钱进培训是哈法地区资深工程师组成的培训机构，通过各位老师的现身说法，帮助各位学员迅速掌握实战知识，为求职打下坚实的基础。电子邮件：jin.qian.canada@gmail.com钱老师报名、答疑微信号：qianjincanada，或扫描以下二维码添加：

钱老师

加拿大

扫一扫上面的二维码图案，加我微信

# Java高级速成班

本课程针对有一定编成基础，希望在短时间内对Java企业级开发有所了解的同学。

通过本课程的学习，学员能够顺利掌握J2EE的体系结构，了解常见的Java框架并熟练使用，具体内容包括:

1. B/s体系结构，HTTP协议栈相关内容（HTML，CSS，JS）
2. 数据库访问的不同方法（Hibernate使用）
3. Spring IOC在企业开发中的应用
4. Restful API/SOAP开发及应用
5. SVN/GIT/MAVEN的应用

# 第三讲：数据库相关

# 复习题

1. access_log, 每个IP访问量，根据访问量排序
    1. POJO
    2. HashMap创建、添加、遍历
    3. Comparable，Comparator

2. HTML
Table/打印一个表格的内容

3.实现一个SMTP客户端

4.JAXB,xml文件和java对象的转换

5.GSON，json文件和java对象的互换

6.MAVEN

相关软件

安装mysql

https://dev.mysql.com/downloads/mysql/

记下root用户的密码

安装heidisql

http://www.heidisql.com/download.php

数据库（Database）是按照数据结构来组织、存储和管理数据的仓库，

每个数据库都有一个或多个不同的API用于创建，访问，管理，搜索和复制所保存的数据。

我们也可以将数据存储在文件中，但是在文件中读写数据速度相对较慢。

所以，现在我们使用关系型数据库管理系统（RDBMS）来存储和管理的大数据量。所谓的关系型数据库，是建立在关系模型基础上的数据库，借助于集合代数等数学概念和方法来处理数据库中的数据。

RDBMS即关系数据库管理系统(Relational Database Management System)的特点：

1.数据以表格的形式出现

2.每行为各种记录名称

3.每列为记录名称所对应的数据域

4.许多的行和列组成一张表单

5.若干的表单组成database

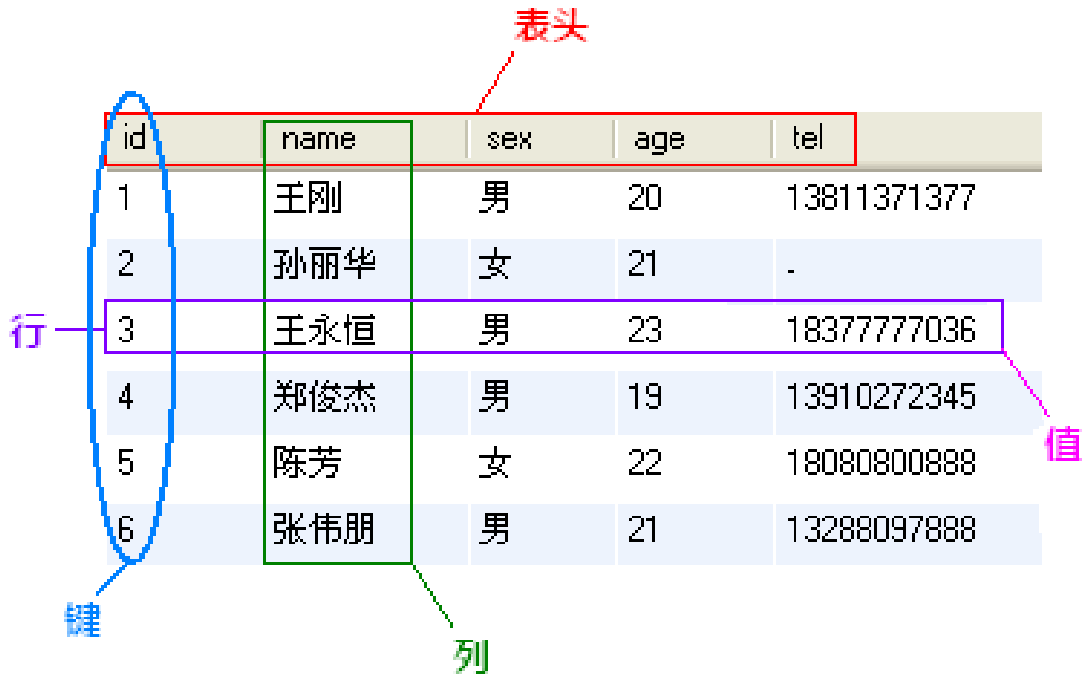
表头(header): 每一列的名称;

列(row): 具有相同数据类型的数据的集合;

行(col): 每一行用来描述某个人/物的具体信息;

值(value): 行的具体信息, 每个值必须与该列的数据类型相同;

键(key): 表中用来识别某个特定的人\物的方法, 键的值在当前列中具有唯一性。




某班级学生信息

**SQL 是用于访问和处理数据库的标准的计算机语言。**

什么是 SQL？

SQL 指结构化查询语言

SQL 使我们有能力访问数据库

SQL 是一种 ANSI 的标准计算机语言

ANSI：美国国家标准学会（AMERICAN NATIONAL STANDARDS INSTITUTE: ANSI）成立于1918年。当时，美国的许多企业和专业技术团体，已开始了标准化工作，但因彼此间没有协调，存在不少矛盾和问题。为了进一步提高效率，数百个科技学会、协会组织和团体，均认为有必要成立一个专门的标准化机构，并制订统一的通用标准。

**SQL 能做什么？**

SQL 面向数据库执行查询

SQL 可从数据库取回数据

SQL 可在数据库中插入新的记录

SQL 可更新数据库中的数据

SQL 可从数据库删除记录

SQL 可创建新数据库

SQL 可在数据库中创建新表

SQL 可在数据库中创建存储过程

SQL 可在数据库中创建视图

SQL 可以设置表、存储过程和视图的权限

RDBMS

RDBMS 指的是关系型数据库管理系统。

RDBMS 是 SQL 的基础，同样也是所有现代数据库系统的基础，比如 MS SQL Server, IBM DB2, Oracle, MySQL 以及 Microsoft Access。

RDBMS 中的数据存储在被称为表（tables）的数据库对象中。

表是相关的数据项的集合，它由列和行组成。

# 关系数据库标准语言SQL

1. SQL的主要标准

SQL-86

SQL-89

SQL-92(SQL2)

SQL-99(SQL3)

SQL-86。SQL的第一个标准是1986年10月由美国国家标准化组织(ANSI)公布的。

SQL-89。ANSI以后通过对SQL-86的不断修改和完善，于1989年第二次公布了SQL标准，即SQL-89，该标准增强了完整性的语言特征。

SQL-92(SQL2)。1992年又公布了SQL-92标准，该标准增加了支持对远程数据库的访问，扩充了数据类型、操作类型、动态SQL等许多新的特征。

SQL-99(SQL3)。完成于1999年的SQL-99修订本具有更高级的特征。引入了支持对象-关系DBMS模型的SQL，扩展了对象、递归、触发等许多新的特征，支持用户自定义函数、自定义数据类型。

SQL的功能特点

功能：

数据定义

数据查询

数据操纵

数据控制

数据库表

一个数据库通常包含一个或多个表。每个表由一个名字标识（例如“客户”或者“订单”）。表包含带有数据的记录（行）。

下面的例子是一个名为 "Persons" 的表：

| Id | LastName | FirstName | Address | City |
|---|---|---|---|---|
| 1 | Adams | John | Oxford Street | London |
| 2 | Bush | George | Fifth Avenue | New York |
| 3 | Carter | Thomas | Changan Street | Beijing |

上面的表包含三条记录（每一条对应一个人）和五个列（Id、姓、名、地址和城市）。

需要在数据库上执行的大部分工作都由 SQL 语句完成。

下面的语句从表中选取 LastName 列的数据：

```
SELECT LastName FROM Persons
```

SQL DML 和 DDL

可以把 SQL 分为两个部分：数据操作语言 (DML) 和 数据定义语言 (DDL)。

SQL (结构化查询语言)是用于执行查询的语法。但是 SQL 语言也包含用于更新、插入和删除记录的语法。

查询和更新指令构成了 SQL 的 DML 部分：

SELECT - 从数据库表中获取数据

UPDATE - 更新数据库表中的数据

DELETE - 从数据库表中删除数据

INSERT INTO - 向数据库表中插入数据

SQL 的数据定义语言 (DDL) 部分使我们有能力创建或删除表格。我们也可以定义索引（键），规定表之间的链接，以及施加表间的约束。

SQL 中最重要的 DDL 语句:

CREATE DATABASE - 创建新数据库

ALTER DATABASE - 修改数据库

CREATE TABLE - 创建新表

ALTER TABLE - 变更（改变）数据库表

DROP TABLE - 删除表

CREATE INDEX - 创建索引（搜索键）

DROP INDEX - 删除索引

SQL SELECT 语句

SELECT 语句用于从表中选取数据。

结果被存储在一个结果表中（称为结果集）。

SQL SELECT 语法

SELECT 列名称 FROM 表名称

以及：

SELECT * FROM 表名称

注释：SQL 语句对大小写不敏感。SELECT 等效于 select。

https://www.sqlteaching.com/

https://www.codecademy.com/zh/learn/learn-sql

Mysql是最流行的关系型数据库管理系统，在WEB应用方面MySQL是最好的RDBMS(Relational Database Management System：关系数据库管理系统)应用软件之一。

数据库: 数据库是一些关联表的集合。.

数据表: 表是数据的矩阵。在一个数据库中的表看起来像一个简单的电子表格。

列: 一列(数据元素) 包含了相同的数据, 例如邮政编码的数据。

行：一行（=元组，或记录）是一组相关的数据，例如一条用户订阅的数据。

冗余：存储两倍数据，冗余可以使系统速度更快。

主键：主键是唯一的。一个数据表中只能包含一个主键。你可以使用主键来查询数据。

外键：外键用于关联两个表。

复合键：复合键（组合键）将多个列作为一个索引键，一般用于复合索引。

索引：使用索引可快速访问数据库表中的特定信息。索引是对数据库表中一列或多列的值进行排序的一种结构。类似于书籍的目录。

MySQL是一个关系型数据库管理系统，由瑞典MySQL AB公司开发，目前属于Oracle公司。MySQL是一种关联数据库管理系统，关联数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。

Mysql是开源的，所以你不需要支付额外的费用。

Mysql支持大型的数据库。可以处理拥有上千万条记录的大型数据库。

MySQL使用标准的SQL数据语言形式。

Mysql可以允许于多个系统上，并且支持多种语言。这些编程语言包括C、C++、Python、Java、Perl、PHP、Eiffel、Ruby和Tcl等。

Mysql对PHP有很好的支持，PHP是目前最流行的Web开发语言。

MySQL支持大型数据库，支持5000万条记录的数据仓库，32位系统表文件最大可支持4GB，64位系统支持最大的表文件为8TB。

Mysql是可以定制的，采用了GPL协议，你可以修改源码来开发自己的Mysql系统

可以通过以下命令来连接到Mysql服务器：

```
[root@host]# mysql -u root -p
Enter password:*******
```

mysql -h 主机名 -u 用户名 -p

-h : 该命令用于指定客户端所要登录的MySQL主机名, 登录当前机器该参数可以省略;
-u : 所要登录的用户名;
-p : 告诉服务器将会使用一个密码来登录, 如果所要登录的用户名密码为空, 可以忽略此选项。

USE 数据库名 :
选择要操作的Mysql数据库，使用该命令后所有Mysql命令都只针对该数据库。

SHOW DATABASES:
列出 MySQL 数据库管理系统的数据库列表。

SHOW TABLES:
显示指定数据库的所有表，使用该命令前需要使用 use 命令来选择要操作的数据库。

MySQL有三大类数据类型, 分别为数字、日期\时间、字符串, 这三大类中又更细致的划分了许多子类型:

数字类型

整数: tinyint、smallint、mediumint、int、bigint

浮点数: float、double、real、decimal

日期和时间: date、time、datetime、timestamp、year

字符串类型

字符串: char、varchar

文本: tinytext、text、mediumtext、longtext

二进制(可用来存储图片、音乐等): tinyblob、blob、mediumblob、longblobd

JDBC

JDBC（Java Data Base Connectivity,java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系数据库提供统一访问，它由一组用Java语言编写的类和接口组成。JDBC为工具/数据库开发人员提供了一个标准的API，据此可以构建更高级的工具和接口，使数据库开发人员能够用纯 Java API 编写数据库应用程序。

有了JDBC，向各种关系数据发送SQL语句就是一件很容易的事。换言之，有了JDBC API，就不必为访问Sybase数据库专门写一个程序，为访问Oracle数据库又专门写一个程序，或为访问Informix数据库又编写另一个程序等等，程序员只需用JDBC API写一个程序就够了，它可向相应数据库发送SQL调用。


问题的重点在于，你的应用程序如何调用这组程序库？不同的数据库通常会有不同的通信协议，用以连接不同数据库的程序库在API 上也会有所不同，如果应用程序直接使用这些程序库，例如：

```
XySqlConnection conn = new XySqlConnection("localhost", "root", "1234");
conn.selectDB("gossip");
XySqlQuery query = conn.query("SELECT * FROM T_USER");
```

假设这段代码中的API 是某Xy 数据库厂商程序库所提供，应用程序中要使用到数据库连接时，都会直接调用这些API，若哪天应用程序打算改用Ab 厂商数据库及其提供的数据库连接API，那就得修改相关的代码。


应用程序
JDBC 标准 API
JDBC 驱动程序
通信协议
数据库

JDBC 标准主要分为两个部分： JDBC 应用程序开发者接口(ApplicationDeveloper Interface)以及JDBC 驱动程序开发者接口(Driver Developer Interface)。如果应用程序需要连接数据库，就是调用JDBC 应用程序开发者接口，相关API 主要在java.sql 与javax.sql 两个包中。JDBC 驱动程序开发者接口则是数据库厂商要实现驱动程序时的规范，一般开发者并不用了解。

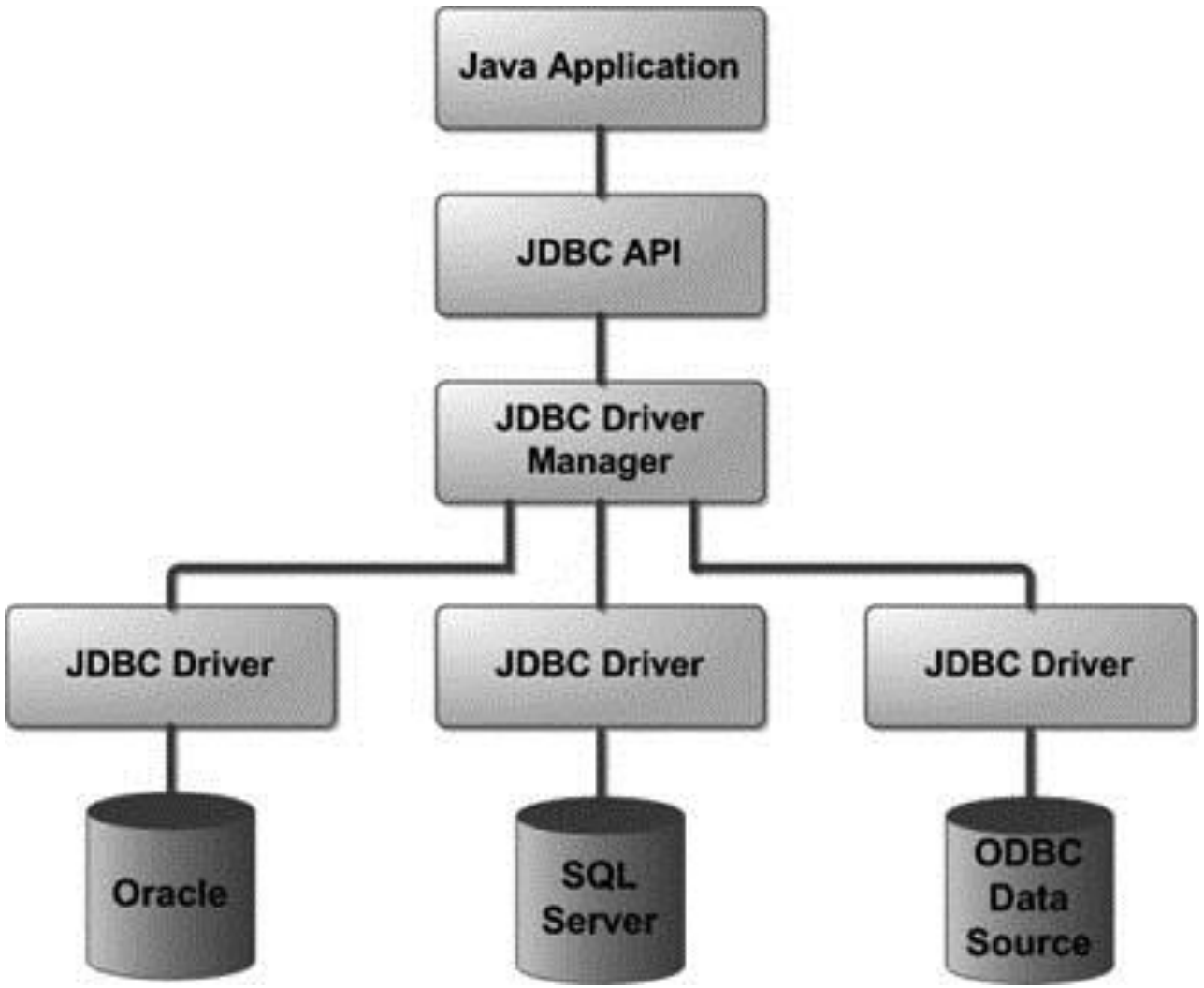
```
Connection conn = DriverManager.getConnection(…);

Statement st = conn.createStatement();

ResultSet rs = st.executeQuery("SELECT * FROM T_USER");
```

假设这段代码是连接MySQL 数据库，你会需要在Classpath 中设置MySQL 对应JDBC的驱动程序。具体来说，就是在Classpath 中设置一个JAR文件，此时应用程序、JDBC 与数据库的关系如下图所示。

dd


Java Application
JDBC API
JDBC Driver Manager
JDBC Driver
JDBC Driver
JDBC Driver
Oracle
SQL Server
ODBC Data Source

1、加载JDBC驱动程序：

在连接数据库之前，首先要加载想要连接的数据库的驱动到JVM（Java虚拟机），

这通过java.lang.Class类的静态方法forName(String  className)实现。

例如：

```
try{
//加载MySql的驱动类
Class.forName("com.mysql.jdbc.Driver") ;
}catch(ClassNotFoundException e){
System.out.println("找不到驱动程序类，加载驱动失败！");
e.printStackTrace() ;
}
```

成功加载后，会将Driver类的实例注册到DriverManager类中。

2、提供JDBC连接的URL

- 连接URL定义了连接数据库时的协议、子协议、数据源标识。
- 书写形式：协议：子协议：数据源标识

协议：在JDBC中总是以jdbc开始

子协议：是桥连接的驱动程序或是数据库管理系统名称。

数据源标识：标记找到数据库来源的地址与连接端口。

例如：（MySql的连接URL）

jdbc:mysql:

//localhost:3306/test?useUnicode=true&characterEncoding=gbk ;

useUnicode=true：表示使用Unicode字符集。如果characterEncoding设置为gb2312或GBK，本参数必须设置为true 。characterEncoding=gbk：字符编码方式

3、创建数据库的连接

- 要连接数据库，需要向java.sql.DriverManager请求并获得Connection对象，该对象就代表一个数据库的连接。
- 使用DriverManager的getConnectin(String url , String username , String password )方法传入指定的欲连接的数据库的路径、数据库的用户名和密码来获得。

例如：

```
//连接MySql数据库，用户名和密码都是root
String url = "jdbc:mysql://localhost:3306/test" ;
String username = "root" ;
String password = "root" ;
try{
Connection con =
        DriverManager.getConnection(url , username , password ) ;
}catch(SQLException se){
System.out.println("数据库连接失败！ ");
se.printStackTrace() ;
}
```

4、创建一个Statement

- 要执行SQL语句，必须获得java.sql.Statement实例，Statement实例分为以下3种类型：

1、执行静态SQL语句。通常通过Statement实例实现。

2、执行动态SQL语句。通常通过PreparedStatement实例实现。

3、执行数据库存储过程。通常通过CallableStatement实例实现。

具体的实现方式：

```
Statement stmt = con.createStatement() ;
PreparedStatement pstmt = con.prepareStatement(sql) ;
CallableStatement cstmt =
        con.prepareCall("{CALL demoSp(? , ?)}") ;
```

5、执行SQL语句

Statement接口提供了三种执行SQL语句的方法：executeQuery 、executeUpdate和execute

1、ResultSet executeQuery(String sqlString)：执行查询数据库的SQL语句，返回一个结果集（ResultSet）对象。

2、int executeUpdate(String sqlString)：用于执行INSERT、UPDATE或DELETE语句以及SQL DDL语句，如：CREATE TABLE和DROP TABLE等

3、execute(sqlString):用于执行返回多个结果集、多个更新计数或二者组合的语句。

具体实现的代码：

```
ResultSet rs = stmt.executeQuery("SELECT * FROM ...") ;
int rows = stmt.executeUpdate("INSERT INTO ...") ;
boolean flag = stmt.execute(String sql) ;
```

6、处理结果

两种情况：

1、执行更新返回的是本次操作影响到的记录数。

2、执行查询返回的结果是一个ResultSet对象。

- ResultSet包含符合SQL语句中条件的所有行，并且它通过一套get方法提供了对这些行中数据的访问。
- 使用结果集（ResultSet）对象的访问方法获取数据：

```
while(rs.next()){
    String name = rs.getString("name") ;
String pass = rs.getString(1) ; // 此方法比较高效
}
```

（列是从左到右编号的，并且从列1开始

7、关闭JDBC对象

操作完成以后要把所有使用的JDBC对象全都关闭，以释放JDBC资源，关闭顺序和声明顺序相反：

1、关闭记录集

2、关闭声明

3、关闭连接对象

```
 if(rs != null){   // 关闭记录集
try{
   rs.close() ;
}catch(SQLException e){
   e.printStackTrace() ;
}
 }
 if(stmt != null){   // 关闭声明
try{
   stmt.close() ;
}catch(SQLException e){
   e.printStackTrace() ;
}
 }
 if(conn != null){  // 关闭连接对象
try{
   conn.close() ;
}catch(SQLException e){
   e.printStackTrace() ;
}
 }
```

对象-关系映射（Object/Relation Mapping，简称ORM），是随着面向对象的软件开发方法发展而产生的。面向对象的开发方法是当今企业级应用开发环境中的主流开发方法，关系数据库是企业级应用环境中永久存放数据的主流数据存储系统。对象和关系数据是业务实体的两种表现形式，业务实体在内存中表现为对象，在数据库中表现为关系数据。内存中的对象之间存在关联和继承关系，而在数据库中，关系数据无法直接表达多对多关联和继承关系。因此，对象-关系映射(ORM)系统一般以中间件的形式存在，主要实现程序对象到关系数据库数据的映射。

传统的Java数据库操作，是使用jdbc将sql语句嵌入在java代码中执行，查询获取到的结果形式类似于数组或者键值对。这样的编写方式将使我们的程序面临如下问题：

每一个操作都要编写对应的sql语句 —— 开发效率低

sql语句是以字符串的形式出现在java代码中，当字段名或表名修改，将会影响所有相关的sql代码 —— 维护效率低

如果编写sql语句时没有注意参照sql标准，极可能编写出针对当前所用数据库的扩展sql语句，这将导致后期更换数据库困难 —— 数据库移植性差

而Hibernate的出现，解决了上面所提出的问题。Hibernate使用对象和表映射的方式，可以采用xml配置和注解配置两种方式来说明对象和表、对象属性和表字段的映射关系。

当操作数据库时，仅需要调用对应api即可实现自动生成sql语句，极大的提高了开发效率。当数据库结构发生改动时，仅需要调整实体属性和xml/注解配置即可，大大降低了维护的工作量。过程中，可以完全避免手动编写sql语句，仅修改配置即可完成切换数据库。

创建hibernate.cfg.xml配置文件

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
 <session-factory>
    <property name="connection.user">root</property>
    <property name="connection.password">root</property>
    <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
    <property
name="connection.url">jdbc:mysql://localhost:3306/hibernate?useUnicode=true&characterEncoding=UTF-
8</property>
    <property name="dialect">org.hibernate.dialect.MySQLDialect</property>

    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <property name="hbm2ddl.auto">update</property>
 </session-factory>
</hibernate-configuration>
```

```
configuration.setProperty("hibernate.connection.driver_class", "org.h2.Driver");
configuration.setProperty("hibernate.connection.url", "jdbc:h2:mem:testdb");
configuration.setProperty("hibernate.connection.username", "sa");
configuration.setProperty("hibernate.connection.password", "");
configuration.setProperty("hibernate.dialect","org.hibernate.dialect.H2Dialect");
configuration.setProperty("hibernate.show_sql", "true");
```

创建持久化类(javabean)

```
@Entity                          //注解Entity表示该类能被Hibernate持久化
@Table(name = "tb_cat")          //指定该Entity对应的数据表名
public class Cat {

   @Id                           //指定该列为主键。主键类型最好不要使用int等原始类型
   @GeneratedValue(strategy = GenerationType.AUTO) //主键类型auto表示该主键为自增长型
   private Integer id;

   @Column(name = "name")        //指定该属性对应的数据库表的列为name，列名与属性名一样时这句注解可省略
   private String name;

   @Column(name = "description")
   private String description;
```

Session session = new HibernateUtil().getSessionFactory().openSession(); //获取session并open，开启一个Hibernate会话

Transaction trans = session.beginTransaction(); //开启一个事务

session.persist(mother); //将mother保存到数据库

trans.commit(); //提交事务

session.close(); //关闭Hibernate会话

- **hibernate的执行流程**

配置对象Configuration

作用：读取配置文档：Hibernate.cfg.xml；

创建SessionFactory对象

作用：读取相应的里面加载的对象关系映射文件

创建session文件

作用：类似于JDBC中的Connection,这个session对象等同于连接对象

执行增删改查：save，delete，update，createQuery等。

执行某个session对象的方法的时候，必须开启一个事物transaction，这些方法需要封装在事物当中。

执行完成方法之后，需要提交事务并且关闭session

transaction.commit();

session.close();

Configuration类（管理Hibernate的配置信息）

Configuration 类负责管理 Hibernate 的配置信息。包括如下内容：

加载hibernate.properties 和 hibernate.cfg.xml
持久化类与数据表的映射关系（*.hbm.xml文件）
创建Configuration 的两种方式：

属性文件（hibernate.properties）：
Configuration cfg = new Configuration(); //手动加载hbm

Xml文件（hibernate.cfg.xml）
Configuration cfg = new Configuration().configure();

通过Configuration对象 addResource方法添加hbm文件映射

```
// 加载位于cn.itcast.domain包下面Customer.hbm.xml文件
configuration.addResource("cn/itcast/domain/Customer.hbm.xml");
```

另一种方式，通过addClass添加持久化类，Hibernate会在类所在包 自动搜索hbm 映射文件

```
// 加载位于cn.itcast.domain.Customer 持久化类，让Hibernate自己搜索配置文件
configuration.addClass(Customer.class);
```

SessionFactory接口（获取Session对象）

Configuration对象根据当前的配置信息生成SessionFactory对象

SessionFactory对象中保存了当前数据库配置信息和所有映射关系以及预定义的SQL语句

SessionFactory对象是线程安全的

SessionFactory还负责维护Hibernate的二级缓存

Configuration configuration = new Configuration().configure();

创建sessionFactory

SessionFactory sessionFactory = configuration.buildSessionFactory();

可以通过SessionFactory对象 获得Session对象

Session session = sessionFactory.openSession();

或者

Session session = sessionFactory.getCurrentSession();

# openSession与getCurrentSession的区别

openSession 每次使用都是创建一个新的session；getCurrentSession 是获取当前session对象，连续使用多次时，得到的session都是同一个对象。

getCurrentSession在事物提交或者回滚之后会自动关闭；而openSession需要手动关闭，如果使用openSession而没有手动关闭，多次使用之后会导致连接池溢出。

一般在实际开发中，往往使用getCurrentSession多，因为一般是处理同一个事务，所以在一般情况下比较少使用openSession。

构造SessionFactory 很消耗资源，一般情况下一个应用只初始化一个

# Session接口（CRUD操作）

调用Session里面的方法，实现crud操作。

Session是单线程对象，只能有一个操作时候，不能同时多个操作使用，不要把Session变量定义成成员变量，每次使用都创建新对象，相当于JDBC的Connection。

Session是应用程序与数据库之间交互操作的一个单线程对象，是 Hibernate 运作的中心；Session是线程不安全的。

所有持久化对象必须在Session 的管理下才可以进行持久化操作。

Session 对象有一个一级缓存，显式执行 flush 之前，所有的持久化操作的数据都缓存在 session 对象处。

持久化类与 Session 关联起来后就具有了持久化的能力。

| 方法 | 作用 |
| --- | --- |
| save()/persist()、update()、saveOrUpdate() | 增加和修改对象 |
| delete() | 删除对象 |
| get()/load() | 根据主键查询 |
| createQuery()、createSQLQuery() | 数据库操作对象 |
| createCriteria | 条件查询 |