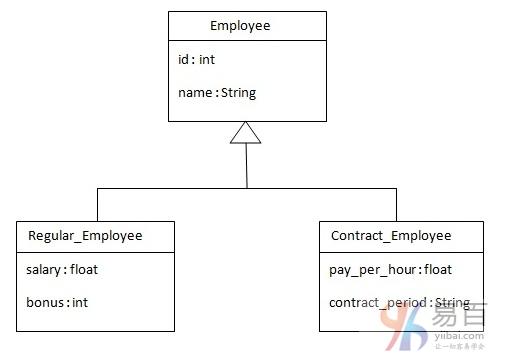
运行上面程序，并打数据库表查看执行结果。

# Hibernate每个具体类一张表映射(使用XML)

在每个具体类一个表中，数据库中将有三个表但彼此之间没有关系(关联)。 根据具体类策略将表格映射到表有两种方法。

* 由union-subclass元素指定
* 通过自我为每个类创建表

我们来了解映射的层次结构。



下面来看看看我们如何通过union-subclass元素映射这个层次结构， employee.hbm.xml文件的内容如下

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

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<!--// by http://www.yiibai.com/hibernate -->

<hibernate-mapping>

<class name="com.yiibai.Employee" table="emp122">

<id name="id">

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

</id>

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

<union-subclass name="com.yiibai.Regular\_Employee"

table="regemp122">

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

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

</union-subclass>

<union-subclass name="com.yiibai.Contract\_Employee"

table="contemp122">

<property name="pay\_per\_hour"></property>

<property name="contract\_duration"></property>

</union-subclass>

</class>

</hibernate-mapping>

XML

在每个具体类一张表的情况下，数据库中将有三个表，每个表表示一个特定的类。  
类的union-subclass子元素指定子类。 它将父表的列添加到此表中。换句话说，将它们作为一个联合。

**每个表的格结构如下：**  
Employee类的表结构 -

CREATE TABLE `emp122` (

`id` int(11) NOT NULL,

`name` varchar(255) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

SQL

Regular\_Employee类的表结构 -

CREATE TABLE `regemp122` (

`id` int(11) NOT NULL,

`name` varchar(255) DEFAULT NULL,

`salary` float DEFAULT NULL,

`bonus` int(11) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

SQL

Contract\_Employee类的表结构 -

CREATE TABLE `contemp122` (

`id` int(11) NOT NULL,

`name` varchar(255) DEFAULT NULL,

`pay\_per\_hour` float DEFAULT NULL,

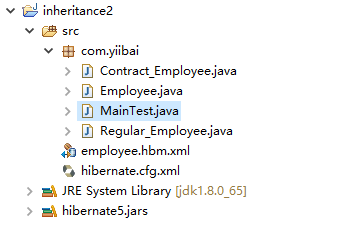
`contract\_duration` varchar(255) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

SQL

## 每个具体类的表示例

在这个例子中，我们创建了三个类，并在employee.hbm.xml文件中提供了这些类的映射。创建一个项目：inheritance2， 完整的项目结构如下 -  


### 1)创建持久类

您需要创建表示继承的持久化类。 让我们为上面的层次结构创建三个类：

文件：Employee.java

package com.yiibai;

public class Employee {

private int id;

private String name;

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;

}

}

Java

文件：Regular\_Employee.java

package com.yiibai;

/\*\*

\*

\* @author by maxsu

\* @copyright http://www.yiibai.com

\* @link download at: http://www.yiibai.com/siteinfo/download.html

\*/

public class Regular\_Employee extends Employee {

private float salary;

private int bonus;

public float getSalary() {

return salary;

}

public void setSalary(float salary) {

this.salary = salary;

}

public int getBonus() {

return bonus;

}

public void setBonus(int bonus) {

this.bonus = bonus;

}

}

Java

文件：Contract\_Employee.java

package com.yiibai;

/\*\*

\*

\* @author by maxsu

\* @copyright http://www.yiibai.com

\* @link download at: http://www.yiibai.com/siteinfo/download.html

\*/

public class Contract\_Employee extends Employee {

private float pay\_per\_hour;

private String contract\_duration;

public float getPay\_per\_hour() {

return pay\_per\_hour;

}

public void setPay\_per\_hour(float payPerHour) {

pay\_per\_hour = payPerHour;

}

public String getContract\_duration() {

return contract\_duration;

}

public void setContract\_duration(String contractDuration) {

contract\_duration = contractDuration;

}

}

Java

### 2)创建持久类的映射文件

上面已经了解了层次结构，接下来看看映射配置。

文件：employee.hbm.xml

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

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name="com.yiibai.Employee" table="emp122">

<id name="id">

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

</id>

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

<union-subclass name="com.yiibai.Regular\_Employee"

table="regemp122">

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

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

</union-subclass>

<union-subclass name="com.yiibai.Contract\_Employee"

table="contemp122">

<property name="pay\_per\_hour"></property>

<property name="contract\_duration"></property>

</union-subclass>

</class>

</hibernate-mapping>

XML

### 3)在配置文件中添加hbm文件的映射

打开hibernate.cfg.xml文件，并添加如下映射资源的项：

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

XML

现在配置文件将如下所示：

文件：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">

<!-- Generated by MyEclipse Hibernate Tools. -->

<hibernate-configuration>

<session-factory>

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

<property name="connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="connection.url">jdbc:mysql://localhost:3306/test</property>

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

<property name="connection.password">123456</property>

<property name="dialect">org.hibernate.dialect.MySQL5InnoDBDialect</property>

<property name="show\_sql">true</property>

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

</session-factory>

</hibernate-configuration>

XML

hbm2ddl.auto属性定义是用于在数据库中创建自动表。

### 4)创建存储持久对象的类

在这个类中，我们只是将Employee 对象存储在数据库表中。

文件：MainTest.java

package com.yiibai;

import org.hibernate.\*;

import org.hibernate.boot.MetadataSources;

import org.hibernate.boot.registry.StandardServiceRegistry;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

/\*\*

\*

\* @author by maxsu

\* @copyright http://www.yiibai.com

\* @link download at: http://www.yiibai.com/siteinfo/download.html

\*/

public class MainTest {

public static void main(String[] args) {

// 但在5.1.0版本汇总，hibernate则采用如下新方式获取：

// 1. 配置类型安全的准服务注册类，这是当前应用的单例对象，不作修改，所以声明为final

// 在configure("cfg/hibernate.cfg.xml")方法中，如果不指定资源路径，默认在类路径下寻找名为hibernate.cfg.xml的文件

final StandardServiceRegistry registry = new StandardServiceRegistryBuilder()

.configure("hibernate.cfg.xml").build();

// 2. 根据服务注册类创建一个元数据资源集，同时构建元数据并生成应用一般唯一的的session工厂

SessionFactory sessionFactory = new MetadataSources(registry)

.buildMetadata().buildSessionFactory();

/\*\*\*\* 上面是配置准备，下面开始我们的数据库操作 \*\*\*\*\*\*/

Session session = sessionFactory.openSession();// 从会话工厂获取一个session

// creating transaction object

Transaction t = session.beginTransaction();

Employee e1 = new Employee();

e1.setName("用户名-01");

Regular\_Employee e2 = new Regular\_Employee();

e2.setName("yiibai su");

e2.setSalary(50002);

e2.setBonus(5);

Contract\_Employee e3 = new Contract\_Employee();

e3.setName("Mina su");

e3.setPay\_per\_hour(1010);

e3.setContract\_duration("15 hours");

session.persist(e1);

session.persist(e2);

session.persist(e3);

t.commit();

session.close();

System.out.println("success");

}

}

Java

执行上面代码运行测试即可，应该会自动创建三张表，并插入数据。